

FB01fi
FB02fi
FB04fi

CERDAS MENGUASAI GIT

CERDAS MENGUASAI GIT

Dalam 24 Jam

Rolly M. Awangga
Informatics Research Center



Kreatif Industri Nusantara

Penulis:

Rolly Maulana Awangga

ISBN : 978-602-53897-0-2

Editor:

M. Yusril Helmi Setyawan

Penyunting:

Syafril Fachrie Pane

Khaera Tunnisa

Diana Asri Wijayanti

Desain sampul dan Tata letak:

Deza Martha Akbar

Penerbit:

Kreatif Industri Nusantara

Redaksi:

Jl. Ligar Nyawang No. 2

Bandung 40191

Tel. 022 2045-8529

Email : awangga@kreatif.co.id

Distributor:

Informatics Research Center

Jl. Sariasih No. 54

Bandung 40151

Email : irc@poltekpos.ac.id

Cetakan Pertama, 2019

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara
apapun tanpa ijin tertulis dari penerbit

*'Jika Kamu tidak dapat
menahan lelahnya
belajar, Maka kamu
harus sanggup menahan
perihnya Kebodohan.'*

Imam Syafi'i

CONTRIBUTORS

ROLLY MAULANA AWANGGA, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia

CONTENTS IN BRIEF

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Listings

FOREWORD

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa

KATA PENGANTAR

Buku ini diciptakan bagi yang awam dengan git sekalipun.

R. M. AWANGGA

*Bandung, Jawa Barat
Februari, 2019*

ACKNOWLEDGMENTS

Terima kasih atas semua masukan dari para mahasiswa agar bisa membuat buku ini lebih baik dan lebih mudah dimengerti.

Terima kasih ini juga ditujukan khusus untuk team IRC yang telah fokus untuk belajar dan memahami bagaimana buku ini mendampingi proses Internship.

R. M. A.

ACRONYMS

ACGIH	American Conference of Governmental Industrial Hygienists
AEC	Atomic Energy Commission
OSHA	Occupational Health and Safety Commission
SAMA	Scientific Apparatus Makers Association

GLOSSARY

git	Merupakan manajemen sumber kode yang dibuat oleh linus torvald.
bash	Merupakan bahasa sistem operasi berbasiskan *NIX.
linux	Sistem operasi berbasis sumber kode terbuka yang dibuat oleh Linus Torvald

SYMBOLS

- A Amplitude
 - $\&$ Propositional logic symbol
 - a Filter Coefficient
- \mathcal{B} Number of Beats

INTRODUCTION

ROLLY MAULANA AWANGGA, S.T., M.T.

Informatics Research Center
Bandung, Jawa Barat, Indonesia

Pada era disruptif saat ini. git merupakan sebuah kebutuhan dalam sebuah organisasi pengembangan perangkat lunak. Buku ini diharapkan bisa menjadi pengantar para programmer, analis, IT Operation dan Project Manajer. Dalam melakukan implementasi git pada diri dan organisasinya.

Rumusnya cuman sebagai contoh aja biar keren.

$$ABC\mathcal{DEF}\alpha\beta\Gamma\Delta \sum_{def}^{abc} \quad (I.1)$$

BAB 1

CHAPTER 1

1.1 1174035 - Luthfi Muhammad Nabil

Kecerdasan buatan merupakan kecerdasan yang dimasukkan ke sistem yang dapat diatur untuk kepentingan ilmiah. Kecerdasan buatan biasa disebut AI (Artificial Intelligence) yang didefinisikan sebagai kecerdasan ilmiah. AI memiliki kemampuan untuk menerjemahkan data dari luar, dan mempelajari data tersebut untuk dipelajari demi mencapai tujuan dan melakukan tugas tertentu sesuai hasil adaptasi berdasarkan data yang didapat.

1.1.1 Sejarah dan perkembangan Kecerdasan Buatan

AI mulai berkembang sesuai dengan konsep yang dikemukakan pada awal abad 17, René Descartes menyebutkan bahwa tubuh hewan bukanlah apa-apa melainkan mesin-mesin yang rumit. Lalu Blaise Pascal menciptakan mesin perhitungan digital mekanis pertama pada 1642. Selanjutnya pada abad ke 19, Charles Babbage dan Ada Lovelace menciptakan sebuah mesin penghitung mekanis yang dapat diprogram.

Pada tahun 1950-an, Program AI pertama yang sudah dapat difungsikan telah ditulis pada 1951 untuk menjalankan mesin Ferranti Mark I di University of Manchester yang merupakan sebuah program permainan naskah yang ditulis oleh Christopher Strachey. John McCarthy menyebutkan istilah kecerdasan buatan pada konferensi pertama yang disediakan untuk persoalan ini. Dilanjut pada tahun 1956, Beliau menemukan bahasa pemrograman yang bernama Lisp.

Jaringan saraf mulai digunakan secara luas pada tahun 1980-an, dimana algoritma perambatan balik pertama kali dijelaskan oleh Paul John Werbos pada tahun 1974. Selanjutnya di tahun 1982, para ahli fisika menganalisis sifat dari penyimpanan dan optimasi pada jaringan saraf menggunakan sistem statistika. Lalu dilanjutkan pada tahun 1985 sedikitnya empat kelompok riset menemukan algoritma pembelajaran propagansi balik. Algoritma ini berhasil diimplementasikan ke ilmu komputer dan psikologi. Dan pada tahun 1990, ditandai perolehan besar dalam berbagai bidang AI dan demonstrasi dari berbagai aplikasi yang sudah mengimplementasi. Seperti Deep Blue, sebuah komputer dari permainan catur yang dapat mengalahkan Garry Kasparov dalam sebuah pertandingan 6 game yang terkenal pada 1997.

1.1.2 Supervised Learning

Supervised learning adalah kondisi yang menggunakan variabel input dan output untuk dapat dilakukan pemetaan input output yang sudah didapat. Disebut Supervised Learning karena proses dari pembelajaran algoritma dari pembelajaran yang disumberkan dengan dataset dapat dipikirkan seperti seorang guru yang mengawasi proses pembelajaran. Proses pembelajaran dari algoritma akan berhenti saat algoritma sudah mendapatkan level dari performansi yang dapat diterima.

Masalah dari Supervised learning dapat dikelompokkan menjadi masalah dengan regresi dan klasifikasi

- Klasifikasi : Masalah dalam klasifikasi yang dimana output dari variable itu adalah kategori, seperti "Laki - laki" atau "Perempuan, dan "Muda" dan "Tua"
- Regresi : Masalah dalam regresi adalah jika pengeluaran dari variabel adalah sebuah nilai asli, seperti "suhu", dan "tinggi"

1.1.3 Unsupervised Learning

Unsupervised learning adalah kondisi dimana kamu hanya memiliki input data tanpa memiliki variabel output yang sesuai. Tujuan dari unsupervised learning adalah untuk memodelkan distribusi pada data untuk mengetahui lebih lanjut mengenai data. Disebut unsupervised learning karena pada metode ini, tidak ada jawaban yang tepat dan tidak ada pengarah. Sehingga algoritma

akan ditinggalkan sesuai rancangan demi menemukan dan dapat mengolah data yang menarik pada saat yang akan datang.

1.1.4 Jenis - Jenis Dataset

Dataset merupakan objek yang merepresentasikan data dan relasinya di memor. Strukturnya dapat mirip sesuai dengan struktur yang ada pada database namun bisa diubah sesuai dengan kebutuhan. Dataset juga berisi koleksi dari tabel data dan relasi data.

- Training set : merupakan sebuah dataset yang digunakan untuk kepentingan pembelajaran. Kepentingan tersebut akan disesuaikan dengan parameter yang ada.
- Test dataset : adalah sebuah dataset yang bersifat independen dibandingkan dengan training dataset, namun mengikuti probabilitas distribusi yang sama dengan training dataset. Jika model sudah sesuai dengan training dataset maka dataset sudah dapat disesuaikan dengan test dataset. Penyesuaian dari training dataset .

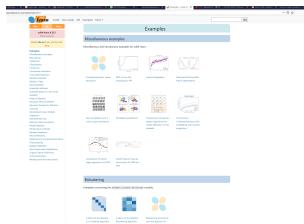
1.1.5 Instalasi dan Percobaan Kompilasi dari Library Scikit-learn

1. Buka anaconda prompt
2. Ketik di anaconda prompt yaitu : "pip install -U scikit-learn" untuk instalasi

```
(base) D:\Wallah\Python\KecerdasanBuatAnda>pip install -U scikit-learn
Collecting scikit-learn
  Downloading scikit-learn-0.22.1-pypi_0-py3-none-any.whl (6.3 MB)
Collecting scipy==1.7.0
  Downloading scipy-1.7.0-cp37-cp37m-win_amd64.whl (30.9 MB)
Collecting numpy==1.18.1
  Downloading numpy-1.18.1-cp37-cp37m-win_amd64.whl (12.3 MB)
Collecting joblib==0.14.1
  Downloading joblib-0.14.1-py3-none-any.whl (294 kB)
Collecting scikit-learn[scipy,numpy]
  Downloading scikit-learn-0.22.1-pypi_0-py3-none-any.whl (204 kB)
Successfully installed joblib-0.14.1 numpy-1.18.1 scikit-learn-0.22.1 scipy-1.4.1
(base) D:\Wallah\Python\KecerdasanBuatAnda>
```

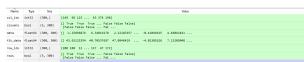
Gambar 1.1 Instalasi Scikit Learn

3. Setelah selesai instalasi, pilih salah satu example dari website Scikit (Contoh :)



Gambar 1.2 Daftar Example

4. Lalu coba jalankan aplikasi tersebut, bisa dicek hasil dari Variable explorernya



Gambar 1.3 Variable Explorer

5. Sample kode

```

1 print(__doc__)
2
3 # Author: Kemal Eren <kemal@kemaleren.com>
4 # License: BSD 3 clause
5
6 import numpy as np
7 from matplotlib import pyplot as plt
8
9 from sklearn.datasets import make_biclusters
10 from sklearn.cluster import SpectralCoclustering
11 from sklearn.metrics import consensus_score
12
13 data, rows, columns = make_biclusters(
14     shape=(300, 300), n_clusters=5, noise=5,
15     shuffle=False, random_state=0)
16
17 plt.matshow(data, cmap=plt.cm.Blues)
18 plt.title("Original dataset")
19
20 # shuffle clusters
21 rng = np.random.RandomState(0)
22 row_idx = rng.permutation(data.shape[0])
23 col_idx = rng.permutation(data.shape[1])
24 data = data[row_idx][:, col_idx]
25
26 plt.matshow(data, cmap=plt.cm.Blues)
27 plt.title("Shuffled dataset")
28
29 model = SpectralCoclustering(n_clusters=5, random_state=0)
30 model.fit(data)
31 score = consensus_score(model.biclusters_,
32                         (rows[:, row_idx], columns[:, col_idx]))
33
34 print("consensus score: {:.3f}".format(score))
35
36 fit_data = data[np.argsort(model.row_labels_)]
37 fit_data = fit_data[:, np.argsort(model.column_labels_)]
38
39 plt.matshow(fit_data, cmap=plt.cm.Blues)
40 plt.title("After biclustering; rearranged to show biclusters")
41

```

```
42 plt.show()
```

1.1.6 Mencoba Loading and example dataset

Disini akan dilakukan percobaan dengan menggunakan beberapa datasets seperti digits dan iris untuk bisa digunakan sebagai training set yang akan dipakai seluruh metode.

- Percobaan 1 (Memuat data iris dan digits dari datasets)

```
1 from sklearn import datasets #Untuk import class/fungsi
  dataset dari scikit-learn library
2 iris = datasets.load_iris() #Untuk memuat dan memasukkan
  dataset iris ke variabel bernama iris
3 digits = datasets.load_digits() #Untuk memuat dan memasukkan
  dataset digits ke variabel digits
```

```
In [18]: """
...: Created on Wed Feb 26 15:55:58 2020
...:
...: @author: Luthfi Muhammad Nabil
...: """
...:
...: from sklearn import datasets
...: iris = datasets.load_iris()
...: digits = datasets.load_digits()
```

Gambar 1.4 Hasil Percobaan 1

- Percobaan 2 (Menampilkan data dari digits)

```
1 print(digits.data) #Menampilkan object berformat Dictionary-
  like yang nanti akan ditampilkan pada console
```

```
In [19]: print(digits.data)
[[ 0.  0.  5. ... 0.  0.  0.]
 [ 0.  0.  0. ... 10. 0.  0.]
 [ 0.  0.  0. ... 16. 9.  0.]
 ...
 [ 0.  0.  1. ... 6.  0.  0.]
 [ 0.  0.  2. ... 12. 0.  0.]
 [ 0.  0. 10. ... 12. 1.  0.]]
```

Gambar 1.5 Hasil Percobaan 2

- Percobaan 3 (Menampilkan digits.target)

```
1 digits.target #Menunjukkan data angka yang berhubungan dengan
  setiap digit gambar yang sedang dipelajari
```

```
In [20]: digits.target
Out[20]: array([0, 1, 2, ..., 8, 9, 8])
```

Gambar 1.6 Hasil Percobaan 3

- Percobaan 4 (Menampilkan data 2 dimensi)

```
1 digits.images[0] #Akan mengambil data dengan berformat array
2 Dimensi, dengan bentuk parameter (n_samples, n_features)
```

```
In [22]: digits.images[0]
Out[22]:
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

Gambar 1.7 Hasil Percobaan 4

- Full sample

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Feb 26 15:55:58 2020
4
5 @author: Luthfi Muhammad Nabil
6 """
7
8 from sklearn import datasets #Untuk import class/fungsi
9      dataset dari scikit-learn library
10 iris = datasets.load_iris() #Untuk memuat dan memasukkan
11      dataset iris ke variabel bernama iris
12 digits = datasets.load_digits() #Untuk memuat dan memasukkan
13      dataset digits ke variabel digits
14 #%%
15 print(digits.data) #Menampilkan object berformat Dictionary-
16      like yang nanti akan ditampilkan pada console
17 #%%
18 digits.target #Menunjukkan data angka yang berhubungan dengan
19      setiap digit gambar yang sedang dipelajari
20 #%%
21 digits.images[0] #Akan mengambil data dengan berformat array
22 Dimensi, dengan bentuk parameter (n_samples, n_features)
```

**Gambar 1.8** Hasil pada variable explorer

1.1.7 Learning and Predicting

Disini akan dicoba untuk melakukan prediksi berupa angka yang inputnya berupa gambaran dataset.

- Percobaan 1

```

1 from sklearn import svm #Untuk mengimport class sv dari
   library sklearn
2 from sklearn import datasets #Untuk import class/fungsi
   dataset dari scikit-learn library
3 clf = svm.SVC(gamma=0.001, C=100) #Memasukkan implementasi
   dari "Support Vector Classification" ke variabel clf
4 iris = datasets.load_iris() #Untuk memuat dan memasukkan
   dataset iris ke variabel bernama iris

```

Gambar 1.9 Hasil Percobaan 1

- Percobaan 2

```

1 digits = datasets.load_digits() #Untuk memuat dan memasukkan
   dataset digits ke variabel digits
2 clf.fit(digits.data[:-1], digits.target[:-1]) #Untuk
   melakukan pengiriman data training set ke method fit

```

Gambar 1.10 Hasil Percobaan 2

- Percobaan 3

```

1 clf.predict(digits.data[-1:]) #Untuk melakukan prediksi nilai
   yang baru berdasarkan gambar terakhir dari digits.data

```

Gambar 1.11 Hasil Percobaan 3

- Full sample

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 10:32:23 2020
4
5 @author: Luthfi Muhammad Nabil
6 """
7
8 from sklearn import svm #Untuk mengimport class sv dari
9     library sklearn
10 from sklearn import datasets #Untuk import class/fungsi
11     dataset dari scikit-learn library
12 clf = svm.SVC(gamma=0.001, C=100) #Memasukkan implementasi
13     dari "Support Vector Classification" ke variabel clf
14 iris = datasets.load_iris() #Untuk memuat dan memasukkan
15     dataset iris ke variabel bernama iris
16 #%%
17 digits = datasets.load_digits() #Untuk memuat dan memasukkan
18     dataset digits ke variabel digits
19 clf.fit(digits.data[:-1], digits.target[:-1]) #Untuk
20     melakukan pengiriman data training set ke method fit
21 #%%
22 clf.predict(digits.data[-1:]) #Untuk melakukan prediksi nilai
23     yang baru berdasarkan gambar terakhir dari digits.data

```



Gambar 1.12 Hasil pada variable explorer

1.1.8 Model Persistence

Disini akan dilakukan persistensi model menggunakan built-in dari Python

- Percobaan 1

```

1 from sklearn import svm #Mengimport class SVM dari library
2     sklearn
3 from sklearn import datasets #Mengimport datasets dari
4     library sklearn
5 clf = svm.SVC() #Memasukkan implementasi dari "Support Vector
6     Classification" ke variabel clf
7 X, y = datasets.load_iris(return_X_y=True) #Untuk memuat dan
8     memasukkan dataset iris ke variabel bernama iris
9 clf.fit(X, y) #Untuk melakukan pengiriman data training set
10    ke method fit

```



Gambar 1.13 Hasil Percobaan 1

- Percobaan 2

```

1 import pickle #Mengimport pickle untuk implementasi protokol
   serializing dan de-serializing
2 s = pickle.dumps(clf) #Untuk serialize hirarki dari object
3 clf2 = pickle.loads(s) #Untuk memuat dan men de-serialize
   hirarki dari object
4 clf2.predict(X[0:1]) #Untuk melakukan prediksi nilai yang
   baru berdasarkan gambar terakhir dari digits.data

```

Gambar 1.14 Hasil Percobaan 2

- Percobaan 3

```
1 y[0] #Untuk menampilkan data iris koordinat y
```

Gambar 1.15 Hasil Percobaan 3

- Percobaan 4

```

1 from joblib import dump, load #Mengambil class dump dan load
   dari library joblib
2 dump(clf, 'filename.joblib') #Untuk memasukkan data ke dalam
   sebuah file

```

Gambar 1.16 Hasil Percobaan 4

- Percobaan 5

```
1 clf = load('filename.joblib') #Untuk memuat data dari file
```

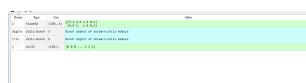
Gambar 1.17 Hasil Percobaan 5

- Full sample

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 10:37:49 2020
4
5 @author:Luthfi Muhammad Nabil
6 """
7
8 from sklearn import svm #Mengimport class SVM dari library
9      sklearn
10 from sklearn import datasets #Mengimport datasets dari
11      library sklearn
12 clf = svm.SVC() #Memasukkan implementasi dari "Support Vector
13      Classification" ke variabel clf
14 X, y = datasets.load_iris(return_X_y=True) #Untuk memuat dan
15      memasukkan dataset iris ke variabel bernama iris
16 clf.fit(X, y) #Untuk melakukan pengiriman data training set
17      ke method fit
18 #%%
19 import pickle #Mengimport pickle untuk implementasi protokol
20      serializing dan de-serializing
21 s = pickle.dumps(clf) #Untuk serialize hirarki dari object
22 clf2 = pickle.loads(s) #Untuk memuat dan men de-serialize
23      hirarki dari object
24 clf2.predict(X[0:1]) #Untuk melakukan prediksi nilai yang
25      baru berdasarkan gambar terakhir dari digits.data
26 #%%
27 y[0] #Untuk menampilkan data iris koordinat y
28 #%%
29 from joblib import dump, load #Mengambil class dump dan load
30      dari library joblib
31 dump(clf, 'filename.joblib') #Untuk memasukkan data ke dalam
32      sebuah file
33 #%%
34 clf = load('filename.joblib') #Untuk memuat data dari file

```



Gambar 1.18 Hasil pada variable explorer

1.1.9 Conventions

Seluruh metode akan dilakukan pengaturan untuk membuat tingkah laku lebih dapat diprediksi

- Percobaan 1

```

1 import numpy as np #Memuat library numpy yang akan
2      diinisialisasikan menjadi np
3 from sklearn import random_projection #Memuat class
4      random_projection dari library sklearn

```

```
4 rng = np.random.RandomState(0) #Memuat angka random dan  
mengekspos angka untuk menghasilkan dari berbagai  
probabilitas distribusi, angka tersebut dimasukkan ke  
variabel rng  
5 X = rng.rand(10, 2000) #Membatasi jarak angka random  
6 X = np.array(X, dtype='float32') #Memuat angka menjadi ke  
dalam array  
7 X.dtype #Mengambil tipe data dari variabel X
```

Gambar 1.19 Hasil Percobaan 1

▪ Percobaan 2

```
1 transformer = random_projection.GaussianRandomProjection() #  
    Mengimplementasikan modul yang simpel dan efisien secara  
    komputasi untuk mengurangi dimensi dari data  
2 X_new = transformer.fit_transform(X) #Untuk membuat  
    penengahan data  
3 X_new.dtype #Mengambil tipe data dari variabel X
```

```
In [20]: transformer = random_projection.GaussianRandomProjection() #Mengimpi  
data  
.... X_new = transformer.fit_transform(X) #Untuk membuat peningkahan data  
.... X_new.dtype #Mengambil tipe data dari variabel X  
Out[20]: dtype('float64')
```

Gambar 1.20 Hasil Percobaan 2

■ Percobaan 3

```
1 from sklearn import datasets #Mengimport datasets dari  
  library sklearn  
2 from sklearn.svm import SVC  
3 iris = datasets.load_iris() #Untuk memuat dan memasukkan  
  dataset iris ke variabel bernama iris  
4 clf = SVC() #Memasukkan implementasi dari "Support Vector  
  Classification" ke variabel clf  
5 clf.fit(iris.data, iris.target) #Untuk melakukan pengiriman  
  data training set ke method fit
```

```
In [25]: #Import library import datasets #Importing datasets dari library sklearn
#Import library
#Import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target
#Data set ini memiliki 3 kelas dan dimana setiap kelas memiliki 50 data
#CIF = SVC() mengambil nilai dari "Support Vector Classification" ke variabel clf
clf = SVC()
#clf.fit(X,y) mengambil nilai dari "fit" ke variabel clf
clf.fit(X,y)
#Dengan memasukkan fitur pada package sklearn.svm.base(ya py153) FutureWarning: The default value of feature_names_in_ will be set again explicitly to None or "scale" to avoid this warning.
#avoid the warning
#clf.fit(X,y)

Out[25]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
         decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
         kernel='rbf', max_iter=-1, probability=False, random_state=None,
         shrinking=True, tol=0.001, verbose=False)
```

Gambar 1.21 Hasil Percobaan 3

- Percobaan 4

```
1 list(clf.predict(iris.data[:3])) #Untuk memuat dan men de-
    serialize hirarki dari object , data tersebut akan
    dimasukkan ke fungsi list
```

Gambar 1.22 Hasil Percobaan 4

- Percobaan 5

```
1 clf.fit(iris.data, iris.target_names[iris.target]) #Untuk
    melakukan pengiriman data training set ke method fit
```

Gambar 1.23 Hasil Percobaan 5

- Percobaan 6

```
1 list(clf.predict(iris.data[:3])) #Untuk memuat dan men de-
    serialize hirarki dari object , data tersebut akan
    dimasukkan ke fungsi list
```

Gambar 1.24 Hasil Percobaan 6

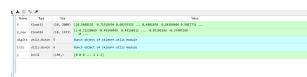
- Full sample

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 10:56:18 2020
4
5 @author: Luthfi Muhammad Nabil
```

```

6 """
7
8 import numpy as np #Memuat library numpy yang akan
9     diinisialisasikan menjadi np
9 from sklearn import random_projection #Memuat class
10    random_projection dari library sklearn
11
11 rng = np.random.RandomState(0) #Memuat angka random dan
12    mengeksplosi angka untuk menghasilkan dari berbagai
13    probabilitas distribusi, angka tersebut dimasukkan ke
14    variabel rng
12 X = rng.rand(10, 2000) #Membatasi jarak angka random
13 X = np.array(X, dtype='float32') #Memuat angka menjadi ke
14    dalam array
14 X.dtype #Mengambil tipe data dari variabel X
15 #%%
16 transformer = random_projection.GaussianRandomProjection() #
17     Mengimplementasikan modul yang simpel dan efisien secara
18     komputasi untuk mengurangi dimensi dari data
17 X_new = transformer.fit_transform(X) #Untuk membuat
18     penengahan data
18 X_new.dtype #Mengambil tipe data dari variabel X
19
20 #%%
21 from sklearn import datasets #Mengimport datasets dari
22     library sklearn
22 from sklearn.svm import SVC
23 iris = datasets.load_iris() #Untuk memuat dan memasukkan
24     dataset iris ke variabel bernama iris
24 clf = SVC() #Memasukkan implementasi dari "Support Vector
25     Classification" ke variabel clf
25 clf.fit(iris.data, iris.target) #Untuk melakukan pengiriman
26     data training set ke method fit
26 #%%
27 list(clf.predict(iris.data[:3])) #Untuk memuat dan men de-
28     serialize hirarki dari object, data tersebut akan
29     dimasukkan ke fungsi list
28 #%%
29 clf.fit(iris.data, iris.target_names[iris.target]) #Untuk
30     melakukan pengiriman data training set ke method fit
30 #%%
31 list(clf.predict(iris.data[:3])) #Untuk memuat dan de-
32     serialize hirarki dari object, data tersebut akan
33     dimasukkan ke fumgsi list

```



Gambar 1.25 Hasil pada variable explorer

1.1.10 Skrinsut Error

Gambar 1.26 Hasil Percobaan 6

1.1.11 Kode error dan jenis error tersebut

Error yang didapat berjenis name error, karena sebuah variabel tidak didefinisikan. Yaitu digits

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 10:56:18 2020
4
5 @author: Luthfi Muhammad Nabil
6 """
7
8 import numpy as np #Memuat library numpy yang akan
9     diinisialisasikan menjadi np
10 from sklearn import random_projection #Memuat class
11     random_projection dari library sklearn
12
13 rng = np.random.RandomState(0) #Memuat angka random dan
14     mengekspos angka untuk menghasilkan dari berbagai
15     probabilitas distribusi, angka tersebut dimasukkan ke
16     variabel rng
17 X = rng.rand(10, 2000) #Membatasi jarak angka random
18 X = np.array(X, dtype='float32') #Memuat angka menjadi ke dalam
19     array
20 X.dtype #Mengambil tipe data dari variabel X
21 """
22 transformer = random_projection.GaussianRandomProjection() #
23     Mengimplementasikan modul yang simpel dan efisien secara
24     komputasi untuk mengurangi dimensi dari data
25 X_new = transformer.fit_transform(X) #Untuk membuat peningangan
26     data
27 X_new.dtype #Mengambil tipe data dari variabel X
28 """
29
30 from sklearn import datasets #Mengimport datasets dari library
31     sklearn
32 from sklearn.svm import SVC
33 iris = datasets.load_iris() #Untuk memuat dan memasukkan dataset
34     iris ke variabel bernama iris
35 clf = SVC() #Memasukkan implementasi dari "Support Vector
36     Classification" ke variabel clf
37 clf.fit(iris.data, iris.target) #Untuk melakukan pengiriman data
38     training set ke method fit
39 """
40
```

```

27 list(clf.predict(iris.data[:3])) #Untuk memuat dan men de-
    serialize hirarki dari object, data tersebut akan dimasukkan
    ke fungsi list
28 #%%
29 clf.fit(iris.data, iris.target_names[iris.target]) #Untuk
    melakukan pengiriman data training set ke method fit
30 #%%
31 list(clf.predict(iris.data[:3])) #Untuk memuat dan men de-
    serialize hirarki dari object, data tersebut akan dimasukkan
    ke fungsi list

```

1.1.12 Penanganan Error

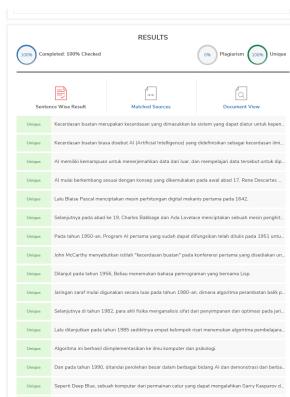
Untuk menangani error tersebut, bisa ditambahkan sesuai instruksi. Yaitu menambahkan sebuah variabel bernama digits. Selain itu, digits harus dapat bekerja sebagaimana mestinya. Berikut full kodingnya :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 10:32:23 2020
4
5 @author: Luthfi Muhammad Nabil
6 """
7
8 from sklearn import svm #Untuk mengimport class sv dari library
    sklearn
9 from sklearn import datasets #Untuk import class/fungsi dataset
    dari scikit-learn library
10 clf = svm.SVC(gamma=0.001, C=100) #Memasukkan implementasi dari "
        Support Vector Classification" ke variabel clf
11 iris = datasets.load_iris() #Untuk memuat dan memasukkan dataset
    iris ke variabel bernama iris
12 #%%
13 digits = datasets.load_digits() #Untuk memuat dan memasukkan
    dataset digits ke variabel digits
14 clf.fit(digits.data[:-1], digits.target[:-1]) #Untuk melakukan
    pengiriman data training set ke method fit
15 #%%
16 clf.predict(digits.data[-1:]) #Untuk melakukan prediksi nilai
    yang baru berdasarkan gambar terakhir dari digits.data

```

1.1.13 Plagiarisme



Gambar 1.27 Hasil pada variable explorer

1.2 1174040 - Hagan Rowlenstino A. S

1.2.1 Teori

1.2.1.1 Definisi Kecerdasan Buatan Artificial Intelligence atau dapat disebut juga dengan kecerdasan buatan merupakan kecerdasan yang ditambahkan kepada suatu sistem yang dapat diatur dalam konteks ilmiah. Michael Haenlein dan Andreas Kaplan mendefinisikan bahawa AI adalah “sistem yang mempunyai kemampuan untuk menguraikan, belajar agar dapat digunakan untuk mencapai tujuan dengan melalui adaptasi yang fleksibel”. Kecerdasan ini dibuat dan dimasukkan ke dalam mesin agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia dengan cepat dan tepat. Bidang-bidang yang menggunakan kecerdasan buatan antara lain logika fuzzy, games, sistem pakar, jaringan saraf tiruan dan robotika.

1.2.1.2 Sejarah Kecerdasan Buatan Sejarah kecerdasan buatan ini dimulai dari zaman kuno, mitos ataupun cerita dan desas-desus tentang sebuah makhluk yang mempunyai kecerdasan serta kesadaran yang diberikan oleh pengrajin. Benih - benih nya mulai ditanam oleh para filsuf klasik yang mencari cara untuk menggambarkan proses berfikir manusia sebagai manipulasi simbol secara mekanis yang memuncak pada penemuan komputer digital di tahun 1940-an, yaitu sebuah mesin yang didasarkan penalaran matematika. Istilah kecerdasan buatan sendiri baru muncul pada tahun 1956, dan teori -teori nya sudah muncul sejak tahun 1941.

1.2.1.3 Perkembangan Kecerdasan Buatan

1. Perkembangan kecerdasan buatan dimulai dari Era Komputer Elektronik pada tahun 1941, dimana ditemukannya alat penyimpanan dan pemros-

esan informasi. Dilanjutkan pada tahun 1949, berhasilnya pembuatan komputer yang mampu menyimpan program yang memudahkan pekerjaan dalam memasukkan program menjadi lebih mudah.

2. Masa - masa persiapan AI terjadi pada tahun 1943 - 1956.
3. Awal Perkembangan AI terjadi pada 1952 - 1969
4. Perkembangan kecerdasan buatan melambat pada tahun 1966-1974
5. Sistem Berbasis Pengetahuan pada tahun 1969-1979
6. Kecerdasan Buatan menjadi sebuah industri pada tahun 1980-1988. Kembarinya Jaringan Syaraf tiruan pada tahun 1986-sekarang.

1.2.2 Instalasi

Membuka <https://scikit-learn.org/stable/tutorial/basic/tutorial.html> lalu mencobanya.

1.2.2.1 Instalasi library scikit, mencoba kompilasi dan ujicoba contoh kode

1. Buka anaconda prompt lalu ketikkan "pip install -U scikit-learn" untuk menginstall library scikit



Gambar 1.28 Install Library Scikit

2. Pilih salah satu example dari website tersebut lalu jalankan

```

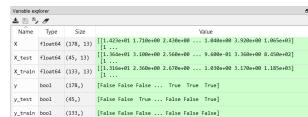
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Feb 26 18:16:44 2020
4
5 @author: User
6 """
7
8 import matplotlib.pyplot as plt
9 from sklearn.svm import SVC
10 from sklearn.ensemble import RandomForestClassifier
11 from sklearn.metrics import plot_roc_curve
12 from sklearn.datasets import load_wine
13 from sklearn.model_selection import train_test_split
  
```

```

14
15 X, y = load_wine(return_X_y=True)
16 y = y == 2
17
18 X_train, X_test, y_train, y_test = train_test_split(X, y,
19 random_state=42)
20 svc = SVC(random_state=42)
21 svc.fit(X_train, y_train)

```

3. buka variable explolernya



Gambar 1.29 Variable Exploler

1.2.2.2 Mencoba loading an example dataset

1. mengambil data iris dan digit dari dataset

```

1 from sklearn import datasets #untuk mengimport dataset dari
    library learn-scikit
2 iris = datasets.load_iris() #membuat sebuah variable iris
    yang mempunyai isi yaitu dataset iris
3 digits = datasets.load_digits() #membuat sebuah variable
    digits yang mempunyai isi yaitu dataset digits

```

2. Menampilkan data digits

```

1 print(digits.data) #memberikan akses ke fitur yang dapat
    digunakan untuk mengklasifikasikan sampel digit dan
    menampilkannya di console

```

```

In [23]: runfile('D:/Semester6/AI/Chapter1/no2.py', wdir='D:/Semester6/AI/Chapter1')
[[ 0.,  0.,  5., ...,  0.,  0.]
 [ 0.,  0.,  10., ...,  0.,  0.]
 [ 0.,  0.,  15., ...,  0.,  0.]
 ...
 [ 0.,  0.,  1., ...,  5.,  0.]
 [ 0.,  0.,  2., ..., 12.,  0.]
 [ 0.,  0.,  0., ..., 12.,  1.]]

```

Gambar 1.30 Data Digits

3. menampilkan digits.target

```

1 digits.target #memberikan informasi tentang data yang
    berhubungan atau juga dapat dijadikan sebagai label

```

```
In [16]: digits.target
Out[16]: array([0, 1, 2, ..., 8, 9, 8])
```

Gambar 1.31 Digits Target

4. menampilkan data bentuk 2D.

```
1 digits.images[0] #Data selalu berupa array 2D, shape (
    n_samples, n_features), meskipun data aslinya mungkin
    memiliki bentuk yang berbeda.
```

```
In [17]: digits.images[0]
Out[17]:
array([[0., 0., 5., 13., 9., 1., 0., 0.],
       [0., 0., 13., 15., 10., 15., 5., 0.],
       [0., 3., 15., 2., 0., 11., 8., 0.],
       [0., 1., 11., 13., 10., 12., 7., 0.],
       [0., 5., 8., 0., 0., 9., 8., 0.],
       [0., 4., 11., 0., 1., 12., 7., 0.],
       [0., 2., 14., 5., 10., 12., 0., 0.],
       [0., 0., 6., 13., 10., 0., 0., 0.]])
```

Gambar 1.32 Data 2D

1.2.2.3 Mencoba Learning and Predicting

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 22:20:55 2020
4
5 @author: User
6 """
7
8 from sklearn.linear_model import LogisticRegression #untuk
    mengimport linear-model dari library sklearn
9 from sklearn.datasets import make_blobs #untuk mengimport library
    datasets dari sklearn
10
11 X, y = make_blobs(n_samples=100, centers=2, n_features=2,
    random_state=1) # untuk generate dataset dengan klasifikasi 2
    D
12 model = LogisticRegression() #menggunakan metode loginstic
    regression
13 model.fit(X, y)
14
15 Xnew, _ = make_blobs(n_samples=3, centers=2, n_features=2,
    random_state=1) # menentukan 1 buah contoh baru dimana
    jawabannya tidak diketahui
16
17 ynew = model.predict_proba(Xnew) # membuat sebuah prediksi dan
    memasukkan nya kedalam variable ynew
18 for i in range(len(Xnew)):
19     print("X=%s, Predicted=%s" % (Xnew[i], ynew[i])) #menampilkan
        hasil prediksi
```

1.2.2.4 Mencoba Model Persistence

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 22:27:13 2020
4
5 @author: User
6 """
7
8 from sklearn import svm #menigimport svm dari library sklearn
9 from sklearn import datasets #menigimport datasets dari library
10    sklearn
11 clf = svm.SVC() #menggunakan method SVC
12 iris = datasets.load_iris() #menggunakan dataset iris
13 X, y = iris.data, iris.target #memasukkan x sebagai iris data ,
14      dan y sebagai iris target
15 clf.fit(X, y) #laalu menggunakan metod fit .

```

1.2.2.5 Mencoba Conventions

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 23:24:21 2020
4
5 @author: User
6 """
7
8 import numpy as npy #mengimport numpy sebagai npy
9 from sklearn import random_projection #mengimport
10    random_projection dari library sklearn
11 rng = npy.random.RandomState(0) #Menggunakan fungsi random dari
12      numpy
13 X = rng.rand(10, 2000) #membuat range random diantara 10 sampai
14      2000
15 X = npy.array(X, dtype='float32') #yang dijadikan array dengan
16      tipe data float32
17 X.dtype

```

1.2.3 Penanganan Error



ValueError: Found array with 0 sample(s) (shape=(0, 2)) while a minimum of 1 is required.

Gambar 1.33 Data 2D

1.2.3.1 Screenshot Error

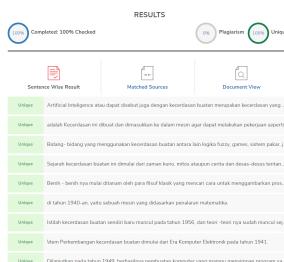
1.2.3.2 Kode error dan jenis error

Jenis errornya adalah value error

```
1 Xnew, _ = make_blobs(n_samples=3, centers=2, n_features=2,
    random_state=1) # menentukan 1 buah contoh baru dimana
    jawabannya tidak diketahui
```

1.2.3.3 Solusi Error Solusinya adalah dengan mengganti nilai n_samples nya agar tidak 0

1.2.4 Cek Plagiarism



Gambar 1.34 Cek Plagiarism

1.3 1174042 Faisal Najib Abdullah

1.3.1 Teori

- Definisi, sejarah, dan perkembangan kecerdasan buatan.

Definisi kecerdasan buatan adalah suatu pengetahuan yang dapat membuat komputer untuk meniru kecerdasan manusia yang berhubungan dengan penangkapan, pemodelan, dan penyimpanan kecerdasan manusia dalam sebuah sistem teknologi. Contohnya yaitu melakukan analisa penalaran untuk mengambil suatu kesimpulan atau penerjemahan atau keputusan dari satu bahasa satu ke bahasa lain.

Sejarah dan perkembangan kecerdasan buatan terjadi pada musim panas tahun 1956 tercatat adanya seminar mengenai AI di Darmouth College. Seminar pada waktu itu dihadiri oleh sejumlah pakar komputer dan membahas potensi komputer dalam meniru kepandaian manusia. Akan tetapi perkembangan yang sering terjadi semenjak diciptakannya LISP, yaitu bahasa kecerdasan buatan yang dibuat tahun 1960 oleh John McCarthy. Istilah pada kecerdasan buatan atau Artificial Intelligence diambil dari Marvin Minsky dari MIT. Dia menulis karya ilmiah berjudul Step towards Artificial Intelligence, The Institute of radio Engineers Proceedings 49, January 1961[?].

- Definisi supervised learning, klasifikasi, regresi, dan unsupervised learning. Data set, training set dan testing set.

Supervised learning merupakan sebuah pendekatan dimana sudah terdapat data yang dilatih, dan terdapat variable yang ditargetkan sehingga tujuan dari pendekatan ini adalah mengelompokan suatu data ke data yang sudah ada. Sedangkan unsupervised learning tidak memiliki data latih, sehingga dari data yang ada, kita mengelompokan data tersebut menjadi 2 bagian atau 3 bagian dan seterusnya.

Klasifikasi adalah salah satu topik utama dalam data mining atau machine learning. Klasifikasi yaitu suatu pengelompokan data dimana data yang digunakan tersebut mempunyai kelas label atau target.

Regresi adalah Supervised learning tidak hanya mempelajari classifier, tetapi juga mempelajari fungsi yang dapat memprediksi suatu nilai numerik. Contoh, ketika diberi foto seseorang, kita ingin memprediksi umur, tinggi, dan berat orang yang ada pada foto tersebut.

Data set adalah cabang aplikasi dari Artificial Intelligence/Kecerdasan Buatan yang fokus pada pengembangan sebuah sistem yang mampu belajar sendiri tanpa harus berulang kali di program oleh manusia.

Training set yaitu jika pasangan objek, dan kelas yang menunjuk pada objek tersebut adalah suatu contoh yang telah diberi label akan menghasilkan suatu algoritma pembelajaran.

Testing set digunakan untuk mengukur sejauh mana classifier berhasil melakukan klasifikasi dengan benar[?].

1.3.2 Instalasi

1.3.2.1 Instalasi Library Scikit dari Pycharm

Masuk pada menu settings terus pilih Project Interpreter kemudian tambah library lalu cari dan install scikit

Mencoba Library

```

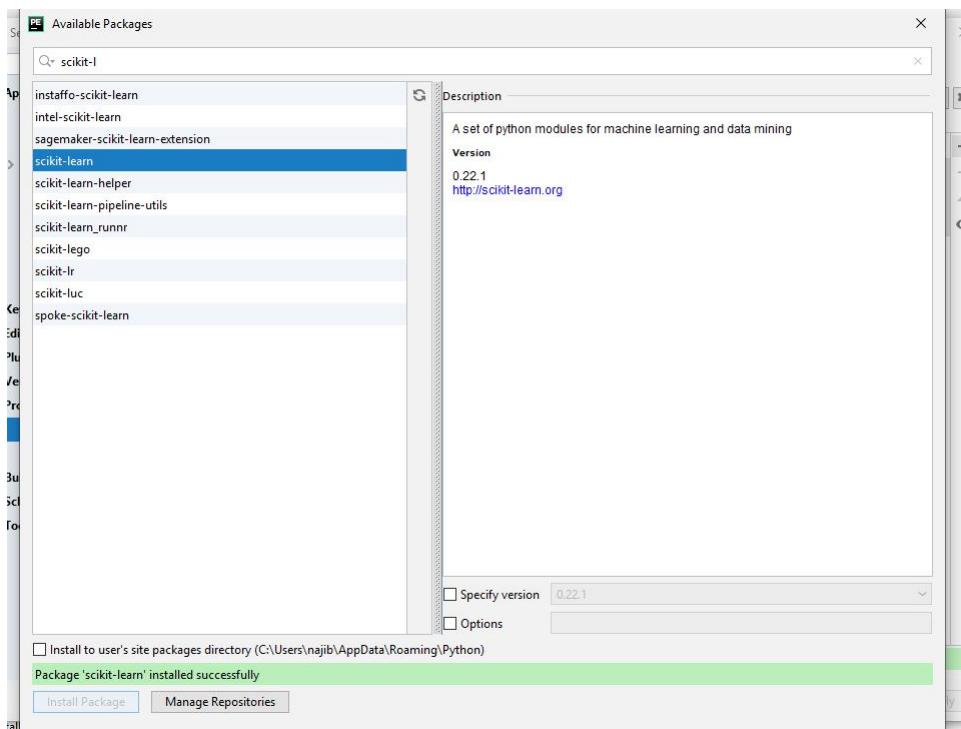
1 from sklearn import datasets
2 #pada baris ini merupakan sebuah perintah untuk mengimport class
   datasets dari packaged sklearn
3 iris = datasets.load_iris()
4 #pada baris kedua ini dimana iris merupakan suatu estimator/
   parameter yang berfungsi untuk mengambil data pada item
   datasets.load_iris
5 digits = datasets.load_digits()
6 #pada baris ketiga ini dimana digits merupakan suatu estimator/
   parameter yang berfungsi untuk mengambil data pada item
   datasets.load_digits

```

```

1 from sklearn import datasets
2 #pada baris ini merupakan sebuah perintah untuk mengimport class
   datasets dari packaged sklearn
3 iris = datasets.load_iris()
4 #pada baris kedua ini dimana iris merupakan suatu estimator/
   parameter yang berfungsi untuk mengambil data pada item
   datasets.load_iris

```



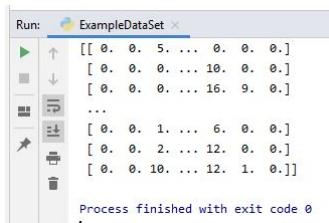
Gambar 1.35 Installasi

```

5 digits = datasets.load_digits()
6 #pada baris ketiga ini dimana digits merupakan suatu estimator/
  # parameter yang berfungsi untuk mengambil data pada item
  # datasets.load_digits
7 print(digits.data)
8 #pada baris keempat ini merupakan perintah yang berfungsi untuk
  # menampilkan estimator/parameter yang dipanggil pada item
  # digits.data dan menampilkan outputannya
9 digits.target
10 #barisan ini untuk mengambil target pada estimator/parameter
    # digits dan menampilkan outputannya
11 digits.images[0]
12 #barisan ini untuk mengambil images[0] pada estimator/parameter
    # digits dan menampilkan outputannya

from sklearn import svm
#pada baris ini merupakan sebuah perintah untuk mengimport class
# svm dari packaged sklearn
clf = svm.SVC(gamma=0.001, C=100.)

```

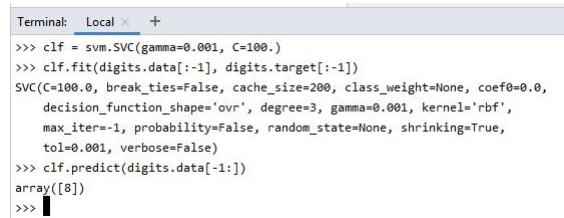


Gambar 1.36 Mencoba Loading an example Dataset

```

4 #pada baris kedua ini clf sebagai estimator/parameter , svm.SVC
   sebagai class , gamma sebagai parameter untuk menetapkan nilai
   secara manual
5 clf.fit(digits.data[:-1], digits.target[:-1])
6 #pada baris ketiga ini clf sebagai estimator/parameter , fit
   sebagai metode , digits.data sebagai item , [-1] sebagai
   syntax pythonnya dan menampilkan outputannya
7 clf.predict(digits.data[-1:])
8 #pada baris terakhir ini clf sebagai estimator/parameter , predict
   sebagai metode lainnya , digits.data sebagai item dan
   menampilkan outputannya

```



Gambar 1.37 Learning and Predicting

```

1 from sklearn import svm
2 #pada baris ini merupakan sebuah perintah untuk mengimport class
   svm dari packaged sklearn
3 from sklearn import datasets
4 #pada baris ini merupakan sebuah perintah untuk mengimport class
   datasets dari packaged sklearn
5 clf = svm.SVC(gamma='scale')
6 #pada baris ketiga ini clf sebagai estimator/parameter , svm.SVC
   sebagai class , gamma sebagai parameter untuk menetapkan nilai
   secara manual dengan nilai scale
7 iris = datasets.load_iris()
8 #pada baris keempat ini iris sebagai estimator/parameter ,
   datasets.load_iris() sebagai item dari suatu nilai
9 X, y = iris.data, iris.target
10 #pada baris kelima ini X, y sebagai estimator/parameter , iris.
    data, iris.target sebagai item dari 2 nilai yang ada

```

```

11 clf.fit(X, y)
12 #pada baris keenam ini clf sebagai estimator/parameter dengan
   menggunakan metode fit untuk memanggil estimator X, y dengan
   outputannya
13
14
15 import pickle
16 #pickle merupakan sebuah class yang di import
17 s = pickle.dumps(clf)
18 #pada baris ini s sebagai estimator/parameter dengan pickle.dumps
   merupakan suatu nilai/item dari estimator/parameter clf
19 clf2 = pickle.loads(s)
20 #pada baris ini clf2 sebagai estimator/parameter, pickle.loads
   sebagai suatu item, dan s sebagai estimator/parameter yang
   dipanggil
21 clf2.predict(X[0:1])
22 #pada baris ini clf2.predict sebagai suatu item dengan
   menggunakan metode predict untuk menentukan suatu nilai dari
   (X[0:1])
23 y[0]
24 #pada estimator/parameter y berapapun angka yang diganti nilainya
   akan selalu konstan yaitu 0
25
26
27 from joblib import dump, load
28 #pada baris berikut ini merupakan sebuah perintah untuk
   mengimport class dump, load dari packaged joblib
29 dump(clf, 'filename.joblib')
30 #pada baris berikutnya dump di sini sebagai class yang didalamnya
   terdapat nilai dari suatu item clf dan data joblib
31 clf = load('filename.joblib')
32 #pada baris terakhir clf sebagai estimator/parameter dengan suatu
   nilai load berfungsi untuk mengulang data sebelumnya

```

```

... clf.fit(X, y)
SVC(C=100.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
     decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
     max_iter=-1, probability=False, random_state=None, shrinking=True,
     tol=0.001, verbose=False)

```

Gambar 1.38 Model Presistence

```

>>> clf2.predict(X[0:1])
array([0])
>>> #pada baris ini clf2.predict
... y[0]
0

```

Gambar 1.39 Model Presistence

```

1 #Type Casting
2 from sklearn import svm

```

The screenshot shows a Jupyter Notebook environment. On the left, there's a file tree with files like 2,4,1.JPG, 2,4.JPG, 2,4.py, 2.JPG, 3.JPG, ExampleDataSet.py, filename.joblib, Teori.tex, and External Libraries. The code cell on the right contains Python code for model persistence:

```

2,4,1.JPG 26 from joblib import dump, load
2,4.JPG 27 #pada baris berikut ini merupakan sel
2,4.py 28 dump(clf, 'filename.joblib')
2.JPG 29 #pada baris berikutnya dump di sini :
3.JPG 30 clf = load('filename.joblib')
ExampleDataSet.py 31 #pada baris terakhir clf sebagai est:
filename.joblib 32 #pada baris terakhir clf sebagai est:
Teori.tex 33 |

```

Gambar 1.40 Model Presistence

```

3 #pada baris ini merupakan sebuah perintah untuk mengimport class
   svm dari packaged sklearn
4 from sklearn import random_projection
5 #pada baris ini merupakan sebuah perintah untuk mengimport class
   random_projection dari packaged sklearn
6 rng = np.random.RandomState(0)
7 #rng sebagai estimator/parameter dengan nilai suatu itemnya yaitu
   np.random.RandomState(0)
8 X = rng.rand(10, 2000)
9 #X sebagai estimator/parameter dengan nilai item rng.rand
10 X = np.array(X, dtype='float32')
11 #X sebagai estimator/parameter dengan nilai item np.array
12 X.dtype
13 #X.dtype sebagai item pemanggil
14 transformer = random_projection.GaussianRandomProjection()
15 #transformer sebagai estimator/parameter dengan memanggil class
   random_projection
16 X_new = transformer.fit_transform(X)
17 #X new di sini sebagai estimator/parameter dan menggunakan metode
   fit
18 X_new.dtype
19 #X new.dtype sebagai item
20
21
22 from sklearn import datasets
23 #pada baris ini merupakan sebuah perintah untuk mengimport class
   datasets dari packaged sklearn
24 from sklearn.svm import SVC
25 #pada baris ini merupakan sebuah perintah untuk mengimport class
   SVC dari packaged sklearn.svm
26 iris = datasets.load_iris()
27 #iris sebagai estimator/parameter dengan item datasets.load_iris
   ()
28 clf = SVC(gamma='scale')
29 #clf sebagai estimator/parameter dengan nilai class SVC pada
   parameter gamma sebagai set penilaian
30 clf.fit(iris.data, iris.target)
31 #estimator/parameter clf menggunakan metode fit dengan itemnya
32 list(clf.predict(iris.data[:3]))
33 #menambahkan item list dengan metode predict
34 clf.fit(iris.data, iris.target_names[iris.target])
35 #estimator/parameter clf menggunakan metode fit dengan itemnya
36 list(clf.predict(iris.data[:3]))
37 #menambahkan item list dengan metode predict
38

```

```
39
40
41 #Refitting and Updating Parameters
42 import numpy as np
43 #pada baris ini merupakan sebuah perintah untuk mengimport class
44     svm dari np
45 from sklearn.svm import SVC
46 #pada baris ini merupakan sebuah perintah untuk mengimport class
47     SVC dari packaged sklearn.svm
48 rng = np.random.RandomState(0)
49 #rng sebagai estimator/parameter dengan nilai suatu itemnya yaitu
50     np.random.RandomState(0)
51 X = rng.rand(100, 10)
52 #X sebagai estimator/parameter dengan nilai item rng.rand
53 y = rng.binomial(1, 0.5, 100)
54 #y sebagai estimator/parameter dengan nilai item rng.binomial
55 X_test = rng.rand(5, 10)
56 #X test sebagai estimator/parameter dengan nilai item rng.rand
57 clf = SVC()
58 #clf sebagai estimator/parameter dan class SVC
59 clf.set_params(kernel='linear').fit(X, y)
60 #set params sebagai item
61 clf.predict(X_test)
62
63
64 #Multiclass vs. Multilabel Fitting
65 from sklearn.svm import SVC
66 #pada baris ini merupakan sebuah perintah untuk mengimport class
67     SVC dari packaged sklearn.svm
68 from sklearn.multiclass import OneVsRestClassifier
69 #pada baris ini merupakan sebuah perintah untuk mengimport class
70     OneVsRestClassifier dari packaged sklearn.multiclass
71 from sklearn.preprocessing import LabelBinarizer
72 #pada baris ini merupakan sebuah perintah untuk mengimport class
73     LabelBinarizer dari packaged sklearn.preprocessing
74 X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]
75 y = [0, 0, 1, 1, 2]
76 classif = OneVsRestClassifier(estimator=SVC(gamma='scale',
77                                     random_state=0))
78 classif.fit(X, y).predict(X)
79 y = LabelBinarizer().fit_transform(y)
80 classif.fit(X, y).predict(X)
81
82
83 from sklearn.preprocessing import MultiLabelBinarizer
84 y = [[0, 1], [0, 2], [1, 3], [0, 2, 3], [2, 4]]
85 y = MultiLabelBinarizer().fit_transform(y)
86 classif.fit(X, y).predict(X)
```

1.3.3 Penanganan eror

```
>>> from joblib import dump, load
>>> #pada baris berikut ini merupakan sebuah perintah untuk mengimport class dump, load dari packaged joblib
... dump(clf, 'filename.joblib')
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
NameError: name 'clf' is not defined
>>> #pada baris berikutnya dump di sini sebagai class yang didalamnya terdapat nilai dari suatu item clf dan data joblib
... clf = load('filename.joblib')
```

Gambar 1.41 Error

1.3.3.1 ScreenShoot Eror

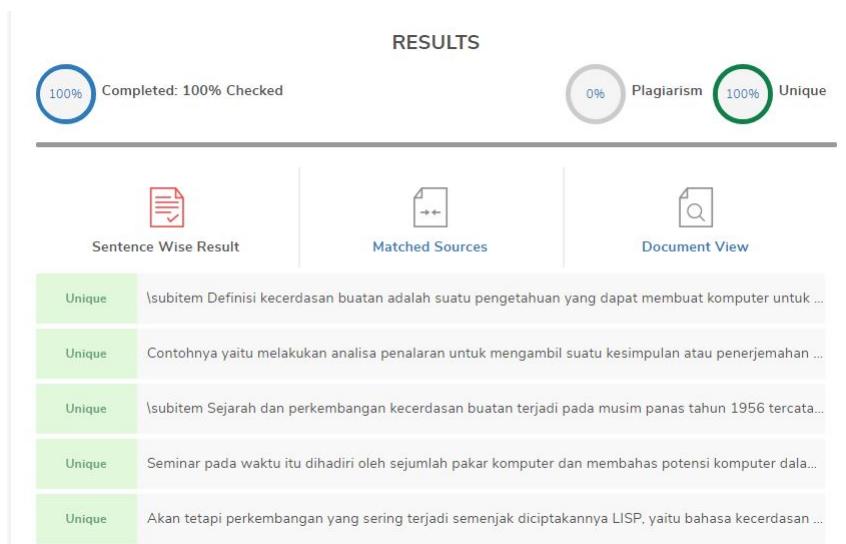
1.3.3.2 Tuliskan Kode Eror dan Jenis Erornya

File "<stdin>", line 2, in <module>
NameError: name 'clf' is not defined

1.3.3.3 Solusi Pemecahan Masalah Error

Ini karna kode di jalankan perbaris perbaris, jika kode dijanlankan bersamaan makan kode berjan sesuai prosedur.

1.3.4 Plagiat



Gambar 1.42 Error

1.3.5 Link

[Youtube](#)

1.4 1174043 - Irwan Rizkiansyah

1.4.1 Definisi Kecerdasan Buatan

Kecerdasan buatan atau biasa disebut AI (Artificial Intelligence) merupakan kecerdasan yang dibuat dan ditambahkan oleh manusia ke suatu sistem teknologi, yang diatur dan dikembangkan dalam konteks ilmiah, yang merupakan bentukan dari kecerdasan entitas ilmiah yang ada. Jadi pada intinya definisi AI dapat terus dikembangkan, namun yang menjadi poin utamanya adalah bagaimana manusia menciptakan sebuah teknologi yang dapat berpikir seperti selayaknya manusia itu sendiri.

1.4.2 Sejarah dan Perkembangan

- Program kecerdasan buatan atau Artificial Intelligence pertama kali diceritakan pada kisaran tahun 1951. Tidak bisa dipungkiri bahwa di tahun tersebut memang sedang gencar-gencarnya pembuatan cikal bakal, konsep, hingga teknologi berbasis AI. Dan, AI sendiri pertama kali digunakan di University of Manchester untuk menjalankan sebuah mesin bernama Ferranti Mark 1.
- Beberapa waktu kemudian, Christopher Strachey melanjutkan konsep kecerdasan buatan untuk menjalankan sebuah permainan catur, dimana bidak catur tersebut dapat berjalan secara otomatis dan mampu bermain melawan manusia sungguhan.
- Berlanjut pada tahun 1956, kecerdasan buatan tidak hanya dibuat untuk memudahkan bermain catur saja. Melainkan pada saat konferensi pertamanya, John McCharty menamai algoritma teknologi tersebut dengan sebutan “Artificial Intelligence”. Istilah tersebut masih digunakan hingga sekarang oleh para pakar teknologi.
- Terakhir, konsep dan teknologi kecerdasan buatan disempurnakan oleh seorang ahli yang namanya masih diingat sampai sekarang sebagai seorang pakar kecerdasan buatan, yaitu Alan Turin. Pada saat itu, Alan Turin meneliti dan menguji coba algoritma AI yang diberi nama dengan “Turing Test”. Hingga seiring berkembangnya waktu, konsep teknologi AI banyak digunakan di berbagai teknologi baik itu multimedia, search engine, dan masih banyak lainnya.

1.4.3 Supervised Learning

Supervised learning (Pembelajaran terawasi), dalam konteks kecerdasan buatan (AI) dan Machine Learning, adalah jenis sistem di mana input dan output data yang diinginkan disediakan. Input dan output data diberi label

untuk klasifikasi untuk memberikan dasar pembelajaran untuk pemrosesan data di masa depan.

1.4.4 Unsupervised Learning

Unsupervised learning merupakan pembelajaran yang tidak terawasi dimana tidak memerlukan target output. Pada metode ini tidak dapat ditentukan hasil seperti apa yang diharapkan selama proses pembelajaran, nilai bobot yang disusun dalam proses range tertentu tergantung pada nilai output yang diberikan. Tujuan metode unsupervised learning ini agar kita dapat mengelompokkan unit-unit yang hampir sama dalam satu area tertentu.

1.4.5 Teknik Klasifikasi

Teknik klasifikasi memprediksi respons diskrit, misalnya seperti apakah email itu asli atau spam, atau apakah tumor itu kanker atau tidak. Model klasifikasi mengklasifikasikan data masukan ke dalam kategori tersebut.

1.4.6 Regresi

Regresi yaitu pengeluaran nilai output yang konstan jika dipicu dengan parameter tertentu biasanya regresi disini berbentuk regresi linier. Regresi linier yaitu metode statistika yang digunakan untuk membentuk model hubungan antara variabel terikat(dependen,respon,Y) dengan satu atau lebih variabel bebas(independent, prdiktor, X). Apabila banyaknya variabel bebas hanya ada satu, disebut sebagai regresi linier sederhana, sedangkan apabila terdapat lebih dari satu variabel bebas, disebut sebagai regresi linier berganda.

1.4.7 Training Set

Training set adalah bagian dari dataset itu sendiri yang dilatih untuk membuat prediksi atau algoritma mesin learning lainnya sesuai keinginan atau tujuan data itu dibuat.

1.4.8 Testing Set

Testing set adalah bagian dari dataset yang di tes atau diujicoba untuk melihat keakuratannya dengan katalain melihat peformanya.

1.4.9 Instalasi dan Percobaan Kompilasi dari Library Scikit-learn

1. Buka anaconda prompt
2. kemudian Ketik di anaconda prompt yaitu : "pip install -U scikit-learn"



Gambar 1.43 Instalasi Scikit Learn


```
1 print(__doc__)
2
3 # Author: Nelle Varoquaux <nelle.varoquaux@gmail.com>
4 #          Alexandre Gramfort <alexandre.gramfort@inria.fr>
5 # License: BSD
6
7 import numpy as np
8 import matplotlib.pyplot as plt
9 from matplotlib.collections import LineCollection
10
11 from sklearn.linear_model import LinearRegression
12 from sklearn.isotonic import IsotonicRegression
13 from sklearn.utils import check_random_state
14
15 n = 100
16 x = np.arange(n)
17 rs = check_random_state(0)
18 y = rs.randint(-50, 50, size=(n,)) + 50. * np.log1p(np.arange(
19     n))
20 #
21 ######
22
23 # Fit IsotonicRegression and LinearRegression models
24
25 ir = IsotonicRegression()
26
27 y_ = ir.fit_transform(x, y)
28
29 lr = LinearRegression()
30 lr.fit(x[:, np.newaxis], y) # x needs to be 2d for
31 # LinearRegression
32 #
33 ######
34
35 # Plot result
36
37 segments = [[i, y[i]], [i, y_[i]]] for i in range(n)]
38 lc = LineCollection(segments, zorder=0)
39 lc.set_array(np.ones(len(y)))
40 lc.set_linewidths(np.full(n, 0.5))
41
42 fig = plt.figure()
```

```
39 plt.plot(x, y, 'r.', markersize=12)
40 plt.plot(x, y_, 'b.-', markersize=12)
41 plt.plot(x, lr.predict(x[:, np.newaxis]), 'b-')
42 plt.gca().add_collection(lc)
43 plt.legend(['Data', 'Isotonic Fit', 'Linear Fit'], loc='lower
        right')
44 plt.title('Isotonic regression')
45 plt.show()
```

5. Kemudian jalankan aplikasi tersebut, dan bisa dicek hasil dari Variable explorernya

Gambar 1.44 Variable Explorer

1.4.10 Mencoba Loading an example dataset


```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Feb 26 19:32:36 2020
4
5 @author: lenovo
6 """
7
8 from sklearn import datasets #import class/fungsi dataset
9         dari scikit-learn library
10
11 import matplotlib.pyplot as plt #import matplotlib
12
13 #Load the digits dataset
14 digits = datasets.load_digits() #memuat dan memasukkan
15         dataset digits ke variabel digits
16
17 #menampilkan digit pertama
18 plt.figure(1, figsize=(3, 3))
19 plt.imshow(digits.images[-1], cmap=plt.cm.gray_r ,
20             interpolation='nearest')
21 plt.show()
```

2. Kemudian jalankan aplikasi tersebut



Gambar 1.45 Dataset

1.4.11 Mencoba Learning and Predicting

1. Sample	kode
-----------	------

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 19:07:08 2020
4
5 @author: lenovo
6 """
7
8 from sklearn.linear_model import LogisticRegression #import
9         dari library sklearn
10 from sklearn.datasets import make_blobs #import dari library
11         sklearn
12
13 X, y = make_blobs(n_samples=100, centers=2, n_features=2,
14                     random_state=1) # generate dataset klasifikasi 2d
15
16 # fit final model
17 model = LogisticRegression()
18 model.fit(X, y)
19
20 Xnew, _ = make_blobs(n_samples=3, centers=2, n_features=2,
21                     random_state=1) # menentukan 1 buah contoh baru dimana
22                     jawabannya tidak diketahui
23
24 ynew = model.predict_proba(Xnew) # membuat prediksi
25 for i in range(len(Xnew)):
26     print("X=%s, Predicted=%s" % (Xnew[i], ynew[i])) #
27             menampilkan hasil prediksi

```

2. Kemudian	jalankan	aplikasi	tersebut
-------------	----------	----------	----------

```

In [18]: runfile('C:/Users/lenovo/Desktop/simple_3.py', wdir='C:/Users/lenovo/Desktop')
X=[-0.3520228  2.38095117], Predicted:[0.9987159  0.0012808]
X=[-2.5290000  3.18773168], Predicted:[0.9999999  0.0000001]
X=[-1.18773168  3.3332125], Predicted:[0.90389073  0.00010917]

```

Gambar 1.46 Predicting

1.4.12 Mencoba Model Persistence

Sample	kode
--------	------

```

1 # -*- coding: utf-8 -*-

```

```

2 """
3 Created on Thu Feb 27 19:36:22 2020
4
5 @author: lenovo
6 """

7
8 from sklearn import svm #import dari library sklearn
9 from sklearn import datasets #import dari library sklearn
10 clf = svm.SVC() #menggunakan Support Vector Classifier
11 iris = datasets.load_iris() #menggunakan iris dataset
12 X, y = iris.data, iris.target
13 clf.fit(X, y)

```

1.4.13 Mencoba Conventions

Sample kode

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Feb 27 20:04:46 2020
4
5 @author: lenovo
6 """

7
8 import numpy as np #import dari library sklearn
9 from sklearn import random_projection #import dari library
   sklearn
10
11 rng = np.random.RandomState(0) #Menggunakan fungsi random
12 X = rng.rand(10, 2000) #mengambil nilai random
13 X = np.array(X, dtype='float32') #membuat array bertipe float32
14 X.dtype

```

1.5 1174050 Dika Sukma Pradana

1.5.1 Teori

1. Definisi, sejarah, dan perkembangan kecerdasan buatan.

Intelejenji buatan (AI) mengacu pada simulasi kecerdasan manusia dalam mesin yang diprogram untuk berpikir seperti manusia dan meniru tindakan mereka. Istilah ini juga dapat diterapkan pada mesin apa pun yang menunjukkan sifat-sifat yang terkait dengan pikiran manusia seperti pembelajaran dan pemecahan masalah.

Sejarah Kecerdasan Buatan (AI) bermula pada saat zaman kuno, dengan sejuta mitos, cerita dan desas-desus tentang makhluk buatan yang diberkahii dengan kecerdasan oleh pengrajin ahli. Benih-benih AI modern ditanam oleh para filsuf klasik yang mencoba menggam-

barkan proses pemikiran manusia sebagai manipulasi simbol secara mekanis. Karya ini memuncak dalam penemuan komputer digital yang dapat diprogram pada tahun 1940-an, sebuah mesin yang didasarkan pada esensi abstrak penalaran matematika. Perangkat ini dan ide-ide di belakangnya menginspirasi segenap ilmuwan untuk mulai serius membahas kemungkinan membangun otak elektronik. Bidang penelitian AI didirikan pada lokakarya yang diadakan di kampus Dartmouth College selama musim panas 1956. Mereka yang hadir akan menjadi pemimpin penelitian AI selama beberapa dekade. Banyak dari mereka meramalkan bahwa sebuah mesin yang secerdas manusia akan hidup tidak lebih dari satu generasi dan mereka diberi jutaan dolar untuk mewujudkan visi atau tujuan ini. Akhirnya, menjadi jelas bahwa mereka meremehkan kesulitan proyek. Pada tahun 1973, sebagai tanggapan atas kritik dari James Lighthill dan tekanan terus-menerus dari kongres, AS dan Pemerintah Inggris menghentikan pendanaan penelitian yang tidak diarahkan pada kecerdasan buatan, dan tahun-tahun sulit berikutnya akan dikenal sebagai musim dingin AI. Tujuh tahun kemudian, sebuah inisiatif visioner oleh Pemerintah Jepang mengilhami pemerintah dan industri untuk menyediakan miliaran dolar dalam AI, tetapi pada akhir 80-an investor menjadi kecewa dengan kurangnya daya komputer (perangkat keras) yang dibutuhkan dan menarik lebih banyak dana. Investasi dan minat pada AI tumbuh pesat pada dekade pertama abad ke-21, ketika pembelajaran mesin berhasil diterapkan pada banyak masalah di dunia akademis dan industri karena metode baru, penerapan perangkat keras komputer yang kuat, dan kumpulan data yang sangat besar.

2. Definisi supervised learning, klasifikasi, regresi, dan unsupervised learning. Data set, training set dan testing set.

Supervised learning adalah sebuah metode pendekatan yang mana sudah terdapat data yang dilatih, dan terdapat variabel yang ditargetkan atau yang menjadi tujuan sehingga tujuan dari pendekatan ini adalah mengelompokan suatu data ke data yang sudah ada. Sedangkan unsupervised learning tidak memiliki data latih, sehingga dari data yang ada, kita mengelompokan data tersebut menjadi dua bagian atau bahkan tiga bagian dan seterusnya.

Klasifikasi adalah salah satu topik utama dalam data mining atau machine learning. Klasifikasi yaitu suatu pengelompokan data dimana data yang digunakan tersebut mempunyai kelas label atau target.

Regresi adalah Supervised learning tidak hanya mempelajari classifier, tetapi juga mempelajari fungsi yang dapat memprediksi suatu nilai numerik.

Data set merupakan sebuah cabang aplikasi dari Artificial Intelligence(AI)/Kecerdasan Buatan yang terfokus kepada pengembangan se-

b. buah sistem yang mampu belajar sendiri tanpa harus berulang kali di program oleh manusia(programmer).

Training set yaitu jika pasangan objek, dan kelas yang menunjuk pada objek tersebut adalah suatu contoh yang telah diberi label akan menghasilkan suatu algoritma pembelajaran.

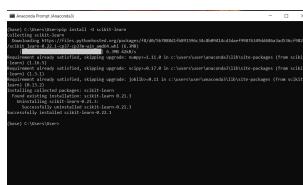
Testing set digunakan untuk mengukur sejauh mana classifier berhasil melakukan klasifikasi dengan benar[?].

1.5.2 Instalasi

1.5.2.1 Instalasi Library Scikit dari Anaconda

1. Download aplikasi Anaconda terlebih dahulu.
 2. Install aplikasi Anaconda yang sudah di download tadi.
 3. Simpan aplikasi sesuai folder yang kita pilih lalu next.
 4. Centang Keduanya lalu tekan tombol install.
 5. Setelah itu tunggu sampai proses instalasi selesai lalu jika sudah tekan tombol finish.
 6. Lalu buka command prompt anda dan tuliskan perintah berikut ini untuk mengecek apakah aplikasinya sudah terinstall.

```
pip install -U scikit-learn
```
 7. Kemudian ketikkan perinta pip install -U scikit-learn seperti gambar berikut.



Gambar 1.47 Instalasi

1.5.3 Percobaan

Mencoba Library

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Feb 26 22:51:53 2020
4
5 @author: User
6 """
```

```

7
8 import matplotlib.pyplot as plt
9 from sklearn.svm import SVC
10 from sklearn.ensemble import RandomForestClassifier
11 from sklearn.metrics import plot_roc_curve
12 from sklearn.datasets import load_wine
13 from sklearn.model_selection import train_test_split
14
15 X, y = load_wine(return_X_y=True)
16 y = y == 2
17
18 X_train, X_test, y_train, y_test = train_test_split(X, y,
19 random_state=42)
20 svc = SVC(random_state=42)
21 svc.fit(X_train, y_train)

```

SVC(random_state=42)

Gambar 1.48 Variabel Explore

```

1 from sklearn import datasets
2 #perintah untuk mengimport class datasets dari packaged sklearn
3 iris = datasets.load_iris()
4 #iris merupakan suatu estimator/parameter yang berfungsi untuk
5 #mengambil data pada item datasets.load_iris
5 digits = datasets.load_digits()
6 #digits merupakan suatu estimator/parameter yang berfungsi untuk
7 #mengambil data pada item datasets.load_digits
7 print(digits.data)
8 #perintah yang berfungsi untuk menampilkan estimator/parameter
9 #yang dipanggil pada item digits.data dan menampilkan
10 #outputannya
10 digits.target
11 #untuk mengambil target pada estimator/parameter digits dan
12 #menampilkan outputannya
11 digits.images[0]
12 #untuk mengambil images[0] pada estimator/parameter digits dan
13 #menampilkan outputannya

```

```

In [13]: runfile('C:/Users/User/Documents/Sayler/1.py', wdir='C:/Users/User/Documents/')
Spiders
[[ 0.,  0.,  5., ...,  0.,  0.,  0.],
 [ 0.,  0.,  0., ..., 10.,  0.,  0.],
 [ 0.,  0.,  0., ...,  0.,  0.,  0.],
 ...,
 [ 0.,  0.,  1., ...,  6.,  0.,  0.],
 [ 0.,  0.,  2., ..., 12.,  0.,  0.],
 [ 0.,  0.,  0., ..., 12.,  1.,  0.]]

```

Gambar 1.49 Datasets

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Feb 26 22:17:01 2020

```

```
4
5 @author: User
6 """
7 from sklearn import svm
8 #merupakan sebuah perintah untuk mengimport class svm dari
9     packaged sklearn
10 # merupakan sebuah perintah untuk mengimport class datasets dari
11     packaged sklearn
12 clf = svm.SVC(gamma='scale')
13 #clf sebagai estimator/parameter, svm.SVC sebagai class , gamma
14     sebagai parameter untuk menetapkan nilai secara manual dengan
15     nilai scale
16 iris = datasets.load_iris()
17 #iris sebagai estimator/parameter, datasets.load_iris() sebagai
18     item dari suatu nilai
19 X, y = iris.data, iris.target
20 #X, y sebagai estimator/parameter, iris.data, iris.target sebagai
21     item dari 2 nilai yang ada
22 clf.fit(X, y)
23 #clf sebagai estimator/parameter dengan menggunakan metode fit
24     untuk memanggil estimator X, y dengan outputannya
```

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Feb 26 23:06:05 2020
4
5 @author: User
6 """
7
8 import pickle
9 #pickle merupakan sebuah class yang di import
10 s = pickle.dumps(clf)
11 #s sebagai estimator/parameter dengan pickle.dumps merupakan
12     suatu nilai/item dari estimator/parameter clf
13 clf2 = pickle.loads(s)
14 #clf2 sebagai estimator/parameter, pickle.loads sebagai suatu
15     item, dan s sebagai estimator/parameter yang dipanggil
16 clf2.predict(X[0:1])
17 #clf2.predict sebagai suatu item dengan menggunakan metode
18     predict untuk menentukan suatu nilai dari (X[0:1])
19 y[0]
20 #pada estimator/parameter y berapapun angka yang diganti nilainya
21     akan selalu konstan yaitu 0
22
23
24 from joblib import dump, load
25 #sebuah perintah untuk mengimport class dump, load dari packaged
26     joblib
27 dump(clf, 'filename.joblib')
28 #dump di sini sebagai class yang didalamnya terdapat nilai dari
29     suatu item clf dan data joblib
30 clf = load('filename.joblib')
31 #clf sebagai estimato/parameter dengan suatu nilai load berfungsi
32     untuk mengulang data sebelumnya
```

1.5.3.4 Conventions Type Casting

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Feb 26 22:45:29 2020
4
5 @author: User
6 """
7
8 #Type Casting
9 from sklearn import svm
10 #pada baris ini merupakan sebuah perintah untuk mengimport class
11 #svm dari packaged sklearn
12 from sklearn import random_projection
13 #pada baris ini merupakan sebuah perintah untuk mengimport class
14 #random projection dari packaged sklearn
15 rng = np.random.RandomState(0)
16 #rng sebagai estimator/parameter dengan nilai suatu itemnya yaitu
17 #np.random.RandomState(0)
18 X = rng.rand(10, 2000)
19 #X sebagai estimator/parameter dengan nilai item rng.rand
20 X = np.array(X, dtype='float32')
21 #X sebagai estimator/parameter dengan nilai item np.array
22 X.dtype
23 #X.dtype sebagai item pemanggil
24 transformer = random_projection.GaussianRandomProjection()
25 #transformer sebagai estimator/parameter dengan memanggil class
26 #random projection
27 X_new = transformer.fit_transform(X)
28 #X new di sini sebagai estimator/parameter dan menggunakan metode
29 #fit
30 X_new.dtype
31 #X new.dtype sebagai item
32
33
34 from sklearn import datasets
35 #pada baris ini merupakan sebuah perintah untuk mengimport class
36 #datasets dari packaged sklearn
37 from sklearn.svm import SVC
38 #pada baris ini merupakan sebuah perintah untuk mengimport class
39 #SVC dari packaged sklearn.svm
40 iris = datasets.load_iris()
41 #iris sebagai estimator/parameter dengan item datasets.load_iris
42 #()
43 clf = SVC(gamma='scale')
44 #clf sebagai estimator/parameter dengan nilai class SVC pada
45 #parameter gamma sebagai set penilaian
46 clf.fit(iris.data, iris.target)
47 #estimator/parameter clf menggunakan metode fit dengan itemnya
48 list(clf.predict(iris.data[:3]))
49 #menambahkan item list dengan metode predict
50 clf.fit(iris.data, iris.target_names[iris.target])
51 #estimator/parameter clf menggunakan metode fit dengan itemnya
52 list(clf.predict(iris.data[:3]))
53 #menambahkan item list dengan metode predict
```

Refitting and updating parameters

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Feb 26 23:08:34 2020
4
5 @author: User
6 """
7
8 #Multiclass vs. Multilabel Fitting
9 from sklearn.svm import SVC
10 #pada baris ini merupakan sebuah perintah untuk mengimport class
11 #SVC dari packaged sklearn.svm
12 from sklearn.multiclass import OneVsRestClassifier
13 #pada baris ini merupakan sebuah perintah untuk mengimport class
14 #OneVsRestClassifier dari packaged sklearn.multiclass
15 from sklearn.preprocessing import LabelBinarizer
16 #pada baris ini merupakan sebuah perintah untuk mengimport class
17 #LabelBinarizer dari packaged sklearn.preprocessing
18 X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]
19 y = [0, 0, 1, 1, 2]
20 classif = OneVsRestClassifier(estimator=SVC(gamma='scale',
21                                     random_state=0))
22 classif.fit(X, y).predict(X)
23 y = LabelBinarizer().fit_transform(y)
24 classif.fit(X, y).predict(X)
25
26 from sklearn.preprocessing import MultiLabelBinarizer
27 y = [[0, 1], [0, 2], [1, 3], [0, 2, 3], [2, 4]]
28 y = MultiLabelBinarizer().fit_transform(y)
29 classif.fit(X, y).predict(X)

```

Multiclass vs. multilabel fitting

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Feb 26 23:10:03 2020
4
5 @author: User
6 """
7
8 #Refitting and Updating Parameters
9 import numpy as np
10 #pada baris ini merupakan sebuah perintah untuk mengimport class
11 #svm dari np
12 from sklearn.svm import SVC
13 #pada baris ini merupakan sebuah perintah untuk mengimport class
14 #SVC dari packaged sklearn.svm
15 rng = np.random.RandomState(0)
16 #rng sebagai estimator/parameter dengan nilai suatu itemnya yaitu
17 #np.random.RandomState(0)
18 X = rng.rand(100, 10)
19 #X sebagai estimator/parameter dengan nilai item rng.rand
20 y = rng.binomial(1, 0.5, 100)
21 #y sebagai estimator/parameter dengan nilai item rng.binomial
22 X_test = rng.rand(5, 10)
23 #X test sebagai estimator/parameter dengan nilai item rng.rand

```

```
21 clf = SVC()
22 #clf sebagai estimator/parameter dan class SVC
23 clf.set_params(kernel='linear').fit(X, y)
24 #set params sebagai item
25 clf.predict(X_test)
26 #menggunakan metode predict
27 clf.set_params(kernel='rbf', gamma='scale').fit(X, y)
28 clf.predict(X_test)
```

1.5.4 Penanganan eror

```
File "C:\Users\User\Anaconda3\lib\site-packages\spyder_kernels\customize
\spydercustomize.py", line 827, in runfile
    execfile(filename, namespace)

File "C:\Users\User\Anaconda3\lib\site-packages\spyder_kernels\customize
\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

File "C:/Users/User/Documents/Spyder/2.py", line 14
    X, y = iris.data, iris.target
          ^
IndentationError: unexpected indent
```

Gambar 1.50 Error

1.5.4.1 ScreenShoot Eror

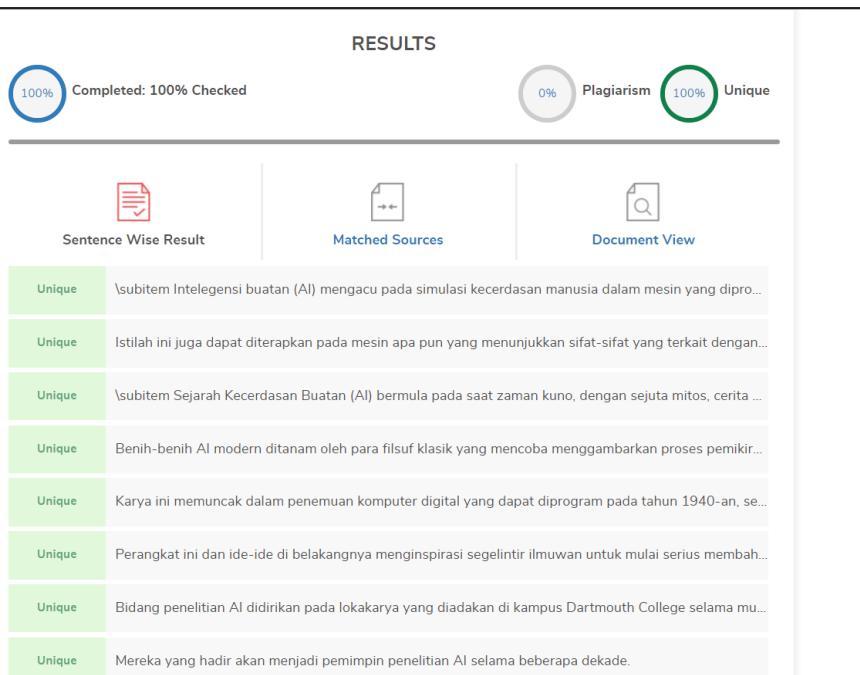
1.5.4.2 Tuliskan Kode Eror dan Jenis Erornya

IndentationError: unexpected indent

1.5.4.3 Solusi Pemecahan Masalah Error

Pastikan semua spasi pada koding sama. Menggunakan spasi atau tab.

1.5.5 Plagiarism



Gambar 1.51 Plagiarism

1.6 1174057 Alit Fajar Kurniawan

1.6.1 Teori

1.6.1.1 Sejarah Perkembangan dan Definisi Artificial Intelligence Kecerdasan buatan merupakan sebuah bidang dalam ilmu computer yang begitu penting di zaman ini dan masa yang akan datang guna mewujudkan sebuah sistem computer yang begitu cerdas. Artificial Intelligence atau biasa disingkat dengan AI berasal dari bahasa latin yang dimana intelligence berarti saya paham.

Pada tahun 1955, Newell dan juga Simon telah mengembangkan The Logic Theorist, yaitu program AI pertama. Dimana program tersebut mempresentasikan sebuah masalah sebagai model pohon, lalu diselesaikan dengan cara memilih cabang yang akan mewujudkan kesimpulan terbenar dan tepat. Program AI tersebut berdampak sangat besar dan dapat mendjadi batu loncatan yang cukup penting dalam mengembangkan bidang AI [?].

Masa Perkembangan AI dimulai pada awal era komputer elektronik pada tahun 1941. dimana ditemukannya alat penyimpanan dan pemrosesan informasi. kemudian dilanjutkan pada masa-masa persiapan AI yang terjadi

pada tahun 1943-1956. Pada sekitaran tahun 1952-1969 merupakan masa awal perkembangan AI terjadi, dan pada tahun 1966-1974 perkembangan AI mengalami penurunan atau melambatnya proses dalam melakukan pengembangan. pada tahun 1969 sampai 10 tahun kedepan kembali terjadi perkembangan yang menciptakan inovasi sistem berbasis pengetahuan. dan sekitaran tahun 1980-an AI kembali menjadi sebuah industri yang terus berkembang sampai sekarang ini.

1.6.1.2 learning, klasifikasi, regresi dan unsupervised learning. Data set, training set dan testing set

1. Definisi Supervised Learning Dan Unsupervised Learning

Supervised Learning merupakan suatu pendekatan yang dimana terdapat data dan variable yang telah ditargetkan sehingga pendekatan tersebut bertujuan untuk dapat mengelompokkan sebuah data ke data yang sudah ada, beda dengan Unsupervised learning yang tidak mempunyai data, sehingga data yang ada harus di kelompokkan menjadi beberapa bagian.

2. Definisi Klasifikasi Dan Regresi

Klasifikasi adalah sebuah kegiatan penggolongan atau pengelompokan. Menurut kamus besar bahasa Indonesia yang dimana klasifikasi merupakan penyusunan sistem di dalam kelompok atau golongan berdasarkan kaidah atau standar yang telah ditetapkan. Regresi adalah sebuah metode analisis statistic yang akan digunakan untuk melihat pengaruh variable.

3. Devinisi Dataset, Training Set, Dan Testing Set

Dataset adalah sebuah objek yang akan mempresentasikan sebuah data dan relasinya di memory. Struktur pada dataset ini mirip dengan data yang ada di dalam database. Training set adalah bagian dari dataset yang berperan dalam membuat prediksi atau algoritma sesuai tujuan masing-masing. Testing set adalah bagian dari dataset yang akan di tes guna melihat keakuratan atau ketepatan datanya.

1.6.2 Praktek

1.6.2.1 Instalasi

1. Melakukan instalasi library scikit pada anaconda, ketik kan pip install -U scikit-learn pada terminal anaconda.

```
(base) C:\Users\alitf>pip install -U scikit-learn
Collecting scikit-learn
  Downloading https://files.pythonhosted.org/packages/f8/d0/5b7088d1fb891596c34c8b09414cd3daef99876349dd686a3ad536cf9820
    /scikit_learn-0.22.1-cp37-cp37m-win_amd64.whl (6.3MB)
      |████████| 6.3MB 297KB/s
Requirement already satisfied, skipping upgrade: numpy>=1.11.0 in c:\users\alitf\anaconda3\lib\site-packages (from scikit-learn) (1.16.4)
Requirement already satisfied, skipping upgrade: joblib>=0.11 in c:\users\alitf\anaconda3\lib\site-packages (from scikit-learn) (0.13.2)
Requirement already satisfied, skipping upgrade: scipy>=0.17.0 in c:\users\alitf\anaconda3\lib\site-packages (from scikit-learn) (1.2.1)
Installing collected packages: scikit-learn
  Found existing installation: scikit-learn 0.21.2
    Uninstalling scikit-learn-0.21.2:
      Successfully uninstalled scikit-learn-0.21.2
Successfully installed scikit-learn-0.22.1

(base) C:\Users\alitf>
```

Gambar 1.52 Instalasi Scikit Learn

- Setelah selesai instalasi, pilih salah satu example dari website Scikit.

Examples

Miscellaneous examples

Miscellaneous and introductory examples for scikit-learn.

Compact estimator representations

ROC Curve with Visualization API

Isotonic Regression

Advanced Plotting With Partial Dependence

Logistic regression coefficients visualization

Decision regions visualization

Cross-validation visualization

Partial dependence visualization

Gambar 1.53 Example

```

1 print(__doc__)
2
3 from sklearn.linear_model import LogisticRegression
4 from sklearn import set_config
5
6
7 lr = LogisticRegression(penalty='l1')
8 print('Default representation:')
```

```

9 print(lr)
10 # LogisticRegression(C=1.0, class_weight=None, dual=False ,
11 #                         fit_intercept=True,
12 #                         max_iter=100,
13 #                         multi_class='auto', n_jobs=None, penalty
14 #                         ='l1',
15 #                         random_state=None, solver='warn', tol
16 #                         =0.0001, verbose=0,
17 #                         warm_start=False)
18
19 set_config(print_changed_only=True)
20 print('\nWith changed_only option:')
21 print(lr)
22 # LogisticRegression(penalty='l1')

```

kemudian coba jalankan, lihat hasilnya

@author: alitf

Default representation:

`LogisticRegression(penalty='l1')`

With changed_only option:

`LogisticRegression(penalty='l1')`

Gambar 1.54 Example

3. latihan 2 Mencoba Loading an example dataset

```

1 from sklearn import datasets #mengimport class dataset dari
2     scikit learn library
3 iris = datasets.load_iris() # memuat dan memasukkan dataset
4     iris ke variabel bernama iris
5 digits = datasets.load_digits() #memuat dan memasukkan
6     dataset digits ke variabel digits
7
8 print(digits.data) #memberikan akses ke fitur yang dapat
9     digunakan untuk mengklasifikasikan sampel digit dan
10    menampilkannya di console
11
12 digits.target #memberikan informasi tentang data yang
13    berhubungan atau juga dapat dijadikan sebagai label
14
15 digits.images[0] #Data selalu berupa array 2D, shape (
16     n_samples, n_features), meskipun data aslinya mungkin
17    memiliki bentuk yang berbeda.

```

hasil dari data digits

```
In [17]: runfile('C:/Users/alitf/OneDrive/Desktop/AI/src/1174057'
              wdir='C:/Users/alitf/OneDrive/Desktop/AI/src/1174057/chapter1')
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0.  10. ... 12.  1.  0.]]
```

Gambar 1.55 Result Data Digits

hasil dari digits.target

```
In [19]: digits.target #memberikan informasi
          dijadikan sebagai Label
Out[19]: array([0, 1, 2, ..., 8, 9, 8])
```

Gambar 1.56 Result digits.target

hasil dari digits.image

```
In [20]: digits.images[0] #Data selalu berupa array 2D, s
          meskipun data aslinya mungkin memiliki bentuk yang berbeda
Out[20]:
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

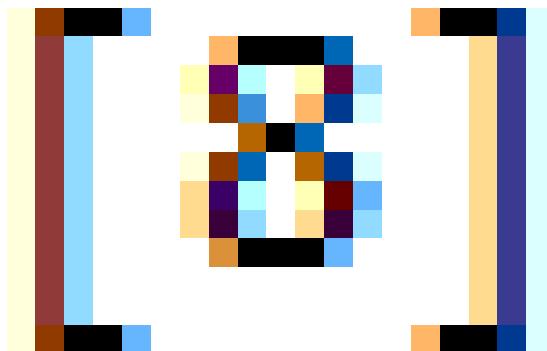
Gambar 1.57 Result digits.image

```

1 from sklearn import svm # perintah untuk mengimport class svm
    dari packaged sklearn
2
3 digits = datasets.load_digits() #memuat dan memasukkan
    dataset digits ke variabel digits
4
5 clf = svm.SVC(gamma=0.001, C=100.) #clf sebagai estimator/
    parameter , svm.SVC sebagai class , gamma sebagai
    parameter untuk menetapkan nilai secara manual
6
7 clf.fit(digits.data[:-1], digits.target[:-1]) # clf sebagai
    estimator/parameter , f i t sebagai metode , digits . data
    sebagai item , [:1] sebagai syntax pythonnya dan
    menampilkan outputannya
8
9 print(clf.predict(digits.data[-1])) #clf sebagai estimator/
    parameter , predict sebagai metode lainnya , digits . data
    sebagai item dan menampilkan outputannya

```

kemudian coba jalankan, lihat hasilnya



Gambar 1.58 Result Learning and predicting

5. latihan 4 Mencoba Model persistence

```

1 from sklearn import svm, datasets #mengimport class dataset
    dari scikit learn library

```

```

2 clf = svm.SVC(gamma=0.001, C=100.) #memanggil class SVC dan
   menyet argument constructor SVC serta ditampung di
   variable clf
3 X, y = datasets.load_iris(return_X_y=True) #meload datasets
   iris dan ditampung di variable x untuk data dan y untuk
   target
4 clf.fit(X, y) #memanggil method fit untuk melakukan training
   data dengan argumen data dan target dari datasets iris
5
6 #Pickle
7 import pickle #mengimport pickle
8 s = pickle.dumps(clf) #memanggil method dumps dengan argumen
   clf dan ditampung di variable s
9 clf2 = pickle.loads(s) #memanggil method loads dengan argumen
   s dan ditampung di variable clf2
10 print(clf2.predict(X[0:1])) #menampilkan hasil dari method
    predict dengan argumen data variable X pertama
11
12 #Joblib
13 from joblib import dump, load #mengimport dump dan load dari
   library joblib
14 dump(clf, '1174057.joblib') #memanggil method dumps dengan
   argumen clf dan nama file joblibnya
15 clf3 = load('1174057.joblib')#memanggil method loads dengan
   argumen nama file joblibnya dan ditampung di variable clf3
16 print(clf3.predict(X[0:1])) #menampilkan hasil dari method
    predict dengan argumen data variable X pertama

```

kemudian coba jalankan, lihat hasilnya

```

In [13]: runfile('D:/Data Alit/Kuliah,
modelpersistence.py', wdir='D:/Data A:
[0]
[0]

```

Gambar 1.59 Result Model persistence

6. latihan 5 Mencoba Conventions

```

1 #Type casting
2 import numpy as np
3 from sklearn import random_projection
4 rng = np.random.RandomState(0)
5 X = rng.rand(10, 2000)
6 X = np.array(X, dtype='float32')
7 print(X.dtype)
8 transformer = random_projection.GaussianRandomProjection()
9 X_new = transformer.fit_transform(X)

```

```
10 print(X_new.dtype)
11
12 from sklearn import datasets
13 from sklearn.svm import SVC
14 iris = datasets.load_iris()
15 clf = SVC(gamma=0.001, C=100.)
16 clf.fit(iris.data, iris.target)
17 print(list(clf.predict(iris.data[:3])))
18 clf.fit(iris.data, iris.target_names[iris.target])
19 print(list(clf.predict(iris.data[:3])))
20
21 #Refitting and updating parameters
22 import numpy as np
23 from sklearn.datasets import load_iris
24 from sklearn.svm import SVC
25 X, y = load_iris(return_X_y=True)
26 clf = SVC(gamma=0.001, C=100.)
27 clf.set_params(kernel='linear').fit(X, y)
28 print(clf.predict(X[:5]))
29 clf.set_params(kernel='rbf').fit(X, y)
30 print(clf.predict(X[:5]))
31
32 #Multiclass vs. Multilabel Fitting
33 from sklearn.svm import SVC
34 from sklearn.multiclass import OneVsRestClassifier
35 from sklearn.preprocessing import LabelBinarizer
36 X = [[1, 2], [2, 4], [4, 5], [3, 2], [3, 1]]
37 y = [0, 0, 1, 1, 2]
38 classif = OneVsRestClassifier(estimator=SVC(random_state=0,
                                              gamma=0.001, C=100.))
39 print(classif.fit(X, y).predict(X))
40 y = LabelBinarizer().fit_transform(y)
41 print(classif.fit(X, y).predict(X))
42
43 from sklearn.preprocessing import MultiLabelBinarizer
44 y = [[0, 1], [0, 2], [1, 3], [0, 2, 3], [2, 4]]
45 y = MultiLabelBinarizer().fit_transform(y)
46 print(classif.fit(X, y).predict(X))
```

kemudian coba jalankan, lihat hasilnya

```
In [20]: runfile('C:/Users/Alit/Desktop/KB3A/src/1174006/chapter1/coba4.py', wdir='C:/Users/Alit/Desktop/KB3A/src/1174006/chapter1')
float32
float64
[0, 0, 0]
['setosa', 'setosa', 'setosa']
[0 0 0 0]
[0 0 0 0]
[0 0 1 1 2]
[[1 0 0]
 [1 0 0]
 [0 1 0]
 [0 0 0]
 [0 0 0]
 [[1 0 1 0 0]
 [1 0 1 0 0]
 [1 0 1 1 0]
 [1 0 1 0 0]
 [1 0 1 0 0]]]
```

Gambar 1.60 Result Conventions

1.6.3 Penanganan Error

1. Screenshot Error

```
File "D:/Data Alit/Kuliah/SEMESTER VI/AI/src/1174057/chapter1/learning.py", line 5, in
<module>
    clf.fit(digits.data[:-1], digits.target[:-1]) # clf sebagai estimator/parameter , f
i t sebagai metode , digits . data sebagai item , [:1] sebagai syntax pythonnya dan
menampilkan outputannya

NameError: name 'digits' is not defined
```

Gambar 1.61 Error

```
File "D:/Data Alit/Kuliah/SEMESTER VI/AI/src/1174057/chapter1/learning.py", line 3
    digits = datasets . load digits () #meload datasets digits dan ditampung di variable
digits
^
SyntaxError: invalid syntax
```

Gambar 1.62 Error

2. Tuliskan kode dan jenis error

is not defined, xception yang terjadi saat syntax melakukan eksekusi terhadap local name atau global name yang tidak terdenisi.

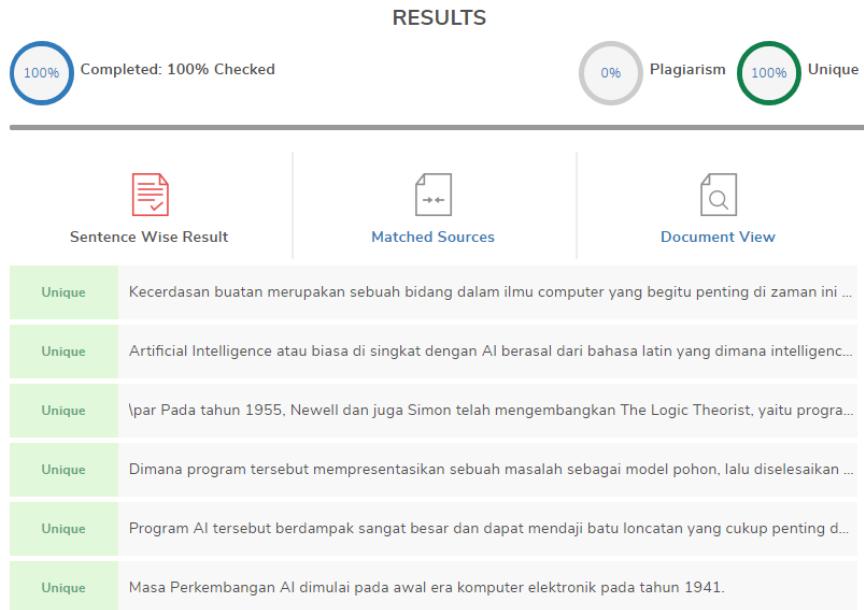
invalid syntax

3. Solusi penanganan error

Solusinya adalah memastikan variabel atau function yang dipanggil ada atau tidak salah ketik.

Periksa kembali syntax yang dibuat, bisa saja ada kesalahan dalam spasi.

1.6.4 Bukti Tidak Plagiat



Gambar 1.63 Plagiarisme

1.7 1174039 - Liyana Majdah Rahma

1.7.1 Teori

1.7.1.1 Definisi Kecerdasan Buatan Artificial Intelligence adalah kecerdasan yang ditambahkan pada suatu sistem yang dapat diatur dalam konteks secara ilmiah. Menurut Michael Haelein menyatakan bahwa AI merupakan "sistem yang mempunyai kemampuan untuk menguraikan, belajar agar dapat dgunakan untuk mencapai tujuan dengan melalui adaptasi yang fleksibel". Kecerdasan buatan juga dapat dibuat dan dimasukkan ke dalam mesin agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia dengan cepat dan tepat.

1.7.1.2 Sejarah Kecerdasan Buatan Sejarah Kecerdasan buatan pertama kali terjadi pada zaman kuno, terdapat mitos ataupun cerita dan desas-desus tentang sebuah makhluk yang mempunyai kecerdasan serta kesadaran yang

diberikan oleh pengrajin. Benih - benih nya mulai ditanam oleh para filsuf klasik yang mencari cara untuk menggambarkan proses berfikir manusia sebagai manipulasi simbol secara mekanis yang memuncak pada penemuan komputer digital di tahun 1940-an, yaitu sebuah mesin yang didasarkan penalaran matematika. Istilah kecerdasan buatan sendiri baru muncul pada tahun 1956, dan teori -teori nya sudah muncul sejak tahun 1941. Dan Pada tahun 1973, sebagai tanggapan atas kritik dari James Lighthill dan tekanan terus-menerus dari kongres, AS dan Pemerintah Inggris menghentikan pendanaan penelitian yang tidak diarahkan pada kecerdasan buatan, dan tahun-tahun sulit berikutnya akan dikenal sebagai musim dingin AI. Tujuh tahun kemudian, sebuah inisiatif visioner oleh Pemerintah Jepang mengilhami pemerintah dan industri untuk menyediakan miliaran dolar dalam AI, tetapi pada akhir 80-an investor menjadi kecewa dengan kurangnya daya komputer (perangkat keras) yang dibutuhkan dan menarik lebih banyak dana.

1.7.1.3 Perkembangan Kecerdasan Buatan

1. Pada tahun 1958, McCarthy di MIT AI Lab Memo orang pertama yang mendefinisikan bahasa pemrograman pada tingkat tinggi yaitu LISP, yang sekarang mendominasi pembuatan program-program kecerdasan buatan.
2. Kemudian Pada tahun 1959, Nathaniel Rochester dari IBM serta mahasiswa-mahasiswanya mengeluarkan program kecerdasan buatan yaitu Geometry Theorem Prover. selain itu juga Program ini dapat mengeluarkan suatu teorema menggunakan aksioma-aksioma yang ada.
3. Sistem Berbasis Pengetahuan pada tahun 1969-1979
4. Kecerdasan Buatan menjadi sebuah industri pada tahun 1980-1988. Kembaliya Jaringan Syaraf tiruan pada tahun 1986-sekarang.

1.7.2 Instalasi

1.7.2.1 Instalasi Library Scikit dari Anaconda

1. Download aplikasi Anaconda terlebih dahulu.
2. Install aplikasi Anaconda yang sudah di download tadi.
3. Simpan aplikasi sesuai folder yang kita pilih lalu next.
4. Centang Keduanya lalu tekan tombol install.
5. Setelah itu tunggu sampai proses instalasi selesai lalu jika sudah tekan tombol finish.
6. Lalu buka command prompt anda dan tuliskan perintah berikut ini untuk mengecek apakah aplikasinya sudah terinstall.

7. Kemudian ketikkan perintah pip install -U scikit-learn seperti gambar berikut.



Gambar 1.64 Instalasi

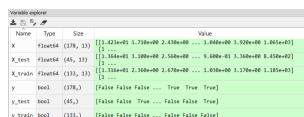
1. Pilih salah satu example dari website tersebut lalu jalankan

```

1 import matplotlib.pyplot as plt
2 from sklearn.svm import SVC
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.metrics import plot_roc_curve
5 from sklearn.datasets import load_wine
6 from sklearn.model_selection import train_test_split
7
8 X, y = load_wine(return_X_y=True)
9 y = y == 2
10
11 X_train, X_test, y_train, y_test = train_test_split(X, y,
12         random_state=42)
13 svc = SVC(random_state=42)
14 svc.fit(X_train, y_train)

```

2. buka variable explolernya



Gambar 1.65 Variable Exploler

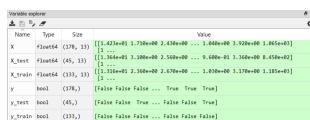
- salah satu example dari website tersebut lalu jalankan

```

1 import matplotlib.pyplot as plt
2 from sklearn.svm import SVC
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.metrics import plot_roc_curve
5 from sklearn.datasets import load_wine
6 from sklearn.model_selection import train_test_split
7
8 X, y = load_wine(return_X_y=True)
9 y = y == 2
10
11 X_train, X_test, y_train, y_test = train_test_split(X, y,
12         random_state=42)
13 svc = SVC(random_state=42)
14 svc.fit(X_train, y_train)

```

buka variable explolernya



Gambar 1.66 Variable Exploler

8fcee3a95c7a660681b429f909ba80ac1a4fa418

1.7.2.2 Mencoba loading an example dataset

1. mengambil data iris dan digit dari dataset

```
1 iris = datasets.load_iris() # Dapat membuat sebuah variable  
    iris yang mempunyai isi yaitu dataset iris  
2 digits = datasets.load_digits() # Digunakan untuk membuat  
    sebuah variable digits yang mempunyai isi yaitu dataset  
    digits
```

- ## 2. Menampilkan data digits



Gambar 1.67 Data Digits

3. menampilkan digits.target

```
In [16]: digits.target  
Out[16]: array([0, 1, 2, ..., 8, 9, 8])
```

Gambar 1.68 Digits Target

4. menampilkan data bentuk 2D.

```
In [17]: digits.images[0]
Out[17]:
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

Gambar 1.69 Data 2D

1.7.2.3 Mencoba Learning and Predicting

```
1 from sklearn.linear_model import LogisticRegression # digunakan
   untuk mengimport linear_model dari library sklearn
2 from sklearn.datasets import make_blobs #digunakan untuk
   mengimport library datasets dari sklearn
3
4 X, y = make_blobs(n_samples=100, centers=2, n_features=2,
   random_state=1) # dapat untuk generate dataset dengan
   klasifikasi 2D
5 model = LogisticRegression() # dapat menggunakan metode loginstic
   regression
6 model.fit(X, y)
7
8 Xnew, _ = make_blobs(n_samples=3, centers=2, n_features=2,
   random_state=1) # dapat menentukan 1 buah contoh baru dimana
   jawabannya tidak dapat diketahui
9
10 ynew = model.predict_proba(Xnew) # membuat sebuah prediksi dan
   memasukkan nya kedalam variable ynew
11 for i in range(len(Xnew)):
12     print("X=%s, Predicted=%s" % (Xnew[i], ynew[i])) #menampilkan
   hasil prediksi
```

1.7.2.4 Mencoba Model Persistence

```
1 from sklearn import svm #dapat mengimport svm dari library
   sklearn
2 from sklearn import datasets #digunakan untuk mengimport datasets
   dari library sklearn
3 clf = svm.SVC() # digunakan dengan menggunakan method SVC
4 iris = datasets.load_iris() # digunakan dengan menggunakan
   dataset iris
5 X, y = iris.data, iris.target #memasukkan x sebagai iris data ,
   dan y sebagai iris target
6 clf.fit(X, y) #laalu menggunakan metod fit .
```

1.7.2.5 Mencoba Conventions

```
1 import numpy as npy #mengimport numpy sebagai npy
2 from sklearn import random_projection #mengimport
   random_projection dari library sklearn
```

```

3
4 rng = npy.random.RandomState(0) #Menggunakan fungsi random dari
    numpy
5 X = rng.rand(10, 2000) #membuat range random diantara 10 sampai
    2000
6 X = npy.array(X, dtype='float32') #yang dijadikan array dengan
    tipe data float32
7 X.dtype

```

1.7.3 Penanganan Error

`ValueError: Found array with 0 sample(s) (shape=(0, 2)) while a minimum of 1 is required.`

Gambar 1.70 Data 2D

1.7.3.1 Screenshot Error

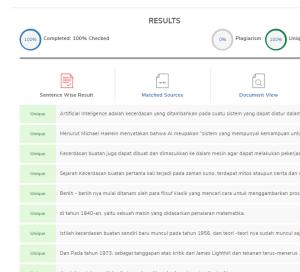
1.7.3.2 Kode error dan jenis error

Jenis errornya adalah value error

1.7.3.3 Solusi Error

Solusinya adalah dengan menggantikan nilainya adalah n_samples nya agar tidak 0

1.7.4 Cek Plagiarism



Gambar 1.71 Cek Plagiarism

BAB 2

CHAPTER 2

2.1 1174042 Faisal Najib Abdullah

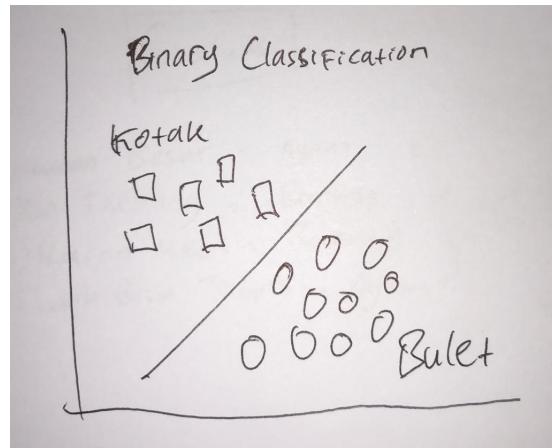
2.1.1 Teori

1. Jelaskan Apa Itu binari classification dilengkapi ilustrasi gambar sendiri.

Binary Classification atau biominal adalah tugas mengklasifikasikan unsur unsur dari himpunan yang diberikan kedalam kedua kelompok berdasarkan aturan klasifikasi yang telah ditetapkan. binari clasification juga dapat diartikan sebagai pembagi yang hanya memberikan dua pilihan contohnya benar dan salah atau klasifikasi tongkat panjang atau pendek. penjelasan lebih singkatnya binari classification merupakan kegiatan mengklasifikasikan yang hanya memberikan dua class.

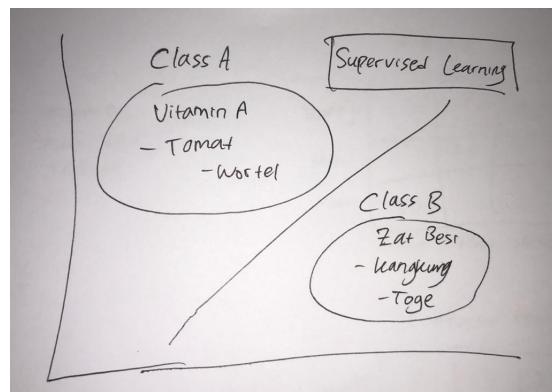
2. Jelaskan Apaitu supervised learning , unsupervised learning dan clustering dengan ilustrasi gambar sendiri.

supervised learning adalah cara untuk mengklasifikasikan suatu objek atau data yang telah di tentukan kelas kelasnya contoh pada sayuran tumbuhan wortel termasuk yang mengandung vitamin A berarti tum-



Gambar 2.1 contoh binari calssification

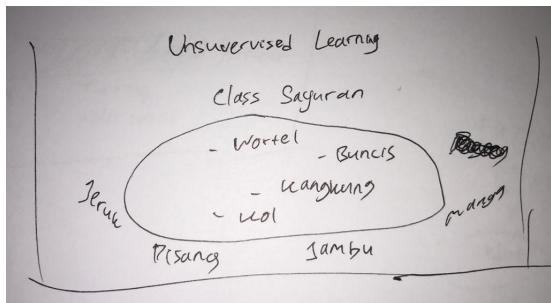
bahan wortel telah di kategorikan kedalam sayuran yang mengandung vitamin A. sedangkan kangkung mengandung zat besi yang berarti tumbuhan kangkung telah di kategorikan kedalam sayuran yang mengandung zat besi untuk lebih jelasnya dapat dilihat pada gambar.



Gambar 2.2 contoh supervised learning

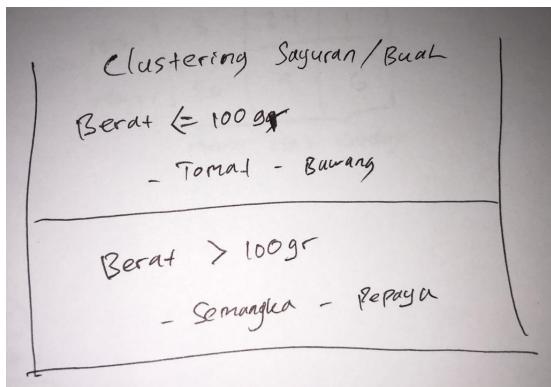
unsupervised learning merupakan cara untuk mengklasifikasi tanpa adanya kelas untuk menentukan jenisnya contoh sayuran berarti semua objek yang memiliki ciri ciri sayuran di kategorikan kedalam sayuran untuk lebih jelasnya dapat dilihat pada gambar.

clustering merupakan peroses mengklasifikasikan yang berdasarkan suatu parameter dalam penentuannya contoh pada berat sayuran sayuran A memiliki berat 100 gr dan sayuran B memiliki berat 120 gr yang berarti



Gambar 2.3 contoh unsupervised learning

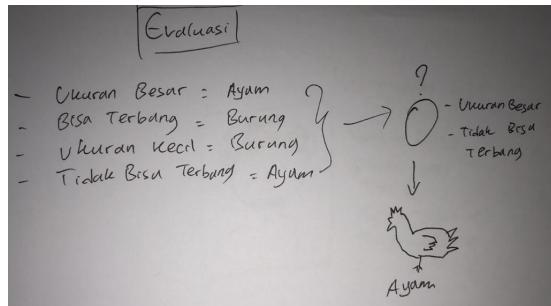
berat sayuran dibagi dua parameter yaitu lebih kecil samadengan 100 gram dan lebih besar dari gram contoh pada gambar.



Gambar 2.4 contoh clustering

3. Jelaskan apa itu evaluasi dan akurasi dan disertai ilustrasi contoh dengan gambar sendiri.

evaluasi adalah pengumpulan pengumpulan dan pengamatan dari berbagai macam bukti untuk mengukur dampak efektifitas dari suatu objek, program, atau proses berkaitan dengan spesifikasi atau persyaratan yang telah ditetapkan sebelumnya. sedangkan akurasi itu sendiri merupakan bagian dari evaluasi yang merupakan ketepatan data terhadap suatu objek berdasarkan kriteria tertentu. kita dapat mengevaluasi seberapa baik model bekerja dengan mengukur akurasinya. ketepatan akan definisikan sebagai persentase kasus yang di klasifikasikan dengan benar. hal ini berkaitan dengan confusion matrix pada materi selanjutnya. contoh evaluasi untuk membedakan burung dengan ayam terdapat parameter yaitu ukuran badan dan fungsi sayap pada hewan tersebut. lebih jelasnya pada gambar berikut:



Gambar 2.5 contoh evaluasi dan akurasi

4. Jelaskan bagaimana cara membuat Confusion Matrix, Buat confusion matrix sendiri.

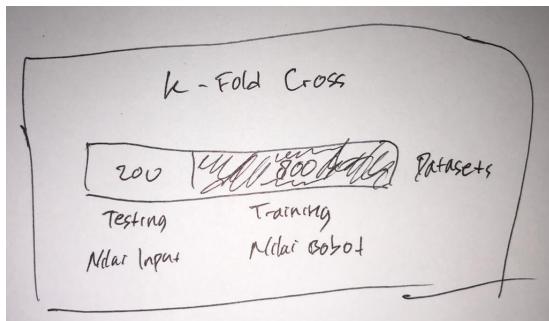
Dalam pembuatan confusion matrix tentukan parameter atau objek yang akan di evaluasi contoh bunga melati , bunga mawar, dan bunga kenangan buat tabel dengan baris dan kolom berjumlah tiga kemudian tentukan nilai miring pada setiap kolom tersebut disini saya memberi nilai 30 dengan ketentuan setiap baris harus berisi nilai 30 nilai tersebut jika terbagi ke kolom lain maka jumlahnya harus bernilai 30 jika tidak berarti data tersebut tidak akurat. untuk lebih jelanya dapat dilihat pada gambar berikut :

Confusion Matrix		
Kembang	Melati	Mawar
30	30	30
Melati	Kembang	Mawar
Mawar	Melati	Kembang

CONFUSION MATRIX		
Kembang	Melati	Mawar
5	2	23
5	24	1
20	4	6
Mawar	Melati	Kembang

Gambar 2.6 contoh Confusion Matrix

5. Jelaskan bagaimana K-fold cross validation bekerja dengan gambar ilustrasi contoh buatan sendiri. K-fold Cross Validation merupakan cara untuk melatih suatu mesin dimana di dalamnya terdapat data set yang dibagi menjadi dua yaitu untuk data testing dan data training contoh 1000 data merupakan data set dan 200 data digunakan untuk data testing kemudian 800 datanya digunakan untuk data training dimana data training tersebut digunakan untuk menentukan nilai bobot yang dimasukan kedalam rumus regresi linier. sedangkan nilai testing akan dijadikan nilai inputan untuk rumus regresi linier. contohnya dapat dilihat pada gambar berikut :



Gambar 2.7 contoh K-fold cross validation

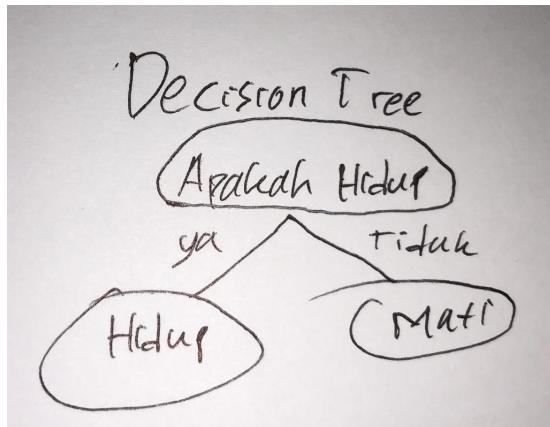
6. Jelaskan Apa itu decision tree dengan gambar ilustrasi contoh buatan sendiri.

Decision tree (pohon keputusan) merupakan implementasi dari binari clasification dimana pada pohon keputusan akan terdapat root atau akar dan cabang cabangnya yang nilainya seperti if contoh pada root berisi nilai jenis kelamin, apakah perempuan pada cabang satu bernilai iya dan pada cabang dua bernilai tidak jika nilainya iya berarti jenis kelamminya perempuan dan jika tidak maka bernilai laki-laki. agar lebih jelas dapat dilihat pada gambar decision tree berikut:

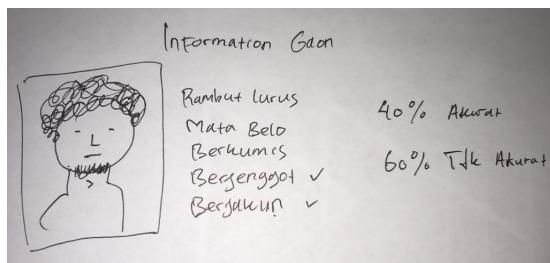
7. jelaskan apa itu information gain dan entropi dengan gambar ilustrasi buatan sendiri.

informasian gain merupakan informasi atau keriteria dalam pembagian sebuah objek contoh information gain pada laki-laki yaitu berrambut lurus, mata belo, berkumis, berjenggot, dan memiliki jakun. untuk lebih jelasnya dapat dilihat pada gambar berikut :

sedangkan entropi merupakan ukuran keacakan dari informasi semakin tinggi entropi maka semakin sulit dalam menentukan keputusan contoh dalam menentukan jenis kelamin semakin detail informasi maka akan semakin susah dalam menentukan keputusan.



Gambar 2.8 contoh decision tree



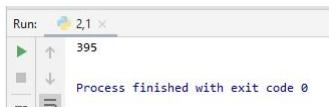
Gambar 2.9 contoh information gain

2.1.2 Sikic-Learn

1. pada surcode pertama yang dapat dilihat pada gambar pada baris pertama di tuliskan yang berarti mengimport library padas. selanjutnya pada baris ke dua codingan tersebut berisi pada code tersebut terdapat variabel muaraenim yang berisi inisialisasi padas dengan aksi untuk membaca vfile csv yang terdapat pada direktori pada komputer kemudian terdapat sep samadengan titik koma yang berarti pemisah field di dalam vile tersebut merupakan titik koma. kemudian pada bagian akhir terdapat code len (nama variabel) yang berarti akan menghotung jumlah baris pada file tersebut.

```

1 # load dataset (student mat pakenya)
2 import pandas as pd
3 muaraenim = pd.read_csv('D:/NAJIB/SEMESTER_6_NAJIB/AI/
Chapter2/dataset/student-mat.csv', sep=';')
4 print(len(muaraenim))
  
```



Gambar 2.10 hasil

2. Pada code selanjutnya akan ditambahkan fungsi untuk lulus atau gagal yang dibuat berupa kolom kolom yang di dalamnya terdapat nilai 0 dan 1 dimana 0 berarti gagal dan 1 berarti lulus. kemudian variabel pada codingan sebelumnya masih digunakan. dimana variabel muaraenim digunakan karena berisi file csv kemudian dilakukan ekseskuasi dengan parameter G1, G2, dan G3 dengan fungsi lambda yang berarti if bersarang atau if didalam if yang berarti nilai yang di eksekusi akan dilempar ke dalam setiap paramater sasuai dengan kriteria dan axis=1 yaitu nilai tersebut akan digunakan dari tiap baris data. sedangkan pada codingan selanjutnya variabel muaraenim di berikan aksi drop yaitu penurunan nilai pada bagian baris data. dan pada code muaraenim.head() yaitu untuk mengeksekusi codingan sebelumnya.

```

1 # generate binary label (pass/fail) based on G1+G2+G3
2 # (test grades, each 0–20 pts); threshold for passing is sum
   >=30
3 muaraenim[ 'pass' ] = muaraenim . apply (lambda row: 1 if (row[ 'G1'
   '] +row[ 'G2' ] +row[ 'G3' ]) >= 35 else 0, axis=1)
4 muaraenim = muaraenim . drop ([ 'G1' , 'G2' , 'G3' ], axis=1)
5 print (muaraenim . head ())

```

	school	sex	age	address	famsize	...	Dalc	Walc	health	absences	pass
0	GP	F	18	U	GT3	...	1	1	3	6	0
1	GP	F	17	U	GT3	...	1	1	3	4	0
2	GP	F	15	U	LE3	...	2	3	3	10	0
3	GP	F	15	U	GT3	...	1	1	5	2	1
4	GP	F	16	U	GT3	...	1	2	5	4	0

[5 rows x 31 columns]

Process finished with exit code 0

Gambar 2.11 hasil

3. pada kodingan selanjutnya diguanakan untuk menambahkan nilai numerik berupa 0 dan 1 yang dirubah dari nilai tidak dan iya hal ini merupakan fungsi dari codingan get_dummies pada baris pertama yang nilainya diambil dari variabel muaraenim yang telah di deklarasikan tadi banyaknya data numerik itu sendiri tergantung pada field dari kolom yang di cattumkan pada codingan dengan catatan field tersebut harus ada dalam data csv yang di import oleh codingan pertama tadi maka hasilnya akan merubah data dalam field tersebut menjadi 0 dan 1.

```

1 # use one-hot encoding on categorical columns
2 muaraenim = pd.get_dummies(muaraenim, columns=['sex', 'school'
    , 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', ,
    reason', 'guardian', 'schoolsups', 'famsup', 'paid', ,
    activities', 'nursery', 'higher', 'internet', 'romantic'])
3 muaraenim.head()

```

	age	Medu	Fedu	...	internet_yes	romantic_no	romantic_yes
0	18	4	4	...	0	1	0
1	17	1	1	...	1	1	0
2	15	1	1	...	1	1	0
3	15	4	2	...	1	0	1
4	16	3	3	...	0	1	0

[5 rows x 59 columns]

Process finished with exit code 0

Gambar 2.12 hasil

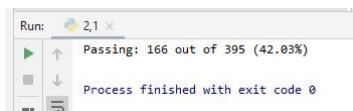
4. selanjutnya pada codingan selanjutnya yaitu menentukan data training dan data testing dari dataset dimana variabel muaraenim yang berisi data csv tadi dibuat perbandingan 500 untuk data training dan sisanya yaitu 149 digunakan untuk data testing hal ini di lakukan pada baris ke 1 sampai ke 3 pada gambar kemudian nilai tersebut di turunkan brdasarkan baris pada data set hal itu dapat dilihat pada baris ke 4 sampai ke 9 pada gambar kemudian selanjutnya mengimport library numpy yang berguna untuk oprasi vektor dan matrix karna data diatas berupa data matrix maka library ini di gunakan.

```

1 # shuffle rows
2 muaraenim = muaraenim.sample(frac=1)
3 # split training and testing data
4 muaraenim_train = muaraenim[:500]
5 muaraenim_test = muaraenim[500:]
6 muaraenim_train_att = muaraenim_train.drop(['pass'], axis=1)
7 muaraenim_train_pass = muaraenim_train['pass']
8 muaraenim_test_att = muaraenim_test.drop(['pass'], axis=1)
9 muaraenim_test_pass = muaraenim_test['pass']
10 muaraenim_att = muaraenim.drop(['pass'], axis=1)
11 muaraenim_pass = muaraenim['pass']
12 # number of passing students in whole dataset:
13 import numpy as np
14 print("Passing: %d out of %d (%.2f%%)" % (np.sum(
        muaraenim_pass), len(muaraenim_pass), 100*float(np.sum(
            muaraenim_pass)) / len(muaraenim_pass)))

```

5. selanjutnya yaitu membuat pohon keputusan pada baris pertama yairu memasukan librari tree kemudian pada baris kedua dibuat variabel palembang dengan nilai DecisionTreeClasifier yang merupakan paket atau fungsi dari scikit-learn yang merupakan class yang mampu melakukan multi

**Gambar 2.13** hasil

class. sedangkan max_depth=5 merupakan untuk penyesuaian data terhadap pohon keputusan itu sendiri.

```

1 # fit a decision tree
2 from sklearn import tree
3 palembang = tree.DecisionTreeClassifier(criterion="entropy",
   max_depth=5)
4 palembang = palembang.fit(muaraeni,_train_att,
   muaraeni_train_pass)
```

**Gambar 2.14** hasil

- selanjutnya yaitu pembuatan gambar dari pohon keputusan yang tadi dibuat pada codingan sebelumnya pada baris pertama codingan ya itu mengimport library graphviz kemudian pada baris ke dua yaitu pemberian nilai pada variabel baru dot data nilainya diambil dari pembuatan pohon keputusan tadi kemudian ditentukannya nilai benar dan salah dari codingan tersebut setelah itu dibuatlah sebuah variabel untuk menampung hasil eksekusi tersebut kemudian variabel tersebut di running.

```

1 # visualize tree
2 import graphviz
3 dot_data = tree.export_graphviz(palembang, out_file=None,
   label="all", impurity=False, proportion=True,
   feature_names=list(
      muaraenim_train_att), class_names=["fail", "pass"],
   filled=True, rounded=True)
5 graph = graphviz.Source(dot_data)
7 graph
```

- selanjutnya pembuatan method untuk menyimpan data pohon dengan menarik data langsung dari pohon keputusan.

```

1 # save tree
2 tree.export_graphviz(palembang, out_file="student-performance
   .dot", label="all", impurity=False, proportion=True,
   feature_names=list(muaraenim_train_att),
   class_names=["fail", "pass"], filled=True, rounded=True)
3
```

```

1  graph TD
2  node [shape=box, style="filled, rounded", color="black", fontname=helvetica];
3  edge [fontname=helvetica];
4  0 [label="failures <= 0.5\nsamples = 100.0%\nvalue = [0.58, 0.42]\nclass = fail", fillcolor="#f8dcc9"];
5  1 [label="schoolsуп_no <= 0.5\nsamples = 79.0%\nvalue = [0.519, 0.481]\nclass = fail", fillcolor="#fdf6f0"];
6  0 -> 1 [label=distance=2.5, labelangle=45, headlabel="True"];
7  2 [label="Mjob_services <= 0.5\nsamples = 10.1%\nvalue = [0.825, 0.175]\nclass = fail", fillcolor="#eb9c63"];
8  1 -> 2 ;
9  3 [label="reason_reputation <= 0.5\nsamples = 6.6%\nvalue = [0.923, 0.077]\nclass = fail", fillcolor="#e78c49"];
10 2 -> 3 ;
11  4 [label="samples = 4.3%\nvalue = [1.0, 0.0]\nclass = fail", fillcolor="#e58139"];
12  3 -> 4 ;
13  5 [label="paid_no <= 0.5\nsamples = 2.3%\nvalue = [0.778, 0.222]\nclass = fail", fillcolor="#eca572"];
14  3 -> 5 ;
15  6 [label="samples = 1.0%\nvalue = [0.5, 0.5]\nclass = fail", fillcolor="#ffffff"];
16  5 -> 6 ;
17  7 [label="samples = 1.3%\nvalue = [1.0, 0.0]\nclass = fail", fillcolor="#e58139"];
18  5 -> 7 ;
19  8 [label="health <= 2.5\nsamples = 3.5%\nvalue = [0.643, 0.357]\nclass = fail", fillcolor="#f3c7a7"];
20  2 -> 8 ;

```

Gambar 2.15 hasil

```

1  graph TD
2  node [shape=box, style="filled, rounded", color="black", fontname=helvetica];
3  edge [fontname=helvetica];
4  0 [label="failures <= 0.5\nsamples = 100.0%\nvalue = [0.58, 0.42]\nclass = fail", fillcolor="#f8dcc9"];
5  1 [label="schoolsуп_no <= 0.5\nsamples = 79.0%\nvalue = [0.519, 0.481]\nclass = fail", fillcolor="#fdf6f0"];
6  0 -> 1 [label=distance=2.5, labelangle=45, headlabel="True"];
7  2 [label="Mjob_services <= 0.5\nsamples = 10.1%\nvalue = [0.825, 0.175]\nclass = fail", fillcolor="#eb9c63"];
8  1 -> 2 ;
9  3 [label="reason_reputation <= 0.5\nsamples = 6.6%\nvalue = [0.923, 0.077]\nclass = fail", fillcolor="#e78c49"];
10 2 -> 3 ;
11  4 [label="samples = 4.3%\nvalue = [1.0, 0.0]\nclass = fail", fillcolor="#e58139"];
12  3 -> 4 ;
13  5 [label="paid_no <= 0.5\nsamples = 2.3%\nvalue = [0.778, 0.222]\nclass = fail", fillcolor="#eca572"];
14  3 -> 5 ;
15  6 [label="samples = 1.0%\nvalue = [0.5, 0.5]\nclass = fail", fillcolor="#ffffff"];
16  5 -> 6 ;
17  7 [label="samples = 1.3%\nvalue = [1.0, 0.0]\nclass = fail", fillcolor="#e58139"];
18  5 -> 7 ;
19  8 [label="health <= 2.5\nsamples = 3.5%\nvalue = [0.643, 0.357]\nclass = fail", fillcolor="#f3c7a7"];
20  2 -> 8 ;

```

Gambar 2.16 hasil

8. selanjutnya pada codingan berikut yaitu digunakan untuk mencetak nilai score rata-rata dari ketepatan data yang telah di olah.

```
1 palembang.score(muaraenim_test_att, muaraenim_test_pass)
```

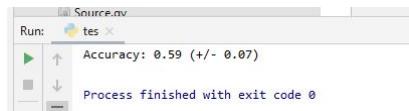
9. selanjutnya yaitu digunakan untuk memeriksa akurasi dari ketepatan hasil pengolahan data tersebut maka akan didapat nilai rata-rata 60 persen dari hasil pengolahan data tersebut, pada codingan tersebut pada baris ke satu melakukan import library dari sklern kemudian pada baris selanjutnya mengisi nilai skor dengan nilai pada variabel palembang setelah hal tersebut dilakukan kemudian data tersebut di eksekusi.

```
1 from sklearn.model_selection import cross_val_score
```

```

2 scores = cross_val_score(palembang, muaraenim_att,
    muaraenim_pass, cv=5)
3 # show average score and +/- two standard deviations away
4 # (covering 95% of scores)
5 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
    std() * 2))

```



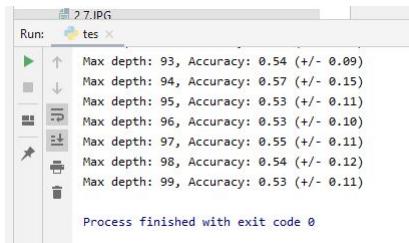
Gambar 2.17 hasil

10. membuat rank akurasi dari 1 sampai 100 untuk melihat akurasi data apakah datatersebut terdapat di rata rata 60 persen atau tidak dengan cara membuat lagi variabel baru dengan nilai tree diadalamnya jadi hampir mirip seperti membuat pohon keputusan namun ini langsung dalam bentuk rata rata akurasi yanlebih spesifik.

```

1 for max_depth in range(1, 100):
2     palembang = tree.DecisionTreeClassifier(criterion="entropy",
        max_depth=max_depth)
3     scores = cross_val_score(palembang, muaraenim_att,
        muaraenim_pass, cv=5)
4     print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (
            max_depth, scores.mean(), scores.std() * 2))

```



Gambar 2.18 hasil

11. untuk selanjutnya yaitu menentukan nilai untuk grafik hampirsama dengan nilai akurasi tadi pertama tentukan rank atau batasan nilai itu disini batasannya di mulai dari 1 sampai 20 dengan menggunakan nilai tree tadi maka hasilnya dapat ditentuka kemudian buat variabel i untuk penomoran tiap record yang keluar atau record hadil dari eksekusi tree tersebut.

```

1 depth_acc = np.empty((19,3), float)
2 i = 0
3 for max_depth in range(1, 20):
4     palembang = tree.DecisionTreeClassifier(criterion="entropy",
        max_depth=max_depth)

```

```

5     scores = cross_val_score(palembang , muaraenim_att ,
6     muaraenim_pass , cv=5)
7     depth_acc[i , 0] = max_depth
8     depth_acc[i , 1] = scores.mean()
9     depth_acc[i , 2] = scores.std() * 2
10    i += 1
11 print(depth_acc)

```

Gambar 2.19 hasil

12. terakhir yaitu pembuatan grafik untuk pembuatan grafik diambil data dari codingan sebelumnya yang 20 record tadi dengan cara mengimport libray matplotlib.pyplot yang di inisialisasi menjadi bakwankemudian inisialisasi tersebut di eksekusi.

```

1 import matplotlib.pyplot as plt
2 fig , ax = plt.subplots()
3 ax.errorbar(depth_acc[:,0] , depth_acc[:,1] , yerr=depth_acc
4 [:,2])
4 plt.show()

```

2.1.3 Penanganan Error

1. skrinsut error
2. kode error dan jenis errornya .

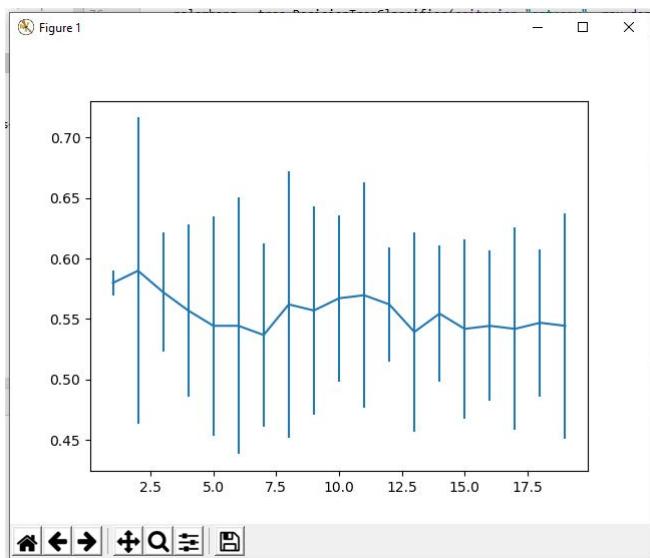
```

import graphviz
dot_data = tree.export_graphviz(palembang, out_file=None, label="all"
                                feature_names=list(muaraenim_train_a
                                filled=True, rounded=True)
graph = graphviz.Source(dot_data)
graph

```

pada codingan tersebut error karena graphiznya belu di istall

3. Solusi pemecahan masalah
buka CMD komputer anda run as administrator koemudian masukan perintah conda install graphviz.



Gambar 2.20 hasil

```
Run: tes
Traceback (most recent call last):
  File "D:/NAJIB/SEMESTER_6 NAJIB/AI/Chapter2/tes.py", line 2, in <module>
    import graphviz
ModuleNotFoundError: No module named 'graphviz'

Process finished with exit code 1
```

Gambar 2.21 Error

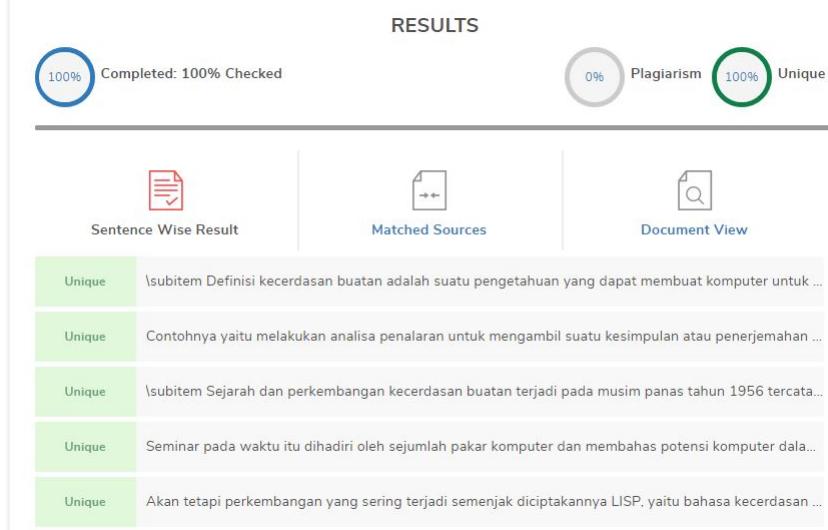
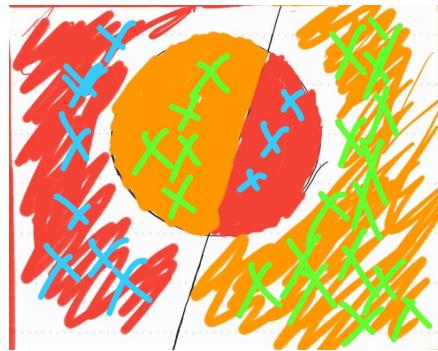
2.1.4 Plagiat

2.2 1174035 Luthfi Muhammad Nabil

2.2.1 Teori

1. Jelaskan Apa Itu Binary Classification dilengkapi ilustrasi gambar sendiri.

Binary Classification adalah sebuah metode untuk menklasifikasikan elemen yang dibentuk seperti grup untuk dibagi menjadi 2 grup. Dari 2 grup tersebut diprediksi setiap anggota pada grup yang mana sesuai dengan yang diatur pada aturan klasifikasi. Data atau konteks yang didapat membutuhkan keputusan dari item tersebut memiliki properti kualitatif, karakteristik yang spesifik, atau tipikal klasifikasi biner.

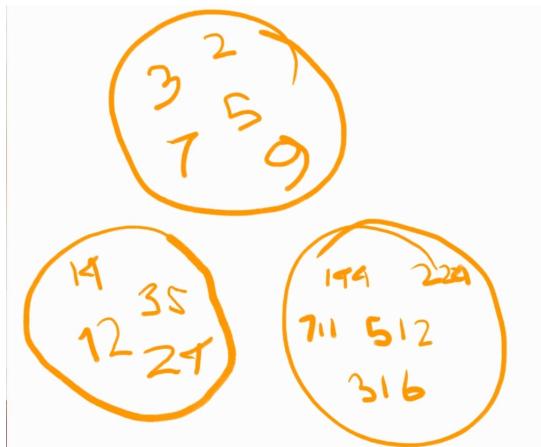
**Gambar 2.22** Error**Gambar 2.23** contoh binary classification

2. Jelaskan Apa itu supervised learning , unsupervised learning dan clustering dengan ilustrasi gambar sendiri.

Supervised learning adalah kondisi yang menggunakan variabel input dan output untuk dapat dilakukan pemetaan input output yang sudah didapat. Disebut Supervised Learning karena proses dari pembelajaran algoritma dari pembelajaran yang disumberkan dengan dataset dapat dipikirkan seperti seorang guru yang mengawasi proses pembelajaran. Proses pembelajaran dari algoritma akan berhenti saat algoritma sudah mendapatkan level dari performansi yang dapat diterima.

Unsupervised learning adalah kondisi dimana kamu hanya memiliki input data tanpa memiliki variabel output yang sesuai. Tujuan dari unsupervised learning adalah untuk memodelkan distribusi pada data untuk mengetahui lebih lanjut mengenai data. Disebut unsupervised learning karena pada metode ini, tidak ada jawaban yang tepat dan tidak ada pengarah. Sehingga algoritma akan ditinggalkan sesuai rancangan demi menemukan dan dapat mengolah data yang menarik pada saat yang akan datang.

Clustering adalah sebuah metode untuk membedakan data - data menjadi kumpulan dari group yang isinya merupakan data yang serupa setiap grupnya. Basisnya dapat berupa kesamaan atau perbedaan dari setiap grup tersebut.



Gambar 2.24 contoh clustering

3. Jelaskan apa itu evaluasi dan akurasi dan disertai ilustrasi contoh dengan gambar sendiri.

Evaluasi adalah kegiatan yang dilakukan untuk menentukan sebuah nilai yang dapat diambil dari suatu hal. Beberapa contoh evaluasi diantaranya menilai sebuah barang, bahaya dari penyakit, dan lain sebagainya dengan parameter yang digunakan yaitu mengetahui faktor - faktor yang menyebabkan atau akibat yang akan terjadi jika dibiarkan.

4. Jelaskan bagaimana cara membuat Confusion Matrix, Buat confusion matrix sendiri.

Confusion Matrix merupakan metode untuk menghitung akurasi pada data mining atau Sistem Pendukung Keputusan. Untuk menggunakan Confusion Matrix, ada 4 istilah sebagai hasil proses dari klasifikasi. Di antaranya adalah :

- True Positive : Data positif yang terdeteksi memiliki hasil benar
- False Positive : Data Positif yang terdeteksi memiliki hasil salah
- True Negative : Data negatif yang terdeteksi memiliki hasil benar
- False Negative : Data negatif yang terdeteksi memiliki hasil salah

5. Jelaskan bagaimana K-fold cross validation bekerja dengan gambar ilustrasi contoh buatan sendiri. k-Fold Cross-Validation merupakan prosedur untuk mengambil sampel ulang yang digunakan untuk mengevaluasi sebuah "machine learning" model pada sampel data yang terbatas. Procedure yang ada memiliki satu parameter yang dipanggil k yang mengacu pada jumlah grup yang memberikan data sampel untuk dipisahkan. K-fold Cross-validation akan melakukan hal berikut :
- (a) Mengacak dataset secara random
 - (b) Memisahkan dataset menjadi k group
 - (c) Untuk setiap grup, akan disesuaikan dan dievaluasi
 - (d) Merekaphasil dari evaluasi dan penyesuaian menggunakan sampel dari model evaluasi



Gambar 2.25 contoh K-fold cross validation

6. Jelaskan Apa itu decision tree disertakan gambar ilustrasi contoh buatan sendiri.

Decision Tree merupakan sebuah struktur yang menentukan keputusan dan setiap konsekuensinya. Hasil dari setiap struktur biasanya menggunakan jawaban (True dan False) atau cabang lain yang akan menjadi pohon selanjutnya. Setiap keputusan diantaranya akan membandingkan

kondisi yang diberikan kepada struktur untuk dibandingkan kondisi apa saja yang sudah didapat pada sistem tersebut.

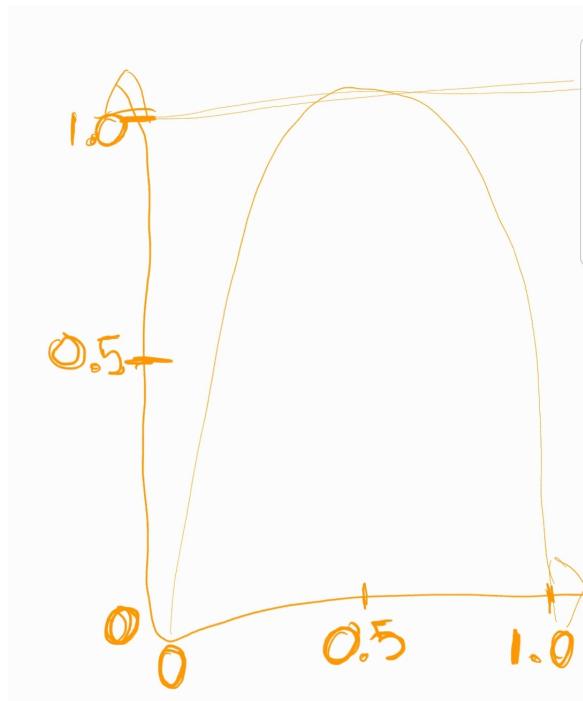
7. jelaskan apa itu information gain dan entropi dengan gambar ilustrasi buatan sendiri.

Information Gain merupakan total data yang didapat dari data - data acak yang data tersebut akan digunakan untuk analisis data lainnya. Information Gain ini digunakan pada decision tree sebagai label setiap aksi - aksi yang perlu dinilai validasinya.

		Coding	
		Ya	Tidak
Outlook	Fokus	3	2
	Normal	4	0
	Tidak Fokus	2	3

Gambar 2.26 contoh information gain

Entropi merupakan pengukuran sebuah data dan validnya data tersebut untuk dapat digunakan sebagai informasi yang akan dimasukkan ke Information Gain. Entropi menilai sebuah obyek berdasarkan kebutuhan di dunia nyata dan pengaruh pada sistem yang akan digunakan



Gambar 2.27 contoh information gain

2.2.2 Scikit-Learn

1.

```
1 import pandas as pd #mengimpor library pandas
2 cakue = pd.read_csv('student-mat.csv', sep=';') #data csv
    dibaca lalu dipisahkan dengan titik koma';
3 len(cakue) #menghitung total nilai(panjang kumpulan nilai/
    array) yang terpisahkan dari csv tersebut
```

In [5]: student-mat.csv
Out[5]: 354

Gambar 2.28 Hasil Percobaan 1

2.

```
1 cakue[ 'pass' ] = cakue .apply(lambda row: 1 if (row[ 'G1' ]+row[ 'G2' ]+row[ 'G3' ]) >= 35 else 0, axis=1)
```

```

2 cakue = cakue.drop(['G1', 'G2', 'G3'], axis=1)
3 cakue.head()#mengambil baris pertama dari cakue

```

In [8]:	cakue[cols] + cakue.apply(lambda row: 1 if (row['G1'] > row['G2']) & (row['G1'] > row['G3']) else 0, axis=1)
Out[8]:	school sex age address family Debt male results absences pass
0	GP G 15 U 0 0 1 1 1 0
1	GP G 15 U 0 0 1 1 1 0
2	GP G 15 U 0 0 1 1 1 0
3	GP G 15 U 0 0 1 1 1 0
4	GP G 15 U 0 0 1 1 1 0
5	GP G 15 U 0 0 1 1 1 0
6	GP G 15 U 0 0 1 1 1 0
7	GP G 15 U 0 0 1 1 1 0
8	GP G 15 U 0 0 1 1 1 0
9	GP G 15 U 0 0 1 1 1 0
10	GP G 15 U 0 0 1 1 1 0
11	GP G 15 U 0 0 1 1 1 0
12	GP G 15 U 0 0 1 1 1 0
13	GP G 15 U 0 0 1 1 1 0
14	GP G 15 U 0 0 1 1 1 0
15	GP G 15 U 0 0 1 1 1 0

Gambar 2.29 Hasil Percobaan 2

3.

```

1 cakue = pd.get_dummies(cakue, columns=['sex', 'school',
   'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason',
   'guardian', 'schoolsup', 'famsup', 'paid', 'activities',
   'nursery', 'higher', 'internet', 'romantic']) #
   Mongkonversi kategori variabel menjadi variabel indikator
2 cakue.head()#mengambil baris pertama dari cakue

```

In [9]:	cakue[cols] + cakue.apply(lambda row: 1 if (row['G1'] > row['G2']) & (row['G1'] > row['G3']) else 0, axis=1)
Out[9]:	school sex age address family Debt male results absences pass
0	GP G 15 U 0 0 1 1 1 0
1	GP G 15 U 0 0 1 1 1 0
2	GP G 15 U 0 0 1 1 1 0
3	GP G 15 U 0 0 1 1 1 0
4	GP G 15 U 0 0 1 1 1 0
5	GP G 15 U 0 0 1 1 1 0
6	GP G 15 U 0 0 1 1 1 0
7	GP G 15 U 0 0 1 1 1 0
8	GP G 15 U 0 0 1 1 1 0
9	GP G 15 U 0 0 1 1 1 0
10	GP G 15 U 0 0 1 1 1 0
11	GP G 15 U 0 0 1 1 1 0
12	GP G 15 U 0 0 1 1 1 0
13	GP G 15 U 0 0 1 1 1 0
14	GP G 15 U 0 0 1 1 1 0
15	GP G 15 U 0 0 1 1 1 0

Gambar 2.30 Hasil Percobaan 3

4.

```

1 # shuffle rows
2 cakue = cakue.sample(frac=1) #mengenerate sampel acak dari
   baris atau kolom dari cakue
3 # split training and testing data
4 cakue_train = cakue[:500]#Mengambil data array dengan batas
   index 500
5 cakue_test = cakue[500:] #Mengambil data array dengan awal
   index 500
6 cakue_train_att = cakue_train.drop(['pass'], axis=1) #
7 cakue_train_pass = cakue_train['pass'] #Mengambil data
   cakue_train dengan index object 'pass' dan dimasukkan ke
   variable cakue_train_pass
8 cakue_test_att = cakue_test.drop(['pass'], axis=1)
9 cakue_test_pass = cakue_test['pass'] #Mengambil data
   cakue_train dengan index object 'pass' dan dimasukkan ke
   variable cakue_test_pass
10 cakue_att = cakue.drop(['pass'], axis=1)
11 cakue_pass = cakue['pass']#Mengambil data cakue_train dengan
   index object 'pass' dan dimasukkan ke variable cakue_pass
12 # number of passing students in whole dataset:
13 import numpy as np #mengimport library numpy
14 print("Passing: %d out of %d (%.2f%%)" % (np.sum(cakue_pass),
   len(cakue_pass), 100*float(np.sum(cakue_pass)) / len(
   cakue_pass))) #Menampilkan hasil integer dan float untuk
   melihat passing datanya

```

```
In [20]: from sklearn import tree #Mengimport class tree dari library sklearn
In [21]: kwetiau = tree.DecisionTreeClassifier(criterion="entropy",
                                             max_depth=5) #Memanggil fungsi untuk melakukan prediksi dengan aturan keputusan yang simpel
In [22]: kwetiau = kwetiau.fit(cakue_train_att, cakue_train_pass) # Untuk melakukan pengiriman data training set ke method fit
```

Gambar 2.31 Hasil Percobaan 4

5.

```
1 from sklearn import tree #Mengimport class tree dari library sklearn
2 kwetiau = tree.DecisionTreeClassifier(criterion="entropy",
3                                     max_depth=5) #Memanggil fungsi untuk melakukan prediksi dengan aturan keputusan yang simpel
4 kwetiau = kwetiau.fit(cakue_train_att, cakue_train_pass) # Untuk melakukan pengiriman data training set ke method fit
```

```
In [20]: dot_data = tree.export_graphviz(kwetiau, out_file=None, label="all", impurity=False, proportion=True, feature_names=list(cakue_train_att), class_names=["fail", "pass"], filled=True, rounded=True) #Untuk mengekspor data ke format graphviz
In [21]: graph = graphviz.Source(dot_data) #Untuk mengambil sumber data berformat graphviz dan dimasukkan ke variable graph
```

Gambar 2.32 Hasil Percobaan 5

6.

```
1 import graphviz #mengimport class graphviz
2 dot_data = tree.export_graphviz(kwetiau, out_file=None, label="all", impurity=False, proportion=True, feature_names=list(cakue_train_att), class_names=["fail", "pass"], filled=True, rounded=True) #Untuk mengekspor data ke format graphviz
3 graph = graphviz.Source(dot_data) #Untuk mengambil sumber data berformat graphviz dan dimasukkan ke variable graph
4 graph #Menampilkan isi variable graph untuk di spyder
```

```
In [20]: dot_data = tree.export_graphviz(kwetiau, out_file=None, label="all", impurity=False, proportion=True, feature_names=list(cakue_train_att), class_names=["fail", "pass"], filled=True, rounded=True) #Untuk mengekspor data ke format graphviz
In [21]: graph = graphviz.Source(dot_data) #Untuk mengambil sumber data berformat graphviz dan dimasukkan ke variable graph
In [22]: graph #Menampilkan isi variable graph untuk di spyder
```

Gambar 2.33 Hasil Percobaan 6

7.

```
1 tree.export_graphviz(kwetiau, out_file="student-performance.dot", label="all", impurity=False, proportion=True, feature_names=list(cakue_train_att), class_names=["fail", "pass"], filled=True, rounded=True) #Untuk mengekspor data ke format graphviz
```

Gambar 2.34 Hasil Percobaan 7

8.

```
1 kwetiau.score(cakue_test_att, cakue_test_pass) #Untuk
   melakukan penilaian sesuai dengan drop – drop pada
   variable cakue_test_att dan cakue_test_pass
```

```
In [12]: kwetiau.score(cakue_test_att, cakue_test_pass)
Out[12]: 0.67
In [13]: #Untuk mengevaluasi nilai menggunakan metode cross-validation
In [14]: scores = cross_val_score(kwetiau, cakue_att, cakue_pass, cv=5)
In [15]: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
Out[15]: Accuracy: 0.67 (+/- 0.21)
```

Gambar 2.35 Hasil Percobaan 8

9.

```
1 from sklearn.model_selection import cross_val_score # Mengambil class cross_val_score dari library sklearn.model_selection
2 scores = cross_val_score(kwetiau, cakue_att, cakue_pass, cv=5) #Untuk mengevaluasi nilai menggunakan metode cross-validation
3 # show average score and +/- two standard deviations away
4 #(covering 95% of scores)
5 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2)) #Menampilkan score
```

```
In [15]: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
Out[15]: Accuracy: 0.67 (+/- 0.21)
```

Gambar 2.36 Hasil Percobaan 9

10.

```
1 for max_depth in range(1, 20): #Looping dengan for berdasarkan nilai dari 1 sampai 20 dengan variable index yaitu max_depth
2     kwetiau = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth) #Memanggil fungsi untuk melakukan prediksi dengan aturan keputusan yang simpel
3     scores = cross_val_score(kwetiau, cakue_att, cakue_pass, cv=5) #Untuk mengevaluasi nilai menggunakan metode cross-validation
4     print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (max_depth, scores.mean(), scores.std() * 2)) #Menampilkan score
```

**Gambar 2.37** Hasil Percobaan 10

11.

```

1 depth_acc = np.empty((19,3), float) #Mengembalikan array
    dengan format bentuk dan tipe
2 i = 0#Inisiasi variable
3 for max_depth in range(1, 20): #Looping dengan for
    berdasarkan nilai dari 1 sampai 20 dengan variable index
    yaitu max_depth
4     kwetiau = tree.DecisionTreeClassifier(criterion="entropy",
        , max_depth=max_depth) #Memanggil fungsi untuk melakukan
        prediksi dengan aturan keputusan yang simpel
5     scores = cross_val_score(kwetiau, cakue_att, cakue_pass,
        cv=5) #Untuk mengevaluasi nilai menggunakan metode cross-
        validation
6     depth_acc[i,0] = max_depth
7     depth_acc[i,1] = scores.mean()
8     depth_acc[i,2] = scores.std() * 2
9     i += 1
10    depth_acc

```

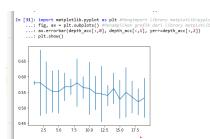
**Gambar 2.38** Hasil Percobaan 11

12.

```

1 import matplotlib.pyplot as plt #Mengimport library
    matplotlib>pyplot
2 fig, ax = plt.subplots() #Menampilkan grafik dari library
    matplotlib
3 ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc
    [:,2])
4 plt.show()

```



Gambar 2.39 Hasil Percobaan 12

2.2.3 Skrinsut Error

Error yang didapat yaitu mengalami error tidak ada library graphviz yang terdeteksi



Gambar 2.40 Error

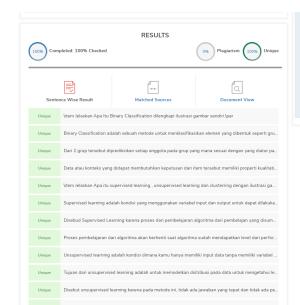
2.2.4 Penanganan Error

Solusinya adalah dengan menginstall library graphviz yang ada



Gambar 2.41 Error

2.2.5 Plagiarisme



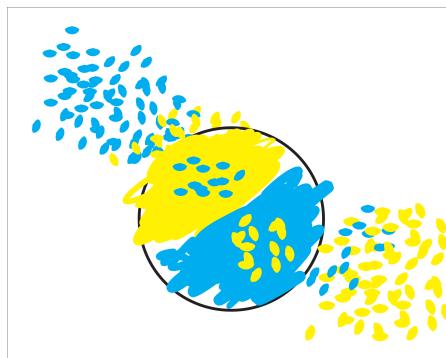
Gambar 2.42 Hasil Pengecekan Plagiarisme

2.3 1174057 - Alit Fajar Kurniawan

2.3.1 Teori

2.3.1.1 Jelaskan apa itu binary classification dilengkapi ilustrasi gambar sendiri

Binary classification merupakan jenis masalah pembelajaran mesin yang paling sederhana. mengkalirkasikan elemen-elemen dari himpunan yang diberikan kedalam dua kelompok berdasarkan aturan klarifikasi. Contoh yang paling sederhana dalam binary classification yaitu mendekripsi dan mendiagnosa. Berikut contoh gambar Binary Classification, gambar 2.1



Gambar 2.43 Klasifikasi Binari

2.3.1.2 Jelaskan apa itu supervised learning dan unsupervised learning dan clustering dengan ilustrasi gambar sendiri

Supervised Learning merupakan paradigma belajar yang berkaitan dengan studi tentang bagaimana komputer dan sistem alami seperti manusia belajar [?]. Tujuannya untuk menyimpulkan pemetaan fungsional berdasarkan serangkaian pelatihan atau mengelompokkan suatu data ke data yang sudah ada. Berikut contoh gambar Supervised Learning

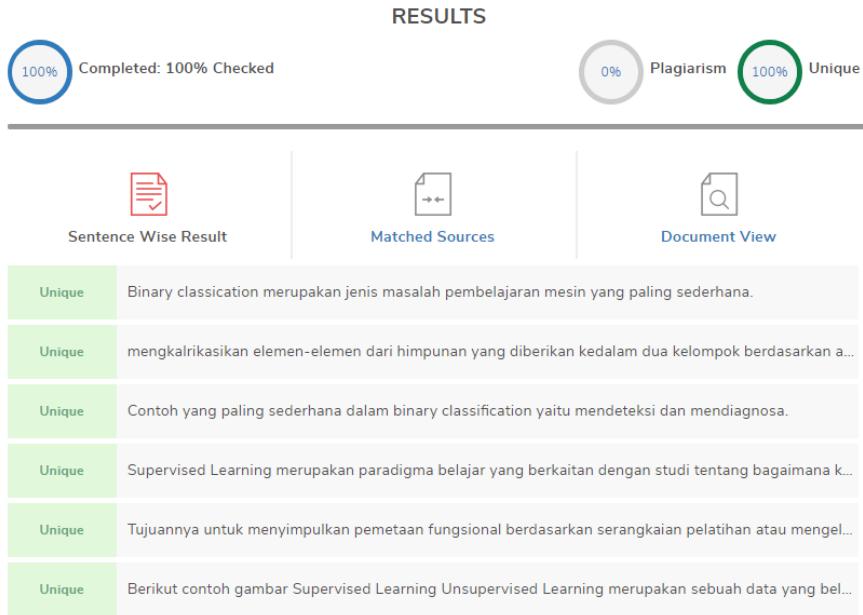
Unsupervised Learning merupakan sebuah data yang belum ditentukan variabelnya jadi hanya berupa data saja. Dalam sebuah kasus Unsupervised Learning adalah aggap saja anda belum pernah membeli buku sama sekali dan pada suatu hari anda telah membeli buku dengan sangat banyak dalam kategori yang berbeda. Sehingga buku tersebut belum di kategorikan dan hanya berupa data buku saja.

Clustering merupakan metode pengelompokan data, clustering juga suatu proses partisi satu set objek data ke dalam himpunan bagian yang disebut dengan cluster. Objek yang pada cluster memiliki kesamaan secara karakteristik antara satu sama lainnya dan berbeda dengan cluster yang lain.

2.3.2 Praktek

2.3.3 Penanganan Error

2.3.4 Bukti Tidak Plagiat



Gambar 2.44 Plagiarisme

2.4 1174050 Dika Sukma Pradana

2.4.1 Teori

1. Jelaskan Apa Itu binari calssification dilengkapi ilustrasi gambar sendiri.

Klasifikasi biner atau binomial adalah tugas untuk mengklasifikasikan elemen-elemen dari himpunan tertentu ke dalam dua kelompok (memprediksi kelompok mana yang masing-masing dimiliki) berdasarkan aturan klasifikasi.



Gambar 2.45 contoh binari calssification

2. Jelaskan Apaitu supervised learning , unsupervised learning dan clustering dengan ilustrasi gambar sendiri.

Supervised learning adalah sebuah metode pendekatan yang mana sudah terdapat data yang dilatih, dan terdapat variabel yang ditargetkan atau yang menjadi tujuan sehingga tujuan dari pendekatan ini adalah mengelompokan suatu data ke data yang sudah ada. contoh pada nasi termasuk yang mengandung karbohidrat berarti nasi telah di kategorikan kedalam karbohidrat. sedangkan ayam mengandung protein yang berarti ayam telah di kategorikan yang mengandung protein untuk lebih jelasnya dapat dilihat pada gambar.



Gambar 2.46 contoh supervised learning

Sedangkan unsupervised learning tidak memiliki data latih, sehingga dari data yang ada, kita mengelompokan data tersebut menjadi dua bagian atau bahkan tiga bagian dan seterusnya. contoh ayam di kategorikan kedalam karbohidrat untuk lebih jelasnya dapat dilihat pada gambar.



Gambar 2.47 contoh unsupervised learning

clustering merupakan proses mengklasifikasikan yang berdasarkan suatu parameter dalam penentuannya contoh pada berat protein memiliki berat 1000 gr dan karbohidrat memiliki berat 1200 gr yang berarti berat dibagi dua parameter yaitu lebih kecil samadengan 1000 gram dan lebih besar dari 1000 gram contoh pada gambar.



Gambar 2.48 contoh clustering

3. Jelaskan apa itu evaluasi dan akurasi dan disertai ilustrasi contoh dengan gambar sendiri.

evaluasi adalah pengumpulan pengumpulan dan pengamatan dari berbagai macam bukti untuk mengukur dampak efektifitas dari suatu objek, program, atau proses berkaitan dengan spesifikasi atau persyaratan yang telah di tetapkan sebelumnya. sedangkan akurasi itu sendiri merupakan bagian dari evaluasi yang merupakan ketepatan data terhadap suatu objek berdasarkan keriteria tertentu. kita dapat mengevaluasi seberapa baik model bekerja dengan mengukur akurasiinya. ketepatan akan definisikan sebagai persentase kasus yang diklasifikasikan dengan benar. hal ini berkaitan dengan confusion matrix pada materi selanjutnya. contoh evaluasi untuk membedakan angsa dengan entok terdapat parameter yaitu panjang leher dan fungsi sifat dari hewan tersebut. lebih jelasnya pada gambar berikut:



Gambar 2.49 contoh evaluasi dan akurasi

4. Jelaskan bagaimana cara membuat Confusion Matrix, Buat confusion matrix sendiri.

Dalam pembuatan confusion matrix tentukan parameter atau objek yang akan di evaluasi contoh angsa, entok, bebek buat tabel dengan baris dan kolom berjumlah tiga kemudian tentukan nilai miring pada setiap kolom tersebut disini saya memberi nilai 15 dengan ketentuan setiap baris harus berisi nilai 15 nilai tersebut jika terbagi ke kolom lain maka jumlahnya harus bernilai 15 jika tidak berarti data tersebut tidak akurat. untuk lebih jelasnya dapat dilihat pada gambar berikut :

		Predicted	
		Angsa	Entok
Actual	Angsa	15	5
	Entok	2	15

Gambar 2.50 contoh Confusion Matrix

5. Jelaskan bagaimana K-fold cross validation bekerja dengan gambar ilustrasi contoh buatan sendiri. K-fold Cross Validation merupakan cara untuk melatih suatu mesin dimana dalamnya terdapat data set yang dibagi menjadi dua yaitu untuk data testing dan data training contoh 1000 data merupakan data set dan 300 data digunakan untuk data testing kemudian 700 datanya digunakan untuk data training dimana data training tersebut digunakan untuk menentukan nilai bobot yang dimasukkan kedalam rumus regresi linier. sedangkan nilai testing akan dijadikan nili

inputan untuk rumus regresi linier. contohnya dapat dilihat pada gambar berikut :



Gambar 2.51 contoh K-fold cross validation

6. Jelaskan Apa itu decision tree dengan gambar ilustrasi contoh buatan sendiri.

Decision tree merupakan implementasi dari binari clasification dimana pada pohon keputusan akan terdapat root atau akar dan cabang cabangnya yang nilainya seperti if contoh pada root berisi nilai warna mawar, apakah merah pada cabang satu bernilai iya dan pada cabang dua bernilai tidak jika nilainya iya berarti mawar dan jika tidak maka bukan mawar. agar lebih jelas dapat dilihat pada gambar decision tree berikut:



Gambar 2.52 contoh decision tree

7. jelaskan apa itu information gain dan entropi dengan gambar ilustrasi buatan sendiri.

informasian gain merupakan informasi atau keriteria dalam pembagian sebuah objek contoh information gain pada llama leher panjang, ekor panjang, berkaki 4, buas, karnivora. untuk lebih jelasnya dapat dilihat pada gambar berikut :



Gambar 2.53 contoh information gain

sedangkan entropi merupakan ukuran keacakan dari informasi semakin tinggi entropi maka semakin sulit dalam menentukan keputusan contoh dalam menentukan satu gen semakin detail informasi maka akan semakin susah dalam menentukan keputusan.

2.4.2 Sikic-Learn

1. pada surcode pertama yang dapat dilihat pada gambar pada baris pertama di tuliskan import pandas as baso yang berarti mengimport library pandas. selanjutnya pada baris ke dua codingan tersebut berisi code tersebut terdapat variabel pecel yang berisi inisialisasi pandas dengan aksi untuk membaca vfile csv yang terdapat pada direktori pada komputer kemudian terdapat sep samadengan titik koma yang berarti pemisah field di dalam file tersebut merupakan titik koma. kemudian pada bagian akhir terdapat code len (nama variabel) yang berarti akan menghitung jumlah baris pada file tersebut. untuk hasilnya dapat dilihat pada gambar dibawah.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 21:32:13 2020
4
5 @author: User
6 """
7
8 # load dataset (student mat pakenya)
9 import pandas as pd
10 pecel = pd.read_csv('D:\\SEMESTER 6\\Python-Artificial-
    Intelligence-Projects-for-Beginners-master\\Chapter01\\
        dataset\\student-mat.csv', sep=';')
11 print(len(pecel))

```

In [1]: runfile('D:/SEMESTER 6/wert/untitled0.py', wdir='D:/SEMESTER 6/wert')
395

Gambar 2.54 hasil

2. Pada code selanjutnya akan ditambahkan fungsi untuk lulus atau gagal yang dibuat berupa kolom kolom yang di dalamnya terdapat nilai 0 dan 1 dimana 0 berarti gagal dan 1 berarti lulus. kemudian variabel pada codingan sebelumnya masih digunakan. dimana variabel pecel digunakan karena berisi nilai file csv kemudian dilakukan ekseskuasi dengan parameter G1, G2, dan G3 dengan fungsi lambda yang berarti if bersarang atau if didalam if yang berarti nilai yang di eksekusi akan dilempar ke dalam setiap paramater sasuai dengan kriteria dan axis=1 yaitu nilai tersebut akan digunakan dari tiap baris data. sedangkan pada codingan variabel pecel di berikan aksi drop yaitu penurunan nilai pada bagian baris data. dan pada code pecel.head yaitu untuk mengeksekusi codingan sebelumnya untuk lebih jelasnya dapat dilihat pada gambar dan hasilnya seperti pada gambar berikut :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 21:36:23 2020
4

```

```

5 @author: User
6 """
7
8 # generate binary label (pass/fail) based on G1+G2+G3
9 # (test grades, each 0–20 pts); threshold for passing is sum
10 # >=30
11 pecel['pass'] = pecelapply(lambda row: 1 if (row['G1']+row['
12 G2']+row['G3']) >= 35 else 0, axis=1)
13 pecel = pecel.drop(['G1', 'G2', 'G3'], axis=1)
14 print(pecel.head())

```

```

In [4]: runfile('D:/SEMESTER 6/wert/2.py', wdir='D:/SEMESTER 6/wert')
   school sex age address famsize ... Dalc Walc health absences pass
0      GP   F  18      U    GT3 ...   1     1     3      6     0
1      GP   F  17      U    GT3 ...   1     1     3      4     0
2      GP   F  15      U    LE3 ...   2     3     3     10     0
3      GP   F  15      U    GT3 ...   1     1     5      2     1
4      GP   F  16      U    GT3 ...   1     2     5      4     0
[5 rows x 31 columns]

```

Gambar 2.55 hasil

3. pada kodingan selanjutnya diguanakan untuk menambahkan nilai numerik berupa 0 dan 1 yang dirubah dari nilai tidak dan iya hal ini merupakan fungsi dari codingan get_dummies pada baris pertama pada gambar yang nilainya diambil dari variabel pecel yang telah di dekralasikan tadi banyaknya data numerik itu sendiri tergantung pada field dari kolom yang di camtumkan pada codingan dengan catatan field tersebut harus ada dalam data csv yang di import oleh codingan pertama tadi maka hasilnya akan merubah data dalam field tersebut menjadi 0 dan 1 untuk lebih jelasnya dapat di lihat pada gambar berikut;

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 21:41:22 2020
4
5 @author: User
6 """
7
8 # use one-hot encoding on categorical columns
9 pecel = pd.get_dummies(pecel, columns=['sex', 'school',
10 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason',
11 'guardian', 'schoolsup', 'famsup', 'paid', 'activities',
12 'nursery', 'higher', 'internet', 'romantic'])
13 pecel.head()

```

```

In [6]: runfile('D:/SEMESTER 6/wert/4.py', wdir='D:/SEMESTER 6/wert')
Passing: 166 out of 395 (42.03%)

```

Gambar 2.56 hasil

4. selanjutnya pada codingan selanjutnya yaitu menentukan data training dan data testing dari dataset dimana variabel pecel yang berisi data csv

tadi dibuat perbandingan 500 untuk data training dan sisanya yaitu 149 digunakan untuk data testing hal ini di lakukan pada baris ke 1 sampai ke 3 pada gambar kemudian nilai tersebut di turunkan brdasarkan baris pada data set hal itu dapat dilihat pada baris ke 4 sampai ke 9 pada gambar kemudian selanjutnya mengimport library numpy yang berguna untuk oprasi vektor dan matrix karna data diatas berupa data matrix maka library ini di gunakan. untuk hasilnya dapat dilihat pada gambar.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 21:42:26 2020
4
5 @author: User
6 """
7
8 # shuffle rows
9 pecel = pecel.sample(frac=1)
10 # split training and testing data
11 pecel_train = pecel[:500]
12 pecel_test = pecel[500:]
13 pecel_train_att = pecel_train.drop(['pass'], axis=1)
14 pecel_train_pass = pecel_train['pass']
15 pecel_test_att = pecel_test.drop(['pass'], axis=1)
16 pecel_test_pass = pecel_test['pass']
17 pecel_att = pecel.drop(['pass'], axis=1)
18 pecel_pass = pecel['pass']
19 # number of passing students in whole dataset:
20 import numpy as np
21 print("Passing: %d out of %d (%.2f%%)" % (np.sum(pecel_pass),
   len(pecel_pass), 100*float(np.sum(pecel_pass)) / len(
   pecel_pass)))

```

	school	sex	age	address	famsize	...	Dalc	Walc	health	absences	pass
0	GP	F	18	U	GT3	...	1	1	3	6	0
1	GP	F	17	U	GT3	...	1	1	3	4	0
2	GP	F	15	U	LE3	...	2	3	3	10	0
3	GP	F	15	U	GT3	...	1	1	5	2	1
4	GP	F	16	U	GT3	...	1	2	5	4	0

Gambar 2.57 hasil

- selanjutnya yaitu membuat pohon keputusan dapat lebih jelasnya dapat di lihat pada gambar. pada baris pertama yairu memasukan librari tree kemudian pada baris kedua dibuat variabel lontong dengan nilai DecisionTreeClasifier yang merupakan paket atau fungsi dari scikit-learn yang merupakan class yang mampu melakukan multi class. sedangkan max_depth=5 merupakan untuk penyesuaian data terhadap pohon keputusan itu sndiri. untuk hasilnya dapat dilihat pada gambar

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 21:56:41 2020
4

```

```

5 @author: User
6 """
7
8 # fit a decision tree
9 from sklearn import tree
10 lontong = tree.DecisionTreeClassifier(criterion="entropy",
11                                     max_depth=5)
12 lontong = lontong.fit(pecel,_train_att, pecel_train_pass)

```

In [18]: runfile('D:/SEMESTER 6/wert/4.py', wdir='D:/SEMESTER 6/wert')
Passing: 166 out of 395 (42.03%)

Gambar 2.58 hasil

6. selanjutnya yaitu pembuatan gambar dari pohon keputusan yang tadi di buta pada codingan sebelumnya pada baris pertama codingan ya itu mengimport library graphviz kemudian pada baris ke dua yaitu pem-berian nilai pada variabel baru dot data nilainya diambil dari pembuatan puhon keputusan tadi kemudian di tentukannya nilai benar dan salah dari codingan tersebut setelah itu dibuatlah sebuah variabel untuk menam-pung hasil eksekusi tersebut kemudian variabel tersebut di running untuk lebih jelasnya dapat di lihat pada gambar.kemudian hasilnya dapat dili-hat pada gambar.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 22:00:57 2020
4
5 @author: User
6 """
7
8 # visualize tree
9 import graphviz
10 dot_data = tree.export_graphviz(lontong, out_file=None, label
11                                 ="all", impurity=False, proportion=True,
12                                 feature_names=list(
13                                   pecel_train_att), class_names=["fail", "pass"],
14                                 filled=True, rounded=True)
15 graph = graphviz.Source(dot_data)
16 graph

```

7. selanjutnya pembuatasn method untuk menyimpan data pohon dengan menarik data langsung dari pohon keputusan di buat tadi untuk code lebih jelasnya dapat dilihat pada gambar. kemudian intuk hasilnya dapat dilihat pada gambar berikut.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 22:03:32 2020
4
5 @author: User
6 """

```

```

graph TD
    node [shape=box, style="filled, rounded", color="black", fontname=helvetica] ;
    edge [fontname=helvetica] ;
    0 [label="failures <= 0.5\nsamples = 100.0%\nvalue = [0.58, 0.42]\nclass = fail", fillcolor="#f8dcc9"] ;
    1 [label="schoolsup_no <= 0.5\nsamples = 79.0%\nvalue = [0.519, 0.481]\nclass = fail", fillcolor="#fdfef0"] ;
    0 --> 1 [label=distance:2.5, labelangle:45, headLabel="true"] ;
    2 [label="Mjob_services <= 0.5\nsamples = 10.1%\nvalue = [0.825, 0.175]\nclass = fail", fillcolor="#eb9c63"] ;
    1 --> 2 ;
    3 [label="Fedu <= 2.5\nsamples = 6.6%\nvalue = [0.923, 0.077]\nclass = fail", fillcolor="#e78c49"] ;
    2 --> 3 ;
    4 [label="famsize_LE3 <= 0.5\nsamples = 2.3%\nvalue = [0.778, 0.222]\nclass = fail", fillcolor="#eeca57"] ;
    3 --> 4 ;
    5 [label="samples = 1.0%\nvalue = [0.5, 0.5]\nclass = fail", fillcolor="#ffffff"] ;
    4 --> 5 ;
    6 [label="samples = 1.3%\nvalue = [1.0, 0.0]\nclass = fail", fillcolor="#e58139"] ;
    4 --> 6 ;
    7 [label="samples = 4.3%\nvalue = [1.0, 0.0]\nclass = fail", fillcolor="#e58139"] ;
    3 --> 7 ;
    8 [label="Walc <= 3.5\nsamples = 3.5%\nvalue = [0.643, 0.357]\nclass = fail", fillcolor="#f3c7a7"] ;
    2 --> 8 ;
    9 [label="famsup_no <= 0.5\nsamples = 2.8%\nvalue = [0.545, 0.455]\nclass = fail", fillcolor="#fbeade"] ;
    8 --> 9 ;
    10 [label="samples = 2.3%\nvalue = [0.667, 0.333]\nclass = fail", fillcolor="#f2c09c"] ;
    9 --> 10 ;
    11 [label="samples = 0.5%\nvalue = [0.0, 1.0]\nclass = pass", fillcolor="#399de5"] ;
    9 --> 11 ;
    12 [label="samples = 0.8%\nvalue = [1.0, 0.0]\nclass = fail", fillcolor="#e58139"] ;
    8 --> 12 ;
    13 [label="famrel <= 1.5\nsamples = 68.9%\nvalue = [0.474, 0.526]\nclass = pass", fillcolor="#ecf5fc"] ;
    1 --> 13 ;
    14 [label="samples = 1.5%\nvalue = [0.0, 1.0]\nclass = pass", fillcolor="#399de5"] ;
    13 --> 14 ;
    15 [label="Mjob_services <= 0.5\nsamples = 67.3%\nvalue = [0.485, 0.515]\nclass = pass", fillcolor="#f3f9fd"] ;
    13 --> 15 ;
    16 [label="Mjob_health <= 0.5\nsamples = 52.7%\nvalue = [0.529, 0.471]\nclass = fail", fillcolor="#fcf1e9"] ;
    15 --> 16 ;
    17 [label="samples = 45.6%\nvalue = [0.567, 0.433]\nclass = fail", fillcolor="#f9e1d0"] ;
    16 --> 17 ;
    18 [label="samples = 7.1%\nvalue = [0.286, 0.714]\nclass = pass", fillcolor="#88c4ee"] ;
    16 --> 18 ;
    19 [label="goout <= 1.5\nsamples = 14.7%\nvalue = [0.328, 0.672]\nclass = pass", fillcolor="#99cdf2"] ;
    15 --> 19 ;
    20 [label="samples = 1.5%\nvalue = [0.833, 0.167]\nclass = fail", fillcolor="#ea9a61"] ;

```

Gambar 2.59 hasil

```

7
8 # save tree
9 tree.export_graphviz(lontong, out_file="student-performance.
10     dot", label="all", impurity=False, proportion=True,
11         feature_names=list(pecel_train_att),
12         class_names=["fail", "pass"], filled=True, rounded=True)

```

8. selanjutnya pada codingan berikut yaitu digunakan untuk mencetak nilai score rata-rata dari ketepatan data yang telah diolah tadi lebih jelsnya dapat dilihat pada gambar kemudian untuk hasilnya dapat dilihat pada gambar tersebut:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 22:03:35 2020
4
5 @author: User
6 """
7
8 lontong.score(pecel_test_att, pecel_test_pass)

```

9. selanjutnya yaitu digunakan untuk memeriksa akurasi dari ketepatan hasil pengolahan data tersebut maka akan didapat nilai rata-rata 60

```

graph TD
    node [shape=box, style="filled, rounded", color="black", fontname=helvetica];
    edge [fontname=helvetica];
    1 [label="failures <= 0.5\nsamples = 100.0%\nvalue = [0.58, 0.42]\nclass = fail", fillcolor="#f8dc09"];
    1 --> 1;
    1 --> 2;
    1 --> 3;
    1 --> 4;
    1 --> 5;
    1 --> 6;
    1 --> 7;
    1 --> 8;
    1 --> 9;
    1 --> 10;
    1 --> 11;
    1 --> 12;
    1 --> 13;
    1 --> 14;
    1 --> 15;
    1 --> 16;
    1 --> 17;
    1 --> 18;
    1 --> 19;
    1 --> 20;
    2 [label="schoolsup_no <= 0.5\nsamples = 79.0%\nvalue = [0.519, 0.481]\nclass = fail", fillcolor="#fdff00"];
    2 --> 1;
    2 --> 3;
    2 --> 4;
    2 --> 5;
    2 --> 6;
    2 --> 7;
    2 --> 8;
    2 --> 9;
    2 --> 10;
    2 --> 11;
    2 --> 12;
    2 --> 13;
    2 --> 14;
    2 --> 15;
    2 --> 16;
    2 --> 17;
    2 --> 18;
    2 --> 19;
    2 --> 20;
    3 [label="Mjob_services <= 0.5\nsamples = 10.1%\nvalue = [0.825, 0.175]\nclass = fail", fillcolor="#eb9c63"];
    3 --> 1;
    3 --> 2;
    3 --> 4;
    3 --> 5;
    3 --> 6;
    3 --> 7;
    3 --> 8;
    3 --> 9;
    3 --> 10;
    3 --> 11;
    3 --> 12;
    3 --> 13;
    3 --> 14;
    3 --> 15;
    3 --> 16;
    3 --> 17;
    3 --> 18;
    3 --> 19;
    3 --> 20;
    4 [label="Fedu <= 2.5\nsamples = 6.6%\nvalue = [0.923, 0.077]\nclass = fail", fillcolor="#e78c49"];
    4 --> 1;
    4 --> 2;
    4 --> 3;
    4 --> 5;
    4 --> 6;
    4 --> 7;
    4 --> 8;
    4 --> 9;
    4 --> 10;
    4 --> 11;
    4 --> 12;
    4 --> 13;
    4 --> 14;
    4 --> 15;
    4 --> 16;
    4 --> 17;
    4 --> 18;
    4 --> 19;
    4 --> 20;
    5 [label="famsize_LE3 <= 0.5\nsamples = 2.3%\nvalue = [0.778, 0.222]\nclass = fail", fillcolor="#eca572"];
    5 --> 1;
    5 --> 2;
    5 --> 3;
    5 --> 4;
    5 --> 6;
    5 --> 7;
    5 --> 8;
    5 --> 9;
    5 --> 10;
    5 --> 11;
    5 --> 12;
    5 --> 13;
    5 --> 14;
    5 --> 15;
    5 --> 16;
    5 --> 17;
    5 --> 18;
    5 --> 19;
    5 --> 20;
    6 [label="samples = 1.0%\nvalue = [0.5, 0.5]\nclass = fail", fillcolor="#ffffff"];
    6 --> 1;
    6 --> 2;
    6 --> 3;
    6 --> 4;
    6 --> 5;
    6 --> 7;
    6 --> 8;
    6 --> 9;
    6 --> 10;
    6 --> 11;
    6 --> 12;
    6 --> 13;
    6 --> 14;
    6 --> 15;
    6 --> 16;
    6 --> 17;
    6 --> 18;
    6 --> 19;
    6 --> 20;
    7 [label="samples = 1.3%\nvalue = [1.0, 0.0]\nclass = fail", fillcolor="#e58139"];
    7 --> 1;
    7 --> 2;
    7 --> 3;
    7 --> 4;
    7 --> 5;
    7 --> 6;
    7 --> 8;
    7 --> 9;
    7 --> 10;
    7 --> 11;
    7 --> 12;
    7 --> 13;
    7 --> 14;
    7 --> 15;
    7 --> 16;
    7 --> 17;
    7 --> 18;
    7 --> 19;
    7 --> 20;
    8 [label="Walc <= 3.5\nsamples = 3.5%\nvalue = [0.643, 0.357]\nclass = fail", fillcolor="#f3c7a7"];
    8 --> 1;
    8 --> 2;
    8 --> 3;
    8 --> 4;
    8 --> 5;
    8 --> 6;
    8 --> 7;
    8 --> 8;
    8 --> 9;
    8 --> 10;
    8 --> 11;
    8 --> 12;
    8 --> 13;
    8 --> 14;
    8 --> 15;
    8 --> 16;
    8 --> 17;
    8 --> 18;
    8 --> 19;
    8 --> 20;
    9 [label="famsup_no <= 0.5\nsamples = 2.8%\nvalue = [0.545, 0.455]\nclass = fail", fillcolor="#fbeade"];
    9 --> 1;
    9 --> 2;
    9 --> 3;
    9 --> 4;
    9 --> 5;
    9 --> 6;
    9 --> 7;
    9 --> 8;
    9 --> 9;
    9 --> 10;
    9 --> 11;
    9 --> 12;
    9 --> 13;
    9 --> 14;
    9 --> 15;
    9 --> 16;
    9 --> 17;
    9 --> 18;
    9 --> 19;
    9 --> 20;
    10 [label="samples = 2.3%\nvalue = [0.667, 0.333]\nclass = fail", fillcolor="#f2c09c"];
    10 --> 1;
    10 --> 2;
    10 --> 3;
    10 --> 4;
    10 --> 5;
    10 --> 6;
    10 --> 7;
    10 --> 8;
    10 --> 9;
    10 --> 11;
    10 --> 12;
    10 --> 13;
    10 --> 14;
    10 --> 15;
    10 --> 16;
    10 --> 17;
    10 --> 18;
    10 --> 19;
    10 --> 20;
    11 [label="samples = 0.5%\nvalue = [0.0, 1.0]\nclass = pass", fillcolor="#399de5"];
    11 --> 1;
    11 --> 2;
    11 --> 3;
    11 --> 4;
    11 --> 5;
    11 --> 6;
    11 --> 7;
    11 --> 8;
    11 --> 9;
    11 --> 10;
    11 --> 12;
    11 --> 13;
    11 --> 14;
    11 --> 15;
    11 --> 16;
    11 --> 17;
    11 --> 18;
    11 --> 19;
    11 --> 20;
    12 [label="samples = 0.8%\nvalue = [1.0, 0.0]\nclass = fail", fillcolor="#e58139"];
    12 --> 1;
    12 --> 2;
    12 --> 3;
    12 --> 4;
    12 --> 5;
    12 --> 6;
    12 --> 7;
    12 --> 8;
    12 --> 9;
    12 --> 10;
    12 --> 11;
    12 --> 12;
    12 --> 13;
    12 --> 14;
    12 --> 15;
    12 --> 16;
    12 --> 17;
    12 --> 18;
    12 --> 19;
    12 --> 20;
    13 [label="famrel <= 1.5\nsamples = 68.9%\nvalue = [0.474, 0.526]\nclass = pass", fillcolor="#ecf5fc"];
    13 --> 1;
    13 --> 2;
    13 --> 3;
    13 --> 4;
    13 --> 5;
    13 --> 6;
    13 --> 7;
    13 --> 8;
    13 --> 9;
    13 --> 10;
    13 --> 11;
    13 --> 12;
    13 --> 13;
    13 --> 14;
    13 --> 15;
    13 --> 16;
    13 --> 17;
    13 --> 18;
    13 --> 19;
    13 --> 20;
    14 [label="samples = 1.5%\nvalue = [0.0, 1.0]\nclass = pass", fillcolor="#399de5"];
    14 --> 1;
    14 --> 2;
    14 --> 3;
    14 --> 4;
    14 --> 5;
    14 --> 6;
    14 --> 7;
    14 --> 8;
    14 --> 9;
    14 --> 10;
    14 --> 11;
    14 --> 12;
    14 --> 13;
    14 --> 14;
    14 --> 15;
    14 --> 16;
    14 --> 17;
    14 --> 18;
    14 --> 19;
    14 --> 20;
    15 [label="Mjob_services <= 0.5\nsamples = 67.3%\nvalue = [0.485, 0.515]\nclass = pass", fillcolor="#f3f9fd"];
    15 --> 1;
    15 --> 2;
    15 --> 3;
    15 --> 4;
    15 --> 5;
    15 --> 6;
    15 --> 7;
    15 --> 8;
    15 --> 9;
    15 --> 10;
    15 --> 11;
    15 --> 12;
    15 --> 13;
    15 --> 14;
    15 --> 15;
    15 --> 16;
    15 --> 17;
    15 --> 18;
    15 --> 19;
    15 --> 20;
    16 [label="Mjob_health <= 0.5\nsamples = 52.7%\nvalue = [0.529, 0.471]\nclass = fail", fillcolor="#fcfcf9"];
    16 --> 1;
    16 --> 2;
    16 --> 3;
    16 --> 4;
    16 --> 5;
    16 --> 6;
    16 --> 7;
    16 --> 8;
    16 --> 9;
    16 --> 10;
    16 --> 11;
    16 --> 12;
    16 --> 13;
    16 --> 14;
    16 --> 15;
    16 --> 16;
    16 --> 17;
    16 --> 18;
    16 --> 19;
    16 --> 20;
    17 [label="samples = 45.6%\nvalue = [0.567, 0.433]\nclass = fail", fillcolor="#f9e1d0"];
    17 --> 1;
    17 --> 2;
    17 --> 3;
    17 --> 4;
    17 --> 5;
    17 --> 6;
    17 --> 7;
    17 --> 8;
    17 --> 9;
    17 --> 10;
    17 --> 11;
    17 --> 12;
    17 --> 13;
    17 --> 14;
    17 --> 15;
    17 --> 16;
    17 --> 17;
    17 --> 18;
    17 --> 19;
    17 --> 20;
    18 [label="samples = 7.1%\nvalue = [0.286, 0.714]\nclass = pass", fillcolor="#88c4ef"];
    18 --> 1;
    18 --> 2;
    18 --> 3;
    18 --> 4;
    18 --> 5;
    18 --> 6;
    18 --> 7;
    18 --> 8;
    18 --> 9;
    18 --> 10;
    18 --> 11;
    18 --> 12;
    18 --> 13;
    18 --> 14;
    18 --> 15;
    18 --> 16;
    18 --> 17;
    18 --> 18;
    18 --> 19;
    18 --> 20;
    19 [label="goutlet <= 1.5\nsamples = 14.7%\nvalue = [0.328, 0.672]\nclass = pass", fillcolor="#99cdf2"];
    19 --> 1;
    19 --> 2;
    19 --> 3;
    19 --> 4;
    19 --> 5;
    19 --> 6;
    19 --> 7;
    19 --> 8;
    19 --> 9;
    19 --> 10;
    19 --> 11;
    19 --> 12;
    19 --> 13;
    19 --> 14;
    19 --> 15;
    19 --> 16;
    19 --> 17;
    19 --> 18;
    19 --> 19;
    19 --> 20;
    20 [label="samples = 1.5%\nvalue = [0.833, 0.167]\nclass = fail", fillcolor="#ea9a61"];
    20 --> 1;
    20 --> 2;
    20 --> 3;
    20 --> 4;
    20 --> 5;
    20 --> 6;
    20 --> 7;
    20 --> 8;
    20 --> 9;
    20 --> 10;
    20 --> 11;
    20 --> 12;
    20 --> 13;
    20 --> 14;
    20 --> 15;
    20 --> 16;
    20 --> 17;
    20 --> 18;
    20 --> 19;
    20 --> 20;

```

Gambar 2.60 hasil

persen dari hasil pengolahan data tersebut untuk lebih jelasnya dapat dilihat pada gambar pada codingan tersebut pada baris ke satu melakukan import library dari sklern kemudian pada baris selanjutnya mengisi nilai skor dengan nilai pada variabel lontong setelah hal tersebut dilakukan kemudian data tersebut di eksekusi. berikut merupakan hasil dari code tersebut dapat dilihat pada gambar.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 22:03:36 2020
4
5 @author: User
6 """
7
8 from sklearn.model_selection import cross_val_score
9 scores = cross_val_score(lontong, pecel_att, pecel_pass, cv
10 =5)
11 # show average score and +/- two standard deviations away
12 #(covering 95% of scores)
13 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
14 std() * 2))
```

```
In [61]: runfile('D:/SEMESTER 6/wert/9.py', wdir='D:/SEMESTER 6/wert')
Accuracy: 0.59 (+/- 0.05)
```

Gambar 2.61 hasil

10. membuat rank akurasi dari 1 sampai 20 untuk melihat akurasi data apakah datatersebut terdapat di rata rata 60 persen atau tidak dengan cara membuat lagi variabel baru dengan nilai tree diadalamnya jadi hampir mirip seperti membuat pohon keputusan namun ini langsung dalam bentuk rata rata akurasi yanlebih spesifik untuk lebih jelasnya dapat dilihat pada gambar codingan berikut. dan untuk hasilnya dapat dilihat pada gambar berikut.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 22:03:38 2020
4
5 @author: User
6 """
7
8 for max_depth in range(1, 100):
9     lontong = tree.DecisionTreeClassifier(criterion="entropy",
10                                         , max_depth=max_depth)
11     scores = cross_val_score(lontong, pecel_att, pecel_pass,
12                               cv=5)
13     print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (
14         max_depth, scores.mean(), scores.std() * 2))
```

```
Max depth: 1, Accuracy: 0.58 (+/- 0.01)
Max depth: 2, Accuracy: 0.58 (+/- 0.06)
Max depth: 3, Accuracy: 0.56 (+/- 0.06)
Max depth: 4, Accuracy: 0.56 (+/- 0.08)
Max depth: 5, Accuracy: 0.58 (+/- 0.05)
Max depth: 6, Accuracy: 0.62 (+/- 0.05)
Max depth: 7, Accuracy: 0.58 (+/- 0.05)
Max depth: 8, Accuracy: 0.60 (+/- 0.07)
Max depth: 9, Accuracy: 0.58 (+/- 0.10)
Max depth: 10, Accuracy: 0.57 (+/- 0.04)
Max depth: 11, Accuracy: 0.58 (+/- 0.06)
Max depth: 12, Accuracy: 0.57 (+/- 0.08)
Max depth: 13, Accuracy: 0.57 (+/- 0.10)
Max depth: 14, Accuracy: 0.55 (+/- 0.07)
Max depth: 15, Accuracy: 0.58 (+/- 0.08)
Max depth: 16, Accuracy: 0.58 (+/- 0.05)
Max depth: 17, Accuracy: 0.57 (+/- 0.09)
Max depth: 18, Accuracy: 0.57 (+/- 0.11)
Max depth: 19, Accuracy: 0.57 (+/- 0.09)
Max depth: 20, Accuracy: 0.57 (+/- 0.07)
Max depth: 21, Accuracy: 0.57 (+/- 0.06)
Max depth: 22, Accuracy: 0.58 (+/- 0.07)
Max depth: 23, Accuracy: 0.58 (+/- 0.10)
```

Gambar 2.62 hasil

11. untuk selanjutnya yaitu menentukan nilai untuk grafik hampirsama dengan nilai akurasi tadi pertama tentukan rank atau batasan nilai itu disini batasannya di mulai dari 1 sampai 20 dengan menggunakan nilai tree tadi maka hasilnya dapat ditentuka kemudian buat variabel i untuk penomoran tiap record yang keluar atau record hadil dari eksekusi tree tersebut. un-

tuk lebih jelasnya dapat dilihat pada gambar dan untuk hasilnya dapat dilihat pada gambar.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 22:07:50 2020
4
5 @author: User
6 """
7
8 depth_acc = np.empty((19,3), float)
9 i = 0
10 for max_depth in range(1, 20):
11     lontong = tree.DecisionTreeClassifier(criterion="entropy",
12                                            max_depth=max_depth)
13     scores = cross_val_score(lontong, pecel_att, pecel_pass,
14                               cv=5)
15     depth_acc[i,0] = max_depth
16     depth_acc[i,1] = scores.mean()
17     depth_acc[i,2] = scores.std() * 2
18     i += 1
19
20 print(depth_acc)

```

```

[[1.0000000e+00 5.79746835e-01 1.01265823e-02]
 [2.0000000e+00 5.69620253e-01 2.77327877e-02]
 [3.0000000e+00 5.82278481e-01 7.33740595e-02]
 [4.0000000e+00 5.77215190e-01 3.03797468e-02]
 [5.0000000e+00 5.84810127e-01 9.39100607e-02]
 [6.0000000e+00 5.56962025e-01 1.01265823e-01]
 [7.0000000e+00 5.92405063e-01 6.28337906e-02]
 [8.0000000e+00 6.0000000e-01 6.71721477e-02]
 [9.0000000e+00 5.79746835e-01 6.86818266e-02]
 [1.0000000e+01 6.10126582e-01 7.74534610e-02]
 [1.1000000e+01 6.07594937e-01 5.77304013e-02]
 [1.2000000e+01 5.94936709e-01 9.47255035e-02]
 [1.3000000e+01 5.97468354e-01 5.16356406e-02]
 [1.4000000e+01 5.82278481e-01 5.77304013e-02]
 [1.5000000e+01 5.77215190e-01 6.11799796e-02]
 [1.6000000e+01 5.82278481e-01 7.67886121e-02]
 [1.7000000e+01 5.82278481e-01 7.51007442e-02]
 [1.8000000e+01 6.02531646e-01 5.68353021e-02]
 [1.9000000e+01 5.92405063e-01 7.05234849e-02]]

```

Gambar 2.63 hasil

12. terakhir yaitu pembuatan grafik untuk pembuatan grafik diambil data dari codingan sebelumnya yang 20 record tadi dengan cara mengimport libray matplotlib.pyplot yang di inisialisasi menjadi bakwankemudian inisialisasi tersebut di eksekusi. untuk lebih jelasnya codingannya seperti gambar dan untuk hasilnya seperti gambar berikut.

```

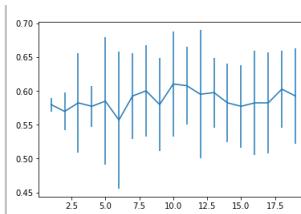
1 # -*- coding: utf-8 -*-
2 """
3 Created on Thu Mar  5 22:07:50 2020
4
5 @author: User
6 """
7
8 import matplotlib.pyplot as plt
9 fig, ax = plt.subplots()

```

```

10 ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc
11 [:,2])
plt.show()

```



Gambar 2.64 hasil

2.4.3 Penanganan Error

1. skrinsut error

```

In [2]: runfile('D:/SEMESTER 6/wert/2.py', wdir='D:/SEMESTER 6/wert')
Traceback (most recent call last):

  File "<ipython-input-2-4b3c06f7fd19>", line 1, in <module>
    runfile('D:/SEMESTER 6/wert/2.py', wdir='D:/SEMESTER 6/wert')

  File "C:\Users\User\Anaconda3\lib\site-packages\spyder_kernels\customize
\spydercustomize.py", line 827, in runfile
    execfile(filename, namespace)

  File "C:\Users\User\Anaconda3\lib\site-packages\spyder_kernels\customize
\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "D:/SEMESTER 6/wert/2.py", line 10, in <module>
    pecel['pass'] = pecelapply(lambda row: 1 if (row['F1']+row['F2']+row['F3'])

NameError: name 'pecelapply' is not defined

```

Gambar 2.65 Error

2. kode error dan jenis errornya .

```

pecel['pass'] = pecelapply(lambda row: 1 if (row['G1']+row['G2']+row
>= 35 else 0, axis=1)
pecel = pecel.drop(['G1', 'G2', 'G3'], axis=1)
print(pecel.head())

```

NameError: name pecelapply is not defined

3. Solusi pemecahan masalah

pada codingan tersebut error karena pecelapply tergabung, seharusnya dipisahkan oleh titik



Date: 2020-03-05

PLAGIARISM SCAN REPORT



Exclude Url : None

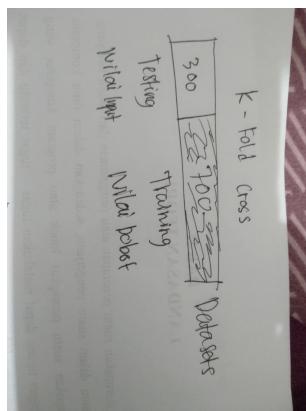
Gambar 2.66 Error

2.4.4 Plagiat

2.5 1174039 Liyana Majdah Rahma

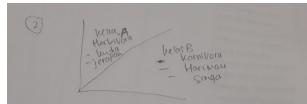
2.5.1 Teori

1. Jelaskan Apa Itu binari classification dilengkapi ilustrasi gambar sendiri.
Klasifikasi biner merupakan tugas yang digunakan untuk mengklasifikasi suatu elemen-elemen dari himpunan tertentu yang terdiri dari dua kelompok yang ditentukan berdasarkan klasifikasi.

**Gambar 2.67** contoh K-fold cross validation

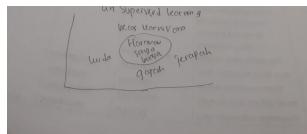
2. Jelaskan Apa itu supervised learning , unsupervised learning dan clustering dengan ilustrasi gambar sendiri.

supervised learning merupakan tipe learning dimana terdapat sebuah metode pendekatan yang mempunyai variable input dan output, serta terdapat variable yang ditargetkan dari pendekatan ini adalah mengelompokan suatu data ke data yang sudah ada sebelumnya. Dimana terdapat kelas A itu dikategorikan sebagai hewan herbivora seperti kuda, dan jerapah. Sedangkan kelas B dikategorikan sebagai hewan karnivora seperti harimau dan singa. Dilihat pada gambar.



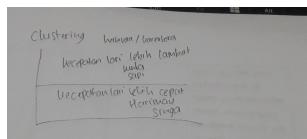
Gambar 2.68 contoh supervised learning

unsupervised merupakan tipe learning yang tidak memiliki data latih sehingga data yang sudah ada, kita kelompokkan data tersebut menjadi dua bagian ataupun menjadi tiga bagian. Untuk lebih jelasnya dapat dilihat pada gambar.



Gambar 2.69 contoh unsupervised learning

clustering merupakan proses yang mengklasifikasi berdasarkan suatu parameter dalam pententuannya. Untuk lebih jelasnya dilihat pada gambar berikut



Gambar 2.70 contoh clustering

3. Jelaskan apa itu evaluasi dan akurasi dan disertai ilustrasi contoh dengan gambar sendiri.

evaluasi merupakan suatu kegiatan yang dilakukan dari pengamatan berbagai macam bukti untuk mengukur dampak efektifitas dari objek atau proses yang berkaitan dengan spesifikasi yang telah ditentukan. sedangkan akurasi merupakan bagian dari evaluasi yang merupakan ketepatan

data terhadap suatu objek yang memiliki kriteria. Dapat dilihat pada gambar berikut

Pada saat ini awal	
Ojeng	Kembar 2 kaki
Burung	Kembar 2 sayap
Elang	Kembar 2 kuku
Elang	Kembar 2 sayap
	terdapat kembar

Gambar 2.71 contoh evaluasi dan akurasi

4. Jelaskan bagaimana cara membuat Confusion Matrix, Buat confusion matrix sendiri.

Pada confusion matrix menentukan parameter yang akan di evaluasi contohnya elang,burung merpati,burung kakatua dengan tabel baris dan kolom berjumlah tida kemudian ditentukan dengan nilai miring pada setiap kolom saya beri nilai 12 dengan ketentuan setiap baris harus bernilai 12 jika kolom lain harus jumlah 15 jika tidak berarti data tidak akurat. Dapat dilihat pada gambar berikut

Confusion matrix		
Ojeng	Burung	Elang
Ojeng	12	12
Burung	12	12
Elang	12	12

Confusion matrix			
Ojeng	Burung	Elang	
Ojeng	6	3	3
Burung	5	6	1
Elang	3	7	2

Gambar 2.72 contoh Confusion Matrix

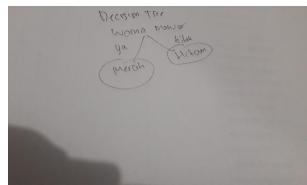
5. Jelaskan bagaimana K-fold cross validation bekerja dengan gambar ilustrasi contoh buatan sendiri. K-fold Cross Validation merupakan cara untuk melatih suatu mesin dimana di dalamnya terdapat data set yang dibagi menjadi dua yaitu untuk data testing dan data training contoh 1000 data merupakan data set dan 300 data digunakan untuk data testing kemudian 700 datanya. Sedangkan nilai testing akan dijadikan nilai inputan untuk rumus regresi linier. contohnya dapat dilihat pada gambar berikut :

1000	700	300
<i>K - fold cross</i>		
Testing Nilai Input	Training Nilai Label	Bila Saja

Gambar 2.73 contoh K-fold cross validation

6. Jelaskan Apa itu decision tree dengan gambar ilustrasi contoh buatan sendiri.

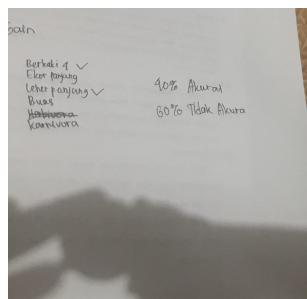
Decision tree merupakan implementasi dari binari clasification dimana pada pohon keputusan akan terdapat root atau akar dan cabang cabangnya yang nilainya seperti if contoh jika pada root berisi nilai warna mawar, apakah merah pada cabang satu bernilai iya dan pada cabang dua bernilai tidak jika nilainya iya berarti mawar dan jika tidak maka bukan mawar. agar lebih jelas dapat dilihat pada gambar decision tree berikut:



Gambar 2.74 contoh decision tree

7. jelaskan apa itu information gain dan entropi dengan gambar ilustrasi buatan sendiri.

information gain merupakan kriteria yang terdapat dalam pembagian sebuah objek seperti gigi tajam, berkaki 4, pemakan daging termasuk karnivora. untuk lebih jelasnya dapat dilihat pada gambar berikut :



Gambar 2.75 contoh information gain

sedangkan entropi merupakan ukuran keacakan dari informasi semakin tinggi entropi maka semakin sulit dalam menentukan keputusan contoh dalam menentukan satu gen semakin detail informasi maka akan semakin susah dalam menentukan keputusan.

2.5.2 Sikic-Learn

1. pada surcode pertama yang dapat dilihat pada gambar pada baris pertama di tuliskan import pandas as baso yang berarti mengimport library pandas. selanjutnya pada baris ke dua codingan tersebut berisi code tersebut terdapat variabel pecel yang berisi inisialisasi pandas dengan aksi untuk membaca vfile csv yang terdapat pada direktori pada komputer kemudian terdapat sep samadengan titik koma yang berarti pemisah field di dalam file tersebut merupakan titik koma. kemudian pada bagian akhir terdapat code len (nama variabel) yang berarti akan menghitung jumlah baris pada file tersebut. untuk hasilnya dapat dilihat pada gambar dibawah.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 22:56:28 2020
4
5 @author: Liyana
6 """
7
8 # load dataset (student mat pakenya)
9 import pandas as pd
10 jatibarang = pd.read_csv('D:\mata kuliah poltekppos\Python-
    Artificial-Intelligence-Projects-for-Beginners-master\
    Chapter01\dataset\student-mat.csv', sep=';')
11 print(len(jatibarang))

```



Gambar 2.76 hasil

2. Pada code selanjutnya akan ditambahkan fungsi untuk lulus atau gagal yang dibuat berupa kolom kolom yang di dalamnya terdapat nilai 0 dan 1 dimana 0 berarti gagal dan 1 berarti lulus. kemudian variabel pada codingan sebelumnya masih digunakan. dimana variabel pecel digunakan karena berisi nilai file csv kemudian dilakukan ekseskuji dengan parameter G1, G2, dan G3 dengan fungsi lambda yang berarti if bersarang atau if didalam if yang berarti nilai yang di eksekusi akan dilempar ke dalam setiap paramater sesuai dengan kriteria dan axis=1 yaitu nilai tersebut akan digunakan dari tiap baris data. sedangkan pada codingan variabel pecel di berikan aksi drop yaitu penurunan nilai pada bagian baris data. dan pada code pecel.head yaitu untuk mengeksekusi codingan sebelumnya untuk lebih jelasnya dapat dilihat pada gambar dan hasilnya seperti pada gambar berikut :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 22:41:20 2020
4
5 @author: Liyana

```

```

6 """
7
8 # generate binary label (pass/fail) based on G1+G2+G3
9 # (test grades, each 0–20 pts); threshold for passing is sum
10 >=30
11 jatibarang[ 'pass' ] = jatibarang .apply(lambda row: 1 if (row[ 'G1' ]+row[ 'G2' ]+row[ 'G3' ]) 
12 >= 35 else 0, axis=1)
13 jatibarang = jatibarang .drop([ 'G1' , 'G2' , 'G3' ], axis=1)
14 print(jatibarang .head())

```

SCHOOL	sex	age	address	famsize	Pstatus	Mjob	Walc	health	absences	pass
G1	F	17	U	GT3 ...	L	T	3	4	0	
G2	F	15	U	LE3 ...	X	S	5	10	0	
G3	F	13	U	LE3 ...	X	S	5	7	1	
G4	F	16	U	GT3 ...	X	S	5	6	0	

Gambar 2.77 hasil

3. pada kodingan selanjutnya diguanakan untuk menambahkan nilai numerik berupa 0 dan 1 yang dirubah dari nilai tidak dan iya hal ini merupakan fungsi dari codingan get_dummies pada baris pertama pada gambar yang nilainya diambil dari variabel pecel yang telah di dekralasikan tadi banyaknya data numerik itu sendiri tergantung pada field dari kolom yang di camtumkan pada codingan dengan catatan field tersebut harus ada dalam data csv yang di import oleh codingan pertama tadi maka hasilnya akam merubah data dalam field tersebut menjadi 0 dan 1 untuk lebih jelasnya dapat di lihat pada gambar berikut;

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 22:54:20 2020
4
5 @author: Liyana
6 """
7
8 # use one-hot encoding on categorical columns
9 jatibarang = pd.get_dummies(jatibarang , columns=[ 'sex' , 'school' , 'address' , 'famsize' , 'Pstatus' , 'Mjob' , 'Fjob' , 'reason' , 'guardian' , 'schoolsups' , 'famsup' , 'paid' , 'activities' , 'nursery' , 'higher' , 'internet' , 'romantic' ])
10 jatibarang .head()

```

```
[15 rows x 31 columns]
Passing: 166 out of 395 (42.03%)
Traceback (most recent call last):
```

Gambar 2.78 hasil

4. selanjutnya pada codingan selanjutnya yaitu menentukan data training dan data testing dari dataset dimana variabel pecel yang berisi data csv tadi dibuat perbandingan 500 untuk data training dan sisanya yaitu 149 digunakan untuk data testing hal ini di lakukan pada baris ke 1 sampai ke 3 pada gambar kemudian nilai tersebut di turunkan brdasarkan baris

pada data set hal itu dapat dilihat pada baris ke 4 sampai ke 9 pada gambar kemudian selanjutnya mengimport library numpy yang berguna untuk oprasi vektor dan matrix karna data diatas berupa data matrix maka library ini di gunakan. untuk hasilnya dapat dilihat pada gambar.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 22:57:59 2020
4
5 @author: Liyana
6 """
7
8 # shuffle rows
9 jatibarang = jatibarang.sample(frac=1)
10 # split training and testing data
11 jatibarang_train = jatibarang[:500]
12 jatibarang_test = jatibarang[500:]
13 jatibarang_train.att = jatibarang_train.drop(['pass'], axis=1)
14 jatibarang_train.pass = jatibarang_train['pass']
15 jatibarang_test.att = jatibarang_test.drop(['pass'], axis=1)
16 jatibarang_test.pass = jatibarang_test['pass']
17 jatibarang.att = jatibarang.drop(['pass'], axis=1)
18 jatibarang.pass = jatibarang['pass']
19 # number of passing students in whole dataset:
20 import numpy as np
21 print("Passing: %d out of %d (%.2f%%)" % (np.sum(
    jatibarang.pass), len(jatibarang.pass), 100*float(np.sum(
        jatibarang.pass)) / len(jatibarang.pass)))

```

	school	sex	age	address	family size	...	Dalc	Walc	health	absences	pass
0	GP	F	18	U	G1	...	1	1	3	6	0
1	GP	F	19	U	G1	...	1	1	3	6	0
2	GP	F	15	U	L1	...	2	1	3	10	0
3	GP	F	15	U	G1	...	1	1	5	2	1
4	GP	F	16	U	G1	...	1	2	5	4	0

Gambar 2.79 hasil

- selanjutnya yaitu membuat pohon keputusan dapat lebih jelasnya dapat di lihat pada gambar. pada baris pertama yairu memasukan librari tree kemudian pada baris kedua dibuat variabel lontong dengan nilai DecisionTreeClasifier yang merupakan paket atau fungsi dari scikit-learn yang merupakan class yang mampu melakukan multi class. sedangkan max_depth=5 merupakan untuk penyesuaian data terhadap pohon keputusan itu sndiri. untuk hasilnya dapat dilihat pada gambar

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 23:01:08 2020
4
5 @author: Liyana
6 """
7
8 # fit a decision tree
9 from sklearn import tree

```

```

10 lobener = tree.DecisionTreeClassifier(criterion="entropy",
   max_depth=5)
11 lobener = lobener.fit(jatibarang_train_att ,
   jatibarang_train_pass)
```

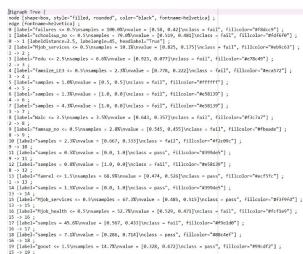
```
[5 rows x 31 columns]
Passing: 166 out of 395 (42.03%)
```

Gambar 2.80 hasil

6. selanjutnya yaitu pembuatan gambar dari pohon keputusan yang tadi di buta pada codingan sebelumnya pada baris pertama codingan ya itu mengimport library graphviz kemudian pada baris ke dua yaitu memberian nilai pada variabel baru dot data nilainya diambil dari pembuatan pohon keputusan tadi kemudian di tentukannya nilai benar dan salah dari codingan tersebut setelah itu dibuatlah sebuah variabel untuk menampung hasil eksekusi tersebut kemudian variabel tersebut di running untuk lebih jelasnya dapat di lihat pada gambar.kemudian hasilnya dapat dilihat pada gambar.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 23:05:14 2020
4
5 @author: Liyana
6 """
7
8 # visualize tree
9 import graphviz
10 dot_data = tree.export_graphviz(lobener, out_file=None, label
11   ="all", impurity=False, proportion=True,
12   feature_names=list(
13     jatibarang_train_att), class_names=["fail", "pass"],
14   filled=True, rounded=True)
15 graph = graphviz.Source(dot_data)
16 graph
```



Gambar 2.81 hasil

7. selanjutnya pembuatan method untuk menyimpan data pohon dengan menarik data langsung dari pohon keputusan di buat tadi untuk code lebih jelasnya dapat dilihat pada gambar. kemudian intuk hasilnya dapat dilihat pada gambar berikut.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 23:08:17 2020
4
5 @author: Liyana
6 """
7
8 tree.export_graphviz(lobener, out_file="student-performance.
9         dot", label="all", impurity=False, proportion=True,
10                     feature_names=list(jatibarang_train_att)
11                     , class_names=["fail", "pass"], filled=True, rounded=True)
```

```

graph TD
    root[jatibarang] --> 1{Jika <math>x_1 \leq 0.4</math>}
    1 --> 2{Jika <math>x_2 \leq 0.3</math>}
    2 --> 3{Jika <math>x_3 \leq 0.3</math>}
    3 --> 4{Jika <math>x_4 \leq 0.5</math>}
    4 --> 5{Jika <math>x_5 \leq 0.5</math>}
    5 --> 6{Jika <math>x_6 \leq 0.5</math>}
    6 --> 7{Jika <math>x_7 \leq 0.5</math>}
    7 --> 8{Jika <math>x_8 \leq 0.5</math>}
    8 --> 9{Jika <math>x_9 \leq 0.5</math>}
    9 --> 10{Jika <math>x_{10} \leq 0.5</math>}
    10 --> 11{Jika <math>x_{11} \leq 0.5</math>}
    11 --> 12{Jika <math>x_{12} \leq 0.5</math>}
    12 --> 13{Jika <math>x_{13} \leq 0.5</math>}
    13 --> 14{Jika <math>x_{14} \leq 0.5</math>}
    14 --> 15{Jika <math>x_{15} \leq 0.5</math>}
    15 --> 16{Jika <math>x_{16} \leq 0.5</math>}
    16 --> 17{Jika <math>x_{17} \leq 0.5</math>}
    17 --> 18{Jika <math>x_{18} \leq 0.5</math>}
    18 --> 19{Jika <math>x_{19} \leq 0.5</math>}
    19 --> 20{Jika <math>x_{20} \leq 0.5</math>}
    20 --> 21{Jika <math>x_{21} \leq 0.5</math>}
    21 --> 22{Jika <math>x_{22} \leq 0.5</math>}
    22 --> 23{Jika <math>x_{23} \leq 0.5</math>}
    23 --> 24{Jika <math>x_{24} \leq 0.5</math>}
    24 --> 25{Jika <math>x_{25} \leq 0.5</math>}
    25 --> 26{Jika <math>x_{26} \leq 0.5</math>}
    26 --> 27{Jika <math>x_{27} \leq 0.5</math>}
    27 --> 28{Jika <math>x_{28} \leq 0.5</math>}
    28 --> 29{Jika <math>x_{29} \leq 0.5</math>}
    29 --> 30{Jika <math>x_{30} \leq 0.5</math>}
    30 --> 31{Jika <math>x_{31} \leq 0.5</math>}
    31 --> 32{Jika <math>x_{32} \leq 0.5</math>}
    32 --> 33{Jika <math>x_{33} \leq 0.5</math>}
    33 --> 34{Jika <math>x_{34} \leq 0.5</math>}
    34 --> 35{Jika <math>x_{35} \leq 0.5</math>}
    35 --> 36{Jika <math>x_{36} \leq 0.5</math>}
    36 --> 37{Jika <math>x_{37} \leq 0.5</math>}
    37 --> 38{Jika <math>x_{38} \leq 0.5</math>}
    38 --> 39{Jika <math>x_{39} \leq 0.5</math>}
    39 --> 40{Jika <math>x_{40} \leq 0.5</math>}
    40 --> 41{Jika <math>x_{41} \leq 0.5</math>}
    41 --> 42{Jika <math>x_{42} \leq 0.5</math>}
    42 --> 43{Jika <math>x_{43} \leq 0.5</math>}
    43 --> 44{Jika <math>x_{44} \leq 0.5</math>}
    44 --> 45{Jika <math>x_{45} \leq 0.5</math>}
    45 --> 46{Jika <math>x_{46} \leq 0.5</math>}
    46 --> 47{Jika <math>x_{47} \leq 0.5</math>}
    47 --> 48{Jika <math>x_{48} \leq 0.5</math>}
    48 --> 49{Jika <math>x_{49} \leq 0.5</math>}
    49 --> 50{Jika <math>x_{50} \leq 0.5</math>}
    50 --> 51{Jika <math>x_{51} \leq 0.5</math>}
    51 --> 52{Jika <math>x_{52} \leq 0.5</math>}
    52 --> 53{Jika <math>x_{53} \leq 0.5</math>}
    53 --> 54{Jika <math>x_{54} \leq 0.5</math>}
    54 --> 55{Jika <math>x_{55} \leq 0.5</math>}
    55 --> 56{Jika <math>x_{56} \leq 0.5</math>}
    56 --> 57{Jika <math>x_{57} \leq 0.5</math>}
    57 --> 58{Jika <math>x_{58} \leq 0.5</math>}
    58 --> 59{Jika <math>x_{59} \leq 0.5</math>}
    59 --> 60{Jika <math>x_{60} \leq 0.5</math>}
    60 --> 61{Jika <math>x_{61} \leq 0.5</math>}
    61 --> 62{Jika <math>x_{62} \leq 0.5</math>}
    62 --> 63{Jika <math>x_{63} \leq 0.5</math>}
    63 --> 64{Jika <math>x_{64} \leq 0.5</math>}
    64 --> 65{Jika <math>x_{65} \leq 0.5</math>}
    65 --> 66{Jika <math>x_{66} \leq 0.5</math>}
    66 --> 67{Jika <math>x_{67} \leq 0.5</math>}
    67 --> 68{Jika <math>x_{68} \leq 0.5</math>}
    68 --> 69{Jika <math>x_{69} \leq 0.5</math>}
    69 --> 70{Jika <math>x_{70} \leq 0.5</math>}
    70 --> 71{Jika <math>x_{71} \leq 0.5</math>}
    71 --> 72{Jika <math>x_{72} \leq 0.5</math>}
    72 --> 73{Jika <math>x_{73} \leq 0.5</math>}
    73 --> 74{Jika <math>x_{74} \leq 0.5</math>}
    74 --> 75{Jika <math>x_{75} \leq 0.5</math>}
    75 --> 76{Jika <math>x_{76} \leq 0.5</math>}
    76 --> 77{Jika <math>x_{77} \leq 0.5</math>}
    77 --> 78{Jika <math>x_{78} \leq 0.5</math>}
    78 --> 79{Jika <math>x_{79} \leq 0.5</math>}
    79 --> 80{Jika <math>x_{80} \leq 0.5</math>}
    80 --> 81{Jika <math>x_{81} \leq 0.5</math>}
    81 --> 82{Jika <math>x_{82} \leq 0.5</math>}
    82 --> 83{Jika <math>x_{83} \leq 0.5</math>}
    83 --> 84{Jika <math>x_{84} \leq 0.5</math>}
    84 --> 85{Jika <math>x_{85} \leq 0.5</math>}
    85 --> 86{Jika <math>x_{86} \leq 0.5</math>}
    86 --> 87{Jika <math>x_{87} \leq 0.5</math>}
    87 --> 88{Jika <math>x_{88} \leq 0.5</math>}
    88 --> 89{Jika <math>x_{89} \leq 0.5</math>}
    89 --> 90{Jika <math>x_{90} \leq 0.5</math>}
    90 --> 91{Jika <math>x_{91} \leq 0.5</math>}
    91 --> 92{Jika <math>x_{92} \leq 0.5</math>}
    92 --> 93{Jika <math>x_{93} \leq 0.5</math>}
    93 --> 94{Jika <math>x_{94} \leq 0.5</math>}
    94 --> 95{Jika <math>x_{95} \leq 0.5</math>}
    95 --> 96{Jika <math>x_{96} \leq 0.5</math>}
    96 --> 97{Jika <math>x_{97} \leq 0.5</math>}
    97 --> 98{Jika <math>x_{98} \leq 0.5</math>}
    98 --> 99{Jika <math>x_{99} \leq 0.5</math>}
    99 --> 100{Jika <math>x_{100} \leq 0.5</math>}
    100 --> 101{Jika <math>x_{101} \leq 0.5</math>}
```

Gambar 2.82 hasil

8. selanjutnya pada codingan berikut yaitu digunakan untuk mencetak nilai score rata-rata dari ketepatan data yang telah diolah tadi lebih jelasnya dapat dilihat pada gambar kemudian untuk hasilnya dapat dilihat pada gambar tersebut:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 23:09:11 2020
4
5 @author: Liyana
6 """
7
8 lobener.score(jatibarang_test_att , jatibarang_test_pass)
```

9. selanjutnya yaitu digunakan untuk memeriksa akurasi dari ketepatan hasil pengolahan data tersebut maka akan didapat nilai rata-rata 60 persen dari hasil pengolahan data tersebut untuk lebih jelasnya dapat dilihat pada gambar pada codingan tersebut pada baris ke satu melakukan import library dari sklern kemudian pada baris selanjutnya mengisi nilai skor dengan nilai pada variabel lontong setelah hal tersebut dilakukan kemudian data tersebut di eksekusi. berikut merupakan hasil dari code tersebut dapat dilihat pada gambar.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 23:09:58 2020
4
5 @author: Liyana
6 """
7
8 from sklearn.model_selection import cross_val_score
9 scores = cross_val_score(lobener, jatibarang_att,
10                         jatibarang_pass, cv=5)
11 # show average score and +/- two standard deviations away
12 #(covering 95% of scores)
13 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
14                                         std() * 2))

```

Accuracy: 0.59 (+/- 0.05)

Gambar 2.83 hasil

10. membuat rank akurasi dari 1 sampai 20 untuk melihat akurasi data apakah datatersebut terdapat di rata rata 60 persen atau tidak dengan cara membuat lagi variabel baru dengan nilai tree diadalamnya jadi ham-pir mirip seperti membuat pohon keputusan namun ini langsung dalam bentuk rata rata akurasi yanlebih spesifik untuk lebih jelasnya dpt dil-ihat pada gambar codingan berikut. dan untuk hasilnya dapat dilihat pada gambar berikut.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 23:10:49 2020
4
5 @author: Liyana
6 """
7
8 for max_depth in range(1, 100):
9     lobener= tree.DecisionTreeClassifier(criterion="entropy",
10                                         max_depth=max_depth)
11     scores = cross_val_score(lobener, jatibarang_att,
12                             jatibarang_pass, cv=5)
13     print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (
14         max_depth, scores.mean(), scores.std() * 2))

```

```

Max depth: 1, Accuracy: 0.58 (+/- 0.01)
Max depth: 2, Accuracy: 0.58 (+/- 0.06)
Max depth: 3, Accuracy: 0.56 (+/- 0.06)
Max depth: 4, Accuracy: 0.56 (+/- 0.08)
Max depth: 5, Accuracy: 0.58 (+/- 0.05)
Max depth: 6, Accuracy: 0.58 (+/- 0.05)
Max depth: 7, Accuracy: 0.56 (+/- 0.05)
Max depth: 8, Accuracy: 0.56 (+/- 0.07)
Max depth: 9, Accuracy: 0.58 (+/- 0.18)
Max depth: 10, Accuracy: 0.57 (+/- 0.04)
Max depth: 11, Accuracy: 0.58 (+/- 0.06)
Max depth: 12, Accuracy: 0.57 (+/- 0.08)
Max depth: 13, Accuracy: 0.57 (+/- 0.10)
Max depth: 14, Accuracy: 0.55 (+/- 0.07)
Max depth: 15, Accuracy: 0.58 (+/- 0.08)
Max depth: 16, Accuracy: 0.56 (+/- 0.05)
Max depth: 17, Accuracy: 0.57 (+/- 0.09)
Max depth: 18, Accuracy: 0.57 (+/- 0.11)
Max depth: 19, Accuracy: 0.57 (+/- 0.09)
Max depth: 20, Accuracy: 0.57 (+/- 0.07)
Max depth: 21, Accuracy: 0.57 (+/- 0.06)
Max depth: 22, Accuracy: 0.58 (+/- 0.07)
Max depth: 23, Accuracy: 0.58 (+/- 0.10)

```

Gambar 2.84 hasil

11. untuk selanjutnya yaitu menentukan nilai untuk grafik hampir sama dengan nilai akurasi tadi pertama tentukan rank atau batasan nilai itu disini batasannya di mulai dari 1 sampai 20 dengan menggunakan nilai tree tadi maka hasilnya dapat ditentuka kemudian buat variabel i untuk penomoran tiap record yang keluar atau recod hadil dari eksekusi tree tersebut. untuk lebih jelasnya dapat dilihat pada gambar dan untuk hasilnya dapat dilihat pada gambar.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 23:11:44 2020
4
5 @author: Liyana
6 """
7
8 depth_acc = np.empty((19,3), float)
9 i = 0
10 for max_depth in range(1, 20):
11     lobener = tree.DecisionTreeClassifier(criterion="entropy",
12                                           max_depth=max_depth)
13     scores = cross_val_score(lobener, jatibarang_att,
14                             jatibarang_pass, cv=5)
15     depth_acc[i,0] = max_depth
16     depth_acc[i,1] = scores.mean()
17     depth_acc[i,2] = scores.std() * 2
18     i += 1
19
20 print(depth_acc)

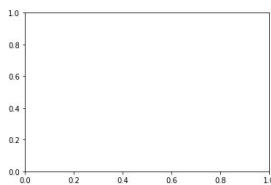
```

```
[[1.0000000e+00 5.79746835e-01 1.01265823e-02]
[2.0000000e+00 5.69620253e-01 2.77327877e-02]
[3.0000000e+00 5.82278481e-01 7.33740595e-02]
[4.0000000e+00 5.77215190e-01 3.03797468e-02]
[5.0000000e+00 5.84810127e-01 9.39106607e-02]
[6.0000000e+00 5.56962025e-01 1.01265823e-01]
[7.0000000e+00 5.92405063e-01 6.28337966e-02]
[8.0000000e+00 6.0000000e+00 6.71721477e-02]
[9.0000000e+00 5.79746835e-01 6.86618226e-02]
[1.0000000e+01 6.10126582e-01 7.74534610e-02]
[1.1000000e+01 6.07536337e-01 5.77304013e-02]
[1.2000000e+01 5.94938709e-01 9.40255035e-02]
[1.3000000e+01 5.82278481e-01 5.16255035e-02]
[1.4000000e+01 5.62278481e-01 5.77304013e-02]
[1.5000000e+01 5.77215190e-01 1.1799796e-02]
[1.6000000e+01 5.82278481e-01 7.67886121e-02]
[1.7000000e+01 5.82278481e-01 7.51007442e-02]
[1.8000000e+01 6.02531546e-01 5.68353021e-02]
[1.9000000e+01 5.92405063e-01 7.05234849e-02]]
```

Gambar 2.85 hasil

12. terakhir yaitu pembuatan grafik untuk pembuatan grafik diambil data dari codingan sebelumnya yang 20 record tadi dengan cara mengimport libray matplotlib.pyplot yang di inisialisasi menjadi bakwankemudian inisialisasi tersebut di eksekusi. untuk lebih jelasnya codingannya seperti gambar dan untuk hasilnya seperti gambar berikut.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar  7 23:12:39 2020
4
5 @author: Liyana
6 """
7
8 import matplotlib.pyplot as plt
9 fig , ax = plt.subplots()
10 ax.errorbar(depth_acc[:,0] , depth_acc[:,1] , yerr=depth_acc
11 [:,2])
12 plt.show()
```

**Gambar 2.86** hasil

2.5.3 Penanganan Error

- skrinsut error

```
Pada: C:\Users\laptop\PycharmProjects\untitled\src\main\java\com\example\untitled\Main.java", line 16, in method
    fig, ax = plt.subplots()
    ^
    No such variable 'plt' available.
  File "C:\Users\laptop\PycharmProjects\untitled\src\main\java\com\example\untitled\Main.java", line 16, in module-
    main
        main();
        ^
        NameError: name 'plt' is not defined
```

Gambar 2.87 Error

- kode error dan jenis errornya .

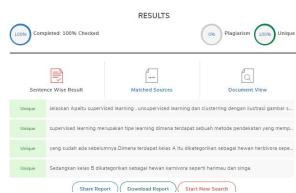
```
jatibarang['pass'] = jatibarangapply(lambda row: 1 if (row['G1']+row['G2']+row['G3'])>= 35 else 0, axis=1)
jatibarang = jatibarang.drop(['G1', 'G2', 'G3'], axis=1)
print(jatibarang.head())
```

NameError: name jatibarangapply is not defined

- Solusi pemecahan masalah

pada codingan tersebut error karena pecelapply tergabung, seharusnya dipisahkan oleh titik

2.5.4 Plagiat



Gambar 2.88 plagiat

2.6 1174043 Irvan Rizkiansyah

2.6.1 Teori

- Jelaskan Apa Itu binari classification dilengkapi ilustrasi gambar sendiri.

Binary Classification merupakan kegiatan yang berguna untuk mengklasifikasikan elemen-elemen dari sebuah himpunan tertentu ke dalam dua kelompok berbeda (memprediksi kelompok mana yang masing-masing dimiliki) berdasarkan dari aturan klasifikasi.



Gambar 2.89 contoh Binary Classification

- Jelaskan Apaitu supervised learning , unsupervised learning dan clustering dengan ilustrasi gambar sendiri.

Supervised learning adalah sebuah metode pendekatan yang mana sudah terdapat data yang dilatih, dan terdapat variabel yang ditargetkan

atau yang menjadi tujuan sehingga tujuan dari pendekatan ini adalah mengelompokan suatu data ke data yang sudah ada. contoh pada jeruk termasuk yang mengandung vitamin c maka jeruk telah di kategorikan kedalam vitamin c. sedangkan salmon mengandung vitamin d yang berarti salmon telah di kategorikan yang mengandung vitamin d untuk lebih jelasnya dapat dilihat pada gambar.



Gambar 2.90 contoh supervised learning

Sedangkan unsupervised learning tidak memiliki data latih, sehingga dari data yang ada, kita mengelompokan data tersebut menjadi dua bagian atau bahkan tiga bagian dan seterusnya. contoh Salmon di kategorikan ke dalam vitamin d untuk lebih jelasnya dapat dilihat pada gambar.



Gambar 2.91 contoh unsupervised learning

clustering merupakan peroses mengklasifikasikan yang berdasarkan suatu parameter dalam penentuannya contoh pada berat vitamin c memiliki berat 500 gr dan vitamin d memiliki berat 1000 gr yang berarti berat dibagi dua parameter yaitu lebih kecil samadengan 500 gram dan lebih besar dari 500 gram contoh pada gambar.



Gambar 2.92 contoh clustering

3. Jelaskan apa itu evaluasi dan akurasi dan disertai ilustrasi contoh dengan gambar sendiri.

evaluasi adalah pengumpulan pengumpulan dan pengamatan dari berbagai macam bukti untuk mengukur dampak efektifitas dari suatu objek, program, atau proses berkaitan dengan spesifikasi atau persyaratan yang telah di tetapkan sebelumnya. sedangkan akurasi itu sendiri merupakan bagian dari evaluasi yang merupakan ketepatan data terhadap suatu objek berdasarkan keriteria tertentu. kita dapat mengevaluasi seberapa baik model bekerja dengan mengukur akurasiinya. ketepatan akan di

definisikan sebagai persentase kasus yang diklasifikasikan dengan benar. hal ini berkaitan dengan confusion matrix pada materi selanjutnya. lebih jelasnya pada gambar berikut:



Gambar 2.93 contoh Evaluasi

- Jelaskan bagaimana cara membuat Confusion Matrix, Buat confusion matrix sendiri.

Dalam pembuatan confusion matrix tentukan parameter atau objek yang akan di evaluasi contoh udang, salmon, tuna buat tabel dengan baris dan kolom berjumlah tiga kemudian tentukan nilai miring pada setiap kolom tersebut disini saya memberi nilai 20 dengan ketentuan setiap baris harus berisi nilai 20 nilai tersebut jika terbagi ke kolom lain maka jumlahnya harus bernilai 20 jika tidak berarti data tersebut tidak akurat. untuk lebih jelanya dapat dilihat pada gambar berikut :

Confusion Matrix				
Actual	Udang	Salmon	Tuna	
Prediction	Udang	120	20	10
Udang	120	20	10	
Salmon	20	10	10	
Tuna	10	10	10	
Total	150	30	30	

Gambar 2.94 contoh Confusion Matrix

- Jelaskan bagaimana K-fold cross validation bekerja dengan gambar ilustrasi contoh buatan sendiri.

K-fold Cross Validation merupakan cara untuk melatih suatu mesin dimana di dalamnya terdapat data set yang dibagi menjadi dua yaitu untuk data testing dan data training contoh 1000 data merupakan data set dan 400 data digunakan untuk data testing kemudian 600 datanya digunakan untuk data training dimana data training tersebut digunakan untuk menentukan nilai bobot yang dimasukan kedalam rumus regresi linier. sedangkan nilai testing akan dijadikan nilai inputan untuk rumus regresi linier. contohnya dapat dilihat pada gambar berikut :



Gambar 2.95 contoh K-fold cross validation

- Jelaskan Apa itu decision tree dengan gambar ilustrasi contoh buatan sendiri.

Decision tree merupakan implementasi dari binari clasification dimana pada pohon keputusan akan terdapat root atau akar dan cabang cabangnya yang nilainya seperti if contoh pada root berisi nilai hidup di air, apakah ikan pada cabang satu bernilai iya dan pada cabang dua bernilai tidak jika nilainya iya berarti hidup di air dan jika tidak maka bukan hidup diair.



Gambar 2.96 contoh decision tree

7. jelaskan apa itu information gain dan entropi dengan gambar ilustrasi buatan sendiri.

informasian gain merupakan informasi atau keriteria dalam pembagian sebuah objek contoh information gain pada ikan yaitu hidup di air, berkoloni, berinsang. untuk lebih jelasnya dapat dilihat pada gambar berikut :



Gambar 2.97 contoh information gain

2.6.2 Scikit-Learn

- 1.

```

1 # load dataset (student mat pakenya)
2 import pandas as pd
3 jeruk = pd.read_csv('D:\student-mat.csv', sep=';')
4 print(len(jeruk))

```

```

In [19]: import pandas as pd
...: jeruk = pd.read_csv('D:\student-mat.csv', sep=';')
...: print(len(jeruk))
395

```

Gambar 2.98 Hasil Percobaan 1

- 2.

```

1 jeruk['pass'] = jeruk.apply(lambda row: 1 if (row['G1']+row['G2'])+row['G3']) >= 35 else 0, axis=1)
2 jeruk = jeruk.drop(['G1', 'G2', 'G3'], axis=1)
3 print(jeruk.head())

```

```
In [20]: jersk = pd.read_csv('jerks.csv')
jersk.groupby(lambda row: 1 if (row['G1']+row['G2']+row['G3'])>=15 else 0, axis=1)
.....
jersk = jersk.drop(['G1', 'G2', 'G3'], axis=1)
.....
print(jersk.head())
school sex age address famsize ... Dale health absences pass
0   GP   F  18    U   GT3 ...     1    1    3    3    0
1   GP   M  22    U   GT3 ...     1    1    3    3    0
2   GP   F  15    U   LT3 ...     2    3    3    3    10    0
3   GP   F  15    U   LT3 ...     1    1    5    2    1
4   GP   F  16    U   GT3 ...     1    2    5    4    0
```

15 rows x 21 columns

Gambar 2.99 Hasil Percobaan 2

3.

```
1 jeruk = pd.get_dummies(jeruk, columns=['sex', 'school', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsupsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic']) #  
Mongkonversi kategori variabel menjadi variabel indikator  
2 jeruk.head()#mengambil baris pertama dari cakue
```

Gambar 2.100 Hasil Percobaan 3

4.

```

1 jeruk = jeruk.sample(frac=1)
2 # split training and testing data
3 jeruk_train = jeruk [:500]
4 jeruk_test = jeruk [500:]
5 jeruk_train_att = jeruk_train.drop(['pass'], axis=1)
6 jeruk_train_pass = jeruk_train ['pass']
7 jeruk_test_att = jeruk_test.drop(['pass'], axis=1)
8 jeruk_test_pass = jeruk_test ['pass']
9 jeruk_att = jeruk.drop(['pass'], axis=1)
10 jeruk_pass = jeruk ['pass']
11 # number of passing students in whole dataset:
12 import numpy as np
13 print("Passing: %d out of %d (%.2f%%)" % (np.sum(jeruk_pass),
14     len(jeruk_pass), 100*float(np.sum(jeruk_pass)) / len(
15         jeruk_pass)))

```

```
In [22]: jeruk = jeruk.sample(frac=1)
jeruk.info()
# output training and testing data
# print(jeruk)
# jeruk.info()
# jeruk_train.att = jeruk_train.drop(['pass'], axis=1)
# jeruk_train_pass = jeruk_train[~jeruk_train['pass']]
# jeruk_test.att = jeruk_test.drop(['pass'], axis=1)
# jeruk_test_pass = jeruk_test['pass'].value_counts()
# print(jeruk_train.att)
# print(jeruk_train_pass)
# print(jeruk_test.att)
# print(jeruk_test_pass)

# import pandas as pd
# print(pd.crosstab(jeruk['student'], jeruk['dataset'],
#                   margins=True))
# print(pd.crosstab(jeruk['student'], jeruk['pass'],
#                   margins=True))

# print(Passing: 3d out of 3d (100%)) # (rp.sum(jeruk_pass), len(jeruk_pass)))
# print(Failing: 166 out of 395 (42%)) # (rp.sum(~jeruk_pass), len(jeruk_pass)))
```

Gambar 2.101 Hasil Percobaan 4

5.

```

1 from sklearn import tree
2 anggur = tree.DecisionTreeClassifier(criterion="entropy",
3                                     max_depth=5)
4 anggur = anggur.fit(jeruk_train_att, jeruk_train_pass)
```

```
[In 28]: from sklearn import tree
...: anggur = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)
...: anggur = anggur.fit(jeruk_train_att, jeruk_train_pass)
```

Gambar 2.102 Hasil Percobaan 5

6.

```

1 import graphviz
2 dot_data = tree.export_graphviz(anggur, out_file=None, label=
3                                 "all", impurity=False, proportion=True,
4                                 feature_names=list(jeruk_train_att), class_names=["fail", "pass"],
5                                 filled=True, rounded=True)
6 graph = graphviz.Source(dot_data)
```

7.

```

1 tree.export_graphviz(anggur, out_file="student-performance.
2                               dot", label="all", impurity=False, proportion=True,
3                               feature_names=list(jeruk_train_att), class_names=["fail", "pass"], filled=True, rounded=True)
```

```
[In 47]: tree.export_graphviz(anggur, out_file="student-performance.dot", label="all",
...: impurity=False, proportion=True,
...: feature_names=list(jeruk_train_att), class_names=["fail",
...: "pass"], filled=True, rounded=True)
```

Gambar 2.103 Hasil Percobaan 7

8.

```
1 anggur.score(jeruk_test_att, jeruk_test_pass)
```

9.

```

1 from sklearn.model_selection import cross_val_score
2 scores = cross_val_score(anggur, jeruk_att, jeruk_pass, cv
3                         =5)
4 # show average score and +/- two standard deviations away
4 #(covering 95% of scores)
5 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
6                                         std() * 2))
```

```
In [52]: from sklearn.model_selection import cross_val_score
... scores = cross_val_score(anggur, jeruk_att, jeruk_pass, cv=5)
... print("Mean score: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
... print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

Accuracy: 0.58 (+/- 0.07)
```

Gambar 2.104 Hasil Percobaan 9

10.

```
1 for max_depth in range(1, 100):
2     anggur = tree.DecisionTreeClassifier(criterion="entropy",
3                                         max_depth=max_depth)
4     scores = cross_val_score(anggur, jeruk_att, jeruk_pass,
5                               cv=5)
5     print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (
#%%%
```

```
In [53]: for max_depth in range(0, 100):
...     anggur = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
...     scores = cross_val_score(anggur, jeruk_att, jeruk_pass, cv=5)
...     print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (max_depth, scores.mean(),
...     scores.std()))
... Max depth: 0, Accuracy: 0.58 (+/- 0.01)
... Max depth: 1, Accuracy: 0.58 (+/- 0.09)
... Max depth: 2, Accuracy: 0.58 (+/- 0.07)
... Max depth: 3, Accuracy: 0.57 (+/- 0.07)
... Max depth: 4, Accuracy: 0.57 (+/- 0.04)
... Max depth: 5, Accuracy: 0.56 (+/- 0.03)
... Max depth: 6, Accuracy: 0.56 (+/- 0.03)
... Max depth: 7, Accuracy: 0.56 (+/- 0.03)
... Max depth: 8, Accuracy: 0.56 (+/- 0.03)
... Max depth: 9, Accuracy: 0.56 (+/- 0.03)
... Max depth: 10, Accuracy: 0.56 (+/- 0.03)
... Max depth: 11, Accuracy: 0.56 (+/- 0.03)
... Max depth: 12, Accuracy: 0.56 (+/- 0.03)
... Max depth: 13, Accuracy: 0.56 (+/- 0.03)
... Max depth: 14, Accuracy: 0.56 (+/- 0.03)
... Max depth: 15, Accuracy: 0.56 (+/- 0.03)
... Max depth: 16, Accuracy: 0.56 (+/- 0.03)
... Max depth: 17, Accuracy: 0.56 (+/- 0.03)
... Max depth: 18, Accuracy: 0.56 (+/- 0.03)
... Max depth: 19, Accuracy: 0.56 (+/- 0.03)
... Max depth: 20, Accuracy: 0.56 (+/- 0.03)
... Max depth: 21, Accuracy: 0.56 (+/- 0.03)
... Max depth: 22, Accuracy: 0.56 (+/- 0.03)
... Max depth: 23, Accuracy: 0.56 (+/- 0.03)
... Max depth: 24, Accuracy: 0.56 (+/- 0.03)
... Max depth: 25, Accuracy: 0.56 (+/- 0.03)
... Max depth: 26, Accuracy: 0.56 (+/- 0.03)
... Max depth: 27, Accuracy: 0.56 (+/- 0.03)
... Max depth: 28, Accuracy: 0.56 (+/- 0.03)
... Max depth: 29, Accuracy: 0.56 (+/- 0.03)
... Max depth: 30, Accuracy: 0.56 (+/- 0.03)
... Max depth: 31, Accuracy: 0.56 (+/- 0.03)
... Max depth: 32, Accuracy: 0.56 (+/- 0.03)
... Max depth: 33, Accuracy: 0.56 (+/- 0.03)
... Max depth: 34, Accuracy: 0.56 (+/- 0.03)
... Max depth: 35, Accuracy: 0.56 (+/- 0.03)
... Max depth: 36, Accuracy: 0.56 (+/- 0.03)
... Max depth: 37, Accuracy: 0.56 (+/- 0.03)
... Max depth: 38, Accuracy: 0.56 (+/- 0.03)
... Max depth: 39, Accuracy: 0.56 (+/- 0.03)
... Max depth: 40, Accuracy: 0.56 (+/- 0.03)
... Max depth: 41, Accuracy: 0.56 (+/- 0.03)
... Max depth: 42, Accuracy: 0.56 (+/- 0.03)
... Max depth: 43, Accuracy: 0.56 (+/- 0.03)
... Max depth: 44, Accuracy: 0.56 (+/- 0.03)
... Max depth: 45, Accuracy: 0.56 (+/- 0.03)
... Max depth: 46, Accuracy: 0.56 (+/- 0.03)
... Max depth: 47, Accuracy: 0.56 (+/- 0.03)
... Max depth: 48, Accuracy: 0.56 (+/- 0.03)
... Max depth: 49, Accuracy: 0.56 (+/- 0.03)
... Max depth: 50, Accuracy: 0.56 (+/- 0.03)
... Max depth: 51, Accuracy: 0.56 (+/- 0.03)
... Max depth: 52, Accuracy: 0.56 (+/- 0.03)
... Max depth: 53, Accuracy: 0.56 (+/- 0.03)
... Max depth: 54, Accuracy: 0.56 (+/- 0.03)
... Max depth: 55, Accuracy: 0.56 (+/- 0.03)
... Max depth: 56, Accuracy: 0.56 (+/- 0.03)
... Max depth: 57, Accuracy: 0.56 (+/- 0.03)
... Max depth: 58, Accuracy: 0.56 (+/- 0.03)
... Max depth: 59, Accuracy: 0.56 (+/- 0.03)
... Max depth: 60, Accuracy: 0.56 (+/- 0.03)
... Max depth: 61, Accuracy: 0.56 (+/- 0.03)
... Max depth: 62, Accuracy: 0.56 (+/- 0.03)
... Max depth: 63, Accuracy: 0.56 (+/- 0.03)
... Max depth: 64, Accuracy: 0.56 (+/- 0.03)
... Max depth: 65, Accuracy: 0.56 (+/- 0.03)
... Max depth: 66, Accuracy: 0.56 (+/- 0.03)
... Max depth: 67, Accuracy: 0.56 (+/- 0.03)
... Max depth: 68, Accuracy: 0.56 (+/- 0.03)
... Max depth: 69, Accuracy: 0.56 (+/- 0.03)
... Max depth: 70, Accuracy: 0.56 (+/- 0.03)
... Max depth: 71, Accuracy: 0.56 (+/- 0.03)
... Max depth: 72, Accuracy: 0.56 (+/- 0.03)
... Max depth: 73, Accuracy: 0.56 (+/- 0.03)
... Max depth: 74, Accuracy: 0.56 (+/- 0.03)
... Max depth: 75, Accuracy: 0.56 (+/- 0.03)
... Max depth: 76, Accuracy: 0.56 (+/- 0.03)
... Max depth: 77, Accuracy: 0.56 (+/- 0.03)
... Max depth: 78, Accuracy: 0.56 (+/- 0.03)
... Max depth: 79, Accuracy: 0.56 (+/- 0.03)
... Max depth: 80, Accuracy: 0.56 (+/- 0.03)
... Max depth: 81, Accuracy: 0.56 (+/- 0.03)
... Max depth: 82, Accuracy: 0.56 (+/- 0.03)
... Max depth: 83, Accuracy: 0.56 (+/- 0.03)
... Max depth: 84, Accuracy: 0.56 (+/- 0.03)
... Max depth: 85, Accuracy: 0.56 (+/- 0.03)
... Max depth: 86, Accuracy: 0.56 (+/- 0.03)
... Max depth: 87, Accuracy: 0.56 (+/- 0.03)
... Max depth: 88, Accuracy: 0.56 (+/- 0.03)
... Max depth: 89, Accuracy: 0.56 (+/- 0.03)
... Max depth: 90, Accuracy: 0.56 (+/- 0.03)
... Max depth: 91, Accuracy: 0.56 (+/- 0.03)
... Max depth: 92, Accuracy: 0.56 (+/- 0.03)
... Max depth: 93, Accuracy: 0.56 (+/- 0.03)
... Max depth: 94, Accuracy: 0.56 (+/- 0.03)
... Max depth: 95, Accuracy: 0.56 (+/- 0.03)
... Max depth: 96, Accuracy: 0.56 (+/- 0.03)
... Max depth: 97, Accuracy: 0.56 (+/- 0.03)
... Max depth: 98, Accuracy: 0.56 (+/- 0.03)
... Max depth: 99, Accuracy: 0.56 (+/- 0.03)
```

Gambar 2.105 Hasil Percobaan 10

11.

```
1 depth_acc = np.empty((19,3), float)
2 i = 0
3 for max_depth in range(1, 20):
4     anggur = tree.DecisionTreeClassifier(criterion="entropy",
5                                         max_depth=max_depth)
5     scores = cross_val_score(anggur, jeruk_att, jeruk_pass,
6                               cv=5)
6     depth_acc[i,0] = max_depth
7     depth_acc[i,1] = scores.mean()
8     depth_acc[i,2] = scores.std() * 2
9     i += 1
10
11 print(depth_acc)
```

Gambar 2.106 Hasil Percobaan 11

12.

```
1 import matplotlib.pyplot as plt
2 fig, ax = plt.subplots()
3 ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc
4 [:,2])
5 plt.show()
```

```
In [55]: import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2])
plt.show()
```

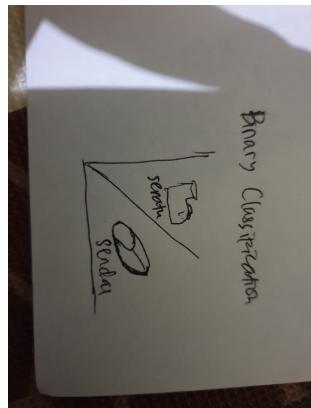
Gambar 2.107 Hasil Percobaan 12

2.7 1174040 - Hagan Rowlenstino A. S.

2.7.1 Teori

1. Jelaskan Apa Itu binari classification dilengkapi ilustrasi gambar sendiri.

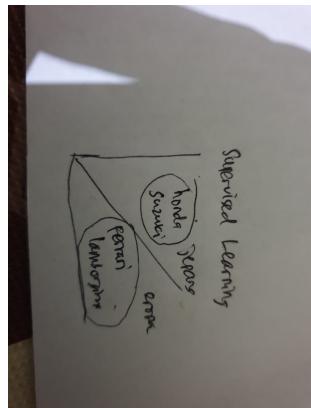
Binary Classification adalah sebuah aksi dimana dilakukannya klasifikasi dari sebuah kumpulan objek tertentu ke dalam dua buah kelompok yang berbeda berdasarkan beberapa fitur atau sifat-sifat.



Gambar 2.108 Binary Classification

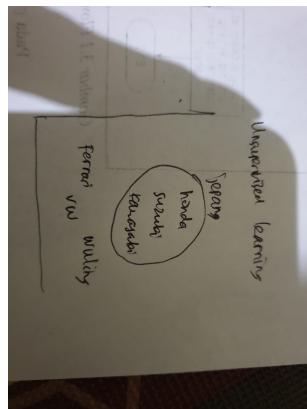
2. Jelaskan Apaitu supervised learning , unsupervised learning dan clustering dengan ilustrasi gambar sendiri.

Supervised learning adalah sebuah cara untuk mengklasifikasikan kumpulan objek yang fitur ataupun sifat-sifatnya dari kelas nya sudah di tentukan sebelumnya. contoh pada merk mobil honda yang merupakan buatan jepang dan ferarri merupakan buatan eropa



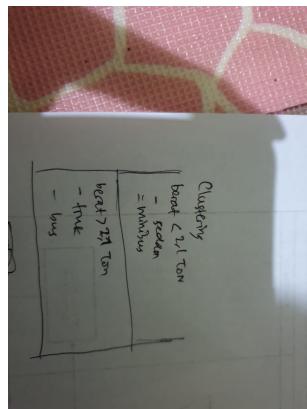
Gambar 2.109 Supervised

Unsupervised learning merupakan teknik pengklasifikasian tanpa perlu di supervised sebelumnya, dimana model tersebut yang akan bekerja sendiri untuk menemukan infomasi yang terkait. seperti pada contoh di gambar.



Gambar 2.110 Unsupervised

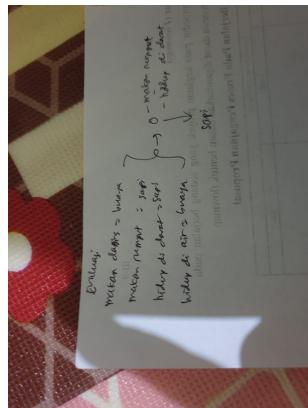
clustering merupakan metode pengelompokan data dengan membagi ke dalam beberapa kelompok. disini saya memberi contoh dengan kriteria kendaraan yang lebih dari 2,1 ton dan kurang dari 2,1 ton.



Gambar 2.111 clustering

3. Jelaskan apa itu evaluasi dan akurasi dan disertai ilustrasi contoh dengan gambar sendiri.

Evaluasi adalah proses yang sistematis yang ditujukan untuk menentukan ataupun membuat keputusan dengan memeriksa pernyataan-pernyataan yang telah ada sebelumnya. Akurasi adalah tingkat ketepatan dari sebuah data yang telah dihasilkan dari evaluasi yang telah dilakukan sebelumnya.

**Gambar 2.112** Evaluasi

4. Jelaskan bagaimana cara membuat Confusion Matrix, Buat confusion matrix sendiri.

Untuk membuat confusion matrix kita perlu menentukan objek yang akan dievaluasi terlebih dahulu, setelah itu tentukan nilai miring pada setiap kolom objek tersebut lalu setiap baris dan jika lainnya di bagi baris dan kolom nya, harus bernilai sesuai nilai yang telah di tetapkan sebelumnya.

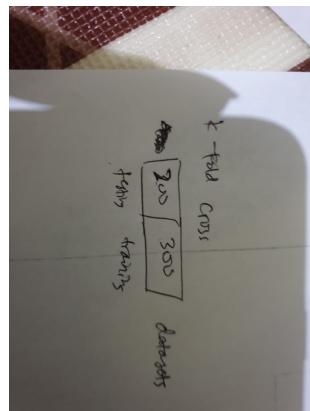
Confusion Matrix
Baris : Prediksi
Kolom : Kategori
Annotations:
1. Baris 1: Benar (1) / Tidak Benar (0)
2. Kolom 1: Benar (1) / Tidak Benar (0)
3. Diagonal: Benar-Benar (1), Tidak-Tidak (0)
4. Off-diagonal: Benar-Tidak (1), Tidak-Benar (0)

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Gambar 2.113 Confussion Matrix

5. Jelaskan bagaimana K-fold cross validation bekerja dengan gambar ilustrasi contoh buatan sendiri.

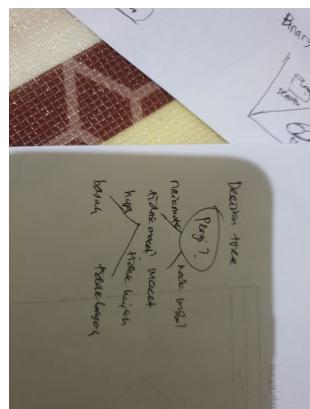
Melatih mesin dengan membagi data set yang akan diolah menjadi dua buah yaitu data testing dan training. sebagai contoh ada 500 data pada dataset, maka 200 sebagai data testing dan 300 akan menjadi data training.



Gambar 2.114 K-Fold

6. Jelaskan Apa itu decision tree dengan gambar ilustrasi contoh buatan sendiri.

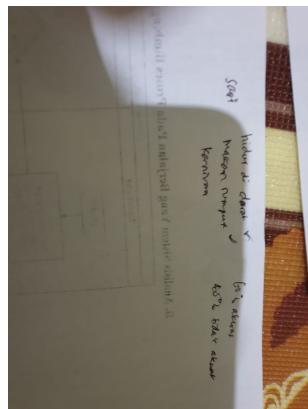
Desicion tree adalah sebuah model prediksi yang menggunakan struktur yang menyerupai pohon ataupun sebuah hirarki



Gambar 2.115 Decision Tree

7. jelaskan apa itu information gain dan entropi dengan gambar ilustrasi buatan sendiri.

information Gain mengukur berapa banyak informasi yang sebuah fitur berikan kepada kita tentang kelas tersebut. sedangkan entropi menentukan bagaimana decision tree memilah data nya.



Gambar 2.116 Information Gain

2.7.2 scikit-learn

1. baris pertama yaitu mengimport library pandas. pada baris kedua ada fungsi read_csv untuk membaca file csv serta datanya dipisah dengan tanda titik koma lalu dimasukkan kedalam variable bernama pisang. fungsi pada baris terakhir yaitu untuk mengetahui panjang data / banyak data yang ada

```
1 import pandas as pd
2 pisang = pd.read_csv('student-mat.csv', sep=';')
3 print(len(pisang))
```

```
In [42]: import pandas as pd
....: pisang = pd.read_csv('student-mat.csv', sep=';')
....: print(len(pisang))
395
```

Gambar 2.117 No 1

2. Disini digunakan fungsi untuk menunjukkan lulus atau gagal dimana jika lulus ditandai dengan angka 1 dan gagal dengan angka 0. data yang diperlukan yaitu data dari csv yang sebelumnya karena itu masih digunakan variable pisang. yang selanjutnya akan dieksekusi.

```
1 pisang[ 'pass' ] = pisang.apply(lambda row: 1 if (row[ 'G1' ]+row
[ 'G2' ]+row[ 'G3' ]) >= 35 else 0, axis=1)
2 pisang = pisang.drop([ 'G1' , 'G2' , 'G3' ], axis=1)
3 print( pisang .head() )
```

	school	sex	age	address	famsize	... Pstatus	Mjob	Fjob	reason	guardian	schoolsup	famsup	paid	activities	nursery	higher	internet	romantic	pass
0	GP	F	15	U	GT3	...	Dalc	Halic	health	absences	
1	GP	F	17	U	GT3	...	1	1	3	6	0	0	0	0	0	0	0	0	
2	GP	F	15	U	LE3	...	2	3	3	10	0	0	0	0	0	0	0	0	
3	GP	F	15	U	GT3	...	1	1	1	5	2	1	1	1	1	1	1	1	
4	GP	F	16	U	GT3	...	1	2	5	5	0	0	0	0	0	0	0	0	

[5 rows x 31 columns]

Gambar 2.118 No 2

3. Disini yaitu mengkonversi kategori variable menjadi variable indikator

```

1 pisang = pd.get_dummies(pisang, columns=['sex', 'school', '',
2   address, 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', '',
3   guardian, 'schoolsup', 'famsup', 'paid', 'activities', '',
4   nursery, 'higher', 'internet', 'romantic']) #  

  Mongkonversi kategori variabel menjadi variabel indikator
2 pisang.head()

```

	age	Medu	Fedu	...	internet_yes	romantic_no	romantic_yes
0	18	4	4	...	0	1	0
1	17	3	3	...	1	1	0
2	15	1	1	...	1	1	0
3	15	2	2	...	1	0	1
4	16	3	2	...	0	1	0

[5 rows x 57 columns]

Gambar 2.119 No 3

4. bagian ini berfungsi untuk menentukan data training dan data testing dari dataset csv yang telah dimasukkan di dalam variable pisang tadi. lalu mengimport library numpy untuk operasi vektor serta matrix karena data diatas berupa matrix.

```

1 pisang = pisang.sample(frac=1)
2 # split training and testing data
3 pisang_train = pisang[:500]
4 pisang_test = pisang[500:]
5 pisang_train_att = pisang_train.drop(['pass'], axis=1)
6 pisang_train_pass = pisang_train['pass']
7 pisang_test_att = pisang_test.drop(['pass'], axis=1)
8 pisang_test_pass = pisang_test['pass']
9 pisang_att = pisang.drop(['pass'], axis=1)
10 pisang_pass = pisang['pass']
11 # number of passing students in whole dataset:
12 import numpy as np
13 print("Passing: %d out of %d (%.2f%%)" % (np.sum(pisang_pass),
14   , len(pisang_pass), 100*float(np.sum(pisang_pass)) / len(
15   pisang_pass)))

```

```

In [46]: pisang = pisang.sample(frac=1)
...: # split training and testing data
...: pisang_train = pisang[:500]
...: pisang_test = pisang[500:]
...: pisang_train_att = pisang_train.drop(['pass'], axis=1)
...: pisang_train_pass = pisang_train['pass']
...: pisang_test_att = pisang_test.drop(['pass'], axis=1)
...: pisang_test_pass = pisang_test['pass']
...: pisang_att = pisang.drop(['pass'], axis=1)
...: pisang_pass = pisang['pass']
...: # number of passing students in whole dataset:
...: import numpy as np
...: print("Passing: %d out of %d (%.2f%%)" % (np.sum(pisang_pass),
...: , len(pisang_pass), 100*float(np.sum(pisang_pass)) / len(pisang_pass)))
Passing: 166 out of 395 (42.05%)

```

Gambar 2.120 No 4

5. pada baris pertam kita mengimport tree dari library sklearn yang berfungsi untuk membuat desicion tree

```
1 from sklearn import tree
2 apel = tree.DecisionTreeClassifier(criterion="entropy",
3                                   max_depth=5)
4 apel = apel.fit(pisang_train_att, pisang_train_pass)
5 print(apel)
```

Gambar 2.121 No 5

- ## 6. mengimport library graphviz

```
1 import graphviz
2 dot_data = tree.export_graphviz(apel, out_file=None, label="all",
3                                 impurity=False, proportion=True, feature_names=list(pisang_train.att), class_names=["fail", "pass"], filled=True, rounded=True)
4 graph = graphviz.Source(dot_data)
5 %%
```

7. untuk menyimpan data dari tree dan menarik data langsung dari desicion tree

1 #%

```
In [54]: tree.export_graphviz(apel, out_file="student-performance.dot", label="all",
                             impurity=False, proportion=True,
                             ...,
                             feature_names=list(pisang_train_att),
                             class_names=["fail", "pass"], filled=True, rounded=True)
```

Gambar 2.122 No 7

8. untuk mencari ketepatan data yang telah diolah

9. untuk menghitung akurasi ketepatan data

```
1 # show average score and +/- two standard deviations away
2 #(covering 95% of scores)
3 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
4 std() * 2))
5 #%%
```

```
In [54]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(apel, pisang_att, pisang_pass, cv=5)
...: print("Meaningful 95% of scores")
...: print((scores.mean() - 30.21) / (-30.21) * (scores.mean(), scores.std() * 2))
Accuracy: 0.58 (+/- 0.06)
```

Gambar 2.123 No 9

10. membuat variable baru dengan nilai tree di dalamnya dengan bentuk akurasi yang spesifik

```
1 scores = cross_val_score(apel, pisang_att, pisang_pass,
2                         cv=5)
3 print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (
4     max_depth, scores.mean(), scores.std() * 2))
#%%

```

```
Max depth: 88, Accuracy: 0.61 (+/- 0.05)
Max depth: 89, Accuracy: 0.61 (+/- 0.06)
Max depth: 90, Accuracy: 0.60 (+/- 0.07)
Max depth: 91, Accuracy: 0.62 (+/- 0.07)
Max depth: 92, Accuracy: 0.61 (+/- 0.09)
Max depth: 93, Accuracy: 0.61 (+/- 0.05)
Max depth: 94, Accuracy: 0.60 (+/- 0.08)
Max depth: 95, Accuracy: 0.60 (+/- 0.05)
Max depth: 96, Accuracy: 0.61 (+/- 0.08)
Max depth: 97, Accuracy: 0.62 (+/- 0.06)
Max depth: 98, Accuracy: 0.61 (+/- 0.06)
Max depth: 99, Accuracy: 0.61 (+/- 0.06)
```

Gambar 2.124 No 10

11. menentukan rank batasannya yaitu 1 sampai 20 dari nilai tree tadi yang digunakan untuk menentukan nilai untuk grafik

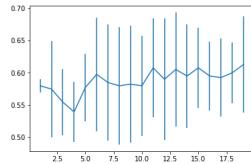
```
1 for max_depth in range(1, 20):
2     apel = tree.DecisionTreeClassifier(criterion="entropy",
3                                         max_depth=max_depth)
4     scores = cross_val_score(apel, pisang_att, pisang_pass,
5                               cv=5)
6     depth_acc[i, 0] = max_depth
7     depth_acc[i, 1] = scores.mean()
8     depth_acc[i, 2] = scores.std() * 2
9     i += 1
10 print(depth_acc)
#%
```

```
[{1.00000000e+00 5.79746835e-01 1.01265823e-02]
[2.00000000e+00 5.74683544e-01 7.44148783e-02]
[3.00000000e+00 5.54430580e-01 5.16356466e-02]
[4.00000000e+00 5.39240506e-01 4.69559304e-02]
[5.00000000e+00 5.24050532e-01 4.22762942e-02]
[6.00000000e+00 5.07468354e-01 3.82814976e-02]
[7.00000000e+00 5.04810127e-01 3.97221747e-02]
[8.00000000e+00 5.07946835e-01 9.11392405e-02]
[9.00000000e+00 5.82278481e-01 9.05749954e-02]
[1.00000000e+01 5.79746835e-01 7.74534610e-02]
[1.10000000e+01 5.80574683e-01 7.68042512e-02]
[1.20000000e+01 5.80574683e-01 7.61826597e-02]
[1.30000000e+01 6.05063201e-01 8.82814976e-02]
[1.40000000e+01 5.04936709e-01 8.00576623e-02]
[1.50000000e+01 6.0750494937e-01 6.20123986e-02]
[1.60000000e+01 5.04936709e-01 5.31042455e-02]
[1.70000000e+01 5.92405636e-01 5.07594937e-02]
[1.80000000e+01 6.00000000e-01 4.69559304e-02]
[1.90000000e+01 6.12658228e-01 7.44148783e-02]]
```

Gambar 2.125 No 11

12. mengimport matplotlib.pyplot yang akan digunakan untuk membuat grafik

```
1 ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc
              [:,2])
2 plt.show()
3 #%%
```

**Gambar 2.126** No 12

2.7.3 Penanganan Error

1. Screenshot

```
KeyError: "[('sex', 'school', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic') not in index"]
```

Gambar 2.127 Screenshot Error

2. Kode dan Jenis Error Jenis error adalah Key Error

```
1 pisang = pd.get_dummies(pisang, columns=['sex', 'school', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic']) #
Mongkonversi kategori variabel menjadi variabel indikator
2 kuda.head()
```

3. Solusi Penanganan Error Mengganti kuda dengan pisang

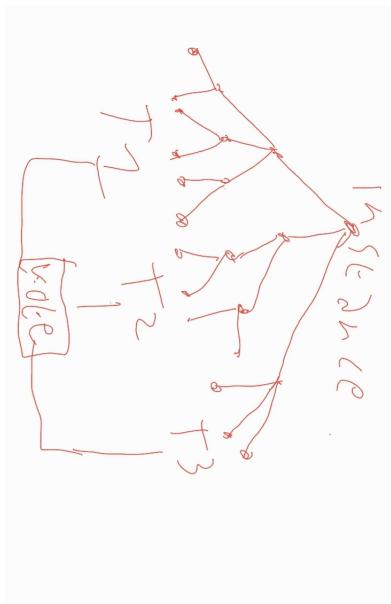
BAB 3

CHAPTER 3

3.0.1 Soal Teori

1. Jelaskan apa itu random forest, sertakan gambar ilustrasi buatan sendiri.

Random Forest atau hutan acak yaitu kumpulan dari pohon keputusan yang difungsikan untuk membaca objek tertentu sesuai dengan yang telah di sepakati untuk di baca pada sistem yang menggunakan pengkondisian seperti Artificial Intelligence. Pohon-pohon keputusan tersebut akan memunculkan hasil yang akan disimpulkan oleh random forest. Pembagian jumlah data yang dimasukan kedalam decision tree pada random forest akan di bagi sama rata sesuai dengan ketentuan tertentu yang disepakati saat sebelum membuat sebuah sistem tersebut.



Gambar 3.1 Contoh Confusion Matrix

2. Jelaskan cara membaca dataset khusus dan artikan makna setiap file dan isi field masing masing file. Yang harus dilakukan untuk membaca dataset adalah mendownload dataset yang sudah disediakan, lalu dibuka menggunakan IDE khusus dari python seperti Spyder untuk mengetahui isi dari dataset yang sudah didownload. Tergantung dari kebutuhan, file dapat berbentuk txt ataupun csv yang sudah sering digunakan karena bentuk datanya seperti tabel dan mudah untuk digunakan. Di dalam file akan mengandung class dari field atau kumpulan data hasil penilaian yang sudah dilakukan. Total datanya sendiri bisa sampai ribuan walaupun hanyalah kategori dan status seperti 0 dan 1.
3. Jelaskan apa itu Cross Validation. Cross Validation merupakan metode untuk mengevaluasi hasil dari sebuah penilaian yang telah digunakan dengan cara membagi dua bagian dari dataset menjadi data training dan data testing. Lalu data akan diolah sehingga muncul tingkat akurasi dari metode yang digunakan contoh pada metode random forest dataset nya dibagi menjadi dua menjadi data training dan data testing kemudian data tersebut diolah oleh mesin untuk melihat tingkat akurasinya maka akan muncul misalkan akurasi kebenaran sebesar 44 % begitu pula dengan menggunakan metode-metode yang lain seperti decision tree dan SVM.
4. Jelaskan apa arti score 44 % pada random forest, 27 % pada decision tree dan 29 % dari SVM. Maksud dari score 44 % tersebut yaitu nilai ketepatan atau kebenaran dari sebuah parameter, tingkat ketepatan

tersebut berpengaruh pada bagaimana mesin tersebut bisa menyatakan jenis burung tersebut dengan akurasi kebenaran 44 %. sedangkan pada metode decision tree yaitu 27 % yang berarti menunjukan bahwa tingkat akurasi ketepatan mesin jika mengerjakan sesuatu atau menyatakan keputusan dengan metode decision tree maka nilai kebenarannya bernilai 27 %. sedangkan dengan menggunakan metode SVM menunjukan hasil 29 % yang berarti nilai ketepatan atau kebenaran dalam memecahkan masalah menggunakan metode SVM ini sebesar 29 % . maka dari itu dapat di simpulkan bahwa dengan menggunakan metode random forest mesin dapat memecahkan masalah lebih akurat dibandingkan dengan menggunakan decision tree dan SVM.

5. Jelaskan bagaimana cara membaca confusion matriks dan contohnya memakai gambar atau ilustrasi sendiri. Cara untuk membaca confusion matrix adalah dengan cara memasukan parameter nilai yang ada pada datasets. Contohnya seperti pada dataset terdapat class yang disandingkan dengan nama burung untuk di normalisasi maka akan menunjukan nilai matrix yang mendekati nilai benar dalam bentuk angka misalkan 0,5 0,2 dan seterusnya mendekati nilai satu. di karenakan susahnya membaca nilai angka maka sering di ubah menjadi bentuk grafik. Untuk nilai - nilai yang ada, adalah sebagai berikut :

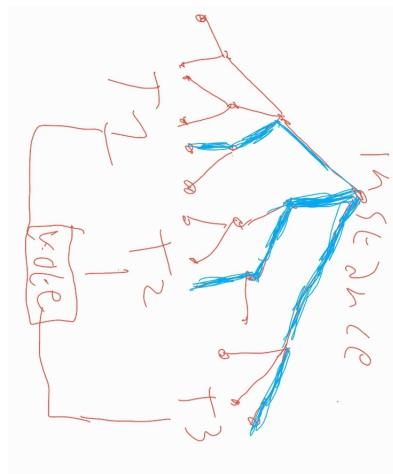
- True Positive : Data positif yang terdeteksi memiliki hasil benar
- False Positive : Data Positif yang terdeteksi memiliki hasil salah
- True Negative : Data negatif yang terdeteksi memiliki hasil benar
- False Negative : Data negatif yang terdeteksi memiliki hasil salah

		C ₁	C ₂
		Predict	Predict
Actual	C ₁	TP	FN
	C ₂	FP	TN

Gambar 3.2 Contoh Confusion Matrix

6. Jelaskan apa itu voting pada random forest disertai dengan ilustrasi gambar sendiri.

Voting merupakan data hasil dari decision tree yang terdapat pada random forest. Dimana hasil data tersebut di gunakan sebagai acuan untuk hasil dari random forest. sebagai contoh misalkan pada satu random forest terdapat enam decision tree untuk menentukan jenis pekerjaan orang, pada decision tree ke satu menyimpulkan bahwa pekerjaanya yaitu dosen , pada decision tree ke dua yaitu dosen kemudian pada decision tiga dosen , pada decision tree ke empat yaitu pekerja kantoran, pada decision tree ke lima yaitu pekerja kantoran dan pada decision tree ke enam yaitu dosen. maka pada random forest dapat menyimpulkan hasilnya yaitu dosen.



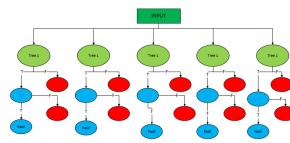
Gambar 3.3 Contoh Random Forest yang sudah Divote

3.1 Faisal Najib Abdullah / 1174042

3.1.1 Teori

1. Jelaskan apa itu random forest, sertakan gambar ilustrasi buatan sendiri.

Random Forest atau hutan acak yaitu kumpulan dari pohon-pohon keputusan yang digunakan untuk membaca objek tertentu yang telah di sepakati untuk di baca dalam AI. pohon-pohon keputusan tersebut akan memunculkan hasil-hasil yang akan disimpulkan oleh random forest. pembagian jumlah data yang dimasukan kedalam decision tree pada random forest akan di bagi sama rata sesuai codingan atau ketentuan tertentu yang di sepakati. misalkan data yang akan digunakan sebanyak 314 jika dalam satu decision tree di putuskan untuk memiliki 50 data maka pada satu random forest akan terdapat enam atau tujuh decision tree.



Gambar 3.4 contoh binari calssification

2. Jelaskan cara membaca dataset khusus dan artikan makna setiap file dan isi field masing masing file. langkah pertama download terlebih dahulu dataset nya kemudian buka menggunakan spyder bawaan anaconda untuk mengetahui isi dari dataset tersebut. biasanya data tersebut berisi

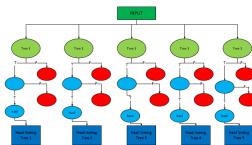
databerekstensi .txt yang di dalamnya terdapat class dari field atau data data yang ada data tersebut. contoh pada data burung ada field index dan angka, index biasanya berisi angka, angka angka tersebut memiliki makna yaitu pengganti nama atau jenis dari burung tersebut sedangkan pada field yang berisi nilai 0 dan 1 berarti menyatakan atau maknanya yaitu memberikan nilai ya dan tidak nilai tersebut di ubah menjadi angka nol dan satu karna data pada field tersebut harus berisi nilai boolean atau pilihan ya dan tidak di karenakan komputer susah membaca nilai dan tidak maka di ubahlah menjadi 0 dan 1 dengan 0 bernilai tidak dan 1 bernilai ya.

3. Jelaskan apa itu Cross Validation. Cross Validation merupakan cara untuk mengevaluasi hasil dari sebuah metode yang telah digunakan dengan cara membagi dua bagian dari dataset menjadi data training dan data testing kemudian data tersebut diolah hingga muncul tingkat akurasi dari metode yang digunakan contoh pada metode random forest dataset nya di bagi menjadi dua menjadi data training dan data testing kemudian data tersebut di olah oleh mesin untuk melihat tingkat akurasinya maka akan muncul misalkan akurasi kebenaran sebesar 44 % begitu pula dengan menggunakan metode-metode yang lain seperti decision tree dan SVM.
4. Jelaskan apa arti score 44 % pada random forest, 27 % pada decision tree dan 29 % dari SVM. maksud dari score 44 % tersebut yaitu nilai ketepatan atau kebenaran atau bisa disebut hasil dari random forest misalkan dengan metode random forest mesin membaca objek burung, mesin tersebut bisa menyatakan jenis burung tersebut dengan akurasi kebenaran 44 %. sedangkan pada metode decision tree yaitu 27 % yang berarti menunjukan bahwa tingkat akurasi ketepatan mesin jika mengerjakan sesuatu atau menyatakan keputusan dengan metode decision tree maka nilai kebenarannya bernilai 27 %. sedangkan dengan menggunakan metode SVM menunjukan hasil 29 % yang berarti nilai ketepatan atau kebenaran dalam memecahkan masalah menggunakan metode SVM ini sebesar 29 % . maka dari itu dapat di simpulkan bahwa dengan menggunakan metode random forest mesin dapat memecahkan masalah lebih akurat dibandingkan dengan menggunakan decision tree dan SVM.
5. Jelaskan bagaimana cara membaca confusion matriks dan contohnya memakai gambar atau ilustrasi sendiri. cara membaca confusio matrix dengan cara memasukan para meter nilai yang ada pada datasets contoh pada dataset terdapat class yang disandingkan dengan nama burung untuk di normalisasi maka akan menunjukan nilai matrix yang mendekati nilai benar dalam bentuk angka misalkan 0,5 0,2 dan seterusnya mendekati nilai satu. di karenakan susahnya membaca nilai angka maka sering di ubah menjadi bentuk grafik.
6. Jelaskan apa itu voting pada random forest disertai dengan ilustrasi gambar sendiri.

	Bueng 1	Bueng 2	Bueng 3	Bueng 4	Bueng 5	Bueng 6	Bueng 7	Bueng 8	Bueng 9	Bueng 10	Bueng 11
Bueng 1	1	0	0	0	0	0	0	0	0	0	0
Bueng 2	0	1	0	0	0	0	0	0	0	0	0
Bueng 3	0	0	1	0	0	0	0	0	0	0	0
Bueng 4	0	0	0	1	0	0	0	0	0	0	0
Bueng 5	0	0	0	0	1	0	0	0	0	0	0
Bueng 6	0	0	0	0	0	1	0	0	0	0	0
Bueng 7	0	0	0	0	0	0	1	0	0	0	0
Bueng 8	0	0	0	0	0	0	0	1	0	0	0
Bueng 9	0	0	0	0	0	0	0	0	1	0	0
Bueng 10	0	0	0	0	0	0	0	0	0	1	0
Bueng 11	0	0	0	0	0	0	0	0	0	0	1

Gambar 3.5 contoh binari calssification

Voting merupakan data hasil dari decision tree yang terdapat pada random forest. Dimana hasil data tersebut di gunakan sebagai acuan untuk hasil dari random forest. sebagai contoh misalkan pada satu random forest terdapat enam decision tree untuk menentukan jenis pekerjaan orang, pada decision tree ke satu menyimpulkan bahwa pekerjaanya yaitu dosen , pada decision tree ke dua yaitu dosen kemudian pada decision tiga dosen , pada decision tree ke empat yaitu pekerja kantoran, pada decision tree ke lima yaitu pekerja kantoran dan pada decision tree ke enam yaitu dosen. maka pada random forest dapat menyimpulkan hasilnya yaitu dosen.

**Gambar 3.6** contoh binari calssification

3.1.2 Praktikum

1. pandas

pada baris ke satu yaitu perintah mengimport library padas pada python atau anaconda kemudian di inisialisasikan menjadi kue. selanjutnya pada baris ke 3 terdapat nama variabel yaitu nama_kue_tradisional = yang di dalamnya terdapat tiga nama field yakni Name Kue, harga satuan dan terbilang kemudian pada baris ke tujuh terdapat variabel baru bernama Data_kue = kemudian didalamnya mendeskripsikan kue berdasarkan tipe DataFrame yang berisi variabel nama_kue_tradisional selanjutnya data tersebut di cetak pada console dengan perintah (Data_kue).

```

1 import pandas as kue
2 nama_kue = { 'Nama Kue' : [ 'Cuhcur' , 'Putri Noong' , 'Bugis' , 'Papais' , 'Ali-Ali' ] ,
3   'Harga Satuan' : [ 2000 , 5000 , 1500 , 2500 , 1000 ] , 'Terbilang' : [
4     'Dua Ribu Rupiah' , 'Lima Ribu Rupiah' ,
5     'Seribu Lima Ratus Rupiah' , 'Dua Ribu Limaratus Rupiah' , 'Seribu Rupiah' ] }
6 Data_kue = kue.DataFrame(nama_kue)
7 print(Data_kue)
  
```

	Nama Kue	Harga Satuan	Terbilang
0	Cucur	2000	Dua Ribu Rupiah
1	Putri Noong	5000	Lima Ribu Rupiah
2	Bugis	1500	Seribu Lima Ratus Rupiah
3	Papais	2500	Dua Ribu Limaratus Rupiah
4	Ali-Ali	1000	Seribu Rupiah

Process finished with exit code 0

Gambar 3.7 hasil

2. numpy

Arti tiap baris codingan pada aplikasi sederhana numpy adalah sebagai berikut : pada baris ke satu yaitu mengimport numpy yang di inisialisasi menjadi np kemudian pada baris ke tiga dibut variabel ali yang berisi numpy bertipekan arrange 6 yang berarti berisi nilai array dari 0 sampai 5 kemudian pada baris ke empat di cetak hasilnya dengan memasukan perintah print (ali) selanjutnya yaitu membuat nilai array tiga dimensi pada baris ke tujuh dengan cara membuat variabel botak yang berisi rank nilainya kemudian dimensinya yaitu 4 3 3 kemudian variabel tersebut di print. selanjutnya pada baris ke 12 dibuat variabel nilai_array_1 dengan isian nilai array 1 2 3 4 kemudian pada baris ke 13 di buat variabe nilai_array_2 dengan nilai array 20 30 40 dan 50 selanjutnya pada baris ke 14 dibut nilai variabel Nilai_array_3 dengan rank 4 yang berarti berisi nilai dari 0 sampai 3 setelah itu di buat variabel hasil dimana isinya yaitu penjumlahan nilai_array_1+Nilai_array_3+Nilai_array_3 setelah itu nilai_array_1 , Nilai_array_3, dan Hasil di prin untuk melihat nilai dari array tersebut.

```

1 import numpy as np
2
3 lala = np.arange(6)
4 print(lala)
5
6 #array 3dimensi
7 lalae = np.arange(36).reshape(4,3,3)
8 print(lalae)
9
10 #penjumlahan array
11 nilai_array_1 = np.array([1,2,3,4])
12 nilai_array_2 = np.array([20,30,40,50])
13 nilai_array_3 = np.arange(4)
14 Hasil = nilai_array_1+nilai_array_3=nilai_array_3
15 print(nilai_array_1)
16 print(nilai_array_3)
17 print(Hasil)

```

3. matplotlib

Arti tiap baris aplikasi sederhana matplotlib pada baris ke satu yaitu memasukan library matplotlib.pyplot yang di definisikan menjadi plt ke-

```

Run: 2,2 <
[0 1 2 3 4 5]
[[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]]]

[[[ 9 10 11]
 [12 13 14]
 [15 16 17]]]

[[[18 19 20]
 [21 22 23]
 [24 25 26]]]

[[[27 28 29]
 [30 31 32]
 [33 34 35]]]
[1 2 3 4]
[0 1 2 3]
[ 1 4 7 10]

Process finished with exit code 0

```

Gambar 3.8 hasil

mudian membuat variabel kelas_ti3 pada baris ke tiga yang berisi label setelah itu di buat variabel jumlah_mhs3 pada baris ke empat yang berisi nilai dari setiap label tersebut. begitu juga pada baris ke emam dan ke tujuh kemudian pada baris ke sembilan matplotlib mendefinisikan gambar dengan ukurannya dan pada baris ke 10 di dekralasikan subplot setelah itu pada baris ke 11 matplotlib mendefinisikan jenis grafik yang digunakan dan dimasukan variabel kelas dan jumlah_mhs. begitujuga oada baris ke 13 14 dan 15 setelah itu di buat title pada baris ke 17 dan matplotlib di show untuk mendapatkan hasil dari grafiknya.

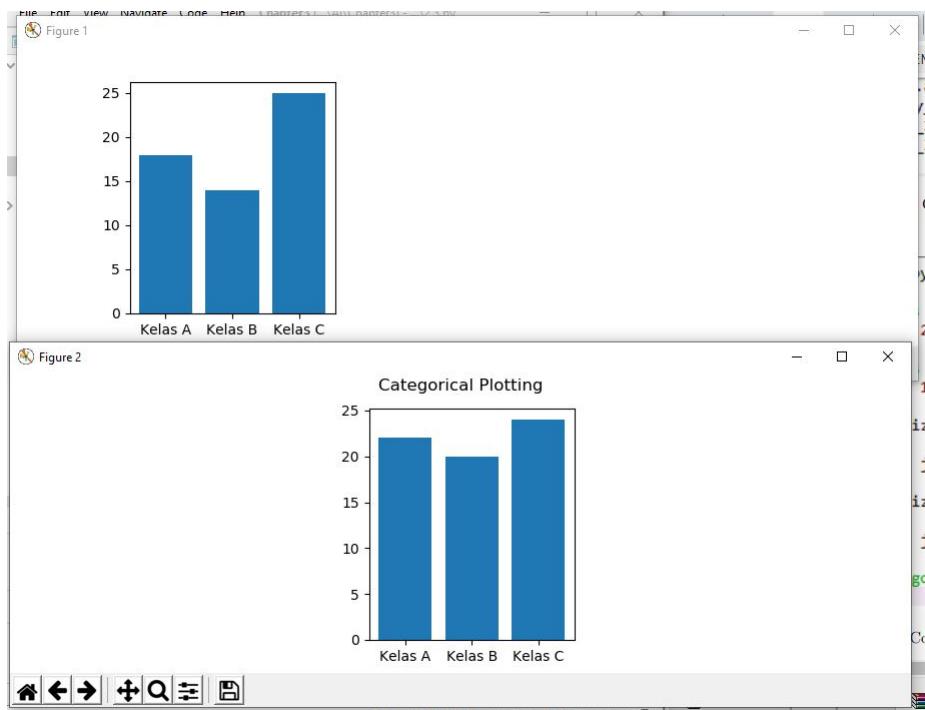
```

1 import matplotlib.pyplot as plt
2
3 kelas_ti3 = ['Kelas A', 'Kelas B', 'Kelas C']
4 jumlah_mhs3 = [18, 14, 25]
5
6 kelas_ti2 = ['Kelas A', 'Kelas B', 'Kelas C']
7 jumlah_mhs2 = [22, 20, 24]
8
9 plt.figure(1, figsize=(9,3))
10 plt.subplot(131)
11 plt.bar(kelas_ti3, jumlah_mhs3)
12 plt.figure(2, figsize=(9,3))
13 plt.subplot(132)
14 plt.bar(kelas_ti2, jumlah_mhs2)
15 plt.suptitle('Categorical Plotting')
16 plt.show()

```

4. Random Forest

Arti tiap baris hasil codingan random forest pada baris pertama random forest di import dari sklearn dengan ketentuan yaitu maksimal isi dari de-



Gambar 3.9 hasil

cision tree berisi 50 data dengan keadaan random dan dengan estimators 100 data ini berada dalam variabel clf kemudian setelah itu variabel clf di running berdasarkan data training dan data label yang telah di definisikan jumlahnya. kemudian variabel clf di running berdasarkan data training paling atas untuk memunculkan hasil data training lima paling atas. setelah itu data tersebut di di running scorenya untuk melihat tingkat akurasi yang dia kerjakan maka tingkat akurasi yang dihasilkan berada pada kisaran 0,437 atau kisaran 43 %.

```

1 from sklearn.ensemble import RandomForestClassifier
2 clf = RandomForestClassifier(max_features=50, random_state=0,
3     n_estimators=100)
4 clf.fit(df_train_att, df_train_label)
5 print(clf.predict(df_train_att.head()))
6 clf.score(df_test_att, df_test_label)

```

5. Confusion Matrix

arti codingan pada hasil tiap codingan confusion matrix pada baris pertama codingan tersebut mendeskripsikan atau mengimport confusion matrix dari sklearn kemudian dibuat variabel pred_labels dengan di isikan clf prdic df_test_att setelah itu membuat variabel cm yang isinya terdapat

```

Terminal: Local + 
>>>
>>> from sklearn.ensemble import RandomForestClassifier
>>> clf = RandomForestClassifier(max_features=50, random_state=0, n_estimators=100)
>>> clf.fit(df_train_att, df_train_label)
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                      criterion='gini', max_depth=None, max_features=50,
                      max_leaf_nodes=None, max_samples=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100,
                      n_jobs=None, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
>>> print(clf.predict(df_train_att.head()))
[[0 0 0 ... 0 0 1]
 [0 0 0 ... 0 0 1]
 [0 0 0 ... 0 0 1]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 0]]
>>> clf.score(df_test_att, df_test_label)
0.009503695881731784
>>> []

```

Gambar 3.10 hasil

data yang di buat confusion matrix berdasarkan data test setelah variabel cm akan di running yang mana akan menghasilkan gambar berupa matrix matrix tersebut berisi nilai nilai kebenaran yang mendekati nilai benar atau mutlak nilai benar.

```

1 from sklearn.metrics import confusion_matrix
2 pred_labels = clf.predict(df_test_att)
3 cm = confusion_matrix(df_test_label, pred_labels)
4 cm

```

```

>>> from sklearn.metrics import confusion_matrix
>>> pred_labels = clf.predict(df_test_att)
>>> cm = confusion_matrix(df_test_label, pred_labels)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "C:\Users\najib\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\metrics\_classification.py", line 270, in confusion_matrix
      raise ValueError("%s is not supported" % y_type)
ValueError: multilabel-indicator is not supported
>>> cm
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cm' is not defined

```

Gambar 3.11 hasil

6. SVM dan Decision Tree

Arti dari setiap baris hasil codingan decision tree dan SVM pada tree masukan terlebih dahulu library tree setelah itu buat variabel clftree yang berisi decision tree setelah itu masukan nilai data training dan label yang telah di deklarasikan tadi setelah itu running variabel tersebut untuk mendapatkan score 0,266 kisaran 26 sampai 27 % akurasinya kemudian pada svm juga hampir sama masukan terlebih dahulu librarynya

setelah itu buat variabel clfsvm yang berarti berisi nilai data training dan data label dan pendeklarasian svm itu sendiri setelah itu di running untuk mendapatkan nilai akurasinya atau score sebesar 0,283 atau dalam kisaran 23 %.

```

1 from sklearn import tree
2 clftree = tree.DecisionTreeClassifier()
3 clftree.fit(df_train_att, df_train_label)
4 clftree.score(df_test_att, df_test_label)
5
6 from sklearn import svm
7 clfsvm = svm.SVC()
8 clfsvm.fit(df_train_att, df_train_label)
9 clfsvm.score(df_test_att, df_test_label)
```

```

>>> from sklearn import tree
>>> clftrree = tree.DecisionTreeClassifier()
>>> clftrree.fit(df_train_att, df_train_label)
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                      max_depth=None, max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
>>> clftrree.score(df_test_att, df_test_label)
0.0008044333214158026
>>> █
```

Gambar 3.12 hasil

7. Cross Validation

arti dari setiap baris hasil cross validation pada gambar?? tersebut diperlihatkan codingan error dikarenakan data training terlalu besar maka untuk mengatasinya dapat dilihat pada sub bab penanganan error

```

1 from sklearn.model_selection import cross_val_score
2 scores = cross_val_score(clf, df_train_att, df_train_label,
                           cv=5)
3 print("Accuracy: %.2f (+/- %.2f)" % (scores.mean(), scores.
                                         std() * 2))
```

```

>>> from sklearn.model_selection import cross_val_score
>>> scores = cross_val_score(clf, df_train_att, df_train_label, cv=5)
>>> print("Accuracy: %.2f (+/- %.2f)" % (scores.mean(), scores.std() * 2))
Accuracy: 0.00 (+/- 0.00)
>>> █
```

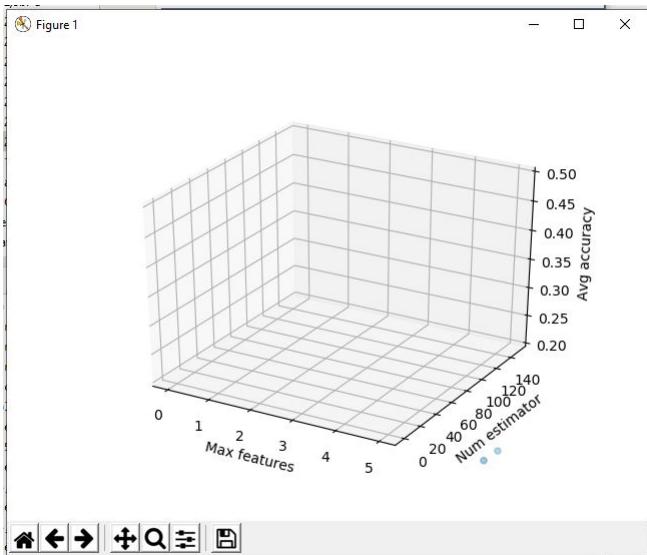
Gambar 3.13 hasil

8. program pengamatan arti dari hasil program pengamatan. perogram pengamatan ini menggunakan library matplotlib supaya hasil dari presentase hasil random forest, svm dan decision tree dapat di bandingkan dengan membuat variabel X Y Z kemudian memberikan label untuk setiap dimensinya untuk lebih jelas dapat dilihat gambar ?? yang menunjukan hasil perbandingan presentase dari tiga metode tersebut.

```

1 import matplotlib.pyplot as plt
2 from mpl_toolkits.mplot3d import Axes3D
3 from matplotlib import cm
4 fig = plt.figure()
5 fig.clf()
6 ax = fig.gca(projection='3d')
7 x = rf_params[:,0]
8 y = rf_params[:,1]
9 z = rf_params[:,2]
10 ax.scatter(x, y, z)
11 ax.set_zlim(0.2, 0.5)
12 ax.set_xlabel('Max features')
13 ax.set_ylabel('Num estimator')
14 ax.set_zlabel('Avg accuracy')
15 plt.show()

```



Gambar 3.14 hasil

3.1.3 Penanganan Error / cokro

Screenshot error

- Untuk gambar screenshot error

Code Errornya

- kode error pada screenshot ke satu yaitu dikarenakan `clfsvm.fit(df_train_att, df_train_label)` dikarenakan data trainingnya terlalu besar sehingga komputernya error.

```
In [48]: scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)
...: print("Accuracy: %.2f (+/- %.2f)" % (scoressvm.mean(), scoressvm.std() * 2))
Traceback (most recent call last):

File "<ipython-input-48-d7a7153ce4e9>", line 1, in <module>
    scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 402, in cross_val_score
    error_score=error_score)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 240, in cross_validate
    for train, test in cv.split(X, y, groups))

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 917, in __call__
    if self.dispatch_one_batch(iterator):

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 759, in dispatch_one_batch
    self._dispatch(tasks)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 716, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel_backends.py", line 182, in apply_async
    result = ImmediateResult(func)

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel_backends.py", line 549, in __init__
    self.results = batch()

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in __call__
    for func, args, kwargs in self.items]

File "C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, in <listcomp>
```

Gambar 3.15 hasil

- untuk kode error pada screen shoot ke 2 sampai ke 4 dikarenakan pada kode berikut scores = cross_val_score(clf, df_train_att, df_train_label, cv=5) scorestree = cross_val_score(clftree, df_train_att, df_train_label, cv=5) dan scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5) hal ini di karenakan data trainingterlalu besar sehingga berdampak pada komputer sehingga library dari python tidak mampu mengolah data dan hasilnya menjadi error.

Solusi Untuk mengatasi Error

- solusinya untuk yang ke satu yaitu dengan cara merestart spyder atau mematikannya kemudian nyalakan kembali setelah itu jalankan code yang error tersebut di CMD cika dalam python CMD jalam maka bisa di running. setelah itu buka kembali spyder dan jalankan codingan dari awal hingga pada bagian SVM tunggu sebenar sampai muncul nilai akurasinya.
- solusi untuk mengatasi error tersebut yaitu dengan cara merubah bobot data pada data training.

```
df = imgatt2.join(imglabels)
df = df.sample(frac=1)
df_att = df.iloc[:, :312]
df_label = df.iloc[:, :312]
df_train_att = df_att[:600]
df_train_label = df_label[:600]
df_test_att = df_att[600:]
df_test_label = df_label[600:]
```

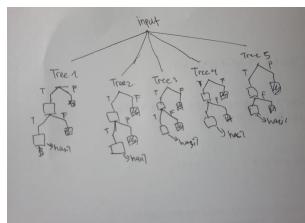
Gambar 3.16 hasil

3.2 1174040 - Hagan Rowlenstino A. S

3.2.1 Teori

- Jelaskan apa itu random forest, sertakan gambar ilustrasi buatan sendiri.

Random forest adalah sebuah algoritma yang biasanya dipakai untuk mengklasifikasikan suatu data yang berjumlah besar. Dimana pengklasifikasianya itu menggunakan pohon atau tree yang digabungkan serta melewati training terlebih dahulu pada data sample nya. Akurasinya pun akan menjadi lebih baik apabila lebih banyak tree nya. Penentuan dari pengklasifikasianya sendiri diambil dari hasil voting yang terbentuk dan pemenangnya adalah tree atau pohon yang mempunyai voting terbanyak.



Gambar 3.17 Random Forest

- Jelaskan cara membaca dataset khusus dan artikan makna setiap le dan isi eld masing masing le

Download terlebih dulu data yang akan dibaca, lalu buka aplikasi spyder dan jalankan kode nya. data yang terdapat pada file tersebut adalah data folder ATTRIBUTE, IMAGES, PARTS yang memiliki kegunaannya sendiri yang dimana pada penggunaannya data yang dipakai adalah data image_attribute_label pada folder attribute, data image_class_labels dan data classes. file image_attribute_label berguna sebagai data awal yang digunakan untuk membaca data attribute yang terdapat pada masing - masing gambar burung yang ada. sedangkan file image_class_label berguna sebagai data yang akan membuat kolom baru pada dataset yang fungsinya adalah untuk memasukan hasil dari semua data yang dimiliki oleh imgatt2. dan file classes berguna sebagai dataset yang akan dipanggil oleh fungsi code untuk menampilkan nama dari data burung yang dimiliki. file image_attribute_label berisi tentang data attribute yang ada pada data gambar file burung yang dimiliki difolder image pada CUB-200-2011 file image_class_label berisi tentang data yang dimiliki oleh attribute dari image_attribute_label dimana data yang bernilai atau memiliki nilai disusun hingga menghasilkan data yang mudah dipahami. file classes berisi tentang data yang berguna untuk menampilkan data nama dari setiap data jenis burung yang dimiliki.

3. Jelaskan apa itu Cross Validation.

Cross Validation adalah teknik untuk memvalidasi sebuah model untuk menilai pengeneralisasian dari kumpulan data independen hasil statistik analis. Biasanya teknik ini digunakan untuk memprediksi dan memperkirakan keakuratan sebuah model pada saat di eksekusi.

4. Jelaskan apa arti score 44 % pada random forest, 27 % pada decision tree dan 29 % dari SVM.

arti score 27% pada decision tree adalah presentasi hasil dari perhitungan dataset acak, dan arti score 29% dari SVM adalah hasil pendekatan neural network. Hasil tersebut didapat dari hasil validasi silang untuk memastikan bahwa membagi training test dengan cara yang berbeda. Jadi 44% untuk random forest, 27% untuk pohon keputusan, dan 29% untuk SVM. Itu merupakan persentase keakuratan prediksi yang dilakukan pada saat testing menggunakan label pada dataset yang digunakan. Score mendefinisikan aturan evaluasi model.

5. Jelaskan bagaimana cara membaca confusion matriks dan contohnya memakai gambar atau ilustrasi sendiri.

Confusion matrix menggunakan rumus perhitungan dengan 4 keluaran, yang pertama adalah :

		Prediksi →	
		Negatif	Positive
Faktal	Mengalih	a	b
	Pompa	c	d

Gambar 3.18 Confusion Matrix

- Recall

$$d/(c + d) \quad (3.1)$$

- Precision

$$d/(b + d) \quad (3.2)$$

- Accuracy

$$(a + c)/(a + b + c + d) \quad (3.3)$$

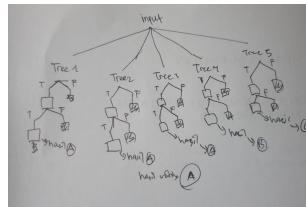
- Error Rate

$$(b + c)/(a + b + c + d) \quad (3.4)$$

6. Jelaskan apa itu voting pada random forest disertai dengan ilustrasi gambar sendiri.

Voting itu adalah hasil dari sebuah desicion tree, yang nanti akhirnya akan menjadi hasil dari random forest. sebagai contoh ada 5 desicion tree. Desicion tree pertama menyimpulkan A, yang ke dua juga A, dan ketiga

pun A, tetapi ke 4 B dan ke 5 C. maka random forest akan menyimpulkan hasilnya adalah A.



Gambar 3.19 Vote

3.2.2 Praktek

- Pandas Pada baris pertama kita mengimport library pandas dan menamainya sebagai pan , lalu memasukkan data kedalam variable data. setelah itu membuat dataframe dengan data yang telah di masukkan tadi, lalu membuat variable bernama cari untuk melihat semua data dari attribute Nama lalu print variable tersebut untuk melihat hasilnya pada console

```
import pandas as pan
data = {'Nama': ['Budi', 'Rina', 'Adi'], 'Umur':[12,13,11]}
pan.DataFrame(data)
cari = data['Nama']
print(cari)
```

Gambar 3.20 Pandas

```
In [6]: runfile('C:/Users/User/.spyder-py3/temp.py', wdir='C:/Users/User/.spyder-py3')
['Budi', 'Rina', 'Adi']
```

Gambar 3.21 hasil Pandas

- numpy Baris pertama adalah untuk mengimport library numpy dan menamainya sebagai num, lalu membuat sebuah variable bernama mat dan menggunakan fungsi arrange untuk membuat angka berurutan dari 1 sampai 25, mengapa ditulis satu sampai 26 karena urutan dimulai dari 0, lalu menggunakan reshape untuk mengubahnya menjadi matrix. dan terakhir print variable mat untuk menampilkan nya pada console

```
# 
# import numpy as num
# mat = num.arange(1,26).reshape(5,5)
# print(mat)
```

Gambar 3.22 Numpy

```
In [20]: runfile('C:/Users/l
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
[11 12 13 14 15]
[16 17 18 19 20]
[21 22 23 24 25]]
```

Gambar 3.23 Hasil Numpy

3. matplotlib Pertama import pyplot dari library matplotlib dan menamainya sebagai plt, lalu memasukkan data yang diinginkan kedalam variable x dan y . lalu menggunakan fungsi bar untuk membuat grafik bar yang isi datanya adalah x, dan y, lalu memberikan label pada sumbu x dan y dan memberikan judul pada grafik tersebut, dan menggunakan show() untuk menampilkan grafik

```
from matplotlib import pyplot as plt
x = [2012,2013,2014,2015,2016,2017]
y = [1000,2000,3000,4000,5000,6000]
plt.bar(x,y)
plt.xlabel('Tahun')
plt.ylabel('Jumlah Produksi')
plt.title('Jumlah Produksi Per Tahun')
plt.show()
```

Gambar 3.24 Matplotlib



Gambar 3.25 Hasil Matplotlib

4. Random Forest

Dari sklearn.ensamble mengimport RandomForestClassifier dan memberikan ketentuan dimana maksimal datanya adalah 50 dengan keadaan random serta estimatornya 100 dan dimasukkan kedalam variable clf lalu itu variabel clf di running berdasarkan data training dan data label yang telah di denisikan jumlahnya. kemudian variabel clf di running berdasarkan data training paling atas untuk memunculkan hasil data training lima paling atas. setelah itu data tersebut di di running scorenya untuk melihat tingkat akurasi yang dia kerjakan maka tingkat akurasi yang dihasilkan berada pada kisaran 0,448 atau kisaran 44%.

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_features=50, random_state=0, n_estimators=100)
clf.fit(df_train_att, df_train_label)
print(clf.predict(df_train_att.head()))
clf.score(df_test_att, df_test_label)
```

Gambar 3.26 Random Forest

```
In [112]: from sklearn.ensemble import RandomForestClassifier
...: clf = RandomForestClassifier(max_features=50, random_state=0,
...: n_estimators=100)
...: clf.fit(df_train_att, df_train_label)
...: print(clf.predict(df_train_att.head()))
...: clf.score(df_test_att, df_test_label)
[ 20  66  56  39 116]
Out[112]: 0.448257655750158
```

Gambar 3.27 Hasil Random Forest

5. Confussion Matrix dari sklearn.metrics mengimport confusion matrix kemudian dibuat variabel pred labels dengan di isikan clf prdic df test att setelah itu membuat variabel cm yang isinya terdapat data yang di buat confusion matrix berdasarkan data test setelah variabel cm akan di running yang mana akan menghasilkan gambar berupa matrix matrix tersebut berisi nilai nilai kebenaran yang mendekati nilai benar atau mutlak nilai benar.

```
from sklearn.metrics import confusion_matrix
pred_labels = clf.predict(df_test_att)
cm = confusion_matrix(df_test_label, pred_labels)
```

Gambar 3.28 Confusion Matrix

```
In [113]: from sklearn.metrics import confusion_matrix
...: pred_labels = clf.predict(df_test_att)
...: cm = confusion_matrix(df_test_label, pred_labels)
...: cm
Out[113]:
array([[ 7,  0,  0, ...,  0,  0,  0],
       [ 1, 12,  0, ...,  0,  1,  0],
       [ 1,  1,  7, ...,  0,  0,  0],
       ...,
       [ 0,  0,  1, ...,  4,  0,  0],
       [ 0,  0,  0, ...,  0, 11,  0],
       [ 0,  0,  0, ...,  0,  0, 16]], dtype=int64)
```

Gambar 3.29 Hasil Coonfusion Matrix

6. Decision Tree dan SVM masukan terlebih dahulu library tree setelah itu buat variabel clftree yang berisi decision tree setelah itu masukan nilai data training dan label yang telah di deklarasikan tadi setelah itu running variabel tersebut untuk mendapatkan score 0,266 kisaran 26 sampai 27% akurasinya

```
from sklearn import tree
clftree = tree.DecisionTreeClassifier()
clftree.fit(df_train_att, df_train_label)
clftree.score(df_test_att, df_test_label)
```

Gambar 3.30 Desicion Tree

```
In [114]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(df_train_att, df_train_label)
...: clftree.score(df_test_att, df_test_label)
Out[114]: 0.26689545934530096
```

Gambar 3.31 Hasil Desicion Tree

Import terlebih dahulu library svm nya setelah itu buat variabel clfsvm yang berarti berisi nilai data training dan data label dan pendeklarasian svm itu sendiri setelah itu di running untuk mendapatkan nilai akurasinya atau score sebesar 0,283 atau dalam kisaran 23%.

```
from sklearn import svm
clfsvm = svm.SVC()
clfsvm.fit(df_train_att, df_train_label)
clfsvm.score(df_test_att, df_test_label)
```

Gambar 3.32 SVM

```
In [115]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(df_train_att, df_train_label)
...: clfsvm.score(df_test_att, df_test_label)
Out[115]: 0.4799366420274551
```

Gambar 3.33 Hasil SVM

- Cross Validation mengimport cross_val_score dari sklearn.model_selection , lalu memasukkan variable variable yang akan di ukur keakuratan dari model yang digunakan lalu menampilkan nya di console, disini digunakan variable clf yaitu random forest.

```
:from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, df_train_att, df_train_label, cv = 5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

Gambar 3.34 Cross Val Rand Forest

```
In [132]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(clf, df_train_att, df_train_label, cv = 5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
Accuracy: 0.45 (+/- 0.03)
```

Gambar 3.35 Hasil Cross Validaiton Random Forest

Disini memasukkan variable variable yang akan di ukur keakuratan dari model yang digunakan lalu menampilkan nya di console, disini digunakan variable clftree yaitu desicion Tree.

```
scoretree = cross_val_score(clftree, df_train_att, df_train_label, cv = 5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scoretree.mean(), scoretree.std() * 2))
```

Gambar 3.36 Cross Vadation Desicion Tree

```
In [137]: scoresvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv = 5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2))
Accuracy: 0.26 (+/- 0.02)
```

Gambar 3.37 Hasil Cross Vadation Desicion Tree

Disini memasukkan variable variable yang akan di ukur keakuratan dari model yang digunakan lalu menampilkan nya di console, disini digunakan variable clfsvm yaitu svm.

```
scoresvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv = 5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2))
```

Gambar 3.38 Cross Vadation SVM

```
In [134]: scoresvm = cross_val_score(classif, df_train_att, df_train_label, cv = 5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2))
Accuracy: 0.47 (+/- 0.03)
```

Gambar 3.39 Hasil Cross Vadation SVM

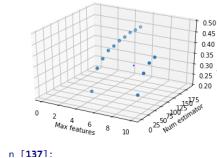
8. Program Pengamatan dan Hasil Pengamatan. perogram pengamatan ini menggunakan library matplotlib supaya hasil dari presentase hasil random forest, svm dan decision tree dapat di bandingkan dengan membuat variabel X Y Z kemudian memberikan label untuk setiap dimensinya untuk lebih jelas dapat dilihat gambar yang menunjukan hasil perbandingan presentase dari tiga metode tersebut.

```
max_estimators_opts = {'n_estimators': 10}
n_estimators_opts = {'n_estimators': 50}
n_estimators_opts = {'n_estimators': 100}
n_estimators_opts = {'n_estimators': 150}
n_estimators_opts = {'n_estimators': 200}
n_estimators_opts = {'n_estimators': 250}
n_estimators_opts = {'n_estimators': 300}
n_estimators_opts = {'n_estimators': 350}
n_estimators_opts = {'n_estimators': 400}
n_estimators_opts = {'n_estimators': 450}
n_estimators_opts = {'n_estimators': 500}
n_estimators_opts = {'n_estimators': 550}
n_estimators_opts = {'n_estimators': 600}
n_estimators_opts = {'n_estimators': 650}
n_estimators_opts = {'n_estimators': 700}
n_estimators_opts = {'n_estimators': 750}
n_estimators_opts = {'n_estimators': 800}
n_estimators_opts = {'n_estimators': 850}
n_estimators_opts = {'n_estimators': 900}
n_estimators_opts = {'n_estimators': 950}
n_estimators_opts = {'n_estimators': 1000}
n_estimators_opts = {'n_estimators': 1050}
n_estimators_opts = {'n_estimators': 1100}
n_estimators_opts = {'n_estimators': 1150}
n_estimators_opts = {'n_estimators': 1200}
n_estimators_opts = {'n_estimators': 1250}
n_estimators_opts = {'n_estimators': 1300}
n_estimators_opts = {'n_estimators': 1350}
n_estimators_opts = {'n_estimators': 1400}
n_estimators_opts = {'n_estimators': 1450}
n_estimators_opts = {'n_estimators': 1500}
n_estimators_opts = {'n_estimators': 1550}
n_estimators_opts = {'n_estimators': 1600}
n_estimators_opts = {'n_estimators': 1650}
n_estimators_opts = {'n_estimators': 1700}
n_estimators_opts = {'n_estimators': 1750}
n_estimators_opts = {'n_estimators': 1800}
n_estimators_opts = {'n_estimators': 1850}
n_estimators_opts = {'n_estimators': 1900}
n_estimators_opts = {'n_estimators': 1950}
n_estimators_opts = {'n_estimators': 2000}
n_estimators_opts = {'n_estimators': 2050}
n_estimators_opts = {'n_estimators': 2100}
n_estimators_opts = {'n_estimators': 2150}
n_estimators_opts = {'n_estimators': 2200}
n_estimators_opts = {'n_estimators': 2250}
n_estimators_opts = {'n_estimators': 2300}
n_estimators_opts = {'n_estimators': 2350}
n_estimators_opts = {'n_estimators': 2400}
n_estimators_opts = {'n_estimators': 2450}
n_estimators_opts = {'n_estimators': 2500}
n_estimators_opts = {'n_estimators': 2550}
n_estimators_opts = {'n_estimators': 2600}
n_estimators_opts = {'n_estimators': 2650}
n_estimators_opts = {'n_estimators': 2700}
n_estimators_opts = {'n_estimators': 2750}
n_estimators_opts = {'n_estimators': 2800}
n_estimators_opts = {'n_estimators': 2850}
n_estimators_opts = {'n_estimators': 2900}
n_estimators_opts = {'n_estimators': 2950}
n_estimators_opts = {'n_estimators': 3000}
n_estimators_opts = {'n_estimators': 3050}
n_estimators_opts = {'n_estimators': 3100}
n_estimators_opts = {'n_estimators': 3150}
n_estimators_opts = {'n_estimators': 3200}
n_estimators_opts = {'n_estimators': 3250}
n_estimators_opts = {'n_estimators': 3300}
n_estimators_opts = {'n_estimators': 3350}
n_estimators_opts = {'n_estimators': 3400}
n_estimators_opts = {'n_estimators': 3450}
n_estimators_opts = {'n_estimators': 3500}
n_estimators_opts = {'n_estimators': 3550}
n_estimators_opts = {'n_estimators': 3600}
n_estimators_opts = {'n_estimators': 3650}
n_estimators_opts = {'n_estimators': 3700}
n_estimators_opts = {'n_estimators': 3750}
n_estimators_opts = {'n_estimators': 3800}
n_estimators_opts = {'n_estimators': 3850}
n_estimators_opts = {'n_estimators': 3900}
n_estimators_opts = {'n_estimators': 3950}
n_estimators_opts = {'n_estimators': 4000}
n_estimators_opts = {'n_estimators': 4050}
n_estimators_opts = {'n_estimators': 4100}
n_estimators_opts = {'n_estimators': 4150}
n_estimators_opts = {'n_estimators': 4200}
n_estimators_opts = {'n_estimators': 4250}
n_estimators_opts = {'n_estimators': 4300}
n_estimators_opts = {'n_estimators': 4350}
n_estimators_opts = {'n_estimators': 4400}
n_estimators_opts = {'n_estimators': 4450}
n_estimators_opts = {'n_estimators': 4500}
n_estimators_opts = {'n_estimators': 4550}
n_estimators_opts = {'n_estimators': 4600}
n_estimators_opts = {'n_estimators': 4650}
n_estimators_opts = {'n_estimators': 4700}
n_estimators_opts = {'n_estimators': 4750}
n_estimators_opts = {'n_estimators': 4800}
n_estimators_opts = {'n_estimators': 4850}
n_estimators_opts = {'n_estimators': 4900}
n_estimators_opts = {'n_estimators': 4950}
n_estimators_opts = {'n_estimators': 5000}
```

Gambar 3.40 Program Pengamatan

```
...: for max_features in max_features_opts:
...:     for n_estimators in n_estimators_opts:
...:         if n_estimators > 1000:
...:             print("No. Estimators: %d, Max Features: %d, Accuracy: %0.2f (+/- %0.2f)" % (n_estimators, max_features, rf_params[1][0], rf_params[1][1]))
...:         else:
...:             scores = cross_val_score(classif, df_train_att, df_train_label, cv = 5)
...:             rf_params[1][0] = n_estimators
...:             rf_params[1][1] = scores.mean()
...:             rf_params[1][2] = scores.std()
...:             rf_params[1][3] = 1 + n_estimators
...:             i += 1
...:             print("No. Estimators: %d, Max Features: %d, Accuracy: %0.2f (+/- %0.2f)" % (n_estimators, max_features, rf_params[1][0], rf_params[1][1]))
```

Gambar 3.41 Hasil Program Pengamatan

**Gambar 3.42** Grafik Program Pengamatan

3.2.3 Penanganan Error

3.2.3.1 Skrinshoot error

1. Error 1

```
FileNotFoundError: File b'D:\Semester6\AI\Chapter3\UB_200_2011\07\Attributes\
\image_attribute_labels.txt' does not exist.
```

Gambar 3.43 Error 1

2. Error 2

```
File "c:\python-input-106-e6d717638a8a", line 1, in <module>
    image_labels = pd.read_csv('D:/Semester6/AI/Chapter3/UB_200_2011/
image_class_labels.txt', sep=' ', header=None, names=['imgId','label'])
NameError: name 'None' is not defined
```

Gambar 3.44 Error 2

3. Error 3

```
In [147]: from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_features=50, random_state=0,
n_estimators=10)
...: clf.fit(df_train_attr, df_train_label)
...: pred_labels = clf.predict(df_test_attr)
...: clf.score(df_test_attr, df_test_label)
...: clf.score(df_test_attr, pred_labels)
C:\Users\Haryo\PycharmProjects\Bab3\temp.py:1: DeprecationWarning: A column-vector y was
passed when a 1D array was expected. Please change the shape of y to (n_samples,), for
example using ravel().
```

Gambar 3.45 Error 3

4. Error 4

```
File "c:\python-input-150-cb3ee086142", line 3, in <module>
    cm = confusion_matrix(df_test_label, pred_labels)
NameError: name 'confusion_matrix' is not defined
```

Gambar 3.46 Error 4

3.2.3.2 Kode Error dan Tipe Error

1. Error 1 type FileNotFoundError

```
input = pd.read_csv("D:\Semester6\AI\Chapter3\UB_200_2011\attributeImage_attributes_labels.txt", sep=",", header=None)
input.head()
```

Gambar 3.47 Kode Error 1

2. Error 2 type NameError

```
imglabels = pd.read_csv("D:\Semester6\AI\Chapter3\UB_200_2011\image_attributes_labels.txt", sep=",", header=None)
imglabels = imglabels.set_index('imgID')
```

Gambar 3.48 Kode Error 2

3. Error 3 type DataConversionWarning

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_features=50, random_state=0, n_estimators=100)
clf.fit(df_train_att, df_train_label)
pred_labels = clf.predict(df_test_att)
clf.score(df_test_att, df_test_label)
```

Gambar 3.49 Kode Error 3

4. Error 4 type NameError

```
from sklearn.metrics import confusion_matrix
pred_labels = clf.predict(df_test_att)
cm = confusion_matrix(df_test_label, pred_labels)
```

Gambar 3.50 Kode Error 4

3.2.3.3 Solusi

1. Mengganti back slash menjadi slash biasa

```
D:\Semester6\AI\Chapter3\UB_200_2011\attributeImage_attributes_labels.txt
```

Gambar 3.51 Fix Error 1

2. Mengubah huruf depan katan none menjadi huruf besar

Gambar 3.52 Fix Error 2

3. Menambahkan label kedalam variable

```

df_train_att = df_att[:8000]
df_train_label = df_label[:8000]
df_test_att = df_att[8000:]
df_test_label = df_label[8000:]

df_train_label = df_train_label['label']
df_test_label = df_test_label['label']

```

Gambar 3.53 Fix Error 3

- Mengilangkan 1 huruf s

```

cm = confusion_matrix(df_test_label, pred_labels)

```

Gambar 3.54 Fix Error 4

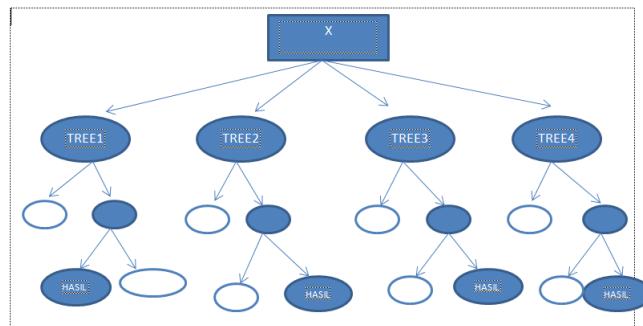
3.3 1174050 Dika Sukma Pradana

3.3.1 Teori

- Jelaskan apa itu random forest, sertakan gambar ilustrasi buatan sendiri.

Random Forest adalah konstruk data yang diterapkan pada machine learning yang mengembangkan sejumlah besar pohon keputusan acak yang menganalisis sekumpulan variabel. Jenis algoritma ini membantu meningkatkan cara teknologi menganalisis data yang kompleks. Juga merupakan algoritma machine learning yang fleksibel, mudah digunakan, bahkan tanpa penyetelan hyper-parameter, dengan hasil yang baik. Ini juga merupakan salah satu algoritma yang paling banyak digunakan, karena kesederhanaan dan faktanya dapat digunakan untuk tugas klasifikasi dan regresi.

Dibawah ini merupakan salah satu ilustrasi penggunaan Random Forest.

**Gambar 3.55** Random Forest

- Jelaskan cara membaca dataset khusus dan artikan makna setiap file dan isi field masing masing file. Dataset adalah kumpulan data. Paling umum

satu data set sesuai dengan isi tabel database tunggal, atau matriks data statistik tunggal, di mana setiap kolom tabel mewakili variabel tertentu, dan setiap baris sesuai dengan anggota tertentu dari dataset yang diperlukan.

- Gunakan librari Pandas pada python untuk dapat membaca dataset dengan format text file.
 - Setelah itu, buat variabel baru "dataset" yang berisikan perintah untuk membaca file csv.
 - Memanggil Librari Panda untuk membaca dataset
 - Membuat variabel "Dataset" yang berisikan pdreadcsv untuk membaca dataset. Pada contoh ini menggunakan txt tapi tetap bisa membaca datasetnya, mengapa? Karena pada saat dijalankan librari panda secara otomatis akan mengubah data dalam bentuk text file ke format csv.
3. Jelaskan apa itu Cross Validation. Cross Validation adalah prosedur resampling yang digunakan untuk mengevaluasi model machine learning pada sampel data yang terbatas. Prosedur ini memiliki parameter tunggal yang disebut k yang mengacu pada jumlah grup tempat sampel data yang akan dibagi. Karena itu, prosedur ini sering disebut k-fold cross-validation. Proses penentuan apakah hasil numerik yang mengukur hubungan yang dihipotesiskan antar variabel, dapat diterima sebagai deskripsi data, dikenal sebagai Validationi. Umumnya, estimasi kesalahan untuk model dibuat setelah training, lebih dikenal sebagai evaluasi residu. Dalam proses ini, estimasi numerik dari perbedaan respons yang diprediksi dan yang asli dilakukan, juga disebut kesalahan training. Namun, ini hanya memberi kita gambaran tentang seberapa baik model kita pada data yang digunakan untuk melatihnya. Sekarang mungkin bahwa model tersebut kurang cocok atau overfitting data. Jadi, masalah dengan teknik evaluasi ini adalah bahwa itu tidak memberikan indikasi seberapa baik pelajar akan menggeneralisasi ke set data independen / tidak terlihat. Model ini dikenal sebagai Cross Validation.
 4. Jelaskan apa arti score 44 % pada random forest, 27 % pada decision tree dan 29 % dari SVM. Itu merupakan presentase keakurasi prediksi yang dilakukan pada saat testing menggunakan label pada dataset yang digunakan. Score merupakan mendefinisikan aturan evaluasi model. Maka pada saat dijalankan akan muncul persentase tersebut yang menunjukkan keakurasi atau keberhasilan dari prediksi yang dilakukan. Jika menggunakan Random Forest maka hasilnya 40%, jika menggunakan Decission Tree hasil prediksinya yaitu 27% dan pada SVM 29% .
 5. Jelaskan bagaimana cara membaca confusion matriks dan contohnya memakai gambar atau ilustrasi sendiri. Perhitungan Confusion Matriks dapat di-

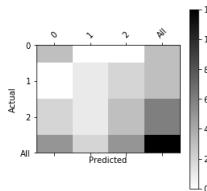
lakukan sebagai berikut. Disini saya menggunakan data yang dibuat sendiri untuk menampilkan data aktual dan prediksi.

- Import librari Pandas, Matplotlib, dan Numpy.
- Buat variabel y actu yang berisikan data aktual.
- Buat variabel y pred berisikan data yang akan dijadikan sebagai prediksi.
- Buat variabel df confusion yang berisikan crosstab untuk membangun tabel tabulasi silang yang dapat menunjukkan frekuensi kemunculan kelompok data tertentu.
- Pada variabel df confusion definisikan lagi nama baris yaitu Actual dan kolomnya Predicted
- Kemudian definisikan suatu fungsi yang diberi nama plot confusion matrix yang berisikan pendefinisian confusion matrix dan juga akan di plotting.

```

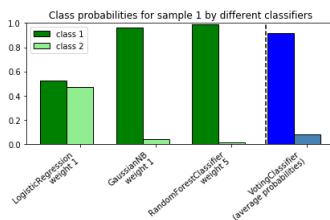
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 16 20:29:09 2020
4
5 @author: User
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10 import pandas as pd
11 y_actu = pd.Series([2, 0, 2, 2, 0, 1, 1, 2, 2, 0, 1, 2],
12                    name='Actual')
13 y_pred = pd.Series([0, 0, 2, 1, 0, 2, 1, 0, 2, 0, 2, 2],
14                    name='Predicted')
15 df_confusion = pd.crosstab(y_actu, y_pred)
16 df_confusion = pd.crosstab(y_actu, y_pred, rownames=['Actual'],
17                           colnames=['Predicted'], margins=True)
18 def plot_confusion_matrix(df_confusion, title='Confusion matrix',
19                           cmap=plt.cm.gray_r):
20     plt.matshow(df_confusion, cmap=cmap) # imshow
21     #plt.title(title)
22     plt.colorbar()
23     tick_marks = np.arange(len(df_confusion.columns))
24     plt.xticks(tick_marks, df_confusion.columns, rotation=45)
25     plt.yticks(tick_marks, df_confusion.index)
26     #plt.tight_layout()
27     plt.ylabel(df_confusion.index.name)
28     plt.xlabel(df_confusion.columns.name)
29     plot_confusion_matrix(df_confusion)
30     plt.show()
```

6. Jelaskan apa itu voting pada random forest disertai dengan ilustrasi gambar sendiri.

**Gambar 3.56** Confusion Matriks

Voting yaitu suara untuk setiap target yang diprediksi pada saat melakukan Random Forest. Pertimbangkan target prediksi dengan voting tertinggi sebagai prediksi akhir dari algoritma random forest.

- Untuk menggunakan Voting pada Random Forest dapat dilihat code berikut. Disini saya mengilustrasikan voting untuk berbagai macam algoritma terutama Random Forest.

**Gambar 3.57** Voting

3.3.2 Praktikum

1. pandas

pada baris ke satu yaitu perintah mengimport library padas pada python atau anaconda kemudian di inisialisasikan menjadi karakter. selanjutnya ada sebuah array yang berisi a b c d. selanjutnya penggunaan array tipe series dan yang terakhir perintah print untuk menampilkan data pada karakter.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 16 21:00:21 2020
4
5 @author: User
6 """
7
8 import pandas as pd
9 data = np.array(['a', 'b', 'c', 'd'])

```

```

10 karakter = pd.Series(data)
11 print(karakter)

```

```

0    a
1    b
2    c
3    d
dtype: object

```

Gambar 3.58 hasil

2. numpy

Arti tiap baris codingan pada aplikasi sederhana numpy adalah sebagai berikut : pada baris ke satu yaitu mengimport numpy yang di inisialisasi menjadi np kemudian pada baris selanjutnya berisikan arange yang berarti membuat data yang berisi 12 dan ada reshape yang berfungsi merubah bentuk dari satu baris menjadi 2 baris data. Lalu yang terakhir ada perintah untuk print yaitu menampilkan data dari dika.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 16 21:08:52 2020
4
5 @author: User
6 """
7
8 import numpy as np
9 dika=np.arange(12).reshape(2,6)
10 print(dika)

```

```

In [11]: runfile('D:/SE'
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]

```

Gambar 3.59 hasil

3. matplotlib

Arti tiap baris aplikasi sederhana matplotlib pada baris ke satu yaitu memasukan library matplotlib.pyplot yang di definisikan menjadi plt kemudian plt.plot untuk menentukan grafik yang akan dibuat. lalu membuat variabel y dengan nama some number yang terakhir untuk menampilkan data pada sebuah grafik.

```

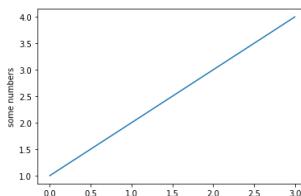
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 16 21:18:45 2020
4
5 @author: User
6 """
7
8 import matplotlib.pyplot as plt
9 plt.plot([1, 2, 3, 4])

```

```

10 plt.ylabel('some numbers')
11 plt.show()

```



Gambar 3.60 hasil

4. Random Forest

Arti tiap baris hasil codingan random forest pada baris pertama random forest di import dari sklearn dengan ketentuan yaitu Nilai default untuk parameter yang mengontrol ukuran pohon (mis. Max_depth, min_samples_leaf, dll.) Mengarah ke pohon yang tumbuh besar dan tidak di-unsumed yang berpotensi sangat besar pada beberapa set data. Untuk mengurangi konsumsi memori, kompleksitas dan ukuran pohon harus dikontrol dengan menetapkan nilai parameter tersebut.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 16 21:32:01 2020
4
5 @author: User
6 """
7
8 from sklearn.ensemble import RandomForestClassifier
9 from sklearn.datasets import make_classification
10 X, y = make_classification(n_samples=1000, n_features=4,
11                           n_informative=2, n_redundant
12                           =0,
13                           random_state=0, shuffle=False)
14 clf = RandomForestClassifier(max_depth=2, random_state=0)
15 clf.fit(X, y)
16 print(clf.feature_importances_)
17 print(clf.predict([[0, 0, 0, 0]]))

```

5. Confusion Matrix

arti codingan pada hasil tiap codingan confusion matrix pada baris pertama codingan tersebut mendeskripsikan atau mengimport confusion matrix dari sklearn kemudian dibuat variabel y_true untuk nilai target ground truth (benar). y_pred untuk Taksiran target seperti yang dikembalikan oleh classifier. lalu menampilkan kedua variabel.

```

1 # -*- coding: utf-8 -*-
2 """

```

```
In [27]: X, y = make_classification(n_samples=1000, n_features=4,
...:                                     n_informative=2, n_redundant=0,
...:                                     random_state=0, shuffle=False)
...: clf = RandomForestClassifier(max_depth=2, random_state=0)
...: clf.fit(X, y)
Out[27]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                 criterion='gini', max_depth=2, max_features='auto',
                                 max_leaf_nodes=None, max_samples=None,
                                 min_impurity_decrease=0.0, min_impurity_split=None,
                                 min_samples_leaf=1, min_samples_split=2,
                                 min_weight_fraction_leaf=0.0, n_estimators=100,
                                 n_jobs=None, oob_score=False, random_state=0, verbose=0,
                                 warm_start=False)

In [28]: print(clf.feature_importances_)
[0.14205973 0.76664038 0.0282433 0.06305659]

In [29]: print(clf.predict([[0, 0, 0, 0]]))
[1]
```

Gambar 3.61 hasil

```
3 Created on Mon Mar 16 21:42:41 2020
4
5 @author: User
6 """
7
8 from sklearn.metrics import confusion_matrix
9 y_true = [2, 0, 2, 2, 0, 1]
10 y_pred = [0, 0, 2, 2, 0, 2]
11 confusion_matrix(y_true, y_pred)
```

```
In [35]: from sklearn.metrics import confusion_matrix
...: y_true = [2, 0, 2, 2, 0, 1]
...: y_pred = [0, 0, 2, 2, 0, 2]
...: confusion_matrix(y_true, y_pred)
Out[35]:
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]], dtype=int64)
```

Gambar 3.62 hasil

6. SVM dan Decision Tree

Seperi pengklasifikasi lainnya, DecisionTreeClassifier mengambil input dua array: array X, jarang atau padat, dengan ukuran n_samples, n_features memegang sampel pelatihan, dan array Y dari nilai integer, ukuran n_samples, Atau, probabilitas setiap kelas dapat diprediksi. Seperti pengklasifikasi lainnya, SVC, NuSVC dan LinearSVC mengambil input dua array: array X ukuran n_samples, n_features memegang sampel pelatihan, dan array y label kelas (string atau bilangan bulat), ukuran n_samples:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 16 21:54:16 2020
4
5 @author: User
6 """
7 from sklearn import tree
```

```

8 X = [[0 , 0] , [1 , 1]]
9 Y = [0 , 1]
10 clf = tree.DecisionTreeClassifier()
11 clf = clf.fit(X, Y)
12 clf.predict([[2. , 2.]])
13
14 from sklearn import svm
15 X = [[0 , 0] , [1 , 1]]
16 y = [0 , 1]
17 clf = svm.SVC()
18 clf.fit(X, y)
19 clf.predict([[2. , 2.]])

```

```

In [40]: from sklearn import svm
...: X = [[0, 0], [1, 1]]
...: y = [0, 1]
...: clf = svm.SVC()
...: clf.fit(X, y)
...: clf.predict([[2., 2.]])
Out[40]: array([1])

In [41]: from sklearn import tree
...: X = [[0, 0], [1, 1]]
...: Y = [0, 1]
...: clf = tree.DecisionTreeClassifier()
...: clf = clf.fit(X, Y)
...: clf.predict([[2., 2.]])
Out[41]: array([1])

```

Gambar 3.63 hasil

7. Cross Validation

digunakan untuk memeriksa akurasi dari ketepatan hasil pengolahan data tersebut maka akan didapat nilai rata-rata 60 persen dari hasil pengolahan data tersebut untuk lebih jelasnya dapat di lihat pada gambar pada codingan tersebut pada baris ke satu melakukan import library dari sklern kemudian pada baris selanjutnya mengisi nilai skor dengan nilai pada variabel lontong setelah hal tersebut dilakukan kemudian data tersebut di eksekusi. berikut merupakan hasil dari code tersebut dapat dilihat pada gambar.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 16 22:05:46 2020
4
5 @author: User
6 """
7
8 from sklearn.model_selection import cross_val_score
9 scores = cross_val_score(lontong, pecel_att, pecel_pass, cv
10 =5)
# show average score and +/- two standard deviations away
11 #(covering 95% of scores)
12 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
std() * 2))

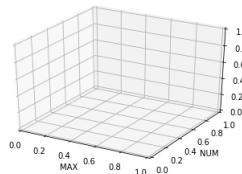
```

```
In [61]: runfile('D:/SEMESTER 6/wert/9.py', wdir='D:/SEMESTER 6/wert')
Accuracy: 0.59 (+/- 0.05)
```

Gambar 3.64 hasil

8. program pengamatan arti dari hasil program pengamatan. perogram pengamatan dapat mengamati dari 3 aspek diatas yaitu svm, random, dan decision tree. Yang memiliki variabel X Y Z dan di tampilkan dalam bentuk grafik.

```
1 import matplotlib.pyplot as plt
2 from mpl_toolkits.mplot3d import Axes3D
3 from matplotlib import cm
4 fig = plt.figure()
5 fig.clf()
6 ax = fig.gca(projection='3d')
7 plt.xlabel('MAX')
8 plt.ylabel('NUM')
9 plt.zlabel('AVG')
10 ax.scatter(x, y, z)
11 ax.set_zlim(0.2, 0.5)
12 ax.set_xlabel('Max features')
13 ax.set_ylabel('Num estimator')
14 ax.set_zlabel('Avg accuracy')
15 plt.show()
```

**Gambar 3.65** hasil

3.3.3 Penanganan Error

Screenshot error

- Untuk gambar screenshot error

Code Errornya

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel(some numbers)
plt.show()
```

```

File "<ipython-input-60-d14a3944647a>", line 1, in <module>
  runfile('D:/SEMESTER 6/wert/1/5.py', wdir='D:/SEMESTER 6/wert/1')

  File "C:\Users\User\Anaconda3\lib\site-packages\spyder_kernels\customize
    \spydercustomize.py", line 827, in runfile
      execfile(filename, namespace)

  File "C:\Users\User\Anaconda3\lib\site-packages\spyder_kernels\customize
    \spydercustomize.py", line 110, in execfile
      exec(compile(f.read(), filename, 'exec'), namespace)

  File "D:/SEMESTER 6/wert/1/5.py", line 10
    plt.ylabel(some numbers)
                           ^
SyntaxError: invalid syntax

```

Gambar 3.66 hasil

pada kode ylabel memiliki nama atau isi some numbers tetapi pada tipe data tertentu harus di awali dan diakhiri dengan tanda petik 2 atau 1. Pada kodingnya hanya kurang tanda petik 1.

3.4 1174057 Alit Fajar Kurniawan

3.4.1 Teori

1. Jelaskan apa itu random forest, sertakan gambar ilustrasi buatan sendiri.

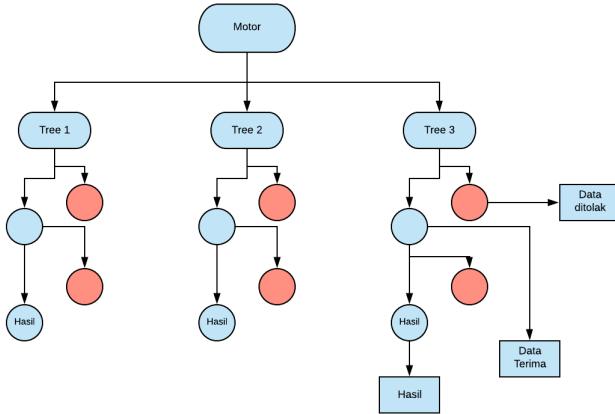
Random Forest merupakan algoritma yang digunakan terhadap klasifikasi data dalam jumlah yang besar. Klasifikasi pada random forest dilakukan dengan penggabungan dicision tree dengan melakuakn training terhadap sempel data yang dimiliki. Semakin banyak dicision tree maka data yang di dapat akan semakin akurat. Dibawah ini merupakan salah satu ilustrasi penggunaan Random Forest..

2. Jelaskan cara membaca dataset khusus dan artikan makna setiap file dan isi field masing masing file

Pertama download dataset terlebih dahulu lalu buka dengan menggunakan software spyder guna melihat isi dari dataset tersebut. Data tersebut memiliki extensi le bernama .txt dan didalamnya terdapat class dari eld. Misalnya saja pada data jenis burung memiliki le index dan angka, dimana index berisi angka yang memiliki makna berupa jenis burung atau bahkan nama burung sedangkan eld memiliki isi nilai berupa 0 dan 1 yang dimana sifatnya boolean atau Ya dan Tidak. Hal ini dikarenakan komputer hanya dapat membaca bilangan biner maka dari itu eld yang di isikan berupa angka. Artinya angka 0 berarti tidak dan angka 1 berarti Ya.

3. Jelaskan apa itu Cross Validation

Cross Validation merupakan sebuah teknik validasi model yang digunakan untuk menilai bagaimana hasil analisis statistik akan digeneral-



Gambar 3.67 Random Forest

isasi ke kumpulan data independen. Cross validation digunakan dengan tujuan prediksi, dan bila kita ingin memperkirakan seberapa akurat model model prediksi yang dilakukan dalam sebuah praktek. Tujuan dari cross validation yaitu untuk mendenisikan dataset guna menguji dalam fase pelatihan untuk membatasi masalah seperti overfitting dan underfitting serta mendapatkan wawasan tentang bagaimana model akan digeneralisasikan ke set data independen.

4. Jelaskan apa arti score 44 % pada random forest, 27 % pada decision tree dan 29 % dari SVM

Dimana Score 44 % diperoleh dari hasil pengelahan dataset jenis burung. Dimana akan dilakukan proses pembagian data testing dan data training lalu diproses dan menghasilkan score sebanyak 44 % dimana menjelaskan bahwa score tersebut digunakan sebagai pembanding dalam tingkat keakuratannya. Pada decision tree akan memperoleh data lebih kecil yaitu sebanyak 27 % hal ini dikarenakan data yang diolah menggunakan decision tree dibagi menjadi beberapa tree dan lalu disimpulkan untuk mendapatkan data yang akurat. Pada SVM akan memperoleh score sebanyak 29 % hal ini dikarenakan data yang dimiliki masih bernilai netral sehingga tingkat keakuratannya masih belum jelas.

5. Jelaskan bagaimana cara membaca confusion matriks dan contohnya memakai gambar atau ilustrasi sendiri. Untuk membaca confusion matriks dapat menggunakan source code sebagai berikut,

```
import numpy as np
```

```
np.set_printoptions(precision=2)
plt.figure(figsize=(60,60), dpi=300)
plot_confusion_matrix(cm, classes=birds, normalize=True)
plt.show()
```

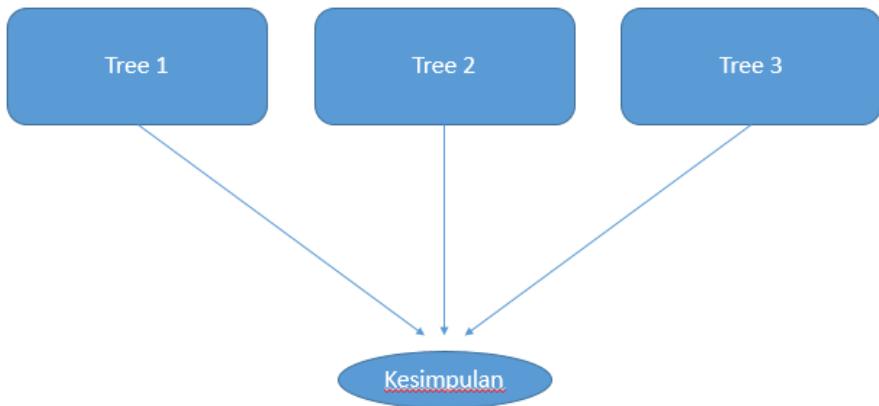
Dimana numpy akan mengurus semua data yang berhubungan dengan matrix. Pada source code tersebut digunakan dalam melakukan read pada dataset burung dengan menggunakan metode confusion matrix. Dalam confusion matrix memiliki 4 istilah yaitu True Positive yang merupakan data posotif yang terditeksi benar, True Negatif yang merupakan data negatif akan tetapi terditeksi benar, False Positif merupakan data negatif namun terditeksi sebagai data positif, False Negatif merupakan data posotif namun terditeksi sebagai data negatif. Adapun contoh hasil read dataset menggunakan confusion matrix dapat dilihat pada figure ??

		True Values	
		True	False
Prediction	True	TP Correct result	FP Unexpected result
	False	FN Missing result	TN Correct absence of result

Gambar 3.68 Confusion Matrix.

6. Jelaskan apa itu voting pada random forest disertai dengan ilustrasi gambar sendiri.

Voting merupakan proses pemilihan dari tree yang dimana akan dimunculkan hasilnya dan disimpulkan menjadi informasi yang pasti. Untuk kebih jelasnya saya akan memberikan sebuah contoh bagaimana voting bekerja.



Gambar 3.69 Voting.

Dimana ditunjukkan pada figure ?? terdapat 3 tree. Dalam tree tersebut akan dilakukan proses voting. Saya akan memberikan contoh kasus, dimana akan diadakan voting untuk menentukan sebuah mobil. Dalam tree akan diberikan sejumlah data misalnya saja data tersebut berupa gambar, yang dimana data tersebut akan dipilih dengan cara voting. Hasil voting akhir dari setiap tree menunjukkan mobil jazz, yang berarti kesimpulan dari data yang telah diberikan menyatakan gambar tersebut adalah mobil jazz. Bagaimana apabila terjadi perbedaan data misalnya saja pada tree 1 dan 2 menyatakan mobil jazz sedangkan pada tree 3 menyatakan mobil yaris, maka kesimpulan yang di ambil adalah mobil jazz dikarenakan hasil voting terbanyak adalah mobil jazz.

3.4.2 Praktikum

1. pandas

pada baris ke satu yaitu perintah mengimport library padas pada python atau anaconda kemudian di inisialisasikan menjadi karakter. selanjutnya ada sebuah array yang berisi nama cewek. selanjutnya penggunaan array tipe series dan yang terakhir perintah print untuk menampilkan data pada karakter ??.

```
1 import pandas as alit
2 cewek = {"List Nama Cewek Alit" : [ 'Triya' , 'Mirla' , 'Alit' , '
   Merita' ]}
3 af = alit.DataFrame(cewek)
4 print('Alit sayang ' + af)
```

```
In [9]: runfile('D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapt
wdir='D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3')
          Nama Cewek
0    Alit sayang Triya
1    Alit sayang Mirla
2    Alit sayang Alit
3    Alit sayang Merita
```

Gambar 3.70 hasil

2. numpy

Source Code baris pertama menjelaskan akan melakukan import pada library numpy dan di rename dengan alit. Selanjutnya kita akan membuat matrix dengan numpy dengan menggunakan fungsi eye. baris kedua selanjutnya kita akan memanggil matrix identitas 10x10. Fungsi eye disini berguna untuk memanggil matrix identitas dengan jumlah colom dan baris sesuai yang ditentukan. Disini saya telah menentukan 10 colom

dan baris maka dari itu hasilnya akan memunculkan matrix identitas 10x10, untuk lebih jelasnya dapat dilihat pada figure ??

```

1 import numpy as alit
2 matrix_one = alit.eye(10)
3 matrix_one
4 print (matrix_one)
```

```
In [11]: runfile('D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3/2.py',
wdir='D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3')
[[1. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 1. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 1. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 1. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 1. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 1. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 1.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Gambar 3.71 hasil

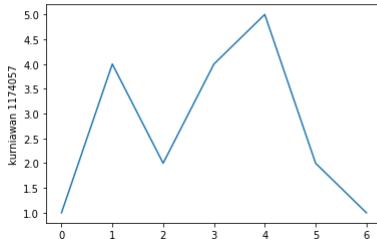
3. matplotlib

Arti tiap baris aplikasi sederhana matplotlib item Pada baris pertama akan melakukan import library Matplotlib dan di rename menjadi fajar. Pada baris kedua kita akan memberikan nilai plot atau grafik pada library fajar. Pada baris ketiga kita akan memberikan nama atau label pada grafik tersebut. Pada baris terakhir kita akan melakukan show sehingga grafik dapat kita lihat. Jalankan source code diatas dengan spyder sehingga hasilnya akan nampak seperti pada figure ??

```

1 import matplotlib.pyplot as fajar
2 fajar.plot([1,4,2,4,5,2,1])
3 fajar.ylabel('kurniawan 1174057')
4 fajar.show()
```

```
In [12]: runfile('D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3/3.py',
wdir='D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3')
```



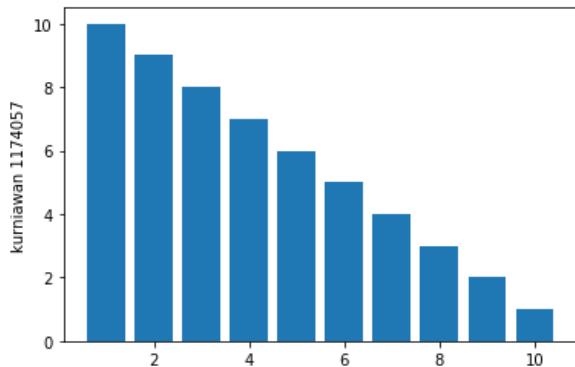
Gambar 3.72 hasil

Selanjutnya perhatikan source code berikut ini,

```
1 import matplotlib.pyplot as alit
2 a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3 f = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
4 alit.bar(a, f)
5 alit.ylabel('kurniawan 1174057')
6 alit.show()
```

Pada baris pertama akan melakukan import library Matplotlib dan di rename menjadi alit. Pada baris kedua dan ketiga kita akan menentukan nilai pada variabel a dan f. Pada baris keempat kita akan membuat diagram batang dengan fungsi bar. Pada baris kelima kita akan memberikan nama atau label pada diagram batang tersebut. Pada baris terakhir kita akan melakukan show sehingga kita dapat melihat hasilnya. Jalankan source code tersebut di dalam spyder maka hasilnya akan nampak seperti pada figure ??

```
In [18]: runfile('D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3/4.py', wdir='D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3')
```



Gambar 3.73 Aplikasi Sederhana Dengan Matplotlib Diagram Batang

4. Random Forest

Arti tiap baris hasil codingan random forest pada baris pertama random forest di import dari sklearn dengan ketentuan yaitu Nilai default untuk parameter yang mengontrol ukuran pohon (mis. Max_depth, min_samples_leaf, dll.) Mengarah ke pohon yang tumbuh besar dan tidak di-unsun yang berpotensi sangat besar pada beberapa set data. Untuk mengurangi konsumsi memori, kompleksitas dan ukuran pohon harus dikontrol dengan menetapkan nilai parameter tersebut. hasil pada gambar ?? .

```
1 from sklearn.ensemble import RandomForestClassifier  
2 from sklearn.datasets import make_classification  
3 X, y = make_classification(n_samples=1000, n_features=4,  
4 n_informative=2, n_redundant  
=0
```

```

5         random_state=0, shuffle=False)
6 clf = RandomForestClassifier(max_depth=2, random_state=0)
7 clf.fit(X, y)
8 print(clf.feature_importances_)
9 print(clf.predict([[0, 0, 0, 0]]))

```

```

In [19]: runfile('D:/Data Alit/Kuliah/SEMESTER VI/KB38/src/1174057/chapter3/5.py',
wdir='D:/Data Alit/Kuliah/SEMESTER VI/KB38/src/1174057/chapter3')
[0.17287856 0.80608704 0.01884792 0.00218648]
[1]
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20 to
100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)

```

Gambar 3.74 hasil

5. Confusion Matrix

arti codingan pada hasil tiap codingan confusion matrix pada baris pertama codingan tersebut mendeskripsikan atau mengimport confusion matrix dari sklearn kemudian dibuat variabel y_true untuk nilai target ground truth (benar). y_pred untuk Taksiran target seperti yang dikembalikan oleh classifier. lalu menampilkan kedua variabel. hasil pada gambar ?? .

```

1 from sklearn.metrics import confusion_matrix
2 y_true = [2, 0, 2, 2, 0, 1]
3 y_pred = [0, 0, 2, 2, 0, 2]
4 confusion_matrix(y_true, y_pred)

```

```

In [26]: from sklearn.metrics import confusion_matrix
...: y_true = [2, 0, 2, 2, 0, 1]
...: y_pred = [0, 0, 2, 2, 0, 2]
...: confusion_matrix(y_true, y_pred)
Out[26]:
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]], dtype=int64)

```

Gambar 3.75 hasil

6. SVM dan Decision Tree

Seperti pengklasifikasi lainnya, DecisionTreeClassifier mengambil input dua array: array X, jarang atau padat, dengan ukuran n_samples, n_features memegang sampel pelatihan, dan array Y dari nilai integer, ukuran n_samples, Atau, probabilitas setiap kelas dapat diprediksi. Seperti pengklasifikasi lainnya, SVC, NuSVC dan LinearSVC mengambil input dua array: array X ukuran n_samples, n_features memegang sampel pelatihan, dan array y label kelas (string atau bilangan bulat), ukuran n_samples:. hasil pada gambar ?? .

```

1 from sklearn import tree
2 X = [[0, 0], [1, 1]]
3 Y = [0, 1]
4 clf = tree.DecisionTreeClassifier()
5 clf = clf.fit(X, Y)

```

```

6 clf.predict([[2., 2.]])
7
8 from sklearn import svm
9 X = [[0, 0], [1, 1]]
10 y = [0, 1]
11 clf = svm.SVC()
12 clf.fit(X, y)
13 clf.predict([[2., 2.]])

```

In [39]: `from sklearn import tree`
`....: X = [[0, 0], [1, 1]]`
`....: Y = [0, 1]`
`....: clf = tree.DecisionTreeClassifier()`
`....: clf = clf.fit(X, Y)`
`....: clf.predict([[2., 2.]])`

Out[39]: `array([1])`

In [40]: `from sklearn import svm`
`....: X = [[0, 0], [1, 1]]`
`....: y = [0, 1]`
`....: clf = svm.SVC()`
`....: clf.fit(X, y)`
`....: clf.predict([[2., 2.]])`

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: default value of gamma will change from 'auto' to 'scale' in version 0.22. This warning is shown because 'gamma' is set to its default value of 'auto'. It will be better for unscaled features. Set gamma explicitly to 'auto' or 'scale'.

"avoid this warning.", FutureWarning)

Out[40]: `array([1])`

Gambar 3.76 hasil

7. Cross Validation

digunakan untuk memeriksa akurasi dari ketepatan hasil pengolahan data tersebut maka akan didapat nilai rata-rata 60 persen dari hasil pengolahan data tersebut untuk lebih jelasnya dapat dilihat pada gambar pada codingan tersebut pada baris ke satu melakukan import library dari sklern kemudian pada baris selanjutnya mengisi nilai skor dengan nilai pada variabel lontong setelah hal tersebut dilakukan kemudian data tersebut di eksekusi. berikut merupakan hasil dari code tersebut dapat dilihat pada gambar ?? .

```

1 from sklearn.model_selection import cross_val_score
2 scores = cross_val_score(bdg, dago_att, dago_pass, cv=5)
3 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
   std() * 2))

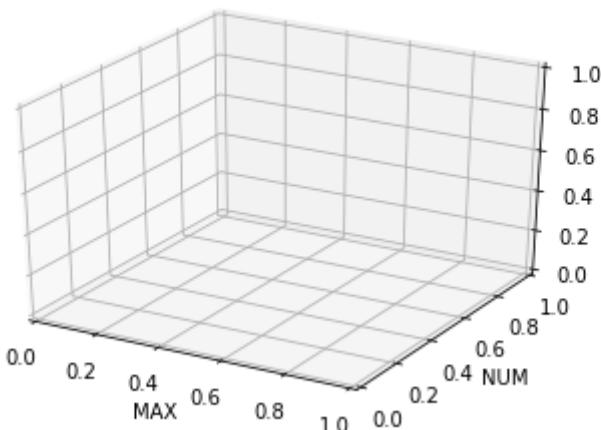
```

```
In [61]: runfile('D:/SEMESTER 3/ML/ML PRACTICE/ML PRACTICE.ipynb', 2)
Accuracy: 0.59 (+/- 0.05)
```

Gambar 3.77 hasil

8. program pengamatan arti dari hasil program pengamatan. perogram pengamatan dapat mengamati dari 3 aspek diatas yaitu svm, random, dan decision tree. Yang memiliki variabel X Y Z dan di tampilkan dalam bentuk grafik pada gambar ??.

```
1 import matplotlib.pyplot as plt
2 from mpl_toolkits.mplot3d import Axes3D
3 from matplotlib import cm
4 fig = plt.figure()
5 fig.clf()
6 ax = fig.gca(projection='3d')
7 plt.xlabel('MAX')
8 plt.ylabel('NUM')
9 plt.zlabel('AVG')
10 ax.scatter(x, y, z)
11 ax.set_zlim(0.2, 0.5)
12 ax.set_xlabel('Max features')
13 ax.set_ylabel('Num estimator')
14 ax.set_zlabel('Avg accuracy')
15 plt.show()
```



Gambar 3.78 hasil

3.4.3 Penanganan Error

Screenshot error

1. Gambar error

```
In [63]: runfile('D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chap-
  wdir='D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3')
Traceback (most recent call last):

  File "C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interacti-
line 3325, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)

  File "<ipython-input-63-6971754af445>", line 1, in <module>
    runfile('D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3'
Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3')

  File "C:\ProgramData\Anaconda3\lib\site-packages\spyder_kernels\custo-
\spydercustomize.py", line 827, in runfile
    execfile(filename, namespace)

  File "C:\ProgramData\Anaconda3\lib\site-packages\spyder_kernels\custo-
\spydercustomize.py", line 110, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter3/1.py"
    print('Alit sayang + af)
                           ^
SyntaxError: EOL while scanning string literal
```

Gambar 3.79 hasil

Code Errornya

```
import pandas as alit
cewek = {"List Nama Cewek Alit" : ['Triya','Mirla','Alit','Merita']}
af = alit.DataFrame(cewek)
print('Alit sayang + af)
```

pada bagian print, perintah ketika akan menjalankan program. terdapat kesalahan pada tanda petik yang kurang sehingga program tidak dapat dijalankan.

perbaiki code error

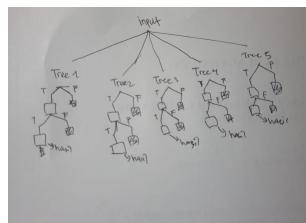
```
import pandas as alit
cewek = {"List Nama Cewek Alit" : ['Triya','Mirla','Alit','Merita']}
af = alit.DataFrame(cewek)
print('Alit sayang ' + af)
```

3.5 1174039 - Liyana Majdah Rahma

3.5.1 Teori

1. Jelaskan apa itu random forest, sertakan gambar ilustrasi buatan sendiri.

Random forest adalah sebuah algoritma yang biasanya dipakai untuk mengklasifikasikan suatu data yang berjumlah besar. Dimana pengklasifikasianya itu dapat menggunakan pohon yang digabungkan serta melewati training terlebih dahulu sebelum pada data sample nya. Dapat dijelaskan dari gambar dibawah bahwa klasifikasi tree ini diambil dari data yang paling banyak.



Gambar 3.80 Random Forest

2. Jelaskan cara membaca dataset khusus dan artikan makna setiap le dan isi eld masing masing le

langkah pertama Download terlebih dulu data yang akan dibaca, lalu buka aplikasi spyder kemudian jalankan kode nya. data yang terdapat pada file tersebut adalah data folder ATTRIBUTE, IMAGES, PARTS yang memiliki kegunaannya sendiri yang dimana pada penggunaannya data yang dipakai adalah data image_attribute_label pada folder attribute, data image_class_labels dan data classes. file image_attribute_label berguna sebagai data awal yang digunakan untuk membaca data attribute yang terdapat pada masing - masing gambar burung yang ada. sedangkan file image_class_label berguna sebagai data yang akan membuat kolom baru pada dataset yang fungsinya adalah untuk memasukan hasil dari semua data yang dimiliki oleh imgatt2. dan file classes berguna sebagai dataset yang akan dipanggil oleh fungsi code untuk menampilkan nama dari data burung yang dimiliki. file image_attribute_label berisi tentang data attribute yang ada pada data gambar file burung yang dimiliki difolder image pada CUB-200-2011 file image_class_label berisi tentang data yang dimiliki oleh attribute dari image_attribute_label dimana data yang bernilai atau memiliki nilai disusun hingga menghasilkan data yang mudah dipahami. file classes berisi tentang data yang berguna untuk menampilkan data nama dari setiap data jenis burung yang dimiliki.

3. Jelaskan apa itu Cross Validation.

Cross Validation adalah suatu teknik yang digunakan untuk memvalidasi sebuah model serta untuk menilai pengeneralisasian dari kumpulan data independen hasil statistik analis.

- Jelaskan apa arti score 44 % pada random forest, 27 % pada decision tree dan 29 % dari SVM.

arti score 27% pada decision tree adalah presentasi hasil dari perhitungan dataset acak, dan arti score 29% dari SVM adalah hasil pendekatan neural network. Hasil tersebut didapat dari hasil validasi silang untuk memastikan bahwa membagi training test dengan cara yang berbeda. Jadi 44% untuk random forest, 27% untuk pohon keputusan, dan 29% untuk SVM. Itu merupakan persentase keakurasi prediksi yang dilakukan pada saat testing menggunakan label pada dataset yang digunakan. Score mendefinisikan aturan evaluasi model.

- Jelaskan bagaimana cara membaca confusion matrix dan contohnya memakai gambar atau ilustrasi sendiri.

Confusion matrix menggunakan rumus perhitungan dengan 4 keluaran, yang pertama adalah :

		Prediksi	
		Negative	Positive
Fakta	Negatif	a	b
	Pozitif	c	d

Gambar 3.81 Confusion Matrix

- Recall

$$d/(c + d) \quad (3.5)$$

- Precision

$$d/(b + d) \quad (3.6)$$

- Accuracy

$$(a + c)/(a + b + c + d) \quad (3.7)$$

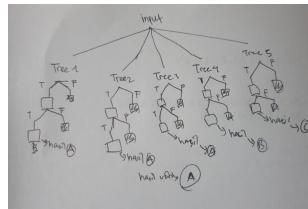
- Error Rate

$$(b + c)/(a + b + c + d) \quad (3.8)$$

- Jelaskan apa itu voting pada random forest disertai dengan ilustrasi gambar sendiri.

Voting itu adalah sebuah hasil dari sebuah decision tree, yang nanti hasil akhirnya akan menjadi hasil dari random forest. Dijelaskan bahwa terdapat 5 buah tree dimana masing-masing terdapat tree satu, tree dua, tree tiga, tree empat, dan tree 5. tree satu menghasilkan A, sedangkan

tree empat dan lima menghasilkan data C. Maka dapat disimpulkan data yang paling banyak diperoleh adalah data A.



Gambar 3.82 Vote

3.5.2 Praktek

1. Pandas Pada baris pertama kita mengimport library pandas dan menamainya sebagai pan , lalu memasukkan data kedalam variable data. setelah itu membuat dataframe dengan data yang telah di masukkan tadi, lalu membuat variable bernama cari untuk melihat semua data dari attribute Nama lalu print variable tersebut.

```
1 import pandas as pd
2 data = {'Film': ['Dora', 'Doraemon', 'Barbie'], 'Tahun': [2011, 2015, 2016]}
3 pd.DataFrame(data)
4 df = data['Film']
5 print(b)
```

2. numpy Baris pertama adalah untuk mengimport library numpy dan menamainya sebagai num, lalu membuat sebuah variable bernama mat dan menggunakan fungsi arrange untuk membuat angka berurutan dari 1 sampai 25, mengapa ditulis satu sampai 26 karena urutan dimulai dari 0, lalu menggunakan reshape untuk mengubahnya menjadi matrix. dan terakhir print variable mat.

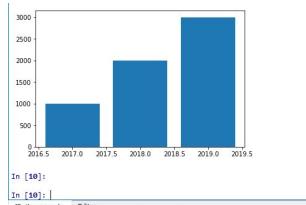
```
8 import numpy as num  
9 mat = num.arange(1,26).reshape(5,5)  
10 print(mat)
```

Gambar 3.85 Numpy

3. matplotlib Pertama import pyplot dari library matplotlib dan menamainya sebagai plt, lalu memasukkan data yang diinginkan kedalam variable x dan y . lalu menggunakan fungsi bar untuk membuat grafik bar yang isi datanya adalah x, dan y, lalu memberikan label pada sumbu x dan y dan memberikan judul pada grafik tersebut, dan menggunakan show() untuk menampilkan grafik

```
1 from matplotlib import pyplot as plt
2 d = [2017, 2018, 2019]
3 x = [1000, 2000, 3000]
4 plt.bar(d, x)
5 plt.xlabel('tahun')
6 plt.ylabel('hasil produksi')
7 plt.title('jelajah hasil produksi per tahun')
8 plt.show()
```

Gambar 3.87 Matplotlib



Gambar 3.88 Hasil Matplotlib

4. Random Forest

Dari sklearn.ensamble mengimport RandomForestClassifier dan memberikan ketentuan dimana maksimal datanya adalah 50 dengan keadaan random serta estimatornya 100 dan dimasukkan kedalam variable clf lalu itu variabel clf di running berdasarkan data training dan data label yang telah di denisikan jumlahnya. kemudian variabel clf di running berdasarkan data training paling atas untuk memunculkan hasil data training lima paling atas. setelah itu data tersebut di di running scorenya untuk melihat tingkat akurasi yang dia kerjakan maka tingkat akurasi yang dihasilkan berada pada kisaran 0,448 atau kisaran 44%.

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_features=50, random_state=0, n_estimators=100)
clf.fit(df_train_att, df_train_label)
print(clf.predict(df_train_att.head()))
clf.score(df_test_att, df_test_label)
```

Gambar 3.89 Random Forest

```
In [122]: from sklearn.ensemble import RandomForestClassifier
...: clf = RandomForestClassifier(max_features=50, random_state=0,
...: n_estimators=100)
...: clf.fit(df_train_att, df_train_label)
...: print(clf.predict(df_train_att.head()))
...: clf.score(df_test_att, df_test_label)
[ 20  56  56  101  101]
Out[122]: 0.44825700375581586
```

Gambar 3.90 Hasil Random Forest

5. Confussion Matrix dari sklearn.metrics mengimport confusion matrix kemudian dibuat variabel pred_labels dengan di isikan clf prdic df test att setelah itu membuat variabel cm yang isinya terdapat data yang di buat confusion matrix berdasarkan data test setelah variabel cm akan di running yang mana akan menghasilkan gambar berupa matrix matrix tersebut berisi nilai kebenaran yang mendekati nilai benar.

```
from sklearn.metrics import confusion_matrix
pred_labels = clf.predict(df_test_att)
cm = confusion_matrix(df_test_label, pred_labels)
```

Gambar 3.91 Confusion Matrix

```
In [113]: from sklearn.metrics import confusion_matrix
...: pred_labels = clf.predict(df_test_att)
...: cm = confusion_matrix(df_test_label, pred_labels)
...: cm
Out[113]:
array([[ 7,  0,  0, ...,  0,  0,  0],
       [ 1, 12,  0, ...,  0,  1,  0],
       [ 1,  1,  7, ...,  0,  0,  8],
       ...,
       [ 0,  0,  1, ...,  4,  0,  0],
       [ 0,  0,  0, ...,  0, 11,  0],
       [ 0,  0,  0, ...,  0,  0, 10]], dtype=int64)
```

Gambar 3.92 Hasil Coonfusion Matrix

6. Decision Tree dan SVM masukan terlebih dahulu library tree setelah itu buat variabel clftree yang berisi decision tree setelah itu masukan nilai data training dan label yang telah di deklarasikan tadi setelah itu running variabel tersebut untuk mendapatkan score 0,266 kisaran 26 sampai 27% akurasinya

```
from sklearn import tree
clftree = tree.DecisionTreeClassifier()
clftree.fit(df_train_att, df_train_label)
clftree.score(df_test_att, df_test_label)
```

Gambar 3.93 Desicion Tree

```
In [114]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(df_train_att, df_train_label)
...: clftree.score(df_test_att, df_test_label)
Out[114]: 0.26689545934530096
```

Gambar 3.94 Hasil Desicion Tree

Import terlebih dahulu library svm nya setelah itu buat variabel clfsvm yang berarti berisi nilai data training dan data label dan pendeklarasian svm itu sendiri setelah itu di running untuk mendapatkan nilai akurasinya atau score sebesar 0,283 atau dalam kisaran 23%.

```
from sklearn import svm
clfsvm = svm.SVC()
clfsvm.fit(df_train_att, df_train_label)
clfsvm.score(df_test_att, df_test_label)
```

Gambar 3.95 SVM

```
In [115]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(df_train_att, df_train_label)
...: clfsvm.score(df_test_att, df_test_label)
Out[115]: 0.4799366420274551
```

Gambar 3.96 Hasil SVM

7. Cross Validation mengimport cross_val_score dari sklearn.model_selection , lalu memasukkan variable variable yang akan di ukur keakuratan dari model yang digunakan lalu menampilkan nya di console, disini digunakan variable clf yaitu random forest.

```
:from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, df_train_att, df_train_label, cv = 5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

Gambar 3.97 Cross Val Rand Forest

```
In [132]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(clf, df_train_att, df_train_label, cv = 5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
Accuracy: 0.45 (+/- 0.01)
```

Gambar 3.98 Hasil Cross Validaiton Random Forest

- Disini memasukkan variable variable yang akan di ukur keakuratan dari model yang digunakan lalu menampilkan nya di console, disini digunakan variable clftree yaitu desicion Tree.

```
scoretree = cross_val_score(clftree, df_train_att, df_train_label, cv = 5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scoretree.mean(), scoretree.std() * 2))
```

Gambar 3.99 Cross Vadation Desicion Tree

```
In [137]: scoretree = cross_val_score(clftree, df_train_att, df_train_label, cv = 5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoretree.mean(), scoretree.std() * 2))
Accuracy: 0.26 (+/- 0.02)
```

Gambar 3.100 Hasil Cross Vadation Desicion Tree

- Disini memasukkan variable variable yang akan di ukur keakuratan dari model yang digunakan lalu menampilkan nya di console, disini digunakan variable clfsvm yaitu svm.

```
scoresvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv = 5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2))
```

Gambar 3.101 Cross Vadation SVM

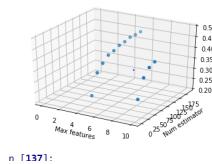
```
In [134]: scoresvsm = cross_val_score(clfsvm, df_train_att, df_train_label, cv = 5)
...: print("Accuracy: %.2f (+/- %.2f)" % (scoresvsm.mean(), scoresvsm.std() * 2))
Accuracy: 0.87 (+/- 0.01)
```

Gambar 3.102 Hasil Cross Vadation SVM

8. Program Pengamatan dan Hasil Pengamatan. perogram pengamatan ini menggunakan library matplotlib supaya hasil dari presentase hasil random forest, svm dan decision tree dapat di bandingkan dengan membuat variabel X Y Z kemudian memberikan label untuk setiap dimensinya untuk lebih jelas dapat dilihat gambar yang menunjukan hasil perbandingan presentase dari tiga metode tersebut.

Gambar 3.103 Program Pengamatan

Gambar 3.104 Hasil Program Pengamatan



Gambar 3.105 Grafik Program Pengamatan

3.5.3 Penanganan Error

3.5.3.1 Skrinshoot error

1. Error 1

FileNotFoundException: File b'0:\\Semester6\\AI\\Chapter3\\(CUB_200_2011)x07ttributes\\

Gambar 3.106 Error 1

3.5.3.2 Kode Error dan Tipe Error

1. Error 1 type FileNotFoundError

```
imgatt = pd.read_csv("D:\Semester1\AI\Chapter9\08_200_2011\attributes\image_attribute_labels.txt", sep='\t', encoding='utf-8')

imgatt.head()
imgatt.shape
```

Gambar 3.107 Kode Error 1

3.5.3.3 Solusi

1. Mengganti back slash menjadi slash biasa

BAB 4

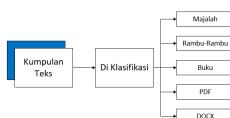
CHAPTER 4

4.1 Faisal Najib Abdullah / 1174042

4.1.1 Teori

1. Jelaskan apa itu klasifikasi teks, sertakan gambar ilustrasi buatan sendiri.

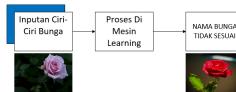
klasifikasi teks adalah cara untuk memilah-milah teks berdasarkan parameter tertentu baik itu jenis teks atau jenis dari dokumen yang terdapat kumpulan teks didalamnya, sedangkan teks itu sendiri merupakan sekumpulan kata yang dapat dibaca. bisa berupa buku, majalah, rambu-rambu dan lain sebagainya.



Gambar 4.1 contoh klasifikasi teks

2. Jelaskan mengapa klasifikasi bunga tidak bisa menggunakan machine learning, sertakan ilustrasi sendiri.

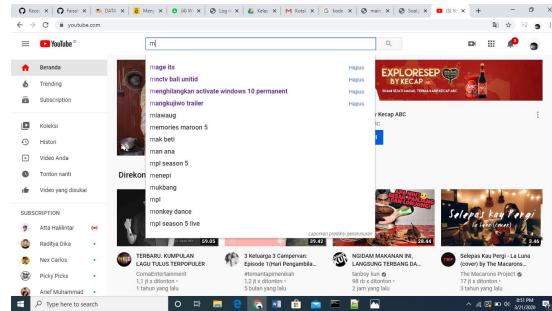
Klasifikasi bunga tidak dapat menggunakan mesin learning dikarenakan jenis-jenis bunga banyak yang mirip bahkan banyak bunga yang serupa tetapi tidak sama. oleh karena itu klasifikasi bunga tidak bisa di gunakan oleh mesin learning dikarenakan jika salah satu inputan ciri-ciri dari siatu bunga di inputkan kemungkinan jawaban dari mesin learning itu tidak tepat contoh dimasukan inputan ciri ciri bunga mawar putih kemudian mesin learning menjawab bahwa itu bunga mawar merah.



Gambar 4.2 contoh klasifikasi bunga

3. Jelaskan bagaimana teknik pembelajaran mesin pada teks pada kata-kata yang digunakan di youtube,jelaskan arti per atribut data csv dan sertakan ilustrasi buatan sendiri.

cara pembelajaran teks yang di gunakan youtube yaitu dengan cara merekam data yang sering di inputkan oleh user pada menu pencarian youtube. sehingga pada saat user akan mencari data yang serupa seringkali youtube menyediakan opsi atau rekomendasi-rekomendasi dari pencaharian. contoh saya menuliskan m maka muncul opsi pilihan master chep dan lainya yang berawalan m rekomendasi yang muncul merupakan kata-kata yang sering di cari oleh banyak user atau sering di buka oleh user itu sendiri.



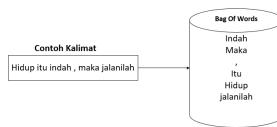
Gambar 4.3 contoh teknik pembelajaran mesin

4. Jelaskan apa yang dimaksud vektorisasi data.

vektorisasi data merupakan pemecahan data menjadi bagian-bagian yang lebih sederhana contoh pada satu paragraf terdiri dari 200 kata kemu-

dian dilakukan vektorisasi dengan cara membagi-bagi kata dalam paragraf tersebut ke dalam kalimat-kalimat yang terpisah kemudian dipecah lagi menjadi data dalam perkata selanjutnya kata-kata tersebut dijemahkan.

- Jelaskan apa itu bag of words dengan kata-kata yang sederhana dan ilustrasi sendiri. bag of words merupakan proses penyederhanaan kata-kata yang asalnya tersirat dalam satu kalimat atau satu paragraf diubah menjadi perkataan kemudian kata-kata tersebut dikumpulkan menjadi satu kelompok tanpa ada arti dari kata-kata yang telah dikumpulkan tersebut lalu dihitung frekuensi kemunculan dari kata tersebut.



Gambar 4.4 contoh bag of words

- Jelaskan apa itu TF-IDF, ilustrasikan dengan gambar sendiri. TF-IDF merupakan metode untuk menghitung bobot dari kata yang sering muncul pada suatu kalimat. metode ini menghitung nilai TF atau Term Frequency dan IDF atau Inverse Document Frequency pada setiap kata pada kalimat yang dijadikan acuan kata pada metode ini sering disebut token adapun rumus dari metode ini.



Gambar 4.5 contoh TF-IDF

4.1.2 Praktek Program

- import data pandas dan 500 baris data dummy kemudian dijelaskan tiap barisnya.

```

1 # In [1]: mengimport librari padas yang di gunakan
2 # untuk membaca file tex atau csv
3 import pandas as pd
4 #membaca file csv menggunakan fungsi read csv dari padas
5 data_forest = pd.read_csv("D:/NAJIB/SEMESTER_6_NAJIB/AI/
  Chapter4/Salaries.csv")
6 # In [2]: untuk melihat jumlah dari baris data yang telah di
  import
  
```

```

7 print(len(data_forest))
8 # In[3]: untuk melihat lima baris pertama data yang telah di
     import
9 print(data_forest.head())
10 # In[4]: untuk mengetahui banyak baris dan kolom dari data
      yang
11 # telah di import.
12 print(data_forest.shape)

```

The screenshot shows a Jupyter Notebook cell titled 'Run: 2.1'. It displays the output of the code, which includes the first five rows of a DataFrame named 'data_forest'. The columns shown are Id, EmployeeName, Agency, and Status. The rows are indexed from 0 to 4. The output also shows the total number of rows (148654) and columns (13). The status of all rows is 'NaN'.

	Id	EmployeeName	Agency	Status
0	1	NATHANIEL FORD	San Francisco	NaN
1	2	GARY JIMENEZ	San Francisco	NaN
2	3	ALBERT PARDINI	San Francisco	NaN
3	4	CHRISTOPHER CHONG	San Francisco	NaN
4	5	PATRICK GARDNER	San Francisco	NaN

[5 rows x 13 columns]
(148654, 13)

Process finished with exit code 0

Gambar 4.6 hasil

2. memecah data prame menjadi dua yag pertama 450 dan kedua sisanya

```

1 # In [5]: membuat data training dan data testing
2 # jumlah baris data training sebanyak 450 baris
3 data_training = data_forest[:450]
4 # jumlah baris data testing dari hasil pengurangan 523 - 450
5 data_testing = data_forest[450:]

```

The screenshot shows a Jupyter Notebook cell titled 'Run: Run1'. It displays the output of the code, which includes the first 450 rows of the 'data_training' DataFrame. The columns shown are Id, EmployeeName, Agency, and Status. The rows are indexed from 443 to 450. The output also shows the total number of rows (450) and columns (13). The status of all rows is 'NaN'.

	Id	EmployeeName	Agency	Status
443	444	SANDRA HUANG	San Francisco	NaN
444	445	SAHIR PUTRUS	San Francisco	NaN
445	446	ELAINE COLEMAN	San Francisco	NaN
446	447	DENISE BAILEY	San Francisco	NaN
447	448	ANNETTE HOBRUCKER-PFEIFER	San Francisco	NaN
448	449	WENDY STILL	San Francisco	NaN
449	450	CHUTEH KOTAKE	San Francisco	NaN

[450 rows x 13 columns]

Process finished with exit code 0

Gambar 4.7 hasil

3. praktek vektorisasi

berikut merupakan codingan untuk melakukan vektorisasi data berupa teks dalam vormat csv

```

1 # In [1]:
2 #melakukan import pandas untuk membaca file csv
3 import pandas as pd
4 data_komen=pd.read_csv("D:/NAJIB/SEMESTER_6_NAJIB/AI/Chapter4
   /Youtube04-Eminem.csv")

```

Run: Run1						
▶	↑	148647	148648	Joann Anderson	...	San Francisco
▼	↓	148648	148649	Leon Walker	...	San Francisco
☰	☷	148649	148650	Roy I Tillary	...	San Francisco
⤒	⤒	148650	148651	Not provided	...	San Francisco
⤓	⤓	148651	148652	Not provided	...	San Francisco
🖨️	🖨️	148652	148653	Not provided	...	San Francisco
🖨️	🖨️	148653	148654	Joe Lopez	...	San Francisco
[148204 rows x 13 columns]						
Process finished with exit code 0						

Gambar 4.8 hasil

```

5 # In [2]:
6 #mengelompokkan komentar spam dan bukan spam
7 spam=data_komen.query('CLASS == 1')
8 nospam=data_komen.query('CLASS == 0')
9 # In [3]: memanggil lib vektorisasi
10 #melakukan fungsi bag of word dengan cara menghitung semua
     kata
11 #yang terdapat dalam file
12 from sklearn.feature_extraction.text import CountVectorizer
13 vectorizer = CountVectorizer()
14 # In [3]: memilih colom CONTENT untuk dilakukan vektorisasi
15 #melakukan bag of word pada dataframe pada colom CONTENT
16 data_vektorisasi = vectorizer.fit_transform(data_komen['
     CONTENT'])
17 # In [4]: melihat isi vektorisasi
18 data_vektorisasi
19 # In [5]: melihat isi data pada baris ke 349
20 print(data_komen['CONTENT'][349])
21 # In [6]: melihat daftar kata yang di vektorisasi
22 #feature_names merupakan digunakan untuk mengambil nama
23 #kolomnya ada apa saja
24 dk=vectorizer.get_feature_names()
25 # In [7]: akan melakukan randomisasi pada database nya supaya
26 #semipurna saat melakukan klasifikasi
27 acak_acak = data_komen.sample(frac=1)
28 # In [8]: membuat data traning dan testing
29 dk_train=acak_acak[:300]
30 dk_test=acak_acak[300:]
31 # In [9]: melakukan training pada data training dan di
     vektorisasi
32 dk_train_att=vectorizer.fit_transform(dk_train['CONTENT'])
33 print(dk_train_att)
34 # In [10]: melakukan testing pada data testing dan di
     vektorisasi
35 dk_test_att=vectorizer.transform(dk_test['CONTENT'])
36 print(dk_test_att)
37 # In [11]: Dimana akan mengambil label spam dan bukan spam
38 dk_train_label=dk_train['CLASS']
39 print(dk_train_label)
40 dk_test_label=dk_test['CLASS']
41 print(dk_test_label)

```

The screenshot shows a Jupyter Notebook interface. On the left, there's a sidebar with icons for file operations like Run, Save, and Cell. A text cell titled '2,1' contains a message from a duo about recording freestyles and encourages users to like their videos. Below it is a code cell with the number '2,3'. This cell contains a list of tuples representing data points, likely for training a machine learning model. The tuples consist of two values separated by a comma, followed by a colon and a numerical value.

```

Run: 2,3
Hi everyone. We are a duo and we are starting to record freestyles and put them on youtube. If any of you could check it out and like we love doing this. We may not have the best recording equipment but if you listen to our lyrics and rhymes I think you'll like because we love making these videos and we want you to like them as much as possible so feel free to comment and give us pointers!
(0, 217) 1
(0, 762) 1
(0, 318) 1
(0, 575) 1
(0, 1049) 1
(0, 987) 1
(0, 519) 1
(0, 283) 1
(2, 217) 1
(2, 762) 1
(2, 519) 1
(2, 487) 1
(2, 452) 1
(2, 618) 1
(2, 523) 3
(2, 139) 1
(2, 947) 1
(2, 113) 4
(2, 546) 2
(2, 789) 1
(2, 775) 2
(2, 736) 1
(2, 192) 1
(2, 795) 1
(2, 992) 1

```

Gambar 4.9 hasil

lakukan import library pandas yang diinisialisasi menjadi pd setelah itu ada dibuat class data_komen dengan method read_csv untuk membaca file berekstensikan csv yang dimasukan alamatnya pada kurung, lakukan klasifikasi atau pemilihan komentar yang berisi spam atau bukan spam dengan parameter class samadengan 1 merupakan spam dan class samadengan 0 bukan spam setelah itu masukan librari CountVektorizer yang digunakan untuk vektorisasi data kemudian dilanjutkan pada bagian In[103] dibuat variabel yang berisi vektorisasi dari data pada data_komen di field content setelah itu variabel tersebut di running hasilnya menunjukan 350 baris di kali 1738 kolom selanjutnya dicoba untuk memunculkan isi record pada baris ke 349 maka akan muncul isian dari baris tersebut. selanjutnya dibuat variabel dk atau daftar yang berisi data hasil vektorisasi setelah yang terdiri dari variabel acak_acak yang berisi data komen yang di dalamnya dibuat random yang nantinya akan dibut data training dan data testing dengan ketentuan data training 300 dan data testing sebanyak 50 setelah itu data training dilakukan vektorisasi dan data testing juga dilakukan vektorisasi setelah itu kedua data training dan testing tersebut dibuat label dengan parameter field CLASS pada tabel.

4. klasifikasi SVM berikut ini merupakan codingan klasifikasi SVM

```

1 # In [18]:
2 from sklearn import svm

```

```

3 clfsvm = svm.SVC()
4 clfsvm.fit(dk_train_att, dk_train_label)
5 clfsvm.score(dk_test_att, dk_test_label)

```

```

Run: Run2
we love doing this. We may not have the best recording equipment but if you like
because we love making these videos and we want you to like them as much as we
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
      decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
      max_iter=-1, probability=False, random_state=None, shrinking=True,
      tol=0.001, verbose=False)
0.918918918918919

Process finished with exit code 0

```

Gambar 4.10 hasil

melakukan verifikasi import librari svm dari sklearn kemudian membuat variabel clfsvm berisikan method svc setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

5. klasifikasi decision tree berikut ini merupakan codingan klasifikasi decision tree

```

1 # In [17]:
2 from sklearn import tree
3 clftree = tree.DecisionTreeClassifier()
4 clftree.fit(dk_train_att, dk_train_label)
5 clftree.score(dk_test_att, dk_test_label)

```

```

Run: Run2
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
0.918918918918919

Process finished with exit code 0

```

Gambar 4.11 hasil

melakukan verifikasi import librari tree dari sklearn kemudian membuat variabel clftree berisikan method DecisionTreeClasifier setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut agar dapat digunakan pada codingan selanjutnya setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

6. plot confusion matrix berikut merupakan codingan untuk confusion matrix

```
1 # In [15]: Membuat confusion Matrix dan menampilkannya
2 from sklearn.metrics import confusion_matrix
3 pred_labels = clf.predict(dk_test_att)
4 cm = confusion_matrix(dk_test_label, pred_labels)
5 cm
```

Gambar 4.12 hasil

lakukan import library comfusion matrix selanjutnya dilakukan prediksi pada pada data tes nya kemudian data tersebut di masukan kedalam variabel cm dengan method confusion matrix yang di dalamnya terdapat data dari variabel perd label dan dk test label setelah itu variabel cm tersebut di running maka akan memunculkan nilai matrixnya.

7. cross valodation berikut merupakan code untuk cross validation pada codingan pertama yaitu melakukan split 5 kali yaoti mengitung tingkat akurasi menggunakan data training.

```
1 # In [16]: Dimana akan melakukan cross validation dengan 5
  split
2 from sklearn.model_selection import cross_val_score
3 scores = cross_val_score(clf, dk_train_att, dk_train_label, cv
  =5)
4 scorerata2=scores.mean()
5 scorersd=scores.std()
6 # In [21]:
7 from sklearn.model_selection import cross_val_score
8 scores = cross_val_score(clf, dk_train_att, dk_train_label,
  cv=5)
9 # show average score and +/- two standard deviations away (covering 95
10 # of scores)
11 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
12 scores.std() * 2))
13 # In [22]:
14 scorestree = cross_val_score(clftree, dk_train_att,
  dk_train_label, cv=5)
15 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
16 scorestree.std() * 2))
17 # In [23]:
18 scoresvm = cross_val_score(clfsvm, dk_train_att,
  dk_train_label, cv=5)
```

```

19 print("Accuracy: %.2f (+/- %.2f)" % (scoressvm.mean() ,
20 scoressvm.std() * 2))

```

```

Run: Run2
Hi everyone. We are a duo and we are sta
we love doing this. We may not have the
because we love making these videos and
Accuracy: 0.95 (+/- 0.04)
Accuracy: 0.96 (+/- 0.02)
Accuracy: 0.98 (+/- 0.06)
Process finished with exit code 0

```

Gambar 4.13 hasil

memunculkan nilai akurasi dari tiga metode yaitu random forest, decision tree, dan klasifikasi svm (support vector machine) di mana akan dibandingkan tingkat akurasi dari semua hasil akurasi mana yang terbaik dan lebih akurat pada hasilnya data yang paling akurat yaitu random forest.

8. Pengamatan program

```

1 # In [24]:
2 max_features_opts = range(1, 10, 1)
3 n_estimators_opts = range(2, 40, 4)
4 rf_params = np.empty((len(max_features_opts)*len(
    n_estimators_opts),4) , float)
5 i = 0
6 for max_features in max_features_opts:
7     for n_estimators in n_estimators_opts:
8         clf = RandomForestClassifier(max_features=
            max_features, n_estimators=n_estimators)
9         scores = cross_val_score(clf, dk_train_att,
        dk_train_label, cv=5)
10        rf_params[i,0] = max_features
11        rf_params[i,1] = n_estimators
12        rf_params[i,2] = scores.mean()
13        rf_params[i,3] = scores.std() * 2
14        i += 1
15        print("Max features: %d, num estimators: %d, accuracy
            : %.2f (+/- %.2f)")
16 % (max_features, n_estimators, scores.mean(), scores.std() *
2))
17 # In [25]:
18 import matplotlib.pyplot as plt
19 from mpl_toolkits.mplot3d import Axes3D
20 from matplotlib import cm
21 fig = plt.figure()
22 fig.clf()
23 ax = fig.gca(projection='3d')
24 x = rf_params[:,0]
25 y = rf_params[:,1]
26 z = rf_params[:,2]
27 ax.scatter(x, y, z)
28 ax.set_zlim(0.6, 1)

```

```

29 ax.set_xlabel('Max features')
30 ax.set_ylabel('Num estimators')
31 ax.set_zlabel('Avg accuracy')
32 plt.show()

```

```

Max features: 7, num estimators: 6, accuracy: 0.91 (+/- 0.08)
Max features: 7, num estimators: 10, accuracy: 0.91 (+/- 0.05)
Max features: 7, num estimators: 14, accuracy: 0.93 (+/- 0.05)
Max features: 7, num estimators: 18, accuracy: 0.94 (+/- 0.04)
Max features: 7, num estimators: 22, accuracy: 0.94 (+/- 0.05)
Max features: 7, num estimators: 26, accuracy: 0.95 (+/- 0.06)
Max features: 7, num estimators: 30, accuracy: 0.93 (+/- 0.06)
Max features: 7, num estimators: 34, accuracy: 0.94 (+/- 0.03)
Max features: 7, num estimators: 38, accuracy: 0.95 (+/- 0.03)
Max features: 8, num estimators: 2, accuracy: 0.83 (+/- 0.10)
Max features: 8, num estimators: 6, accuracy: 0.88 (+/- 0.07)
Max features: 8, num estimators: 10, accuracy: 0.92 (+/- 0.06)
Max features: 8, num estimators: 14, accuracy: 0.93 (+/- 0.05)
Max features: 8, num estimators: 18, accuracy: 0.93 (+/- 0.04)
Max features: 8, num estimators: 22, accuracy: 0.93 (+/- 0.06)
Max features: 8, num estimators: 26, accuracy: 0.94 (+/- 0.03)
Max features: 8, num estimators: 30, accuracy: 0.95 (+/- 0.04)
Max features: 8, num estimators: 34, accuracy: 0.94 (+/- 0.05)
Max features: 8, num estimators: 38, accuracy: 0.94 (+/- 0.04)
Max features: 9, num estimators: 2, accuracy: 0.80 (+/- 0.17)
Max features: 9, num estimators: 6, accuracy: 0.88 (+/- 0.06)
Max features: 9, num estimators: 10, accuracy: 0.93 (+/- 0.05)
Max features: 9, num estimators: 14, accuracy: 0.93 (+/- 0.05)
Max features: 9, num estimators: 18, accuracy: 0.94 (+/- 0.04)
Max features: 9, num estimators: 22, accuracy: 0.94 (+/- 0.03)
Max features: 9, num estimators: 26, accuracy: 0.93 (+/- 0.04)
Max features: 9, num estimators: 30, accuracy: 0.94 (+/- 0.06)
Max features: 9, num estimators: 34, accuracy: 0.93 (+/- 0.06)
Max features: 9, num estimators: 38, accuracy: 0.94 (+/- 0.06)

```

Gambar 4.14 hasil

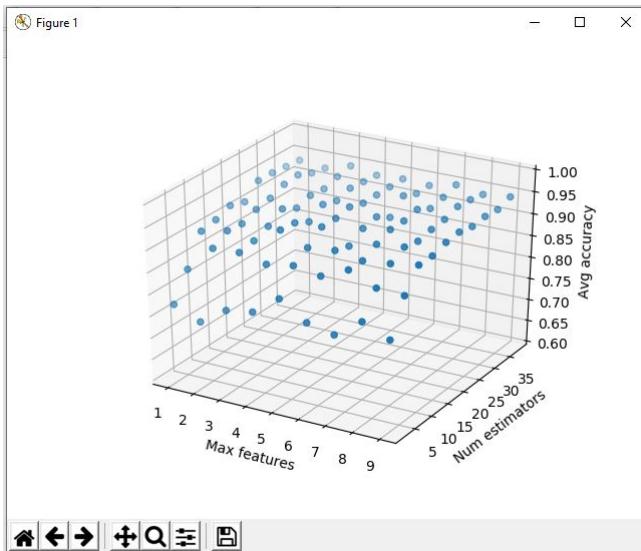
terdapat grafik data yang terdapat dari grafik tersebut di dapat dari codingan dengan cara pengulangan data masing masing 10 kali setelah itu di eksekusi menjadi grafik berbentuk 3D pada gambar tersebut menunjukan rasio dari yang terrendah yaitu data SVM kemudian data decision tree dan hasil random forest.

4.1.3 Penanganan Error

1. screnn shoot error
2. codingan yang errornya terdapat pada

```
clfsvm.fit(dk_train_att, d_train_label)
```

3. solusinya
- ```
clfsvm.fit(dk_train_att, dk_train_label)
```

**Gambar 4.15** hasil

```

Run: Run2
Traceback (most recent call last):
File "D:/NAJIB/SEMESTER_6_NAJIB/AI/Chapter4/Run2.py", line 47, in <module>
 clfsvm.fit(dk_train_att, d_train_label)
NameError: name 'd_train_label' is not defined
Process finished with exit code 1

```

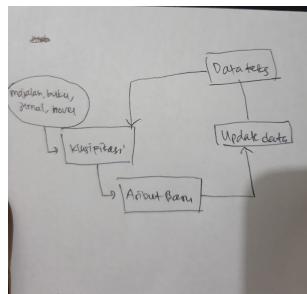
**Gambar 4.16** hasil

## 4.2 1174040 - Hagan Rowlenstino A. S

### 4.2.1 Teori

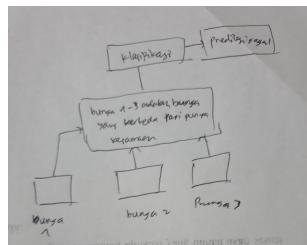
#### 1. Klasifikasi Text

Penggolongan teks atau biasa disebut pengklasifikasian teks biasanya sangat dibutuhkan pada aplikasi yang mempunyai jumlah dokumen yang biasanya bertambah dengan cepat. Klasifikasi teks itu sendiri merupakan cara dalam memilah milah data teks berdasarkan parameter tertentu dengan data yang bersifat dokumen ataupun teks yang memiliki kumpulan teks di dalamnya, serta teks itu sendiri bertipe data char atau string yang mudah untuk diolah.



**Gambar 4.17** Klasifikasi Text

- Mengapa Klasifikasi Bunga tidak dapat menggunakan Machine Learning karena klasifikasinya menggunakan tipe data yang dimana attributnya memiliki nilai data berupa vektor dengan perbandingan masing - masing data yang dimiliki memiliki sedikit perbedaan, sehingga program atau sistem tidak dapat membedakan dengan tepat antara gambar 1 dan gambar 2 dikarenakan memiliki perbedaan yang hampir tidak dapat dilihat pada beberapa contoh gambar. untuk ilustrasi dapat dilihat pada gambar



**Gambar 4.18** Klasifikasi Bunga

- Teknik Pembelajaran machine learning pada teks kata-kata di youtube Contohnya pada saat kita membuka sebuah video maka di sebelah kanan ada list "berikutnya", pada saat itu mesin melakukan ujicoba dan apabila anda menekan salah satu dari video tersebut maka hal tersebut akan direkam dan disimpan oleh mesin tersebut



**Gambar 4.19** Machine Learning Youtube

4. Jelaskan apa yang dimaksud vektorisasi data.

vektorisasi data merupakan pemecahan data menjadi bagian-bagian yang lebih sederhana contoh ada sebuah paragraf, dari paragraf tersebut akan dibagi-bagi menjadi kalimat, yang nantinya akan dibagi-bagi kembali menjadi perkata.

5. Jelaskan apa itu bag of words dengan kata-kata yang sederhana dan ilustrasi sendiri

Bag of words merupakan penyajian sederhana yang biasanya digunakan pada aplikasi text mining pada saat mengenalkan struktur ke sebuah kumpulan dokumen yang berbasis teks untuk diklasifikasikan kembali menjadi dua atau lebih kelas yang telah ditentukan. Dalam model ini teks, paragraf, dokumen, ataupun kalimat dimuat dalam kumpulan kata yang mengabaikan tata bahasa ataupun urutan dari kata-kata tersebut muncul.



Gambar 4.20 Bag of Words

6. Jelaskan apa itu TF-IDF, ilustrasikan dengan gambar sendiri.

TF-IDF merupakan metode untuk menghitung bobot dari kata yang sering muncul pada suatu kalimat. Metode ini menghitung nilai TF atau Term Frequency dan IDF atau Inverse Document Frequency pada setiap kata pada kalimat yang dijadikan acuan kata pada metode ini sering disebut token adapun rumus dari metode ini dapat dilihat pada gambar.

|                                  | $n$   | $d$  | $tf$  | $df$ | $df+1$ | $WtfIDF(n)$ |
|----------------------------------|-------|------|-------|------|--------|-------------|
| Q                                | 43    | 43   | 1     | 2    | 3      | 0.136       |
| apple                            | 0     | 2    | 0     | 1    | 2      | 0.136       |
| sheer                            | 0     | 0    | 0     | 3    | 4      | 0.477       |
| truck                            | 3     | 1    | 1     | 2    | 3      | 0.370       |
|                                  |       |      |       |      |        | 1.176       |
|                                  |       |      |       |      |        | 1.176       |
|                                  |       |      |       |      |        | 2.352       |
| <b>Total Bobot Tep Dokumen =</b> |       |      |       |      |        |             |
|                                  | 1.176 | 4.13 | 2.352 |      |        |             |

Gambar 4.21 TF-IDF

#### 4.2.2 Praktek

1. import data pandas dan 500 baris data dummy kemudian di jelaskan tiap barisnya.

```
1 import pandas as pd #mengimport library pandas dan menamainya
pd
2 #%%
3 hagan = pd.read_csv("D:/ Semester6/AI/Chapter4/1174040.csv") #
membuat variable bernama hagan dan mengisinya dengan data
dari dataset dummy yang telah dibuat
```

```

4 a = hagan.head() #untuk melihat 5 baris pertama dari data
 hagan
5 hagan.shape #untuk mengetahui berapa banyak baris data
6 print(a) #menampilkan isi dari varibale a pada console

```

```

In [42]:
Removing all variables...

In [42]:
...: Created on Mon Mar 23 13:20:30 2020
...:
...: Author: User
...:
...: Import pandas as pd
...: import matplotlib.pyplot as plt
...: %matplotlib inline
...: df = pd.read_csv("D:/Semester6/AI/Chapter4/Youtube02-KatyPerry.csv")
...: dtra = df[:450]
...: dtes = df[450:]
...: print(a)
...: print(hagan)

a
 id first_name last_name email gender
0 1 Godard Adelie gaddelein@list.com Male
1 2 Angèle Bobigny ktdone@spipolice.fr Male
2 3 Sophie Toubon ktdone@spipolice.fr Female
3 4 Gianna Orteg portegals@medialy.com Female
4 5 Odile Vilt odile@spipolice.fr Female

```

Gambar 4.22 Data Dummy

2. memecah data prame menjadi dua yag pertama 450 dan kedua sisanya

```

1 #%%
2 dtra = hagan[:450] #memasukkan 450 data pertama ke dalam
 variable dtra
3 dtes = hagan[450:] #memasukkan 50 data terakhir kedalam
 variable dtes

```

```

dtes DataFrame (50, 5) [Column names: id, first_name, last_name, email, gender]
dtra DataFrame (450, 5) [Column names: id, first_name, last_name, email, gender]

```

Gambar 4.23 Pisah Data

3. praktek vektorisasi

```

1 #%% memasukkan data dari file csv tersebut ke dalam variable
 data
2 data=pd.read_csv("D:/ Semester6/AI/Chapter4/Youtube02-
 KatyPerry .csv")
3 spam=data.query('CLASS == 1')
4 nospam=data.query('CLASS == 0')
5 #%% melakukan fungsi bag of word dengan cara menghitung semua
 kata
6 from sklearn.feature_extraction.text import CountVectorizer
7 vectorizer = CountVectorizer()
8 #%% melakukan bag of word pada dataframe pada colom CONTENT
9 data_vektorisasi = vectorizer.fit_transform(data['CONTENT'])
10 #%% melihat isi vektorisasi
11 data_vektorisasi

```

```

12 #%% melihat isi data pada baris ke 345
13 print(data['CONTENT'][345])
14 #%% untuk mengambil apa saja nama kolom yang tersedia
15 dk=vectorizer.get_feature_names()
16 #%%: melakukan randomisasi agar hasil sempurna pada saat
 klasifikasi
17 dshuf = data.sample(frac=1)
18 #%%: membuat data tranning dan testing
19 dk_train=dshuf[:300]
20 dk_test=dshuf[300:]
21 #%%: melakukan training pada data training dan di vektorisasi
22 dk_train_att=vectorizer.fit_transform(dk_train['CONTENT'])
23 print(dk_train_att)
24 #%% melakukan testing pada data testing dan di vektorisasi
25 dk_test_att=vectorizer.transform(dk_test['CONTENT'])
26 print(dk_test_att)
27 #%%: Dimana akan mengambil label spam dan bukan spam
28 dk_train_label=dk_train['CLASS']
29 print(dk_train_label)
30 dk_test_label=dk_test['CLASS']
31 print(dk_test_label)

```

lakukan import library pandas yang di inisialisasi menjadi pd setelah itu ada dibuat variable data dengan method read\_csv untuk membaca file berekstensikan csv yang di masukan alamatnya pada kurung, lakukan klasifikasi atau pemilihan komentar yang berisi spam atau bukan spam dengan parameter class samadengan 1 merupakan spam dan class samadengan 0 bukan spam setelah itu masukan librari CountVektorizer yang digunakan untuk vektorisasi data kemudian dilanjutkan pada bagian In[103] dibuat variabel yang berisi vektorisasi dari data pada data di field content setelah itu variabel tersebut di running hasilnya menunjukan 350 baris di kali 1738 kolom selanjutnya dicoba untuk memunculkan isi recod pada baris ke 345 maka akan muncul isian dari baris tersebut. selanjutnya dibuat variabel dk atau daftar yang berisi data hasil vektorisasi setelah yang terdiri dari variabel dshuf yang berisi data komen yang di dalamnya di buat random yang nantinya akan dibut data training dan data testing dengan ketentuan data training 300 dan data testing sebanyak 50 setelah itu data training di lakukan vektorisasi dan data testing juga dilakukan vektorisasi setelah itu kedua data training dan testing tersebut dibuat label dengan parameter field CLASS pada tabel.

```

... : dk_train_label=dk_train['CLASS']
... : dk_test_label=dk_test['CLASS']
... :
who is going to reach the billion first : katy or taylor ?
(9, 1557) 1
(9, 1558) 1
(9, 1559) 1
(9, 1560) 1
(9, 1561) 1
(9, 1562) 1
(9, 1563) 1
(9, 1564) 1
(9, 1565) 1
(9, 1566) 1
(9, 1567) 1
(9, 1568) 1
(9, 1569) 1
(9, 1570) 1
(9, 1571) 1
(9, 1572) 1
(9, 1573) 1
(9, 1574) 1
(9, 1575) 1
(9, 1576) 1
(9, 1577) 1
(9, 1578) 1
(9, 1579) 1
(9, 1580) 1
(9, 1581) 1
(9, 1582) 1
(9, 1583) 1
(9, 1584) 1
(9, 1585) 1
(9, 1586) 1
(9, 1587) 1
(9, 1588) 1
(9, 1589) 1
(9, 1590) 1
(9, 1591) 1
(9, 1592) 1
(9, 1593) 1
(9, 1594) 1
(9, 1595) 1
(9, 1596) 1
(9, 1597) 1
(9, 1598) 1
(9, 1599) 1
(9, 1600) 1
(9, 1601) 1
(9, 1602) 1
(9, 1603) 1
(9, 1604) 1
(9, 1605) 1
(9, 1606) 1
(9, 1607) 1
(9, 1608) 1
(9, 1609) 1
(9, 1610) 1
(9, 1611) 1
(9, 1612) 1
(9, 1613) 1
(9, 1614) 1
(9, 1615) 1
(9, 1616) 1
(9, 1617) 1
(9, 1618) 1
(9, 1619) 1
(9, 1620) 1
(9, 1621) 1
(9, 1622) 1
(9, 1623) 1
(9, 1624) 1
(9, 1625) 1
(9, 1626) 1
(9, 1627) 1
(9, 1628) 1
(9, 1629) 1
(9, 1630) 1
(9, 1631) 1
(9, 1632) 1
(9, 1633) 1
(9, 1634) 1
(9, 1635) 1
(9, 1636) 1
(9, 1637) 1
(9, 1638) 1
(9, 1639) 1
(9, 1640) 1
(9, 1641) 1
(9, 1642) 1
(9, 1643) 1
(9, 1644) 1
(9, 1645) 1
(9, 1646) 1
(9, 1647) 1
(9, 1648) 1
(9, 1649) 1
(9, 1650) 1
(9, 1651) 1
(9, 1652) 1
(9, 1653) 1
(9, 1654) 1
(9, 1655) 1
(9, 1656) 1
(9, 1657) 1
(9, 1658) 1
(9, 1659) 1
(9, 1660) 1
(9, 1661) 1
(9, 1662) 1
(9, 1663) 1
(9, 1664) 1
(9, 1665) 1
(9, 1666) 1
(9, 1667) 1
(9, 1668) 1
(9, 1669) 1
(9, 1670) 1
(9, 1671) 1
(9, 1672) 1
(9, 1673) 1
(9, 1674) 1
(9, 1675) 1
(9, 1676) 1
(9, 1677) 1
(9, 1678) 1
(9, 1679) 1
(9, 1680) 1
(9, 1681) 1
(9, 1682) 1
(9, 1683) 1
(9, 1684) 1
(9, 1685) 1
(9, 1686) 1
(9, 1687) 1
(9, 1688) 1
(9, 1689) 1
(9, 1690) 1
(9, 1691) 1
(9, 1692) 1
(9, 1693) 1
(9, 1694) 1
(9, 1695) 1
(9, 1696) 1
(9, 1697) 1
(9, 1698) 1
(9, 1699) 1
(9, 1700) 1
(9, 1701) 1
(9, 1702) 1
(9, 1703) 1
(9, 1704) 1
(9, 1705) 1
(9, 1706) 1
(9, 1707) 1
(9, 1708) 1
(9, 1709) 1
(9, 1710) 1
(9, 1711) 1
(9, 1712) 1
(9, 1713) 1
(9, 1714) 1
(9, 1715) 1
(9, 1716) 1
(9, 1717) 1
(9, 1718) 1
(9, 1719) 1
(9, 1720) 1
(9, 1721) 1
(9, 1722) 1
(9, 1723) 1
(9, 1724) 1
(9, 1725) 1
(9, 1726) 1
(9, 1727) 1
(9, 1728) 1
(9, 1729) 1
(9, 1730) 1
(9, 1731) 1
(9, 1732) 1
(9, 1733) 1
(9, 1734) 1
(9, 1735) 1
(9, 1736) 1
(9, 1737) 1
(9, 1738) 1

```

Gambar 4.24 Vektorisasi

#### 4. klasifikasi SVM

```

1 from sklearn import svm
2 clfsvm = svm.SVC()
3 clfsvm.fit(dk_train_att, dk_train_label)
4 clfsvm.score(dk_test_att, dk_test_label)

```

import librari svm dari sklearn kemudian membuat variabel clfsvm berisikan method svc setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

```

In [66]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(dk_train_att, dk_train_label)
...: clfsvm.score(dk_test_att, dk_test_label)
Out[66]: 0.94

```

**Gambar 4.25** SVM

#### 5. klasifikasi decision tree

```

1 from sklearn import tree
2 clftree = tree.DecisionTreeClassifier()
3 clftree.fit(dk_train_att, dk_train_label)
4 clftree.score(dk_test_att, dk_test_label)

```

import librari tree dari sklearn kemudian membuat variabel clftree berisikan method DecisionTreeClasifier setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut agar dapat digunakan pada codingan selanjutnya setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

```

In [67]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(dk_train_att, dk_train_label)
...: clftree.score(dk_test_att, dk_test_label)
Out[67]: 0.96

```

**Gambar 4.26** Desicion Tree

#### 6. plot comfusion matrix

```

1 from sklearn.metrics import confusion_matrix
2 pred_labels = clftree.predict(dk_test_att)
3 cm = confusion_matrix(dk_test_label, pred_labels)
4 cm

```

import library comfusion matrix selanjutnya dilakukan prediksi pada pada data tes nya kemudian data tersebut di masukan kedalam variabel cm dengan method confusion matrix yang di dalamnya terdapat data dari variabel perd label dan dk test label setelah itu variabel cm tersebut di running maka akan memunculkan nilai matrixnya.

```
In [80]: from sklearn.metrics import confusion_matrix
...: pred_labels = clftree.predict(dk_test_att)
...: cm = confusion_matrix(dk_test_label, pred_labels)
...: cm
Out[80]:
array([[25, 1],
 [1, 23]], dtype=int64)
```

**Gambar 4.27** Confussion Matrix

## 7. cross valodation

```
1 #%%%
2 from sklearn.model_selection import cross_val_score
3 scores = cross_val_score(clftree, dk_train_att, dk_train_label,
 cv=5)
4 scorerata2=scores.mean()
5 scorersd=scores.std()
6 #%%:
7 from sklearn.model_selection import cross_val_score
8 scores = cross_val_score(clftree, dk_train_att,
 dk_train_label, cv=5)
9 # show average score and +/- two standard deviations away (
 covering 95
10 #% of scores)
11 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
12 scores.std() * 2))
13 #%%:
14 scorestree = cross_val_score(clftree, dk_train_att,
 dk_train_label, cv=5)
15 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
16 scorestree.std() * 2))
17 #%%:
18 scoressvm = cross_val_score(clfsvm, dk_train_att,
 dk_train_label, cv=5)
19 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(),
20 scoressvm.std() * 2))
```

memunculkan nilai akurasi dari tiga metode yaitu random forest, decision tree, dan klasifikasi svm (suport vector machine) diamana akan di bandingkan tingkat akurasi dari semua hasil akurasi mana yang terbaik dan lebih akurat pada hasilnya data yang paling akurat yaitu random forest.

```
In [82]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(ciftree, dk_train_att, dk_train_label, cv=5)
...: scores = np.mean(scores)
...: scores+=scores.std()
In [83]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(ciftree, dk_train_att, dk_train_label, cv=5)
...: scores = np.mean(scores)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
...: scores.std() * 2))
Accuracy: 0.73 (+/- 0.07)

In [84]: scores = cross_val_score(lfnew, dk_train_att, dk_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
...: scores.std() * 2))
Accuracy: 0.85 (+/- 0.07)

In [85]: scoresmean = cross_val_score(lfnew, dk_train_att, dk_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoresmean.mean(),
...: scoresmean.std() * 2))
Accuracy: 0.91 (+/- 0.08)
```

**Gambar 4.28** Cross Validation

## 8. Pengamatan program

```
1 #%%%
2 import numpy as np
3 max_features_opts = range(1, 10, 1)
4 n_estimators_opts = range(2, 40, 4)
5 rf_params = np.empty((len(max_features_opts)*len(
6 n_estimators_opts),4) , float)
7 i = 0
8 for max_features in max_features_opts:
9 for n_estimators in n_estimators_opts:
10 clf = RandomForestClassifier(max_features=
11 max_features , n_estimators=n_estimators)
12 scores = cross_val_score(clf , dk_train_att ,
13 dk_train_label , cv=5)
14 rf_params[i,0] = max_features
15 rf_params[i,1] = n_estimators
16 rf_params[i,2] = scores.mean()
17 rf_params[i,3] = scores.std() * 2
18 i += 1
19 print("Max features: %d, num estimators: %d, accuracy
20 : %0.2f (+/- %0.2f)"
21 % (max_features , n_estimators , scores.mean() , scores.std() *
22))
```

terdapat grafik data yang terdapat dari grafik tersebut di dapat dari codingan dengan cara pengulangan data masing masing 10 kali setelah itu di eksekusi menjadi grafik berbentuk 3D pada gambar tersebut menunjukan rasio dari yang terrendah yaitu data SVM kemudian data decision tree dan hasil random forest.

```
...: clf = RandomForestClassifier(max_features=max_features,
...: n_estimators=n_estimators)
...: scores = cross_val_score(clf, dk_train_att, dk_train_label, cv=5)
...: rf_params[i,0] = max_features
...: rf_params[i,1] = n_estimators
...: rf_params[i,2] = scores.mean()
...: rf_params[i,3] = scores.std() * 2
...: i += 1
...: print("Max features: %d, num estimators: %d, accuracy: %0.2f (+/- %0.2f)"
...: % (max_features, n_estimators, scores.mean(), scores.std() * 2))
Max features: 1, num estimators: 2, accuracy: 0.73 (+/- 0.06)
Max features: 1, num estimators: 4, accuracy: 0.73 (+/- 0.06)
Max features: 1, num estimators: 10, accuracy: 0.86 (+/- 0.05)
Max features: 1, num estimators: 14, accuracy: 0.88 (+/- 0.12)
Max features: 1, num estimators: 18, accuracy: 0.88 (+/- 0.05)
Max features: 1, num estimators: 22, accuracy: 0.87 (+/- 0.07)
Max features: 1, num estimators: 26, accuracy: 0.86 (+/- 0.05)
Max features: 1, num estimators: 30, accuracy: 0.86 (+/- 0.05)
Max features: 1, num estimators: 34, accuracy: 0.89 (+/- 0.03)
Max features: 2, num estimators: 2, accuracy: 0.69 (+/- 0.03)
Max features: 2, num estimators: 10, accuracy: 0.89 (+/- 0.03)
Max features: 2, num estimators: 18, accuracy: 0.86 (+/- 0.05)
Max features: 2, num estimators: 22, accuracy: 0.86 (+/- 0.05)
Max features: 2, num estimators: 26, accuracy: 0.86 (+/- 0.05)
Max features: 2, num estimators: 30, accuracy: 0.86 (+/- 0.05)
Max features: 2, num estimators: 34, accuracy: 0.86 (+/- 0.05)
```

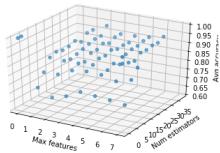
**Gambar 4.29** Pengamatan Program

Berikut adalah untuk grafik

```

1 import matplotlib.pyplot as plt
2 from mpl_toolkits.mplot3d import Axes3D
3 from matplotlib import cm
4 fig = plt.figure()
5 fig.clf()
6 ax = fig.gca(projection='3d')
7 x = rf_params[:,0]
8 y = rf_params[:,1]
9 z = rf_params[:,2]
10 ax.scatter(x, y, z)
11 ax.set_zlim(0.6, 1)
12 ax.set_xlabel('Max features')
13 ax.set_ylabel('Num estimators')
14 ax.set_zlabel('Avg accuracy')
15 plt.show()

```



**Gambar 4.30** Grafik

### 4.2.3 Penanganan Error

#### 4.2.3.1 Screenshoot Error

##### 1. Error 1

```

rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),4), float)
NameError: name 'np' is not defined

```

**Gambar 4.31** Error1

##### 2. Error 2

```

File "pandas_libs\hashtable_class_helper.pxi", line 1492, in
pandas._libs.hashtable.PyObjectHashTable.get_item
File "pandas_libs\hashtable_class_helper.pxi", line 1500, in
pandas._libs.hashtable.PyObjectHashTable.get_item
KeyError: 'CONTENT'

```

**Gambar 4.32** Error2

##### 3. Error 3

```

File "pandas_libs\parsers.pyx", line 695, in
pandas._libs.parsers.read_csv_cython._read_file
FileNotFoundError: File b'D:\Semester6\AI\Chapter404080.csv' does not exist

```

**Gambar 4.33** Error3

#### 4. Error 4

```
File "pandas\libs\hashtable_class_helper.pxi", line 964, in
pandas._libs.hashtable.Int64HashTable.get_item
KeyError: 5000
```

**Gambar 4.34** Error4

##### 4.2.3.2 Kode Error dan Jenisnya

###### 1. Kode Error 1 jenis Name Error

```
max_features_opts = range(1, 10, 1)
n_estimators_opts = range(2, 40, 2)
rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts), 4), float)
i = 0
```

**Gambar 4.35** Error1

###### 2. Kode Error 2 jenis Key Error

```
data_vektorisasi = vectorizer.fit_transform(data['CONTENTz'])
```

**Gambar 4.36** Error2

###### 3. Kode Error 3 jenis FileNotFoundError

```
hagen = pd.read_csv("D:\\Semester5\\AI\\Chapter4\\L174040.csv") #membuat
a = hagen.head() #ambil sebanyak 5 baris pertama dari data hagen
hagen.shape #untuk mengetahui berapa banyak baris data
print(a) #menampilkan isi dari variabel a pada console
```

**Gambar 4.37** Error3

###### 4. Kode Error 4 Key Error

```
#%%
print(data['CONTENT'][5000])
```

**Gambar 4.38** Error4

##### 4.2.3.3 Solusi

###### 1. Mengimport library numpy sebagai np

```
import numpy as np
max_features_opts = range(1, 10, 1)
n_estimators_opts = range(2, 40, 2)
rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts), 4), float)
i = 0
```

**Gambar 4.39** Solusi 1

###### 2. Mengganti CONTENTz menjadi CONTENT

**Gambar 4.40** Solusi 2

- Merubah backslash menjadi slash biasa

```
data_vektorisasi = vectorizer.fit_transform(data['CONTENT'])
data_vektorisasi
```

**Gambar 4.41** Solusi 3

- Merubah data menjadi dibawah 350

```
#%%%
print(data['CONTENT'][349])
%%%
```

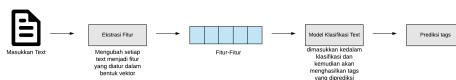
**Gambar 4.42** Solusi 4

### 4.3 Alit Fajar Kurniawan 1174057

#### 4.3.1 Teori

- Jelaskan apa itu klasifikasi teks, sertakan gambar ilustrasi buatan sendiri.

Klasifikasi teks adalah proses pemberian tag atau kategori ke teks sesuai dengan isinya. Teks dapat menjadi sumber informasi yang sangat kaya, tetapi mengekstraksi wawasan darinya bisa sulit dan memakan waktu karena sifatnya yang tidak terstruktur. berikut contoh gambar ?? .

**Gambar 4.43** contoh klasifikasi teks

- Jelaskan mengapa klasifikasi bunga tidak bisa menggunakan machine learning, sertakan ilustrasi sendiri.

Untuk klasifikasi bunga tidak dapat menggunakan machine learning dikarenakan memiliki masalah input yang sama namun keluarannya (output) yang berbeda, biasanya output atau error ini disebut dengan istilah 'noise'. Noise sendiri merupakan output yang disimpan / ditangkap maupun direkam bukan seperti seharusnya ( keluaran yang diinginkan ).

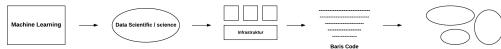
Apabila diberikan contoh, maka contohnya yaitu kita berasumsi secara implisit bahwa klarifikasi bunga yang kita lakukan sudah tepat dan kita melakukannya seperti seorang ahli tanaman. Namun pada hasilnya masih saja terjadi kesalahan. Selain itu, selalu ada peluang untuk memperkenalkan kesalahan saat merekam ataupun menyimpan data, maka harus dilakukan penelitian yang lebih rinci sehingga tidak menimbulkan 'noise' itu sendiri. Berikut ilustrasi gambar ??



**Gambar 4.44** gambar bunga

3. Jelaskan bagaimana teknik pembelajaran mesin pada teks pada kata-kata yang digunakan di youtube,jelaskan arti per atribut data csv dan sertakan ilustrasi buatan sendiri.

Kita ambil sebuah kasus yang semua orang telah ketahui dan juga pahami. Kasus tersebut yaitu perekendasian video dari pencarian menggunakan "text / kata" di Youtube. Pada saat menggunakan Youtube terdapat Machine Learning yang bekerja dan memproses perintah ataupun aktivitas tersebut, dimana akan memfilter secara otomatis video yang disesuaikan dengan "keyword" yang kita masukkan sehingga memberikan keluaran video dengan keyword yang benar. Adapula tur yang dapat diperoleh ketika sedang menonton Youtube. Tampilan sebelah kanan terdapat pilihan 'Next' ataupun 'Suggestion' yang menampilkan beberapa video serupa sesuai dengan yang dicari atau sedang ditonton. Ketika mengklik salah satu video dari baris tersebut, maka Youtube akan mengingat dan menggunakan kata yang tertera sebagai referensi kembali sehingga akan memberikan kemudahan pada pencarian yang lainnya, Dan disitu lah mesin belajar sendiri dan menyimpan data secara berkala sehingga berkembang. contoh pada gambar ??



**Gambar 4.45** pembelajaran mesin

4. Jelaskan apa yang dimaksud vektorisasi data.

Pembagian dan pemecahan data, kemudian dilakukan perhitungan. Vektorisasi juga dapat dimaksudkan dengan setiap data yang mungkin dipetakan ke integer tertentu. jika kita memiliki array yang cukup besar maka setiap kata / data cocok dengan slot unik dalam array (nilai pada indeks adalah nomor satu kali kata itu muncul).

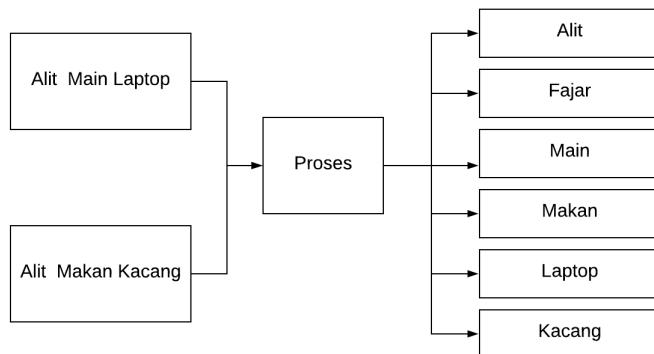
Array angka floating point ( Mewakili data ) :

- teks
- audio
- gambar

Contoh : -[1.0, 0.0, 1.0, 0.5]

5. Jelaskan apa itu bag of words dengan kata-kata yang sederhana dan ilustrasi sendiri.

bag-of-words adalah representasi penyederhanaan yang digunakan dalam pemrosesan bahasa alami dan pengambilan informasi. Model bag-of-words sederhana untuk dipahami dan diterapkan dan telah melihat kesuksesan besar dalam masalah seperti pemodelan bahasa dan klasifikasi dokumen. Pada model ini, tiap kalimat dalam dokumen digambarkan sebagai token, mengabaikan tata bahasa dan bahkan urutan kata namun menghitung frekuensi kejadian atau kemunculan kata dari dokumen. berikut contoh gambar ??

**Gambar 4.46** bag of words

6. Jelaskan apa itu TF-IDF, ilustrasikan dengan gambar sendiri

TF-IDF memberi kita frekuensi kata dalam setiap dokumen dalam korpus atau mengganti data jadi number. Ini adalah rasio berapa kali kata itu muncul dalam dokumen dibandingkan dengan jumlah total kata dalam dokumen itu. Itu meningkat seiring jumlah kemunculan kata itu di dalam dokumen meningkat. Setiap dokumen memiliki tf sendiri.

|       | Alit | Fajar | Main | Makan | Laptop | Kacang |
|-------|------|-------|------|-------|--------|--------|
| Doc 1 | 1    |       | 1    |       |        | 1      |
| Doc 2 | 1    | 1     | 1    | 1     |        |        |
| Doc 3 |      | 1     |      |       | 1      |        |

**Gambar 4.47** TF-IDF

### 4.3.2 Praktek Program

- import data pandas dan 500 baris data dummy kemudian di jelaskan tiap barisnya.

```

1 #mengimport library pandas dan menamainya pd
2 import pandas as pd
3 #%%
4 #membuat variable bernama alit dan mengisinya dengan data
 dari dataset dummy yang telah dibuat
5 alit = pd.read.csv("D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src
 /1174057/chapter4/1174057.csv")
6 #untuk melihat 5 baris pertama dari data alit
7 c = alit.head()
8 #untuk mengetahui berapa banyak baris data

```

```

9 alit.shape
10 #menampilkan isi dari varibale c pada console
11 print(c)

```

```

In [15]: runfile('D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter4/1.py',
wdir='D:/Data Alit/Kuliah/SEMESTER VI/KB3B/src/1174057/chapter4')
first_name ... web
0 James ... http://www.bentonjohnbjr.com
1 Josephine ... http://www.chanayjeffreyaesq.com
2 Art ... http://www.chemeljameslcpa.com
3 Lenna ... http://www.feltzprintingservice.com
4 Donette ... http://www.printingdimensions.com

[5 rows x 12 columns]

```

**Gambar 4.48** hasil

2. dari dataframe tersebut dipecah menjadi dua dataframe yaitu 450 row pertama dan 50 row sisanya

```

1
2
3 #%%
4 #memasukkan 450 data pertama ke dalam variable dtra
5 dtra = alit [:450]

```

```

In [78]: data=pd.read_csv("D:/Data Alit/Kuliah/SEMESTER VI/AI_KB3B/
Python-Artificial-Intelligence-Projects-for-Beginners-master/Chapter04/
Youtube03-LMFAO.csv")
....: spam=data.query('CLASS == 1')
....: nospam=data.query('CLASS == 0')

```

**Gambar 4.49** hasil

|      |           |           |                                                                            |
|------|-----------|-----------|----------------------------------------------------------------------------|
| dtes | DataFrame | (50, 12)  | Column names: first_name, last_name, company_name, address, city, coun ... |
| dtra | DataFrame | (450, 12) | Column names: first_name, last_name, company_name, address, city, coun ... |

**Gambar 4.50** hasil

3. praktek vektorisasi

```

1 #%% memasukkan data dari file Youtube03-LMFAO.csv tersebut ke
 dalam variable data
2 data=pd.read_csv("D:/Data Alit/Kuliah/SEMESTER VI/AI_KB3B/
 Python-Artificial-Intelligence-Projects-for-Beginners-
 master/Chapter04/Youtube03-LMFAO.csv")
3 spam=data .query('CLASS == 1')
4 nospam=data .query('CLASS == 0')
5 #%% melakukan fungsi bag of word dengan cara menghitung semua
 kata

```

```
6 from sklearn.feature_extraction.text import CountVectorizer
7 vectorizer = CountVectorizer()
8 %% melakukann bag of word pada dataframe pada colom CONTENT
9 data_vektorisasi = vectorizer.fit_transform(data['CONTENT'])
10 %% melihat isi vektorisasi
11 data_vektorisasi
12 %% menampilkan isi data pada baris ke 300
13 print(data['CONTENT'][300])
14 %% untuk mengambil apa saja nama kolom yang tersedia
15 dk=vectorizer.get_feature_names()
16 %%: randomisasi agar hasil sempurna pada saat klasifikasi
17 dshuf = data.sample(frac=1)
18 %%: membuat data tranning dan testing
19 dk_train=dshuf[:300]
20 dk_test=dshuf[300:]
21 %%: melakukan training pada data training dan di vektorisasi
22 dk_train_att=vectorizer.fit_transform(dk_train['CONTENT'])
23 print(dk_train_att)
24 %%: melakukan testing pada data testing dan di vektorisasi
25 dk_test_att=vectorizer.transform(dk_test['CONTENT'])
26 print(dk_test_att)
27 %%: Dimana akan mengambil label spam dan bukan spam
28 dk_train_label=dk_train['CLASS']
29 print(dk_train_label)
30 dk_test_label=dk_test['CLASS']
31 print(dk_test_label)
```

lakukan import library pandas yang di inisialisasi menjadi pd setelah itu ada dibuat variable data dengan method read\_csv untuk membaca file berekstensikan csv yang di masukan alamatnya pada kurung, lakukan klasifikasi atau pemilihan komentar yang berisi spam atau bukan spam dengan parameter class samadengan 1 merupakan spam dan class samadengan 0 bukan spam setelah itu masukan librari CountVektorizer yang digunakan untuk vektorisasi data kemudian dilanjutkan pada bagian In[103] dibuat variabel yang berisi vektorisasi dari data pada data di field content setelah itu variabel tersebut di running hasilnya menunjukan 350 baris di kali 1738 kolom selanjutnya dicoba untuk memunculkan isi record pada baris ke 345 maka akan muncul isian dari baris tersebut. selanjutnya dibuat variabel dk atau daftar yang berisi data hasil vektorisasi setelah yang terdiri dari variabel dshuf yang berisi data komen yang di dalamnya di buat random yang nantinya akan dibut data training dan data testing dengan ketentuan data training 300 dan data testing sebanyak 50 setelah itu data training di lakukan vektorisasi dan data testing juga dilakukan vektorisasi setelah itu kedua data training dan testing tersebut dibuat label dengan parameter field CLASS pada tabel.

```
In [41]: dk_train_label=dk_train['CLASS']
...: print(dk_train_label)
...: dk_test_label=dk_test['CLASS']
...: print(dk_test_label)
228 1
212 1
344 1
326 1
270 1
322 1
304 1
332 1
33 0
361 1
205 1
365 1
195 0
285 1
370 1
124 0
298 1
51 0
333 1
187 1
426 0
215 1
101 0
106 1
```

IPython console History log

Gambar 4.51 hasil

#### 4. klasifikasi SVM

```
1 #import librari svm dari sklearn
2 from sklearn import svm
3 #membuat variabel clfsvm berisikan method svc
4 clfsvm = svm.SVC()
5 #variabel tersebut di berikan method fit dengan isian data
#train vektorisasi dan data training label
6 clfsvm.fit(dk_train_att, dk_train_label)
7 clfsvm.score(dk_test_att, dk_test_label)
```

```
In [45]: from sklearn import svm
...: #membuat variabel clfsvm berisikan method svc
...: clfsvm = svm.SVC()
...: #variabel tersebut di berikan method fit dengan isian data train vektorisasi
dan data training label
...: clfsvm.fit(dk_train_att, dk_train_label)
...: clfsvm.score(dk_test_att, dk_test_label)
Out[45]: 0.855072463768116
```

Gambar 4.52 hasil

#### 5. klasifikasi decision tree

```
1 #import librari tree dari sklearn
2 from sklearn import tree
3 clftree = tree.DecisionTreeClassifier()
```

```

4 clftree.fit(dk_train_att, dk_train_label)
5 clftree.score(dk_test_att, dk_test_label)

```

import librari tree dari sklearn kemudian membuat variabel clftrree berisikan method DecisionTreeClasifier setelah itu variabel tersebut di berikan method fit dengan isian data train vektorissasi dan data training label yang berguna untuk melatih data tersebut agar dapat digunakan pada codingan selanjutnya setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

```

In [49]: from sklearn import tree
...: clftrree = tree.DecisionTreeClassifier()
...: clftrree.fit(dk_train_att, dk_train_label)
...: clftrree.score(dk_test_att, dk_test_label)
Out[49]: 0.9637681159420289

```

**Gambar 4.53** hasil

## 6. plot confusion matrix

```

1 %%##plot comfusion matrix
2 from sklearn.metrics import confusion_matrix
3 pred_labels = clftrree.predict(dk_test_att)
4 cm = confusion_matrix(dk_test_label, pred_labels)
5 cm

```

import library comfusion matrix selanjutnya dilakukan prediksi pada pada data tes nya kemudian data tersebut di masukan kedalam variabel cm dengan method confusion matrix yang di dalamnya terdapat data dari variabel perd label dan dk test label setelah itu variabel cm tersebut di running maka akan memunculkan nilai matrixnya.

```

In [51]: from sklearn.metrics import confusion_matrix
...: pred_labels = clftrree.predict(dk_test_att)
...: cm = confusion_matrix(dk_test_label, pred_labels)
...: cm
Out[51]:
array([[61, 2],
 [3, 72]], dtype=int64)

```

**Gambar 4.54** hasil

## 7. cross valodation

```

1 from sklearn.model_selection import cross_val_score
2 scores = cross_val_score(clftrree, dk_train_att, dk_train_label,
 cv=5)
3 scorerata2=scores.mean()
4 scorersd=scores.std()
5 %%%

```

```

6 from sklearn.model_selection import cross_val_score
7 scores = cross_val_score(clftree, dk_train_att,
8 dk_train_label, cv=5)
show average score and +/- two standard deviations away (covering 95
9 #%% of scores)
10 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
11 scores.std() * 2))
12 #%%:
13 scorestree = cross_val_score(clftree, dk_train_att,
14 dk_train_label, cv=5)
15 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
16 scorestree.std() * 2))
17 #%%:
18 scoressvm = cross_val_score(clfsvm, dk_train_att,
19 dk_train_label, cv=5)
20 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(),
21 scoressvm.std() * 2))

```

memunculkan nilai akurasi dari tiga metode yaitu random forest, decision tree, dan klasifikasi svm (suport vector machine) diamanakan dibandingkan tingkat akurasi dari semua hasil akurasi mana yang terbaik dan lebih akurat pada hasilnya data yang paling akurat yaitu random forest.

```
In [57]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(clftree, dk_train_att, dk_train_label, cv=5)
...: # show average score and +/- two standard deviations away (covering 95
...: #%% of scores)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
...: scores.std() * 2))
Accuracy: 0.94 (+/- 0.08)
```

```
In [58]: from sklearn.model_selection import cross_val_score
...: scores = cross_val_score(clftree, dk_train_att, dk_train_label, cv=5)
...: # show average score and +/- two standard deviations away (covering 95
...: #%% of scores)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(),
...: scores.std() * 2))
Accuracy: 0.95 (+/- 0.06)
```

```
In [59]: scorestree = cross_val_score(clftree, dk_train_att, dk_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
...: scorestree.std() * 2))
Accuracy: 0.95 (+/- 0.05)
```

```
In [60]: scoressvm = cross_val_score(clfsvm, dk_train_att, dk_train_label, cv=5)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(),
...: scoressvm.std() * 2))
Accuracy: 0.54 (+/- 0.01)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The
default value of gamma will change from 'auto' to 'scale' in version 0.22 to account
better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this
```

Gambar 4.55 hasil

## 8. Pengamatan program

```

1 import numpy as np
2 from sklearn.ensemble import RandomForestClassifier
```

```

3 max_features_opts = range(1, 10, 1)
4 n_estimators_opts = range(2, 40, 4)
5 rf_params = np.empty((len(max_features_opts)*len(
6 n_estimators_opts),4) , float)
7 i = 0
8 for max_features in max_features_opts:
9 for n_estimators in n_estimators_opts:
10 clf = RandomForestClassifier(max_features=
11 max_features , n_estimators=n_estimators)
12 scores = cross_val_score(clf , dk_train_att ,
13 dk_train_label , cv=5)
14 rf_params[i,0] = max_features
15 rf_params[i,1] = n_estimators
16 rf_params[i,2] = scores.mean()
17 rf_params[i,3] = scores.std() * 2
18 i += 1
19 print("Max features: %d, num estimators: %d, accuracy
20 : %.02f (+/- %.02f)" %
21 (max_features , n_estimators , scores.mean() , scores.std() *
22))

```

terdapat grafik data yang terdapat dari grafik tersebut di dapat dari codingan dengan cara pengulangan data masing masing 10 kali setelah itu di eksekusi menjadi grafik berbentuk 3D pada gambar tersebut menunjukan rasio dari yang terrendah yaitu data SVM kemudian data decision tree dan hasil random forest.

```

dk_train_label, cv=5)
...: rf_params[i,0] = max_features
...: rf_params[i,1] = n_estimators
...: rf_params[i,2] = scores.mean()
...: rf_params[i,3] = scores.std() * 2
...: i += 1
...: print("Max features: %d, num estimators: %d, accuracy
%0.2f (+/- %.02f)"
...:
...: % (max_features, n_estimators, scores.mean(), scores.std() *
2))
Max features: 1, num estimators: 2, accuracy: 0.85 (+/- 0.08)
Max features: 1, num estimators: 6, accuracy: 0.89 (+/- 0.03)
Max features: 1, num estimators: 10, accuracy: 0.90 (+/- 0.04)
Max features: 1, num estimators: 14, accuracy: 0.93 (+/- 0.04)
Max features: 1, num estimators: 18, accuracy: 0.91 (+/- 0.06)
Max features: 1, num estimators: 22, accuracy: 0.95 (+/- 0.03)
Max features: 1, num estimators: 26, accuracy: 0.93 (+/- 0.06)
Max features: 1, num estimators: 30, accuracy: 0.95 (+/- 0.06)
Max features: 1, num estimators: 34, accuracy: 0.93 (+/- 0.04)
Max features: 1, num estimators: 38, accuracy: 0.95 (+/- 0.03)
Max features: 2, num estimators: 2, accuracy: 0.84 (+/- 0.09)
Max features: 2, num estimators: 6, accuracy: 0.90 (+/- 0.06)
Max features: 2, num estimators: 10, accuracy: 0.93 (+/- 0.03)
Max features: 2, num estimators: 14, accuracy: 0.94 (+/- 0.04)
Max features: 2, num estimators: 18, accuracy: 0.91 (+/- 0.06)
Max features: 2, num estimators: 22, accuracy: 0.93 (+/- 0.05)
Max features: 2, num estimators: 26, accuracy: 0.95 (+/- 0.02)

```

**Gambar 4.56** hasil

Berikut adalah untuk grafik

```

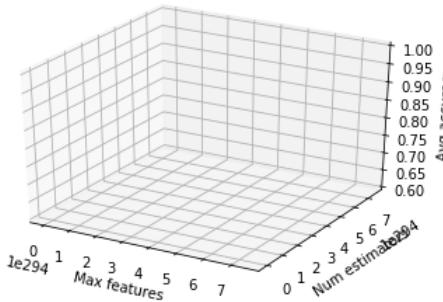
1 #%%
2 import matplotlib.pyplot as plt
3 from mpl_toolkits.mplot3d import Axes3D
4 from matplotlib import cm
5 fig = plt.figure()
6 fig.clf()
7 ax = fig.gca(projection='3d')
8 x = rf_params[:,0]
9 y = rf_params[:,1]
10 z = rf_params[:,2]
11 ax.scatter(x, y, z)
12 ax.set_zlim(0.6, 1)
13 ax.set_xlabel('Max features')
14 ax.set_ylabel('Num estimators')
15 ax.set_zlabel('Avg accuracy')

```

```

....: ax = fig.gca(projection='3d')
....: x = rf_params[:,0]
....: y = rf_params[:,1]
....: z = rf_params[:,2]
....: ax.scatter(x, y, z)
....: ax.set_zlim(0.6, 1)
....: ax.set_xlabel('Max features')
....: ax.set_ylabel('Num estimators')
....: ax.set_zlabel('Avg accuracy')
....: plt.show()

```



**Gambar 4.57** hasil

### 4.3.3 Penanganan Error

#### 4.3.3.1 Screenshoot Error

1. Error 1

```
FileNotFoundException: [Errno 2] File b'D:/Data Alit/Kuliah/SEMESTER VI/AI_KB3B/Chapter04/Youtube03-LMFAO.csv' does not exist: b'D:/Data Alit/Kuliah/SEMESTER VI/AI_KB3B/Chapter04/Youtube03-LMFAO.csv'
```

Gambar 4.58 error 1

## 2. Error 2

```
FileNotFoundException: [Errno 2] File b'D:/Data Alit/Kuliah/SEMESTER VI/AI_KB3B/Chapter04/Youtube03-LMFAO.csv' does not exist: b'D:/Data Alit/Kuliah/SEMESTER VI/AI_KB3B/Chapter04/Youtube03-LMFAO.csv'
```

Gambar 4.59 error 1

### 4.3.3.2 Penanganan Error

1. penanganan 1, kesalahan pada alamat direktori yang dituju, jadi harus dipastikan alamat direktori yang benar

```
data=pd.read_csv("D:/Data Alit/Kuliah/SEMESTER VI/AI_KB3B/Python-Artifi
```

Gambar 4.60 penanganan1

2. penanganan 2, RandomForestClassifier belum diimport, sehingga melakukan import library terlebih dahulu agar program jalan

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier
```

Gambar 4.61 penanganan2

## 4.4 Luthfi Muhammad Nabil (1174035)

### 4.4.1 Teori

1. Jelaskan apa itu klasifikasi teks, sertakan gambar illustrasi buatan sendiri.

Klasifikasi teks merupakan sebuah metode untuk memilah setiap kata yang ada dan mengelompokkannya berdasarkan kelompok yang telah ditentukan. Seperti halnya pengelompokan surat ke kategori spam atau sms, memfilter berita sesuai kategori yang ada. Biasanya klasifikasi teks

digunakan untuk memfilter dan mengelompokkan banyak konten menjadi beberapa yang dapat dibedakan berdasarkan kelompok yang sudah ditentukan. Untuk contoh penyaringan yaitu jika ada sekitar sekian persen isi dari konten didominasi dengan kategori tertentu (Misal : Komputer), maka konten yang dikirim merupakan konten dengan tema komputer.

| Bahasa     | OOP   | Berbasis Web | Berbasis Mobile | Menghubungkan Mesin (BOT) |
|------------|-------|--------------|-----------------|---------------------------|
| Assembly   | Tidak | Tidak        | Tidak           | Ya                        |
| Python     | Ya    | Tidak        | Tidak           | Tidak                     |
| C++        | Tidak | Tidak        | Tidak           | Tidak                     |
| C#         | Ya    | Tidak        | Tidak           | Tidak                     |
| JavaScript | Ya    | Ya           | Tidak           | Tidak                     |
| Java       | Ya    | Tidak        | Ya              | Tidak                     |
| Android    | Ya    | Tidak        | Ya              | Tidak                     |
| Kotlin     | Ya    | Tidak        | Ya              | Tidak                     |
| Dart       | Ya    | Tidak        | Ya              | Tidak                     |
| PHP        | Ya    | Ya           | Tidak           | Tidak                     |

**Gambar 4.62** Illustrasi Klasifikasi Teks

2. Jelaskan mengapa klasifikasi bunga tidak bisa menggunakan machine learning, sertakan illustrasi sendiri

Klasifikasi bunga tidak dapat menggunakan metode machine learning karena banyaknya jenis - jenis dari bunga yang sama namun makna dari bunga tersebut dapat berbeda - beda. Sehingga akan membingungkan sebuah mesin saat memfilter data yang ada. Jika ada pun akan terjadi ketidak tepatan data karena ciri - ciri dari sebuah bunga yang dominan hampir mirip bahkan bisa sama namun dengan makna yang berbeda - beda.

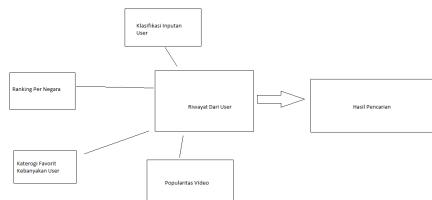


**Gambar 4.63** Maksud klasifikasi bunga tidak bisa

3. Jelaskan bagaimana teknik pembelajaran mesin pada teks pada kata-kata yang digunakan di youtube, jelaskan arti per atribut data csv dan sertakan illustrasi buatan sendiri

Cara youtube mempelajari pemecahan teks untuk dapat membedakan mana yang merupakan sebuah kategori yaitu mengelompokkan kandidat - kandidat yang dimiliki untuk dapat ditampilkan pada user tersebut. Dimulai dari dimana user tinggal, ranking video berdasarkan daerah, riwayat dari pencarian user, dan lain sebagainya. Saat user menginputkan sebuah pencarian, youtube akan merekam pencarian tersebut untuk menjadikan referensi dari apa yang pengguna sering cari sehingga tau apa konteks dari yang dicari pengguna tersebut. Hal itu dilakukan karena terkadang satu kata memiliki banyak makna sehingga youtube akan menyesuaikan makna yang dicari dengan apa yang dimaksud dengan user tersebut.

4. Jelaskan apa yang dimaksud vektorisasi data



**Gambar 4.64** Teknik filtering pada youtube

Vektorisasi data merupakan proses konversi data raster yang diubah menjadi data vektor. Parameter yang digunakan biasanya berupa data beberapa persentase atau kondisi ya/tidak yang berbentuk numerik untuk membedakan setiap data itu memiliki makna apa pada parameter yang dimaksud. Konversi itu wujudnya disebut digitasi. Vektorisasi data perlu dilakukan karena untuk menghitung sebuah nilai pada Artificial Intelligence, diperlukan angka yang dapat dihitung dan kategori dari nilai tersebut sehingga memudahkan dalam menghitung dan menentukan apa yang perlu dilakukan.

5. Jelaskan apa itu bag of words dengan kata-kata yang sederhana dan ilustrasi sendiri.

Bag of Words merupakan model representasi sederhana yang dipakai untuk pengambilan informasi dan pemrosesan bahasa umum atau alami. Pada model ini, sebuah teks akan dikategorikan sebagai bungkusan dari kumpulan kata tanpa perlu diolah ulang sekalipun. Bag of words juga digunakan pada klasifikasi dokumen yang biasanya setiap kata digunakan sebagai fitur dari pelatihan klasifikasi.

| Label      | Total Kata |
|------------|------------|
| Assembly   | 10         |
| Python     | 2          |
| C++        | 5          |
| C#         | 20         |
| JavaScript | 11         |
| Java       | 52         |
| Android    | 22         |
| Kotlin     | 42         |
| Dart       | 35         |
| PHP        | 1          |

**Gambar 4.65** Illustrasi Bag of Words

6. Jelaskan apa itu TF-IDF, illustrasikan dengan gambar sendiri.. TF-IDF merupakan pengukuran statistik yang mengevaluasi seberapa bernilai kata yang dicek di dalam dokumen tersebut. Parameter yang digunakan biasanya yaitu berapa banyak kata tersebut muncul pada sebuah dokumen. Biasanya metode ini digunakan untuk menilai sebuah kata yang digunakan pada algoritma machine learning.

|     | D1  | D2  | D3    | ...   | Dn |
|-----|-----|-----|-------|-------|----|
| t1  | 2)  | 4)  | 5 ... |       | 1  |
| t2  | 1   | 2   | 2 ... |       | 2  |
| t3  |     | 2   | 1     | 5 ... | 2  |
| ... | ... | ... | ...   | ...   |    |
| tn  | 0   | 15  | 2 ... |       | 2  |

Gambar 4.66 Illustrasi TF-IDF

#### 4.4.2 Praktek

- Buat aplikasi sederhana menggunakan pandas, buat data dummy format csv sebanyak 500 baris dan melakukan load ke dataframe pandas. Jelaskan arti setiap baris kode yang dibuat.

```

1 # In [1]: Import library dari pandas
2 import pandas as pan
3 #Membaca file csv menggunakan pandas
4 data_forest = pan.read_csv('1_praktek.csv')
5 # In [2]: untuk melihat jumlah dari baris data yang telah di
 import
6 print(len(data_forest))
7 # In [3]: untuk melihat lima baris pertama data yang telah di
 import
8 print(data_forest.head())
9 # In [4]: untuk mengetahui banyak baris dan kolom dari data
 yang
10 # telah di import.
11 print(data_forest.shape)

```

- Dari dataframe tersebut, dipecah menjadi dua dataframe yaitu 450 row pertama dan 50 row sisanya(Harus beda dengan teman sekelas)

```

1 #%%
2 data_450 = data_forest [:450]
3 #%%
4 data_50 = data_forest [450:]

```

- Praktekkan vektorisasi dan klasifikasi dari data Shakira dengan decision tree. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud dari setiap luaran yang didapatkan.

```

1 npm = 1174035%4
2 print(npm)
3 #%%
4 #Import pandas dan meload file Youtube05-Shakira
5 import pandas as pan
6 data_komentar = pan.read_csv('Youtube05-Shakira.csv')
7
8 #Mengelompokkan spam dan bukan spam ke 2 variabel yang
 berbeda
9 spam=data_komentar.query('CLASS == 1')
10 no_spam=data_komentar.query('CLASS == 0')
11
12 #Melakukan fungsi bag of word dengan cara menghitung semua
 kata yang ada pada file

```

```

13 from sklearn.feature_extraction.text import CountVectorizer
14 vectorizer = CountVectorizer()
15
16 #Membuat variabel dengan perintah untuk memproses kolom
17 CONTENT pada dataset , lalu membaca variable yang ada
18 data_vektor = vectorizer.fit_transform(data_komentar['CONTENT'])
19 data_vektor
20
21 #Menampilkan sampel komentar spam yang didapat
22 print(data_komentar['CONTENT'][304])
23
24 #Menampilkan daftar nama kolom
25 dk=vectorizer.get_feature_names()
26
27 #Untuk melakukan randomisasi dari setiap komentar
28 train_test = data_komentar.sample(frac=1)
29 #Memuat data training dan testing dari data yang sudah ada
30 dk_train=train_test[:300]
31 dk_test=train_test[300:]
32
33 #Melakukan training pada data training dan memvektorisasi
34 data tersebut
35 dk_train_att = vectorizer.fit_transform(dk_train['CONTENT'])
36 dk_train_att
37
38 #Melakukan Testing pada data testing dan memvektorisasi data
39 tersebut
40 dk_test_att=vectorizer.transform(dk_test['CONTENT'])
41 dk_test_att
42
43 #Mengambil label spam dan bukan spam
44 dk_train_label=dk_train['CLASS']
45 dk_test_label = dk_test['CLASS']

```

```

www.earnmoneyonlinewithoutinvestment....just visit this link
http://www.earnmoneyonlinewithoutinvestment.com/visitthislink.php?tk_id=16
D:\MK\import_pesan.csv
data_vektor = pd.read_csv('YoutubeShaming.csv')
#membaca file csv
#memperbaiki karakter spesial >< /> & < /> yang berada
#di dalam komentar query ('>< /> & < />') agar tidak mengganggu saat kita peng
#ada proses
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
#membuat variabel dengan perintah untuk memproses kolom
data_vektor = vectorizer.fit_transform(data_komentar['CONTENT'])
data_vektor
print(data_komentar['CONTENT'][304])
print(data_komentar['CONTENT'][304])
vectorizer.get_feature_names()
#untuk melakukan randomisasi dari setiap komentar
dk_train = data_komentar.sample(frac=1)
#memuat data training dan testing dari data yang sudah ada
dk_train_att = vectorizer.fit_transform(dk_train['CONTENT'])
#melakukan training pada data training dan memvektorisasi data
#tersebut
dk_train_att=vectorizer.fit_transform(dk_train['CONTENT'])
dk_train_att
#memuat data testing dan memvektorisasi data tersebut
dk_test = data_komentar.sample(frac=1)
#memuat data testing dan memvektorisasi data tersebut
dk_test_att=vectorizer.transform(dk_test['CONTENT'])
dk_test_att
#mengambil label spam dan bukan spam
dk_train_label=dk_train['CLASS']
dk_test_label = dk_test['CLASS']
www.earnmoneyonlinewithoutinvestment....just visit this link
http://www.earnmoneyonlinewithoutinvestment.com/visitthislink.php?tk_id=16

```

**Gambar 4.67** Melakukan vektorisasi dan mengambil salah satu contoh

4. Cobalah klasifikasikan dari data vektorisasi yang ditentukan di nomor sebelumnya dengan klasifikasi SVM. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1 from sklearn import svm

```

```

2 clfsvm = svm.SVC()
3 clfsvm.fit(dk_train_att, dk_train_label)
4 clfsvm.score(dk_test_att, dk_test_label)

```

```

In [40]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(dk_train_att, dk_train_label)
...: clfsvm.score(dk_test_att, dk_test_label)
...: Out[40]: 0.77072429774285

```

**Gambar 4.68** Melakukan prediksi CVM berdasarkan nilai

5. Coba klasifikasikan dari data vektorisasi yang ditentukan di nomor sebelumnya dengan klasifikasi Decision Tree. Tunjukkan kelaurannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1 #Melakukan klasifikasi Decision Tree
2 from sklearn import tree
3 clftree = tree.DecisionTreeClassifier()
4 clftree.fit(dk_train_att, dk_train_label)
5 clftree.score(dk_test_att, dk_test_label)

```

```

In [41]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(dk_train_att, dk_train_label)
...: clftree.score(dk_test_att, dk_test_label)
...: Out[41]: 0.9142057142057142

```

**Gambar 4.69** Klasifikasi data dari vektorisasi yang ada dengan decision tree

6. Plotlah confusion matrix dari praktek modul ini menggunakan matplotlib. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

```

1 %%%
2 #Melakukan confusion matrix
3 from sklearn.metrics import confusion_matrix
4 pred_labels = clftree.predict(dk_test_att)
5 cm = confusion_matrix(dk_test_label, pred_labels)
6 cm

```

```

In [42]:
In [42]: from sklearn.metrics import confusion_matrix
...: pred_labels = clftree.predict(dk_test_att)
...: cm = confusion_matrix(dk_test_label, pred_labels)
...: Out[42]:
array([[30, 1],
 [5, 39]], dtype='int64')

```

**Gambar 4.70** Penggunaan confusion matrix

7. Jalankan program cross validation pada bagian teori bab ini. Tunjukkan keluarannya dari komputer sendiri dan artikan mnaksud setiap luaran yang didapatkan.

```

1 from sklearn.model_selection import cross_val_score
2 scores = cross_val_score(clftree, dk_train_att, dk_train_label,
 cv=5)
3 scorerata2=scores.mean()

```

```

4 scorersd=scores.std()
5
6 from sklearn.model_selection import cross_val_score
7 scores = cross_val_score(clftree, dk_train_att,
8 dk_train_label, cv=5)
9 # show average score and +/- two standard deviations away (
10 # covering 95
11 #%% of scores)
12 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
13 std() * 2))
14
15 scorestree = cross_val_score(clftree, dk_train_att,
16 dk_train_label, cv=5)
17 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
18 scorestree.std() * 2))
19
20 scoressvm = cross_val_score(clfsvm, dk_train_att,
21 dk_train_label, cv=5)
22 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(),
23 scoressvm.std() * 2))

```

```

In [8]: from sklearn.model_selection import cross_val_score
... scores = cross_val_score(clftree, dk_train_att, dk_train_label, cv=5)
... scorestree = scores.std()
...
... from sklearn.model_selection import cross_val_score
... scores = cross_val_score(clfsvm, dk_train_att, dk_train_label, cv=5)
... scoressvm = scores.std()
... print("Average accuracy of tree classifier: %0.2f (%0.2f)" % (scorestree.mean(),
... scorestree.std() * 2))
... print("Average accuracy of SVM classifier: %0.2f (%0.2f)" % (scoressvm.mean(),
... scoressvm.std() * 2))
... scorestree = cross_val_score(clftree, dk_train_att, dk_train_label,
... cv=5)
... print("Accuracy: %0.2f (%0.2f)" % (scorestree.mean(),
... scorestree.std() * 2))
... scoressvm = cross_val_score(clfsvm, dk_train_att, dk_train_label,
... cv=5)
... print("Accuracy: %0.2f (%0.2f)" % (scoressvm.mean(),
... scoressvm.std() * 2))
... Accuracy: 0.81 (+/- 0.01)
... Accuracy: 0.80 (+/- 0.01)
... Accuracy: 0.80 (+/- 0.01)

```

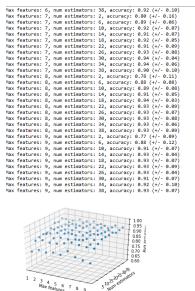
**Gambar 4.71** Mengambil tingkat akurasi dari klasifikasi data dan prediksi

8. Buatlah program pengamatan komponen informasi pada bagian teori bab ini. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

```

1 #%%
2 from sklearn.ensemble import RandomForestClassifier
3 import numpy as np
4 max_features_opts = range(1, 10, 1)
5 n_estimators_opts = range(2, 40, 4)
6 rf_params = np.empty((len(max_features_opts)*len(
7 n_estimators_opts),4), float)
8 i = 0
9 for max_features in max_features_opts:
10 for n_estimators in n_estimators_opts:
11 clf = RandomForestClassifier(max_features=
12 max_features,n_estimators=n_estimators)
13 scores = cross_val_score(clf, dk_train_att,
14 dk_train_label, cv=5)
15 rf_params[i,0] = max_features
16 rf_params[i,1] = n_estimators
17 rf_params[i,2] = scores.mean()
18 rf_params[i,3] = scores.std() * 2
19 i += 1
20 print("Max features: %d, num estimators: %d, accuracy
21 : %0.2f (+/- %0.2f)"
```

```
18 % (max_features , n_estimators , scores . mean () , scores . std ()
19 * 2))
20
21 import matplotlib . pyplot as plt
22 from mpl_toolkits.mplot3d import Axes3D
23 from matplotlib import cm
24 fig = plt . figure ()
25 fig . clf ()
26 ax = fig . gca (projection = ' 3d ')
27 x = rf_params [: , 0]
28 y = rf_params [: , 1]
29 z = rf_params [: , 2]
30 ax . scatter (x , y , z)
31 ax . set_zlim (0.6 , 1)
32 ax . set_xlabel (' Max features ')
33 ax . set_ylabel (' Num estimators ')
34 ax . set_zlabel (' Avg accuracy ')
35 plt . show ()
```



**Gambar 4.72** Melakukan regresi data dengan numpy dan mengambil fitur - fitur yang ada

#### **4.4.3 Penanganan Error**

## 1. Skrinsut Error

**Gambar 4.73** Error

## 2. Tipe Error : EOF

### 3. Penanganan : Meluruskan String

#Awalnya

```

print("Max
features: %d, num estimators: %d, accuracy: %.2f (+/- %.2f)
#Sekarang
print("Max features: %d, num estimators: %d, accuracy: %.2f

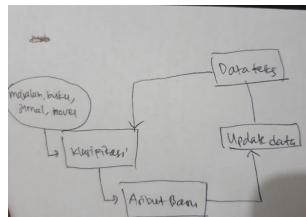
```

## 4.5 1174039 - Liyana majdah rahma

### 4.5.1 Teori

#### 1. Klasifikasi Text

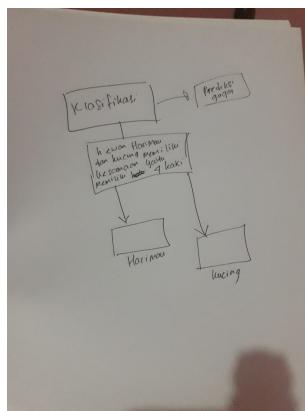
Pengolahan teks sangat dibutuhkan pada aplikasi yang memiliki jumlah dokumen yang berukuran besar. klasifikasi teks sendiri merupakan cara dalam memilah data teks berdasarkan parameter serta dengan data yang bersifat dokumen. Data tersebut dapat berupa char atau string.



**Gambar 4.74** Klasifikasi Text

#### 2. Mengapa Klasifikasi Bunga tidak dapat menggunakan Machine Learning

dikarenakan klasifikasi itu menggunakan tipe data yang dimana atributnya memiliki nilai data yang berupa vektor dengan perbandingan masing-masing data yang memiliki sedikit perbedaan. Dapat dilihat pada gambar berikut.

**Gambar 4.75** Klasifikasi Hewan

3. Teknik Pembelajaran machine learning pada teks kata-kata di youtube

Contohnya pada saat kita membuka sebuah video maka di sebelah kanan ada list "berikutnya", pada saat itu mesin melakukan ujicoba dan apabila anda menekan salah satu dari video tersebut maka hal tersebut akan direkam dan disimpan oleh mesin tersebut

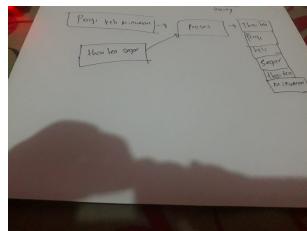
**Gambar 4.76** Machine Learning Youtube

4. Jelaskan apa yang dimaksud vektorisasi data.

Vektorisasi data merupakan pemecahan data menjadi bagian-bagian yang lebih sederhana. Contoh ada sebuah paragraf, dari paragraf tersebut akan dibagi-bagi menjadi kalimat, yang nantinya akan dibagi-bagi kembali menjadi perkata.

5. Jelaskan apa itu bag of words dengan kata-kata yang sederhana dan ilustrasi sendiri

Bag of words merupakan penyajian sederhana yang biasanya digunakan pada aplikasi text mining pada saat mengenalkan struktur ke sebuah kumpulan dokumen yang berbasis teks untuk diklasifikasikan kembali menjadi dua atau lebih kelas yang telah ditentukan.



Gambar 4.77 Bag of Words

6. Jelaskan apa itu TF-IDF, ilustrasikan dengan gambar sendiri.

TF-IDF merupakan metode untuk menghitung bobot dari kata yang sering muncul pada suatu kalimat. metode ini menghitung nilai Term Frequency atau Inverse Document Frequency pada setiap kata pada kalimat yang dijadikan acuan kata pada metode ini sering disebut token adapun rumus dari metode ini dapat dilihat pada gambar.

|        | tf |    |    | df | tf/df | IDF(x) | WeightedIDF(x) |       |       |
|--------|----|----|----|----|-------|--------|----------------|-------|-------|
|        | d1 | d2 | d3 |    |       |        | d2             | d3    |       |
| gold   | 0  | 1  | 1  | 2  | 1.5   | 0.378  | 1.379          | 1.379 | 1.379 |
| silver | 0  | 2  | 0  | 1  | 3     | 0.477  | 1.477          | 0     | 2.954 |
| truck  | 1  | 1  | 1  | 2  | 1.5   | 0.378  | 1.379          | 1.379 | 1.379 |

Nilai Bobot tag Dokumen =  $\frac{\text{WeightedIDF}(d1)}{\text{sum}(d1)} + \frac{\text{WeightedIDF}(d2)}{\text{sum}(d2)} + \frac{\text{WeightedIDF}(d3)}{\text{sum}(d3)}$  =  $1.379 + 4.33 + 2.952$

Gambar 4.78 TF-IDF

#### 4.5.2 Praktek

1. import data pandas dan 500 baris data dummy kemudian di jelaskan tiap barisnya.

```

1 # In [1]: Import library dari pandas
2 import pandas as pd
3 #Membaca file csv menggunakan pandas
4 data_forest = pd.read_csv('1174039.csv')
5 # In [2]: untuk melihat jumlah dari baris data yang telah di
 import
6 print(len(data_forest))

```



Gambar 4.79 Data Dummy

2. memecah data frame menjadi dua yang pertama 450 dan kedua sisanya

```

1
2 #Mengelompokkan spam dan bukan spam ke 2 variabel yang
 berbeda
3 spam=data_komentar.query('CLASS == 1')

```

|      |                    |                                                        |
|------|--------------------|--------------------------------------------------------|
| dtes | DataFrame (50, 5)  | Column names: id, first_name, last_name, email, gender |
| dtra | DataFrame (450, 5) | Column names: id, first_name, last_name, email, gender |

**Gambar 4.80** Pisah Data

### 3. praktik vektorisasi

```

1 no_spam=data_komentar.query('CLASS == 0')
2
3 #Melakukan fungsi bag of word dengan cara menghitung semua
4 #kata yang ada pada file
5 from sklearn.feature_extraction.text import CountVectorizer
6 vectorizer = CountVectorizer()
7
8 #Membuat variabel dengan perintah untuk memproses kolom
9 #CONTENT pada dataset, lalu membaca variable yang ada
10 data_vektor = vectorizer.fit_transform(data_komentar['CONTENT'])
11 data_vektor
12
13 #Menampilkan sampel komentar spam yang didapat
14 print(data_komentar['CONTENT'][304])
15
16 #Menampilkan daftar nama kolom
17 dk=vectorizer.get_feature_names()
18
19 #Untuk melakukan randomisasi dari setiap komentar
20 train_test = data_komentar.sample(frac=1)
21 #Memuat data training dan testing dari data yang sudah ada
22 dk_train=train_test[:300]
23 dk_test=train_test[300:]
24
25 #Melakukan training pada data training dan memvektorisasi
26 #data tersebut
27 dk_train_att = vectorizer.fit_transform(dk_train['CONTENT'])
28 dk_train_att
29
30 #Melakukan Testing pada data testing dan memvektorisasi data
31 #tersebut
32 dk_test_att=vectorizer.transform(dk_test['CONTENT'])
33 dk_test_att
34
35 #Mengambil label spam dan bukan spam

```

lakukan import library pandas yang diinisialisasi menjadi pd setelah itu ada dibuat variable data dengan method read\_csv untuk membaca file berekstensi csv yang dimasukan alamatnya pada kurung, lakukan klasifikasi atau pemilihan komentar yang berisi spam atau bukan spam dengan parameter class sama dengan 1 merupakan spam dan class sama dengan 0 bukan spam setelah itu masukan librari CountVektorizer yang digunakan untuk vektorisasi data kemudian dilanjutkan pada bagian In[103] dibuat variabel yang berisi vektorisasi dari data pada data di field content setelah itu variabel tersebut di running hasilnya menunjukkan 350 baris

di kali 1738 kolom selanjutnya dicoba untuk memunculkan isi record pada baris ke 345 maka akan muncul isian dari baris tersebut. selanjutnya dibuat variabel dk atau daftar yang berisi data hasil vektorisasi setelah yang terdiri dari variabel dshuf yang berisi data komen yang di dalamnya di buat random yang nantinya akan dibutuhkan data training dan data testing dengan ketentuan data training 300 dan data testing sebanyak 50 setelah itu data training dilakukan vektorisasi dan data testing juga dilakukan vektorisasi setelah itu kedua data training dan testing tersebut dibuat label dengan parameter field CLASS pada tabel.

**Gambar 4.81** Vektorisasi

#### 4. klasifikasi SVM

```
1 dk_test_label = dk_test['CLASS']
2
3 #%%
4 #Coding untuk melakukan klasifikasi SVM
5 from sklearn import svm
```

import librari svm dari sklearn kemudian membuat variabel clfsvm berisikan method svc setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

```
In [41]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(dk_train_att, dk_train_label)
...: clftree.score(dk_test_att, dk_test_label)
Out[41]: 0.9142857142857143
```

**Gambar 4.82** SVM

## 5. klasifikasi decision tree

```
1 clfsvm.fit(dk_train_att, dk_train_label)
2 clfsvm.score(dk_test_att, dk_test_label)
```

import librari tree dari sklearn kemudian membuat variabel clftree berisikan method DecisionTreeClasifier setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut agar dapat digunakan pada codingan selanjutnya setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

```
In [67]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(dk_train_att, dk_train_label)
...: clftree.score(dk_test_att, dk_test_label)
Out[67]: 0.96
```

**Gambar 4.83** Desicion Tree

## 6. plot comfusion matrix

```
1 from sklearn import tree
2 clftree = tree.DecisionTreeClassifier()
3 clftree.fit(dk_train_att, dk_train_label)
4 clftree.score(dk_test_att, dk_test_label)
```

import library comfusion matrix selanjutnya dilakukan prediksi pada pada data tes nya kemudian data tersebut di masukan kedalam variabel cm dengan method confusion matrix yang di dalamnya terdapat data dari variabel perd label dan dk test label setelah itu variabel cm tersebut di running maka akan memunculkan nilai matrixnya.

```
In [43]:
In [43]: from sklearn.metrics import confusion_matrix
...: pred_labels = clftree.predict(dk_test_att)
...: cm = confusion_matrix(dk_test_label, pred_labels)
Out[43]:
array([[38, 1],
 [5, 26]], dtype=int64)
```

**Gambar 4.84** Confussion Matrix

## 7. cross valodation

```
1 #%%
2 #Melakukan confussion matrix
3 from sklearn.metrics import confusion_matrix
4 pred_labels = clftree.predict(dk_test_att)
5 cm = confusion_matrix(dk_test_label, pred_labels)
6 cm
7
8 #%%
9 from sklearn.model_selection import cross_val_score
10 scores = cross_val_score(clftree, dk_train_att, dk_train_label,
 cv=5)
```

```

11 scorerata2=scores.mean()
12 scorersd=scores.std()
13
14 from sklearn.model_selection import cross_val_score
15 scores = cross_val_score(clftree , dk_train_att ,
16 dk_train_label , cv=5)
17 # show average score and +/- two standard deviations away (
18 covering 95
19 ##% of scores)
20 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean() , scores.
21 std() * 2))
22
23 scorestree = cross_val_score(clftree , dk_train_att ,
24 dk_train_label , cv=5)

```

memunculkan nilai akurasi dari tiga metode yaitu random forest, decision tree, dan klasifikasi svm (suport vector machine) diamana akan di bandingkan tingkat akurasi dari semua hasil akurasi mana yang terbaik dan lebih akurat pada hasilnya data yang paling akurat yaitu random forest.



**Gambar 4.85** Cross Validation

## 8. Pengamatan program

```

1 scoressvm = cross_val_score(clfsvm , dk_train_att ,
2 dk_train_label , cv=5)
3 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean() ,
4 scoressvm.std() * 2))
5
5 ##%
6 from sklearn.ensemble import RandomForestClassifier
7 import numpy as np
8 max_features_opts = range(1, 10, 1)
9 n_estimators_opts = range(2, 40, 4)
10 rf_params = np.empty((len(max_features_opts)*len(
11 n_estimators_opts),4) , float)
12 i = 0
13 for max_features in max_features_opts:
14 for n_estimators in n_estimators_opts:
15 clf = RandomForestClassifier(max_features=
16 max_features ,n_estimators=n_estimators)
17 scores = cross_val_score(clf , dk_train_att ,
18 dk_train_label , cv=5)
19 rf_params[i,0] = max_features
20 rf_params[i,1] = n_estimators
21 rf_params[i,2] = scores.mean()
22

```

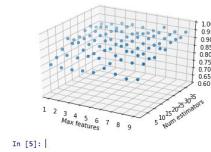
terdapat grafik data yang terdapat dari grafik tersebut di dapat dari codingan dengan cara pengulangan data masing masing 10 kali setelah itu

di eksekusi menjadi grafik berbentuk 3D pada gambar tersebut menunjukkan rasio dari yang terrendah yaitu data SVM kemudian data decision tree dan hasil random forest.

### Gambar 4.86 Pengamatan Program

Berikut adalah untuk grafik

```
1 % (max_features , n_estimators , scores . mean () , scores . std ()
2 * 2))
3
4 import matplotlib . pyplot as plt
5 from mpl_toolkits . mplot3d import Axes3D
6 from matplotlib import cm
7 fig = plt . figure ()
8 fig . clf ()
9 ax = fig . gca (projection = ' 3d ')
10 x = rf_params [: , 0]
11 y = rf_params [: , 1]
12 z = rf_params [: , 2]
13 ax . scatter (x , y , z)
14 ax . set_zlim (0.6 , 1)
15 ax . set_xlabel (' Max features ')
16 ax . set_ylabel (' Num estimators ')
```



**Gambar 4.87** Grafik

### 4.5.3 Penanganan Error

#### 4.5.3.1 Screenshoot Error

- ## 1. Error 1

```
...
rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),4) , float)
NameError: name 'rp' is not defined
```

Gambar 4.88 Error1

#### 4.5.3.2 Kode Error dan Jenisnya

1. Kode Error 1 jenis Name Error

```
max_features_opts = range(1, 10, 1)
n_estimators_opts = range(2, 40, 4)
rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),4) , float)
l = 0
```

Gambar 4.89 Error1

#### 4.5.3.3 Solusi

1. Mengimport library numpy sebagai np

```
import numpy as np
max_features_opts = range(1, 10, 1)
n_estimators_opts = range(2, 40, 4)
rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),4) , float)
l = 0
```

Gambar 4.90 Solusi 1

## 4.6 1174050 Dika Sukma Pradana

### 4.6.1 Teori

1. Jelaskan apa itu klasifikasi teks, sertakan gambar ilustrasi buatan sendiri.

Klasifikasi teks adalah proses pemberian tag atau kategori ke teks sesuai dengan isinya. Ini salah satu tugas mendasar dalam Natural Language Processing (NLP) dengan aplikasi luas seperti analisis sentimen, pelabelan topik, deteksi spam, dan deteksi maksud.



Gambar 4.91 contoh klasifikasi teks

2. Jelaskan mengapa klasifikasi bunga tidak bisa menggunakan machine learning, sertakan ilustrasi sendiri.

Dikarenakan tidak semua bunga memiliki ciri - ciri yang sama. Atau dalam kata lain terdapat data noise dalam klasifikasi bunga sehingga tidak bisa menggunakan machine learning.

Contohnya Anggrek memiliki warna ungu, dengan jumlah kelopak 5. Kemudian ada bunga warna ungu dengan jumlah kelopak yang sama namun ternyata bukan anggrek dan kategorinya banyak sekali. Bahkan ada

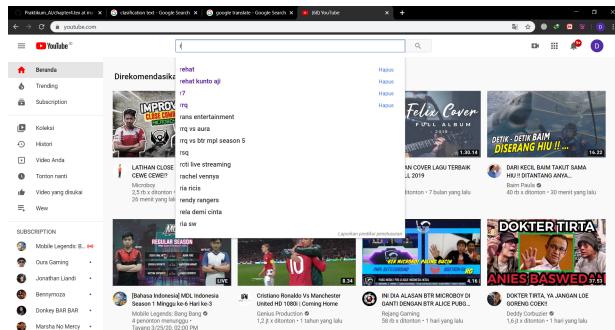


**Gambar 4.92** contoh klasifikasi bunga

bunga yang tidak jelas apakah warnanya sesuai atau tidak, sehingga bisa menyebabkan data noise.

- Jelaskan bagaimana teknik pembelajaran mesin pada teks pada kata-kata yang digunakan di youtube, jelaskan arti per atribut data csv dan sertakan ilustrasi buatan sendiri.

Youtube akan mengingat kata - kata yang sering kita gunakan pada bagian search. seperti pada saat kita menginputkan huruf r maka akan muncul rekomendasi dari youtube baik yang sudah pernah kita cari atau yang belum pernah.



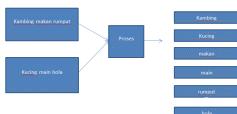
**Gambar 4.93** contoh teknik pembelajaran mesin

- Jelaskan apa yang dimaksud vektorisasi data.

Pembagian dan pemecahan data, kemudian dilakukan perhitungan. Vektorisasi juga dapat dimaksudkan dengan setiap data yang mungkin dipetakan ke integer tertentu. jika kita memiliki array yang cukup besar maka setiap kata / data cocok dengan slot unik dalam array (nilai pada indeks adalah nomor satu kali kata itu muncul).

- Jelaskan apa itu bag of words dengan kata-kata yang sederhana dan ilustrasi sendiri. Bag-of-words ialah sebuah gambaran sederhana digunakan dalam pengolahan bahasa alami dan pencarian informasi. Dikenal sebagai model ruang vektor. Pada model ini, tiap kalimat dalam dokumen

digambarkan sebagai token, mengabaikan tata bahasa dan bahkan urutan kata namun menghitung frekuensi kejadian atau kemunculan kata dari dokumen.



**Gambar 4.94** contoh bag of words

6. Jelaskan apa itu TF-IDF, ilustrasikan dengan gambar sendiri. TF-IDF memberi kita frekuensi kata dalam setiap dokumen dalam korpus atau mengganti data jadi number. Ini adalah rasio berapa kali kata itu muncul dalam dokumen dibandingkan dengan jumlah total kata dalam dokumen itu. Itu meningkat seiring jumlah kemunculan kata itu di dalam dokumen meningkat. Setiap dokumen memiliki tf sendiri.

|       | Kuning | Makan | Racing | Main | Bola |
|-------|--------|-------|--------|------|------|
| Doc 1 | 2      | 1     | 1      | 1    | 1    |
| Doc 2 | 3      | 1     | 1      | 1    | 1    |

**Gambar 4.95** contoh TF-IDF

#### 4.6.2 Praktek Program

1. import data pandas dan 500 baris data dummy kemudian di jelaskan tiap barisnya.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 23 21:14:37 2020
4 @author: User
5 """
6
7 import pandas as pd# untuk membaca file tex atau csv
8 data_forest = pd.read_csv("D:/SEMESTER 6/New folder/
9 datakaryawan.csv")#membaca file csv menggunakan fungsi
10 read_csv dari padas
11 print(len(data_forest))#untuk melihat jumlah dari baris data
12 yang telah di import
13 print(data_forest.head()) #untuk melihat lima baris pertama
14 data yang telah di import
15 print(data_forest.shape)#untuk mengetahui banyak baris dan
16 kolom dari data yang telah di import.

```

2. memecah data prame menjadi dua yang pertama 450 dan kedua sisanya 50

```

1 # -*- coding: utf-8 -*-
2 """

```

```
In [6]: runfile('D:/SEMESTER 6/New folder/1174050/1.py', wdir='D:/SEMESTER 6/New folder/1174050')
148654
 Id EmployeeName ... Agency Status
0 1 NATHANIEL FORD ... San Francisco NaN
1 2 GARY JIMENEZ ... San Francisco NaN
2 3 ALBERT PARDINI ... San Francisco NaN
3 4 CHRISTOPHER CHONG ... San Francisco NaN
4 5 PATRICK GARDNER ... San Francisco NaN

[5 rows x 13 columns]
(148654, 13)
```

Gambar 4.96 hasil

```
3 Created on Mon Mar 23 21:35:59 2020
4
5 @author: User
6 """
7
8 data_train = data_forest [:450] # memasukkan data training
9 sebanyak 450 baris
10
10 data_tes = data_forest [:450] # memasukkan data tes sebanyak 50
```

|               |                        |                                                                            |
|---------------|------------------------|----------------------------------------------------------------------------|
| acak_acak     | DataFrame (448, 5)     | Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS                     |
| data_forest   | DataFrame (148654, 13) | Column names: Id, EmployeeName, JobTitle, BasePay, OvertimePay, OtherP ... |
| data_komen    | DataFrame (448, 5)     | Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS                     |
| data_tes      | DataFrame (148204, 13) | Column names: Id, EmployeeName, JobTitle, BasePay, OvertimePay, OtherP ... |
| data_testing  | DataFrame (148204, 13) | Column names: Id, EmployeeName, JobTitle, BasePay, OvertimePay, OtherP ... |
| data_train    | DataFrame (450, 13)    | Column names: Id, EmployeeName, JobTitle, BasePay, OvertimePay, OtherP ... |
| data_komentar | DataFrame (450, 13)    | Column names: Id, EmployeeName, JobTitle, BasePay, OvertimePay, OtherP ... |

Gambar 4.97 hasil

### 3. praktik vektorisasi

berikut merupakan codingan untuk melakukan vektorisasi data berupa teks dalam format csv

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 23 21:49:14 2020
4
5 @author: User
6 """
7
8
9 #melakukan import pandas untuk membaca file csv
10 import pandas as pd
11 data_komen=pd.read_csv("D:\SEMESTER 6\Python-Artificial-
12 Intelligence-Projects-for-Beginners-master\Chapter03\
13 Youtube04-Eminem.csv")
14 #mengelompokkan komentar spam dan bukan spam
15 spam=data_komen.query('CLASS == 1')
16 nospam=data_komen.query('CLASS == 0')
17 # memanggil lib vektorisasi
```

```

16 #melakukan fungsi bag of word dengan cara menghitung semua
17 kata
18 #yang terdapat dalam file
19 from sklearn.feature_extraction.text import CountVectorizer
20 vectorizer = CountVectorizer()
21 # memilih colom CONTENT untuk dilakukan vektorisasi
22 #melakukan bag of word pada dataframe pada colom CONTENT
23 data_vektorisasi = vectorizer.fit_transform(data_komen[,
24 'CONTENT'])
25 # melihat isi vektorisasi
26 data_vektorisasi
27 # melihat isi data pada baris ke 349
28 print(data_komen['CONTENT'][349])
29 # melihat daftar kata yang di vektorisasi
30 #feature_names merupakan digunakan untuk mengambil nama
31 #kolomnya ada apa saja
32 dk=vectorizer.get_feature_names()
33 # akan melakukan randomisasi pada database nya supaya
34 #semurnya saat melakukan klasifikasi
35 acak_acak = data_komen.sample(frac=1)
36 # membuat data traning dan testing
37 dk_train=acak_acak[:300]
38 dk_test=acak_acak[300:]
39 # melakukan training pada data training dan di vektorisasi
40 dk_train_att=vectorizer.fit_transform(dk_train['CONTENT'])
41 print(dk_train_att)
42 #melakukan testing pada data testing dan di vektorisasi
43 dk_test_att=vectorizer.transform(dk_test['CONTENT'])
44 print(dk_test_att)
45 # Dimana akan mengambil label spam dan bukan spam
46 dk_train_label=dk_train['CLASS']
47 print(dk_train_label)
48 dk_test_label=dk_test['CLASS']
49 print(dk_test_label)

```

---

because we love doing this. We may not have the best recording equipment but if you  
listen to our lyrics and rhymes I think you'll like it. If you do then please  
subscribe and share because we love making these videos and we want you to like them as  
much as possible so feel free to comment and give us pointers! Thank you!

|           |   |
|-----------|---|
| (0, 1012) | 1 |
| (0, 620)  | 1 |
| (0, 111)  | 1 |
| (0, 1043) | 1 |
| (0, 1106) | 1 |
| (0, 746)  | 1 |
| (0, 1195) | 1 |
| (1, 925)  | 1 |
| (1, 622)  | 1 |
| (1, 602)  | 1 |
| (1, 662)  | 1 |
| (1, 395)  | 1 |
| (1, 1179) | 1 |
| (2, 957)  | 1 |
| (2, 536)  | 2 |
| (2, 1169) | 1 |
| (2, 326)  | 1 |
| (2, 21)   | 1 |
| (3, 957)  | 1 |
| (3, 131)  | 1 |

Gambar 4.98 hasil

import library pandas yang di inisialisasi menjadi pd setelah itu ada dibuat class dan method untuk membaca file csv yang di masukan alamat-

nya pada kurung, lakukan klasifikasi samadengan 1 merupakan spam dan class samadengan 0 bukan spam setelah itu masukan librari CountVektorizer yang digunakan untuk vektorisasi data, lalu dilanjutkan pada bagian dibuat variabel yang berisi vektorisasi dari data di field content setelah itu variabel tersebut di running hasilnya menunjukan 350 baris di kali 1738 kolom selanjutnya dicoba untuk memunculkan isi record pada baris ke 349 maka akan muncul isian dari baris tersebut. selanjutnya dibuat variabel yang berisi data hasil vektorisasi setelah yang terdiri dari variabel yang berisi data komen yang di dalamnya di buat random yang nantinya akan dibut data training dan data testing dengan ketentuan data training 300 dan data testing sebanyak 50 setelah itu data training di lakukan vektorisasi dan data testing juga dilakukan vektorisasi setelah itu kedua data training dan testing tersebut dibuat label dengan parameter field CLASS.

#### 4. klasifikasi SVM berikut kodingnya :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 23 22:00:11 2020
4
5 @author: User
6 """
7
8 #Coding untuk melakukan klasifikasi SVM
9 from sklearn import svm
10 clfsvm = svm.SVC()
11 clfsvm.fit(dk_train_att, dk_train_label)
12 clfsvm.score(dk_test_att, dk_test_label)
```

```

In [19]: runfile('D:/SEMESTER 6/New folder/1174050/4.py', wdir='D:/SEMESTER 6/New
folder/1174050')
In [20]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(dk_train_att, dk_train_label)
...: clfsvm.score(dk_test_att, dk_test_label)
Out[20]: 0.9054054054054054
```

**Gambar 4.99** hasil

melakukan verifikasi import librari svm dari sklearn kemudian membuat variabel clfsvm berisikan method svc setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

#### 5. klasifikasi decision tree berikut ini merupakan codingan klasifikasi decision tree

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 23 22:11:57 2020
```

```

4
5 @author: User
6 """
7
8 #Melakukan klasifikasi Decision Tree
9 from sklearn import tree
10 clftree = tree.DecisionTreeClassifier()
11 clftree.fit(dk_train_att, dk_train_label)
12 clftree.score(dk_test_att, dk_test_label)

```

```

In [21]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(dk_train_att, dk_train_label)
...: clftree.score(dk_test_att, dk_test_label)
Out[21]: 0.8986486486486487

```

**Gambar 4.100** hasil

import library tree dari sklearn kemudian membuat variabel clftree berisikan method DecisionTreeClasifier setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut agar dapat digunakan pada codingan selanjutnya setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

6. plot confusion matrix berikut merupakan codingan untuk confusion matrix

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 23 22:13:25 2020
4
5 @author: User
6 """
7
8 #Melakukan confusion matrix
9 from sklearn.metrics import confusion_matrix
10 pred_labels = clftree.predict(dk_test_att)
11 cm = confusion_matrix(dk_test_label, pred_labels)
12 cm

```

```

In [22]: from sklearn.metrics import confusion_matrix
...: pred_labels = clftree.predict(dk_test_att)
...: cm = confusion_matrix(dk_test_label, pred_labels)
...: cm
Out[22]:
array([[66, 2],
 [13, 67]], dtype=int64)

```

**Gambar 4.101** hasil

import library comfusion matrix selanjutnya dilakukan prediksi pada pada data tes nya kemudian data tersebut di masukan kedalam variabel cm dengan method confusion matrix yang di dalamnya terdapat data

dari variabel perd label dan dk test label setelah itu variabel cm tersebut di running maka akan memunculkan nilai matrixnya.

7. cross valodation berikut merupakan code untuk cross validation pada codingan pertama yaitu melakukan split 5 kali yaoti mengitung tingkat akurasi menggunakan data training.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 23 22:14:06 2020
4
5 @author: User
6 """
7
8 from sklearn.model_selection import cross_val_score
9 scores = cross_val_score(clftree, dk_train_att, dk_train_label,
10 cv=5)
11 scorerata2=scores.mean()
12 scorsersd=scores.std()
13
14 from sklearn.model_selection import cross_val_score
15 scores = cross_val_score(clftree, dk_train_att,
16 dk_train_label, cv=5)
17 # show average score and +/- two standard deviations away (
18 # covering 95
19 #%% of scores)
20 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
21 std() * 2))
22
23 scorestree = cross_val_score(clftree, dk_train_att,
24 dk_train_label, cv=5)
25 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
26 scorestree.std() * 2))
27
28 scoressvm = cross_val_score(clfsvm, dk_train_att,
29 dk_train_label, cv=5)
30 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(),
31 scoressvm.std() * 2))

```

```

In [24]: runfile('D:/SEMESTER 6/New folder/1174050/7.py', wdir='D:/SEMESTER 6/New
folder/1174050')
Accuracy: 0.96 (+/- 0.03)
Accuracy: 0.95 (+/- 0.02)
Accuracy: 0.94 (+/- 0.03)

```

**Gambar 4.102** hasil

akan di bandingkan tingkat akurasi dari semua hasil akurasiya yang akan memunculkan nilai akurasi dari tiga metode yaitu random forest, decision tree, dan klasifikasi svm (suport vector machine) diamana mana yang terbaik dan lebih akurat pada hasilnya data yang paling akurat yaitu random forest. terdapat grafik data yang terdapat dari grafik tersebut di dapat dari codingan dengan cara pengulangan data masing masing 10 kali setelah itu di eksekusi menjadi grafik berbentuk 3D pada gambar

tersebut menunjukkan rasio dari yang terrendah yaitu data SVM kemudian data decision tree dan hasil random forest.

## 8. Pengamatan program

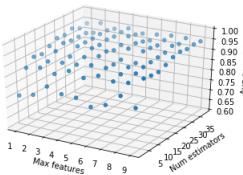
```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Mar 23 22:19:22 2020
4
5 @author: User
6 """
7
8 from sklearn.ensemble import RandomForestClassifier
9 import numpy as np
10 max_features_opts = range(1, 10, 1)
11 n_estimators_opts = range(2, 40, 4)
12 rf_params = np.empty((len(max_features_opts)*len(
13 n_estimators_opts),4), float)
13 i = 0
14 for max_features in max_features_opts:
15 for n_estimators in n_estimators_opts:
16 clf = RandomForestClassifier(max_features=
17 max_features,n_estimators=n_estimators)
18 scores = cross_val_score(clf, dk_train_att,
19 dk_train_label, cv=5)
20 rf_params[i,0] = max_features
21 rf_params[i,1] = n_estimators
22 rf_params[i,2] = scores.mean()
23 rf_params[i,3] = scores.std() * 2
24 i += 1
25 print("Max features: %d, num estimators: %d, accuracy
26 : %0.2f (+/- %0.2f)" %
27 (max_features, n_estimators, scores.mean(), scores.std()
28 * 2))
29
30 import matplotlib.pyplot as plt
31 from mpl_toolkits.mplot3d import Axes3D
32 from matplotlib import cm
33 fig = plt.figure()
34 fig.clf()
35 ax = fig.gca(projection='3d')
36 x = rf_params[:,0]
37 y = rf_params[:,1]
38 z = rf_params[:,2]
39 ax.scatter(x, y, z)
40 ax.set_zlim(0.6, 1)
41 ax.set_xlabel('Max features')
42 ax.set_ylabel('Num estimators')
43 ax.set_zlabel('Avg accuracy')
44 plt.show()
```

### 4.6.3 Penanganan Error

#### 1. screenshoot error

```
Max features: 9, num estimators: 6, accuracy: 0.92 (+/- 0.06)
Max features: 9, num estimators: 10, accuracy: 0.92 (+/- 0.08)
Max features: 9, num estimators: 14, accuracy: 0.91 (+/- 0.03)
Max features: 9, num estimators: 18, accuracy: 0.93 (+/- 0.03)
Max features: 9, num estimators: 22, accuracy: 0.94 (+/- 0.03)
Max features: 9, num estimators: 26, accuracy: 0.95 (+/- 0.02)
Max features: 9, num estimators: 30, accuracy: 0.94 (+/- 0.03)
Max features: 9, num estimators: 34, accuracy: 0.95 (+/- 0.02)
Max features: 9, num estimators: 38, accuracy: 0.94 (+/- 0.03)
```

**Gambar 4.103** hasil

```
Max features: 9, num estimators: 38, accuracy: 0.94 (+/- 0.04)
Traceback (most recent call last):
 File "<ipython-input-28-a024febdaf7>", line 1, in <module>
 runfile('D:/SEMESTER 6/New folder/1174050/8.py', wdir='D:/SEMESTER 6/New folder/1174050')
 File "C:\Users\>User\Anaconda3\lib\site-packages\spyder_kernels\customize\spydercustomize.py", line 827, in runfile
 execfile(filename, namespace)
 File "C:\Users\>User\Anaconda3\lib\site-packages\spyder_kernels\customize\spydercustomize.py", line 110, in execfile
 exec(compile(f.read(), filename, 'exec'), namespace)
 File "D:/SEMESTER 6/New folder/1174050/8.py", line 29, in <module>
 fig = plt.figures()
AttributeError: module 'matplotlib.pyplot' has no attribute 'figures'
```

**Gambar 4.104** hasil

## 2. codingan yang error

```
-*- coding: utf-8 -*-
"""
Created on Mon Mar 23 22:19:22 2020

```

```
@author: User
"""
```

```
from sklearn.ensemble import RandomForestClassifier
import numpy as np
max_features_opts = range(1, 10, 1)
n_estimators_opts = range(2, 40, 4)
rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),
i = 0
for max_features in max_features_opts:
 for n_estimators in n_estimators_opts:
 clf = RandomForestClassifier(max_features=max_features,n_est
```

```
scores = cross_val_score(clf, dk_train_att, dk_train_label,
rf_params[i,0] = max_features
rf_params[i,1] = n_estimators
rf_params[i,2] = scores.mean()
rf_params[i,3] = scores.std() * 2
i += 1
print("Max features: %d, num estimators: %d, accuracy: %.2f"
% (max_features, n_estimators, scores.mean(), scores.std() * 2))

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
fig = plt.figure()
fig.clf()
ax = fig.gca(projection='3d')
x = rf_params[:,0]
y = rf_params[:,1]
z = rf_params[:,2]
ax.scatter(x, y, z)
ax.set_zlim(0.6, 1)
ax.set_xlabel('Max features')
ax.set_ylabel('Num estimators')
ax.set_zlabel('Avg accuracy')
plt.show()
```

3. solusinya Didalam library matplotlib tidak ada attribute figures tetapi figure makanya pada coding diatas error karena menggunakan figures yang harusnya menggunakan figure.

## BAB 5

---

# CHAPTER 5

---

### 5.1 1174042 Faisal Najib Abdullah

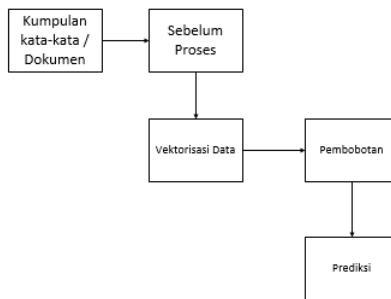
#### 5.1.1 Teori

1. Jelaskan Kenapa Kata-Kata harus dilakukan vektorisasi lengkapi dengan ilustrasi gambar.

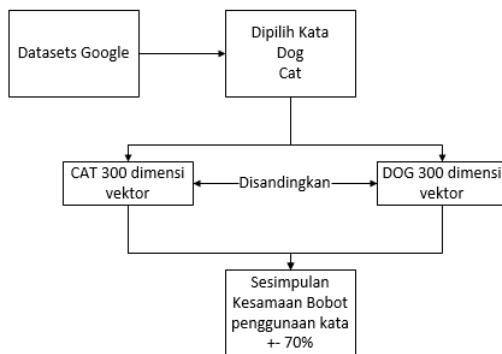
Kata kata harus dilakukan vektorisasi dikarenakan atau bertujuan utk mengukur nilai kemunculan suatau kata yang serupa dari sebuah kalimat sehingga kata-kata tersebut dapat di prediksi kemunculannya. atau juga di buatnya vektorisasi data digunakan untuk memprediksi bobot dari suatu kata misalkan ayam dan kucing sama-sama hewan maka akan dibuat prediksi apakah kata tersebut akan muncul pada kalimat yang kira-kira memiliki bobot yang sama.

2. Jelaskan Mengapa dimensi dari vektor dataset google bisa mencapai 300 lengkapi dengan ilustrasi gambar.

Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap kata,

**Gambar 5.1** Teori 1

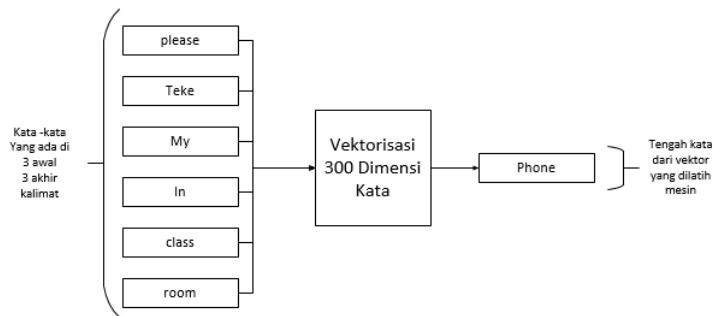
misalkan terdapat kata dog dan cat pada dataset google tersebut setiap kata tersebut di buat dimensi vektor 300 untuk kata dog dan 300 dimensi vektor juga untuk kata cat kemudian kata tersebut di bandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70 persen kesamaan bobot dikarenakan kata dog dan cat sama sama di gunakan untuk hewan priharaan.

**Gambar 5.2** Teori 2

3. Jelaskan Konsep vektorisasi untuk kata . dilengkapi dengan ilustrasi atau gambar.

Vektorisasi untuk kata untuk mengetahui kata tengah dari suatu kalimat atau kata utama atau objek utama pada suatu kalimat contoh ( Jangan lupa subscribe channel saya ya sekian treimakasih ) kata tengah tersebut merupakan channel yang memiliki bobot sebagai kata tengah dari suatu kalimat atau bobot sebagai objek dari suatu kalimat. hal ini sangat berkaitan dengan dimensi vektor pada dataset google yang 300 tadi

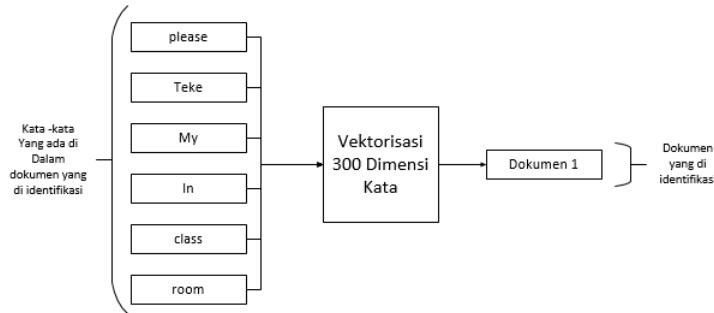
karena untuk mendapatkan nilai atau bobot dari kata tengah tersebut di dapatkan dari proses dimensiasi dari kata tersebut.



**Gambar 5.3** Teori 3

4. Jelaskan Konsep vektorisasi untuk dokumen. dilengkapi dengan ilustrasi atau gambar.

Vektorisasi untuk dokumen hampir sama seperti vektorisasi untuk kata hanya saja pemilihan kata utama atau kata tengah terdapat pada satu dokumen jadi mesin akan membuat dimensi vektor 300 untuk dokumen dan nanti kata tengahnya akan di sandingkan pada dokumen yang terdapat pada dokumen tersebut.

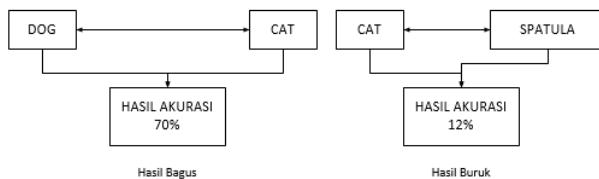


**Gambar 5.4** Teori 4

5. Jelaskan apa mean dan standar deviasi, lengkapi dengan ilustrasi atau gambar.

mean merupakan petunjuk terhadap kata-kata yang diolah jika kata-kata itu akurasinya tinggi berarti kata tersebut sering muncul begitu juga sebaliknya, sedangkan standar deviation merupakan standar untuk menimbang kesalahan. sehingga kesalahan tersebut dianggap wajar misalkan

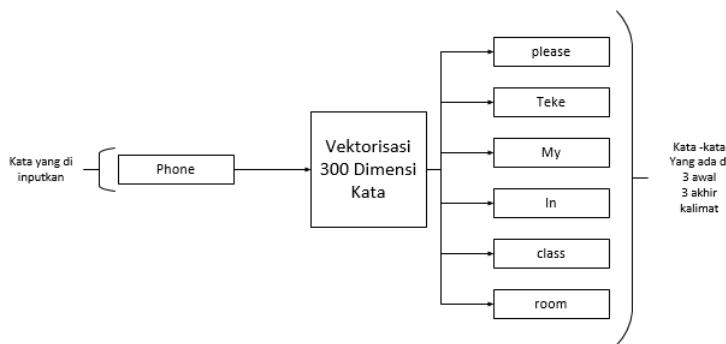
kita memperkirakan kedalaman dari dataset merupakan 2 atau 3 tapi pada kenyataanya merupakan 5 itu merupakan kesalahan tapi masih bisa dianggap wajar karna masih mendekati perkiraan awal.



**Gambar 5.5** Teori 5

6. Jelaskan Apa itu Skip-Gram sertakan contoh ilustrasi.

Skip-Gram adalah kebalikan dari konsep vektorisasi untuk kata dimana kata tengah menjadi acuan terhadap kata-kata pelengkap dalam suatu kalimat.



**Gambar 5.6** Teori 6

### 5.1.2 Praktikum

1. mencoba dataset google dan penjelasan vektor dari kata love, faith, fall, sick, clear, shine, bag, car, wash, motor, dan cycle.
  - berikut merupakan code import gensim digunakan untuk membuat data model atau rangcangan data yang akan dibuat. selanjutnya dibuat variabel gmodel yang berisi data vektor negatif. selanjutnya data tersebut di load agar data tersebut dapat ditampilkan dan diolah.

```
In [2]: import gensim
gmodel = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary = True)
```

**Gambar 5.7** Praktek 1

- berikut merupakan hasil lpengolahan kata love pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [3]: gmodel['love']
```

```
Out[3]: array([0.10302734, -0.15234375, 0.02587891, 0.16503906, -0.16503906,
 0.06689453, 0.29296875, -0.26367188, -0.140625 , 0.20117188,
 -0.02624512, -0.08203125, -0.02770996, -0.04394531, -0.23535156,
 0.16992188, 0.12898625, 0.15722656, 0.00756836, -0.06982422,
 -0.03857422, -0.07958984, 0.22949219, -0.14355469, 0.16796875,
 -0.03515625, 0.05517578, 0.10693359, 0.11181641, -0.16308594,
 -0.11181641, 0.13964844, 0.01556396, 0.12792969, 0.15429688,
 0.07714844, 0.26171875, 0.08642578, -0.02514648, 0.33398438,
 0.18652344, -0.20996094, 0.07080078, 0.02600098, -0.10644531,
 -0.10253906, 0.12304688, 0.04711914, 0.02209473, 0.05834961,
 -0.10986328, 0.14941406, -0.10693359, 0.01556396, 0.08984375,
 0.11230469, -0.04378117, -0.11376953, -0.0037384 , -0.01818848,
 0.24316406, 0.08447266, -0.07080078, 0.18066406, 0.03515625,
 -0.09667969, -0.21972656, -0.00328064, -0.03198242, 0.18457031,
 0.28515625, -0.0859375 , -0.11181641, 0.0213623 , -0.30664062,
 -0.09228516, -0.18945312, 0.01513672, 0.18554688, 0.34375 ,
 -0.31054688, 0.22558594, 0.08740234, -0.2265625 , -0.29492188,
 0.08251953, -0.38476562, 0.25390625, 0.26953125, 0.06298828,
 -0.00958252, 0.23632812, -0.17871094, -0.12451172, -0.17285156,
 -0.11767578, 0.19726562, -0.03466797, -0.104000391, -0.1640625 ,
 -0.19726562, 0.19824219, 0.09521484, 0.00561523, 0.12597656,
 0.00073624, -0.0402832 , -0.030863965, 0.01623535, -0.1640625 ,
 -0.22167969, 0.171875 , 0.12011719, -0.01965332, 0.4453125 ,
 0.06494141, 0.05932617, -0.1640625 , -0.01367188, 0.18945312,
 0.05566406, -0.05004883, -0.01422119, 0.15917969, 0.07421875,
 -0.31640625, -0.0534668 , -0.02355957, -0.16992188, 0.0625 ,
 -0.140625 , -0.13183594, -0.12792969, 0.12060547, 0.05883789,
```

**Gambar 5.8** Praktek 1

- berikut merupakan hasil lpengolahan kata faith pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata fall pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata sick pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata clear pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [4]: gmodel['faith']
```

```
Out[4]: array([0.26367188, -0.04150391, 0.1953125 , 0.13476562, -0.14648438,
 0.11962891, 0.04345703, 0.10351562, 0.12207031, 0.13476562,
 0.06640625, 0.18945312, -0.16601562, 0.21679688, -0.27148438,
 0.3203125 , 0.10449219, 0.36132812, -0.1953125 , -0.18164062,
 0.15332031, -0.10839844, 0.10253906, -0.01367188, 0.23144531,
 -0.05957031, -0.22949219, -0.00604248, 0.26171875, 0.10302734,
 -0.1328125 , 0.21484375, 0.01135254, 0.02111816, 0.18554688,
 0.04125977, 0.12011719, 0.17480469, -0.22167969, -0.13476562,
 0.3125 , 0.06640625, -0.17675781, -0.01708984, -0.1640625 ,
 -0.02819824, 0.01257324, -0.09521484, -0.18066406, -0.140625 ,
 -0.02258301, 0.16308594, -0.13183594, -0.08007812, 0.13085938,
 0.27539062, -0.20605469, 0.10351562, -0.20214844, -0.1875 ,
 0.16992188, 0.13574219, 0.13769531, 0.16308594, -0.03881836,
 -0.11132812, 0.05688477, 0.12255859, 0.09814453, -0.04956055,
 -0.02331543, -0.04248047, -0.08203125, 0.16015625, 0.04150391,
 -0.16601562, -0.13671875, 0.09619141, 0.32617188, 0.08251953,
 -0.20800781, 0.04199219, 0.05834961, -0.27734375, 0.09130859,
 -0.17382812, -0.22460938, 0.03466797, 0.19824219, -0.08837891,
 0.18359375, 0.07324219, 0.1171875 , -0.33984375, 0.16796875,
 -0.13574219, -0.30078125, -0.00469971, 0.06005859, -0.29296875,
 0.15234375, 0.02966309, 0.33203125, 0.28320312, 0.09375 ,
 -0.20605469, -0.09082031, 0.0534668 , 0.05834961, -0.03222656,
 -0.29296875, 0.25585938, 0.00430298, 0.140625 , 0.05810547,
 0.21582031, 0.0291748 , 0.02929688, 0.20019531, 0.34960938,
 0.10449219, -0.01940918, 0.04077148, 0.32226562, -0.1953125 ,
 -0.05688477, 0.10498847, -0.04785156, -0.15136719, -0.07714844,
 -0.45898438, -0.29492188, -0.328125 , -0.20996094, 0.38671875,
 0.0039978 . 0.07373047. -0.01220703. 0.22460938. 0.14550781.
```

**Gambar 5.9** Praktek 1

- berikut merupakan hasil lpengolahan kata shine pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata bag pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata car pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata wash pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata motor pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata cycle pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil dari similaritas kata kata yang di olah menjadi matrix tadi adapun persentase untuk perbandingan setiap katanya

```
In [5]: gmodel['fall']
```

```
Out[5]: array([-0.04272461, 0.10742188, -0.09277344, 0.16894531, -0.1328125 ,
-0.10693359, 0.04321289, 0.01904297, 0.14648438, 0.15039062,
-0.08691406, 0.04492188, 0.0145874, 0.08691406, -0.19824219,
-0.11035156, 0.01092529, -0.08300781, -0.0189209, -0.1953125 ,
-0.1015625 , 0.13671875, 0.09228516, -0.12109375, 0.12695312,
0.03417969, 0.2109375, 0.01977539, 0.125 , 0.01544189,
-0.26953125, -0.0098877 , -0.07763672, -0.15527344, -0.03393555,
0.04199219, -0.29828212, -0.18554688, 0.08496094, -0.02087402,
0.13574219, -0.22558594, 0.33789062, -0.03564453, -0.10839844,
-0.19335938, 0.0546875, -0.04956055, 0.3671875 , -0.03295898,
0.10205078, -0.15136719, -0.00445557, 0.04003906, 0.27539062,
-0.06935394, 0.05834961, 0.01422119, -0.01397705, -0.05395508,
-0.0255127 , 0.06298828, 0.07080078, -0.07617188, 0.06542969,
-0.01672363, -0.047111914, 0.19628906, -0.08984375, 0.078125 ,
0.2189375 , 0.0612793 , 0.08789062, 0.19628906, 0.11376953,
0.06542969, 0.03125 , 0.12988281, 0.02270508, 0.14550781,
-0.06225586, -0.37695312, -0.05737305, -0.06396484, 0.08984375,
0.00448608, -0.14160156, -0.04541016, -0.0703125 , 0.06005859,
0.26757812, 0.02001953, -0.12695312, -0.04882812, 0.18945312,
-0.03466797, 0.04638672, 0.1484375 , 0.01708984, -0.08789062,
-0.14941406, -0.02331543, -0.03955078, -0.10400391, -0.14160156,
-0.18261719, -0.03076172, 0.04589844, -0.2890625 , -0.093540039,
0.12890625, -0.10595783, 0.17578125, 0.06689453, 0.34960938,
0.04296875, 0.09863281, -0.08956641, -0.06298828, 0.12255859,
0.0234375 , -0.06494141, 0.09667969, -0.04589844, 0.04956055,
0.08007812, -0.00482178, -0.1646025 , -0.03271484, 0.0703125 ,
-0.07958984, -0.12890625, -0.01879883, -0.17773438, 0.01293945,
-0.20019531, 0.08886719, -0.18847656, -0.23828125, 0.02758789,
-0.14453125, -0.10058594, -0.0859375 , 0.10205078, -0.00396729,
```

**Gambar 5.10** Praktek 1

yaitu 9 persen untuk kata wash dan clear 7 persen untuk kata bag dan love 48 persen untuk kata motor dan car 12 persen untuk kata sick dan faith dan terakhir yaitu 6 persen untuk kata cycle dan shine.

2. pada code berikut merupakan hasil dari running code untuk ekstrak word dimana pada baris ke tiga dimasukan perintah untuk menghapus tag html yang terdapat dalam file tersebut selanjutnya pada baris ke 4 yaitu perintah untuk menghilangkan tanda kutip satu selanjutnya pada baris ke 5 yaitu perintah untuk menghapus tanda baca pada file tersebut dan yang terakhir yaitu perintah untuk menghapus double sepsi atau sepsi berurutan. setelah itu dibuat class bari dari random yang bertujuan untuk mengkocok data yang ada pada file tersebut kemudian class permute sentence tersebut akan digunakan untuk mengolah data selanjutnya. untuk lebih jelasnya dapat dilihat pada gambar ??.
3. gensim merupakan library untuk memodelkan topik unsupervised atau memodelkan bahasa dengan model unsupervised. Saat ini taget dokumen digunakan untuk TaggetDokumen berarti memasukan dokumen untuk diolah oleh mesin. Kemudian Doc2Vect digunakan untuk membandingkan dokumen apakah isi dari dokumen itu bobotnya sama dengan dokumen yang di sandingkannya. menunjukkan di baris ke satu dilakukan dari librari gensim dokumen mengimport method taggedDocument lalu

```
In [6]: gmodel['sick']
```

```
Out[6]: array([-1.82617188e-01, 1.49414062e-01, -4.05273438e-02, 1.64062500e-01,
-2.59765625e-01, 3.22265625e-01, 1.73828125e-01, -1.47460938e-01,
1.01874219e-01, 5.46875000e-02, 1.66992188e-01, -1.68945312e-01,
2.24304199e-03, 9.66796875e-02, -1.66015625e-01, -1.12304688e-01,
1.66015625e-01, 1.79687500e-01, 5.92041016e-03, 2.45117188e-01,
8.74023438e-02, -2.56347656e-02, 3.41796875e-01, 4.98046875e-02,
1.78710938e-01, -9.91821289e-04, 8.88671875e-02, -1.95312500e-01,
1.81640625e-01, -2.65625000e-01, -1.45597812e-01, 1.00585938e-01,
9.42382812e-02, -3.12500000e-02, 1.98974609e-02, -6.39648438e-02,
1.18652344e-01, 1.23046875e-01, -6.03027344e-02, 4.68750000e-01,
9.13085938e-02, -3.12500000e-01, 1.84570312e-01, -1.51367188e-01,
5.78613281e-02, -1.04980469e-01, -1.68945312e-01, -8.00781250e-02,
-2.01171875e-01, 1.06201172e-02, -1.29882812e-01, -1.25976562e-01,
-5.56640625e-02, 3.14453125e-01, 5.61523438e-02, -1.20117188e-01,
7.12890625e-02, 4.37011719e-02, 2.05078125e-01, 5.71289062e-02,
8.44726562e-02, 2.15820312e-01, -1.26953125e-01, 8.78906250e-02,
2.48846875e-01, -6.54296875e-02, -2.02636719e-02, 1.52343750e-01,
-3.57421875e-01, 3.02124023e-03, -2.08007812e-01, -5.05371094e-02,
2.81982422e-02, 1.73828125e-01, -2.08007812e-01, -5.93261719e-02,
-6.49414062e-02, 3.63769531e-02, 1.91406250e-01, 2.77343750e-01,
3.54003906e-02, 1.56250000e-01, -1.03857422e-02, 2.26562500e-01,
-4.66308594e-02, -5.17578125e-02, -1.63085938e-01, 4.17480469e-02,
2.01171875e-01, -2.01171875e-01, -1.50756836e-02, 2.61718750e-01,
-1.10839844e-01, -4.21875000e-01, 2.22167969e-02, 1.46484375e-01,
4.19921875e-01, -6.88476562e-02, 9.42382812e-02, -1.96289062e-01,
-9.42382812e-02, -3.12500000e-02, 6.34765625e-02, 2.47802734e-02,
-1.61132812e-01, -1.53320312e-01, 1.31835938e-01, -1.81640625e-01,
```

**Gambar 5.11** Praktek 1

pada baris kedua dari librari gensim model melakukan import metod Doc2Vec.

4. cara memasukan data traning file pertama tentukan terlebih dahulu tempat file dokumen tersebut disimpan kemudian import librari os setelah itu buat variabel unsup\_sentences yang berisikan array kosong, lalu tentukan file yang akan dimasukan setelah itu lakukan os.listdir pada data yang akan dimasukan kemudian semua data tersebut di inisialisasi menjadi f kemudian nama f tersebut dimasukan ke variabel unsup. pada codingan tersebut merupakan praktikum untuk memasukan data doc2vec
5. kenapa harus dilakukan pengocokan data atau randomisasi ? hal ini harus dilakukan supaya data lebis gambang untuk di olah dan meningkatkan tingkat akurasi dari proses pengolahan data Doc2Vec. kemudian harus dilakukan pembersihan data agar memori pc atau laptop yang di gunakan untuk mengolah data menjadi ringan dan menambah peforma dari mesin itu sendiri untuk codingan pertama lakukan terlebih dahulu randomisasi. selanjutnya membuat variabel baru dengan nama mute yang di isi data class random dan data unsup\_sentence. kemudian setelah pengolahan data dilakukan pembersihan dengan melakukan code.
6. kenapa model harus di save ? suapaya dalam pengolahan data tidak perlu menjalankan kembali data vektorisasi serta untuk meringankan beban

```
In [7]: gmodel['clear']
```

```
Out[7]: array([-2.44140625e-04, -1.02050781e-01, -1.49414062e-01, -4.24804688e-02,
-1.67968750e-01, -1.46484375e-01, 1.76757812e-01, 1.46484375e-01,
2.26562500e-01, 9.76562500e-02, -2.67578125e-01, -1.29882812e-01,
1.24511719e-01, 2.23632812e-01, -2.13867188e-01, 3.10058594e-02,
2.00195312e-01, -4.76074219e-02, -6.83593750e-02, -1.21093750e-01,
3.22265625e-02, 3.14453125e-01, -1.11816406e-01, 8.00781250e-02,
-2.75878906e-02, -6.04248047e-03, -7.37304688e-02, -1.72851562e-01,
9.66796875e-02, -4.91333008e-03, -1.78710938e-01, -1.40380859e-03,
7.91015625e-02, 1.07910156e-01, -1.10351562e-01, -8.34960938e-02,
1.98974609e-02, -3.14331055e-03, 1.30615234e-02, 3.34472656e-02,
2.55859375e-01, -7.12890625e-02, 2.83203125e-01, -2.75390625e-01,
-8.49689375e-02, -3.73535156e-02, -9.17968750e-02, -1.30859375e-01,
3.49121094e-02, 7.32421875e-02, 2.17773438e-01, 5.00488281e-02,
7.47070312e-02, -1.25000000e-01, -5.73730469e-02, -2.74658203e-02,
-1.37329102e-02, -2.13867188e-01, -1.12792969e-01, -4.98046875e-02,
-1.92382812e-01, 1.32812500e-01, -5.00488281e-02, -1.66015625e-01,
-1.57470703e-02, -1.37695312e-01, 3.88183594e-02, 1.57226562e-01,
-1.52343750e-01, -1.64794922e-02, -2.27539062e-01, 3.34472656e-02,
1.55273438e-01, 1.65039062e-01, -1.66992188e-01, 3.14941406e-02,
2.85156250e-01, 2.69531250e-01, 3.32031250e-02, 2.41210938e-01,
1.39160156e-02, 5.12695312e-02, 9.76562500e-02, 3.14941406e-02,
2.51464844e-02, -2.85156250e-01, -1.24023438e-01, -6.88476562e-02,
-5.29785156e-02, 2.06054688e-01, -2.07031250e-01, -1.60156250e-01,
-2.61230469e-02, -3.01513672e-02, 8.66699219e-03, -1.30859375e-01,
3.88183594e-02, 8.60595703e-03, 5.31005859e-03, -6.05468750e-02,
1.03759766e-02, 1.33789062e-01, -1.89971924e-03, -4.27246094e-02,
-1.14746094e-01, -1.47705078e-02, 9.71679688e-02, -1.66992188e-01,
5.63964844e-02, -2.76184082e-03, -1.05468750e-01, -1.03027344e-01,
```

**Gambar 5.12** Praktek 1

ram. kemudian temporary harus dihapus guna meningkatkan peforma komputer.

7. inver\_code digunakan untuk membandingkan data doc2vec yang telah diolah dengan kata yang baru atau data yang ada dalam perintah vector itu sendiri contoh membandingkan kata (i will go home) untuk lebih jelasnya dapat di lihat pada gambar ???. kemudian untuk hasil running code tersebut dapat di lihat pada gambar ?? pada hasil gambar tersebut terdapat hasil vektor yang rata rata berada pada kisaran 0,2 an yang berarti kata yang dimasukan pada inter\_vec datanya ada pada doc2vec atau ada data yang bobotnya menyamai kata-kata di dalam dokumen tersebut.
8. consine\_simirarity setelah melakukan pengolahan data doc2vec dilakukan consine simirarity yang bertujuan untuk membandingkan data berisikan bahasa inggris dengan data yang telah di olah tadi apakah hasilnya mirip atau tidak untuk caranya yaitu dengan cara mencobacodingan yang terdapat pada gambar ?? berikut maka akan muncul hasilnya berapa persen dengan tulisan 0,4 sekian yang berarti tingkat kemiripan dokumen yang di uji tadi untuk hasilnya dapat dilihat pada gambar ?? berikut.
9. untuk melakukan cross validation pertama masukan terlebih dahulu metode KNeighborsClasifier dan RandomForestClasifier dari library sklearn ke-

```
In [8]: gmodel['shine']
```

```
Out[8]: array([-0.12402344, 0.25976562, -0.15917969, -0.27734375, 0.30273438,
 0.09960938, 0.39257812, -0.22949219, -0.18359375, 0.3671875 ,
 -0.10302734, 0.13671875, 0.25390625, 0.07128906, 0.02539062,
 0.21777344, 0.24023438, 0.5234375 , 0.12304688, -0.19335938,
 -0.05883789, 0.0612793 , -0.01949918, 0.07617188, 0.05102539,
 0.20019531, 0.38085938, 0.00162506, -0.05029297, 0.14648438,
 -0.34765625, 0.02563477, -0.23925781, -0.04516682, -0.00479126,
 -0.24121094, -0.18945312, -0.15234375, -0.05493164, 0.01434326,
 0.390625 , -0.2109375 , 0.1484375 , -0.13183594, 0.24511719,
 -0.24023438, -0.36132812, -0.12792969, 0.10595703, 0.09912109,
 -0.0246582 , 0.32226562, 0.11376953, 0.18164062, 0.04931641,
 -0.10253906, -0.00283813, 0.29882812, -0.171875 , -0.18945312,
 -0.01367188, -0.20898438, -0.07861328, -0.0859375 , 0.05395508,
 -0.14257812, -0.149625 , 0.03027344, -0.14453125, 0.359375 ,
 0.16113281, 0.22265625, 0.265625 , -0.06347656, -0.02807617,
 0.04760742, 0.08837891, -0.04272461, 0.05988203, 0.07128906,
 0.01519775, -0.11621994, 0.07128906, 0.01403889, -0.10644531,
 0.08886719, 0.11523438, 0.09667969, -0.11083984, 0.16015625,
 0.3359375 , -0.1875 , 0.14550781, 0.00463867, 0.07617188,
 -0.09521484, 0.08447266, 0.20117188, 0.11230469, -0.33984375,
 -0.25390625, 0.05200195, 0.27539062, -0.08398438, -0.31054688,
 -0.22949219, 0.14941406, -0.1953125 , 0.08496094, -0.00753784,
 0.078125 , 0.05908203, 0.02355957, 0.06347656, 0.32617188,
 -0.08740234, 0.10058594, -0.11474609, -0.18164062, 0.13378906,
 0.11230469, -0.00080109, 0.08691406, 0.03888594, 0.0300293 ,
 -0.03417969, -0.00830078, 0.14160156, -0.09619141, -0.11328125,
 0.00823975, -0.15234375, 0.19042969, 0.04980469, 0.25 ,
 -0.00171661. 0.04589844. -0.05102539. -0.04052734. -0.03491211.
```

**Gambar 5.13** Praktek 1

mudian dilakukan cross validation setelah itu buat variabel clf dengan isi KNeighborsClasifier dan variabel clfrf dengan isi RandomForestClasifier kemudian di buat skor menggunakan cross validation dengan menggunakan variabel clf dan data sentvecs dan sentiments kemudian dengan numpy dibuat mean dari scores begitu pula untuk variabel clfrf selanjutnya melakukan import metode make\_pipeline yang dilakukan untuk membuat skor dari vektorisasi tfidf dan rf. untuk lebih jelasnya dapat di lihat pada gambar ?? maka akan muncul hasil rata-rata 0,76 sekian atau 76 persen untuk clf yang dapat dilihat pada gambar ?? dan untuk hasil clfrf menghasilkan hasil rata-rata di 71 persen yang dapat dilihat pada gambar ?? dan untuk hasil rata-rata keseluruhan cros validation sebesar 0,74 atau 74 persen yang dapat dilihat pada gambar ??.

```
1 from sklearn.neighbors import KNeighborsClassifier
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn.model_selection import cross_val_score
4 import numpy as np
5
6 clf = KNeighborsClassifier(n_neighbors=9)
7 clfrf = RandomForestClassifier()
8
9 scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
10 print((np.mean(scores), np.std(scores)))
11
12 scores = cross_val_score(clfrf, sentvecs, sentiments, cv=5)
```

```
In [9]: gmodel['bag']

Out[9]: array([-0.03515625, 0.15234375, -0.12402344, 0.13378906, -0.11328125,
 -0.0133667 , -0.16113281, 0.14648438, -0.06835938, 0.140625 ,
 -0.06005859, -0.3046875 , 0.20996694, -0.04345703, -0.2109375 ,
 -0.05957031, -0.05053711, 0.10253906, 0.19042969, -0.09423828,
 0.18847656, -0.07958984, -0.11035156, -0.07910156, 0.06347656,
 -0.1527344 , -0.18945312, 0.11132812, 0.27539062, -0.06787109,
 0.01806641, 0.06689453, 0.2578125 , 0.0324707 , -0.24609375,
 -0.05541992, 0.01813184, 0.24121094, -0.21875 , 0.07568359,
 -0.09814453, -0.16113281, 0.16503906, -0.09521484, -0.16601562,
 -0.41796875, 0.0300293 , 0.19433594, 0.2890625 , 0.12695312,
 -0.19824219, -0.05517578, 0.04296875, -0.10107422, 0.07324219,
 -0.13378906, 0.265625 , -0.00466919, 0.19628906, -0.10839844,
 0.14941406, 0.1484375 , 0.09619141, 0.21777344, -0.08544922,
 -0.02819824, 0.02539662, -0.03759766, 0.23242188, 0.19628906,
 0.27539062, 0.09130859, 0.23738469, 0.09033203, -0.28515625,
 0.05932617, 0.06591797, -0.01794434, -0.00055313, -0.1796875 ,
 0.05615234, -0.12207831, -0.09863281, -0.05786133, -0.09375 ,
 -0.30273438, -0.06396484, -0.00744629, -0.17871094, 0.08544922,
 -0.20401056, 0.33789062, 0.00228882, -0.39453125, -0.14453125,
 -0.328125 , -0.12695312, -0.08544922, 0.15234375, 0.03662109,
 -0.1484375 , 0.05566406, 0.02844238, 0.07519531, -0.21484375,
 -0.15722656, 0.3359375 , -0.04736328, -0.00405884, -0.19726562,
 0.27929688, 0.05566406, -0.10058594, -0.00811768, -0.20703125,
 0.03295898, -0.14550781, -0.15917969, 0.16503906, 0.234375 ,
 0.03588867, 0.04296875, -0.25 , 0.1171875 , -0.07714844,
 0.00521851, 0.125 , 0.08886719, 0.15527344, -0.02185059,
 -0.15234375, -0.12890625, -0.34765625, -0.13769531, -0.18164062,
```

Gambar 5.14 Praktek 1

```
13 print((np.mean(scores), np.std(scores)))
14
15 # bag-of-words comparison
16 from sklearn.pipeline import make_pipeline
17 from sklearn.feature_extraction.text import CountVectorizer,
 TfIdfTransformer
18 pipeline = make_pipeline(CountVectorizer(), TfIdfTransformer()
 (), RandomForestClassifier())
19 scores = cross_val_score(pipeline, sentences, sentiments, cv
 =5)
20 print((np.mean(scores), np.std(scores)))
```

### 5.1.3 Penanganan Error

## 5.2 1174040 - Hagan Rowlenstino A. S

### 5.2.1 Teori

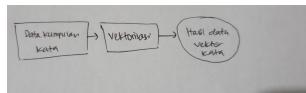
#### 1. Why words need to be Vectorizer

karena kata-kata yang digunakan untuk memproses data agar dapat menjadi bagian dari kumpulan data atau atribut yang dapat dibaca oleh sistem machine learning karena sistem tersebut tidak dapat memproses data text secara langsung dan harus di convert terlebih dahulu kedalam bilangan.untuk ilustrasinya dapat dilihat pada gambar

```
In [10]: gmodel['car']
```

```
Out[10]: array([-0.13085938, 0.00842285, 0.03344727, -0.05883789, 0.04003966,
 -0.14257812, 0.04931641, -0.16894531, 0.20898438, 0.11962891,
 0.18066406, -0.25 , -0.10400391, -0.10742188, -0.01879883,
 0.05200195, -0.00216675, 0.06445312, 0.14453125, -0.04541016,
 0.16113281, -0.01611328, -0.03088379, 0.08447266, 0.16210938,
 0.04467773, -0.15527344, 0.25390625, 0.33984375, 0.00756836,
 -0.25585938, -0.01733398, -0.03295898, 0.16308594, -0.12597656,
 -0.09912109, 0.16503906, 0.06884766, -0.18945312, 0.02832031,
 -0.0534668 , -0.03063965, 0.11083984, 0.24121094, -0.234375 ,
 0.12353516, -0.002944495, 0.1484375 , 0.33203125, 0.05249023,
 -0.20019531, 0.37695312, 0.12255859, 0.11425781, -0.17675781,
 0.10009766, 0.0030365 , 0.26757812, 0.28117188, 0.03710938,
 0.11083984, -0.09814453, -0.3125 , 0.03515625, 0.02832031,
 0.26171875, -0.08642578, -0.02258301, -0.05834961, -0.00787354,
 0.11767578, -0.04296875, -0.17285156, 0.04394531, -0.23046875,
 0.1640625 , -0.11474609, -0.06030273, 0.01196289, -0.24707031,
 0.32617188, -0.04492188, -0.11425781, 0.22851562, -0.01647949,
 -0.15039062, -0.13183594, 0.12597656, -0.17480469, 0.02209473,
 -0.1815625 , -0.00817871, 0.10791816, -0.24609375, -0.109375 ,
 -0.09375 , -0.01623535, -0.20214844, 0.23144531, -0.05444336,
 -0.05541992, -0.20898438, 0.26757812, 0.27929688, 0.17089844,
 -0.17578125, -0.02770996, -0.20410156, 0.02392578, 0.03125 ,
 -0.25390625, -0.125 , -0.05493164, -0.17382812, 0.28515625,
 -0.23242188, 0.0234375 , -0.20117188, -0.13476562, 0.26367188,
 0.00769043, 0.20507812, -0.01708984, -0.12988281, 0.04711914,
 0.22070312, 0.02099609, -0.29101562, -0.02893066, 0.17285156,
 0.04272461, -0.19824219, -0.04003906, -0.16992188, 0.10058594,
 -0.09326172, 0.15820312, -0.16503906, -0.06054688, 0.19433594,
 -0.07080078, -0.06884766, -0.09619141, -0.07226562, 0.04882812,
```

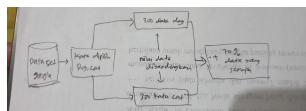
**Gambar 5.15** Praktek 1



**Gambar 5.28** Vektorisasi Kata

## 2. Why Dimension of Google dataset can reach 300

Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap data kata yang diproses. ilustrasi dapat dilihat pada gambar



**Gambar 5.29** Dataset Google

## 3. Concept of Vectorizer on words

pada vektorisasi dengan menggunakan Word2Vec memiliki kelebihan yang dapat dibedakan dengan penggunaan bag of words yang biasanya. pada bag of word pemrosesan data tidak dapat menganalisa

```
In [11]: gmodel['wash']
```

```
Out[11]: array([- 9.46004492e-03, 1.41601562e-01, - 5.46875000e-02, 1.34765625e-01,
 - 2.38281250e-01, 3.24218750e-01, - 8.44726562e-02, - 1.29882812e-01,
 1.07910156e-01, 2.53906250e-01, 1.13525391e-02, - 1.66992188e-01,
 - 2.79541016e-02, 2.08007812e-01, - 4.27246094e-02, 1.05468750e-01,
 - 7.42187500e-02, 3.04687500e-01, 2.11914062e-01, - 8.88671875e-02,
 2.67578125e-01, 2.12890625e-01, 1.74569547e-02, 2.02941895e-03,
 6.29882812e-02, 1.62109375e-01, 1.93359375e-01, 2.17285156e-02,
 - 2.67028809e-03, - 9.13085938e-02, - 2.38281250e-01, 2.23632812e-01,
 - 8.00781250e-02, - 3.80859375e-02, - 1.00097656e-01, - 1.39648438e-01,
 1.74804688e-01, 6.78710938e-02, 1.11328125e-01, 1.65039962e-01,
 - 1.05468750e-01, 2.30712891e-02, 2.00195312e-01, - 6.03027344e-02,
 - 3.43750000e-01, - 1.02050781e-01, - 3.80859375e-01, - 5.05371894e-02,
 5.07812500e-02, 1.45507812e-01, 2.81250000e-01, 7.03125000e-02,
 2.84423828e-02, - 2.29492188e-01, - 5.81054688e-02, 4.51660156e-02,
 - 3.56445312e-02, 1.77734375e-01, 1.22070312e-01, 3.71093750e-02,
 - 1.10839844e-01, 6.83593750e-02, - 2.52685547e-02, - 1.27929688e-01,
 4.21875000e-01, 5.32226562e-02, - 3.92578125e-01, 1.74804688e-01,
 1.77001953e-02, - 2.05078125e-02, 2.21679688e-01, 3.18359375e-01,
 1.088398438e-01, - 4.30297852e-03, - 2.45117188e-01, - 2.08984375e-01,
 3.58867191e-02, 8.30078125e-02, 1.68945312e-01, 2.79541016e-02,
 1.04980469e-01, - 3.47656250e-01, - 5.20019531e-02, 2.24609375e-01,
 1.69677734e-02, 1.69921875e-01, - 1.46484375e-01, 2.65625000e-01,
 2.17285156e-02, 1.12304688e-02, - 1.14257812e-01, 7.22656250e-02,
 4.32128906e-02, 1.11694336e-02, 5.07354736e-04, - 7.91815625e-02,
 - 5.98144531e-02, - 5.44433594e-02, 3.73046875e-01, 5.62500000e-01,
 - 2.26562500e-01, - 5.39550781e-02, 1.13769531e-01, - 5.83496094e-02,
 - 1.53320312e-01, - 4.37500000e-01, 2.59765625e-01, - 1.49414062e-01,
 5.66406250e-02, 2.13867188e-01, - 2.86865234e-02, - 1.70898438e-01,
```

**Gambar 5.16** Praktek 1

data yang memiliki makna sama namun penulisannya berbeda, namun pada penggunaan Word2Vec proses tersebut dapat berjalan dengan lebih mudah contohnya adalah penulisan kata please dengan plz. untuk ilustrasi datanya bisa dilihat dalam gambar



**Gambar 5.30** Concept of Vectorizer on words

#### 4. Concept of Vectorizer on documents

vektorisasi pada Doct2Vec dimana data yang terdapat pada file document tersebut diolah dengan melakukan pemrosesan yang mengutamakan nilai data filenamenya atau atribut utama dimana nilai data inputnya tidak terlalu diproses. ilustrasinya dapat dilihat pada gambar



**Gambar 5.31** Concept of Vectorizer on Documents

```
In [12]: gmodel['motor']
```

```
Out[12]: array([5.73730469e-02, 1.50390625e-01, -4.61425781e-02, -1.32812500e-01,
 -2.59765625e-01, -1.77734375e-01, 3.68652344e-02, -4.37500000e-01,
 2.34375000e-02, 2.57812500e-01, 1.74804688e-01, 2.44140625e-02,
 -2.51953125e-01, -5.76171875e-02, 8.15429688e-02, 1.86767578e-02,
 -3.83300781e-02, 1.58203125e-01, -5.85937500e-02, 1.12304688e-01,
 1.56250000e-01, -4.24804688e-02, -1.32812500e-01, 2.11914062e-01,
 1.23046875e-01, 1.69921875e-01, -1.55273438e-01, 4.58984375e-01,
 3.02734375e-01, 1.53320312e-01, -1.69921875e-01, -1.01074219e-01,
 -3.26538806e-03, 2.28515625e-01, 8.98437500e-02, -7.12890625e-02,
 1.54296875e-01, -8.88671875e-02, -2.36328125e-01, 5.61523438e-03,
 -4.46777344e-02, -3.06640625e-01, 7.42187500e-02, 5.58593750e-01,
 -1.30859375e-01, 1.00585938e-01, -3.34472656e-02, 2.10937500e-01,
 3.10058594e-02, -6.50024414e-03, 6.34765625e-02, 4.02832031e-02,
 -2.78320312e-02, 1.07421875e-02, 1.47468938e-01, 2.88761719e-02,
 -1.50390625e-01, -1.37695312e-01, 9.96093750e-02, 1.28906250e-01,
 -3.34472656e-02, -1.08032227e-02, -2.14843750e-01, -9.52148438e-02,
 -6.39648438e-02, 7.51953125e-02, -3.06640625e-01, 2.17773438e-01,
 -2.21679688e-01, 2.33398438e-01, 5.05371094e-02, -3.37890625e-01,
 1.53320312e-01, -7.12890625e-02, -3.68652344e-02, 7.66601562e-02,
 -8.00781250e-02, -1.14257812e-01, -9.71679688e-02, -2.61718750e-01,
 3.84765625e-01, -1.87500000e-01, -1.10351562e-01, 1.00585938e-01,
 1.08398438e-01, 9.57031250e-02, -8.20312500e-02, 1.54296875e-01,
 -2.40234375e-01, 8.34960938e-02, 4.19921875e-02, -1.91650391e-02,
 9.71679688e-02, 2.52685547e-02, -5.46875000e-02, -5.88378906e-02,
 8.20312500e-02, -3.32031250e-01, 3.27148438e-02, 5.71289062e-02,
 1.77734375e-01, -9.57031250e-02, 2.45117188e-01, 6.88476562e-02,
 2.63671875e-01, -8.15429688e-02, 1.25976562e-01, 1.20849609e-02,
 4.00390625e-01, 8.69140625e-02, -3.00781250e-01, -1.99218750e-01,
```

**Gambar 5.17** Praktek 1

## 5. What is mean and deviation standart

Mean adalah nilai rata-rata dari beberapa buah data. Nilai mean dapat ditentukan dengan membagi jumlah data dengan banyaknya data.

Standar deviasi adalah nilai statistik yang digunakan untuk menentukan bagaimana sebaran data dalam sampel, dan seberapa dekat titik data individu ke mean – atau rata-rata – nilai sampel.

untuk ilustrasi data mean dan deviation standart bisa dilihat pada gambar

| Contoh |       |
|--------|-------|
| nama   | nilai |
| budi   | 70    |
| ani    | 80    |
| ina    | 90    |

mean = 80  
 Standar deviasi = 15,275

**Gambar 5.32** Mean and Deviation Standard

```
In [13]: gmodel['cycle']
```

```
Out[13]: array([0.04541015, 0.21679688, -0.02709961, 0.12353516, -0.20783125,
 -0.13281015, 0.26367188, -0.12890625, -0.125 , 0.15332031,
 -0.18261719, -0.15820312, -0.06176758, 0.21972656, -0.15820312,
 0.02563477, -0.07568359, -0.0625 , 0.04614258, -0.31054688,
 -0.13378906, -0.11669922, -0.3359375 , 0.078125 , 0.08447266,
 0.07226562, -0.06445312, 0.05517578, 0.14941406, 0.13671875,
 0.19302734, 0.02172852, -0.10693359, 0.024980234, -0.10644531,
 -0.05541992, -0.29492188, -0.40039062, 0.06347656, -0.08447266,
 0.17871094, 0.01165771, -0.01696777, 0.13671875, -0.1640625 ,
 0.11425781, 0.20800781, -0.06079102, -0.07275391, 0.15039062,
 0.18066406, -0.28515625, -0.04052734, 0.01806641, 0.00331116,
 0.00872803, 0.03564453, -0.29882812, 0.09960938, -0.1484375 ,
 -0.06787109, 0.05957031, -0.05517578, -0.19628906, 0.2265625 ,
 0.03173828, -0.07080078, 0.1484375 , -0.20214844, -0.03393555,
 0.09863281, -0.02038574, -0.08789062, -0.07226562, -0.09423828,
 -0.17089844, 0.1484375 , 0.10546875, 0.06445312, 0.01031494,
 -0.02636719, -0.03686523, -0.125 , 0.06787109, 0.14257812,
 0.37109375, -0.15722656, 0.09326172, 0.34960938, -0.00091553,
 -0.03613281, 0.16894531, -0.02856445, 0.10791016, -0.32421875,
 -0.14355469, 0.03173828, -0.07421875, 0.34179688, 0.140625 ,
 0.00433335, -0.12890625, -0.34960938, -0.02929688, -0.19628906,
 -0.2578125 , -0.3671875 , 0.01483154, 0.20703125, 0.09667969,
 -0.10351562, -0.31054688, 0.02844238, 0.18400391, 0.17773438,
 0.06689453, -0.1796875 , 0.02783203, 0.15625 , 0.02026367,
 0.0324707 , -0.13476562, 0.15527344, 0.11132812, -0.01055908,
 0.07958984, 0.01989746, 0.25585938, 0.13378906, 0.02539062,
 0.10986328, -0.20605469, 0.07275391, -0.35546875, -0.02746582,
```

Gambar 5.18 Praktek 1

```
In [15]: gmodel.similarity('wash', 'clear')
```

```
Out[15]: 0.09019176
```

```
In [16]: gmodel.similarity('bag', 'love')
```

```
Out[16]: 0.07536096
```

```
In [17]: gmodel.similarity('motor', 'car')
```

```
Out[17]: 0.4810173
```

```
In [18]: gmodel.similarity('sick', 'faith')
```

```
Out[18]: 0.123073205
```

```
In [19]: gmodel.similarity('cycle', 'shine')
```

```
Out[19]: 0.061617922
```

Gambar 5.19 Praktek 1

## 6. What is skip-gram

Skip-gram merupakan teknik yang digunakan di area speech processing, dimana n-gram yang dibentuk kemudian ditambahkan juga dengan tindakan “skip” pada token-tokennya. contohnya terdapat pada gambar

```
In [20]: import re
def extract_words(sent):
 sent = sent.lower()
 sent = re.sub(r'<[^>]+>', ' ', sent) #hapus tag html
 sent = re.sub(r'(\w)\'(\w)', ' ', sent) #hapus petik satu
 sent = re.sub(r'\W', ' ', sent) #hapus tanda baca
 sent = re.sub(r'\s+', ' ', sent) #hapus spasi yang berurutan
 return sent.split()
```

```
In [21]: import random
class PermuteSentences(object):
 def __init__(self, sents):
 self.sents = sents

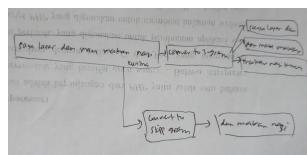
 def __iter__(self):
 shuffled = list(self.sents)
 random.shuffle(shuffled)
 for sent in shuffled:
 yield sent
```

Gambar 5.20 Praktek 2

```
In [22]: from gensim.models.doc2vec import TaggedDocument
from gensim.models import Doc2Vec
```

```
In []:
```

Gambar 5.21 Praktek 3



Gambar 5.33 Skip-Gram

## 5.2.2 Praktek

### 1. Try datasets GoogleNews-vectors

- berikut adalah hasil dari code yang digunakan untuk memanggil data library GENSIM dengan menggunakan perintah import, lalu dari library tersebut diambil data yang akan digunakan untuk memproses data dari GoogleNews-vector. ilustrasi dapat dilihat pada gambar

```
In [4]: import gensim
In [5]: gmodel = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)
```

Gambar 5.34 import gensim dan olah data GoogleNews-vector

```
In [53]: import os
unsup_sentences = []

In [54]: for dirname in ["train/pos", "train/neg", "train/unsup", "test/pos", "test/neg"]:
 for fname in sorted(os.listdir("aclimdb/" + dirname)):
 if fname[-4:] == '.txt':
 with open("aclimdb/" + dirname + "/" + fname, encoding='UTF-8') as f:
 sent = f.read()
 words = extract_words(sent)
 unsup_sentences.append(TaggedDocument(words, [dirname + "/" + fname]))

In [55]: for dirname in ["txt_sentoken/pos", "txt_sentoken/neg"]:
 for fname in sorted(os.listdir(dirname)):
 if fname[-4:] == '.txt':
 with open("aclimdb/" + dirname + "/" + fname, encoding='UTF-8') as f:
 for i, sent in enumerate(f):
 words = extract_words(sent)
 unsup_sentences.append(TaggedDocument(words, ["%s/%s-%d" % (dirname, fname, i)]))

In [56]: with open("stanforSentimentTreebank/original_rt_snippets.txt", encoding='UTF-8') as f:
 for i, sent in enumerate(f):
 words = extract_words(sent)
 unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))
```

Gambar 5.22 Praktek 4

```
In [77]: import re
def extract_words(sent):
 sent = sent.lower()
 sent = re.sub(r'<[^>]+>', ' ', sent) #hapus tag html
 sent = re.sub(r'(\w)\'(\w)', ' ', sent) #hapus petik satu
 sent = re.sub(r'\W', ' ', sent) #hapus tanda baca
 sent = re.sub(r'\s+', ' ', sent) #hapus spasi yang berurutan
 return sent.split()

import random
class PermuteSentences(object):
 def __init__(self, sents):
 self.sents = sents

 def __iter__(self):
 shuffled = list(self.sents)
 random.choice(shuffled)
 for sent in shuffled:
 yield sent

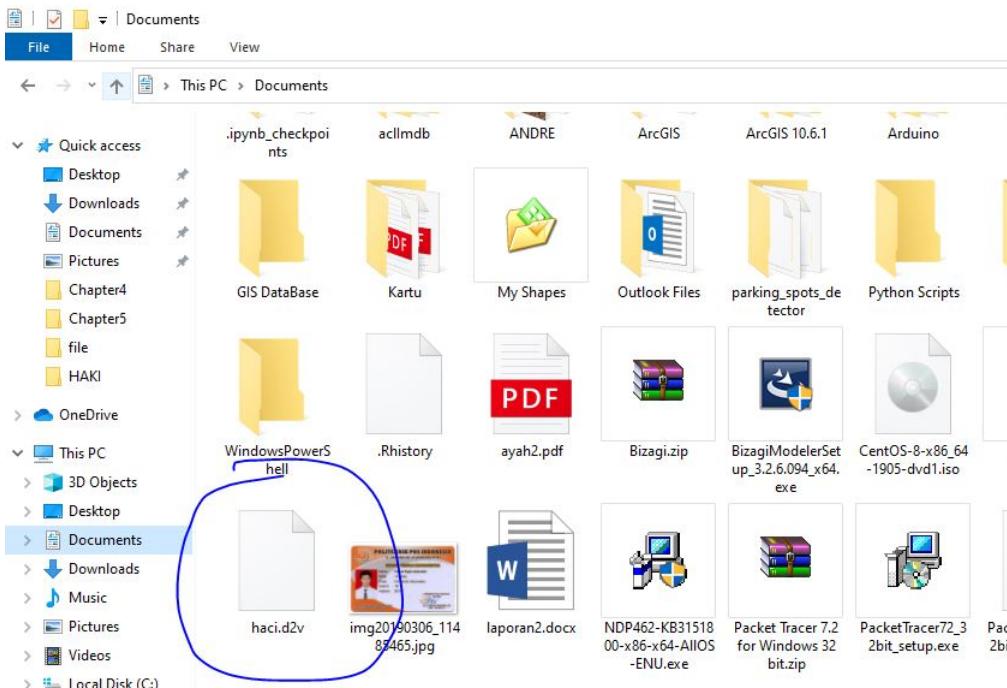
mute=PermuteSentences(unsup_sentences)
model = Doc2Vec(mute, dm=0, hs=1, size=50)
model.delete_temporary_training_data(keep_inference=True)
```

Gambar 5.23 Praktek 5

- lalu pada penggunaan code berikut ini akan mengolah data LOVE yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [78]: model.delete_temporary_training_data(keep_inference=True)
 model.save('haci.d2v')
```

**Gambar 5.24** Praktek 6



### Gambar 5.25 Praktek 6

```
In [8]: genmod['love']
Out[8]: array([0.10307324, -0.15234375, 0.02587891, -0.16503906, -0.16503906,
 0.06884543, 0.2996875, -0.26357188, -0.140625, 0.20171188,
 0.02624512, 0.0383125, -0.0593888, 0.04394031, -0.03535156,
 0.02500001, 0.02500001, 0.02500001, 0.02500001, 0.02500001,
 0.03857422, 0.07859864, 0.22649212, 0.14355469, 0.17679468,
 0.03515625, 0.05175578, 0.10639359, 0.11181641, 0.16505984,
 -0.11181641, 0.13964484, 0.01553693, 0.17292969, 0.15426688,
 0.02635755, 0.02635755, 0.02635755, 0.02635755, 0.02635755,
 0.16552344, 0.23946886, 0.04711914, 0.02929473, 0.05835684])
```

**Gambar 5.35** hasil olah data LOVE pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data FAITH yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [79]: model.infer_vector(extract_words("I will go home"))
```

```
Out[79]: array([-1.7533980e-01, 1.7654952e-01, 8.7283678e-02, 1.7998287e-02,
 6.2501184e-03, -3.2540500e-02, 1.6391091e-01, -1.3786003e-02,
 5.1714227e-02, 4.6473891e-02, 1.6999269e-02, 2.6260551e-02,
 -2.5963100e-02, 7.7627085e-02, -1.5409573e-02, 1.3232067e-01,
 -4.3878101e-02, 4.8159737e-02, -5.1153481e-02, -7.2681673e-02,
 -3.1889844e-02, -2.4982829e-02, 1.1039837e-01, 7.9181477e-02,
 6.1015221e-03, 6.1555382e-02, -2.2586747e-01, 1.3859421e-01,
 1.6294651e-02, -6.1351705e-02, 1.3722880e-01, -7.6827303e-02,
 -1.4081554e-01, 4.0382754e-02, -1.2432394e-01, -3.1883363e-02,
 6.4639568e-02, 7.1126200e-02, -4.4149000e-02, -5.9015196e-02,
 1.0176090e-04, 1.0674181e-02, 6.5928474e-02, 3.7513527e-03,
 5.5374358e-02, -1.3092439e-02, -3.5841893e-02, 4.9701292e-02,
 -1.1787581e-01, 9.5303327e-02], dtype=float32)
```

Gambar 5.26 Praktek 7

```
In [83]: from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(
 [model.infer_vector(extract_words("she going to school, after wash hand"))],
 [model.infer_vector(extract_words("Services sucks2."))])
```

```
Out[83]: array([[0.04744839]], dtype=float32)
```

```
In [84]: from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(
 [model.infer_vector(extract_words("Dia pergi ke sekolah tadi siang"))],
 [model.infer_vector(extract_words("Services sucks2."))])
```

```
Out[84]: array([[0.2095491]], dtype=float32)
```

Gambar 5.27 Praktek 8

```
In [9]: genmod(['faith'])
Out[9]:
array([-0.26367188, -0.04150891, 0.1953125, 0.13476562, -0.14648488,
 0.11962891, 0.04345793, 0.10551662, 0.12297931, 0.13476562,
 0.13293125, -0.198449219, 0.36132812, -0.1953125, -0.18154682,
 0.15332831, -0.18839844, 0.18153986, -0.01367188, 0.23144531,
 -0.1328125, 0.21484375, 0.01137524, 0.02111816, 0.18554688,
 0.04125977, 0.12011719, 0.17480669, -0.22167969, -0.13476562,
 0.0128125, 0.0128125, 0.0128125, -0.0128125, -0.0128125,
 -0.02138234, 0.01257324, -0.09721469, -0.18066469, -0.146825 ,
```

Gambar 5.36 hasil olah data FAITH pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data FALL yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [10]: genmod(['fall'])
Out[10]:
array([-0.04773441, 0.15742189, -0.05977344, -0.14059451, -0.13231125,
 0.16993359, 0.04121289, 0.01949237, 0.14648488, 0.15898602,
 -0.08691486, 0.04492188, 0.015874, 0.08691486, -0.19824219,
 -0.18154625, 0.13571075, 0.09228516, -0.12139975, 0.12695312,
 0.04137669, 0.21893775, 0.01977539, 0.125, 0.01544105,
 -0.0128125, 0.0128125, 0.0128125, -0.0128125, -0.0128125,
 0.0499219, -0.29882812, -0.18554688, 0.08696984, -0.02087462,
 0.13574219, -0.22558594, 0.33789862, -0.03564453, -0.18839844,
```

Gambar 5.37 hasil olah data FALL pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data SICK yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [11]: genmed ['sick']
Out[11]:
array([-1.8211758e-01, 1.4691480e-01, -4.9527455e-02, 1.46480250e-01,
 3.5935525e-01, 3.7274552e-01, -1.73958125e-01, -1.43646801e-01,
 1.0187421e-01, 5.4687500e-02, 1.66991250e-01, -1.68945531e-01,
 2.6661525e-01, 1.79687500e-01, 5.92081056e-03, 2.4511718e-01,
 9.7492353e-02, -2.56347056e-02, 3.41798751e-01, 4.0886875e-02,
 1.2661525e-01, 1.79687500e-01, 5.92081056e-03, 2.4511718e-01,
 1.81546025e-01, -2.65025000e-01, -1.45957812e-01, 1.09055935e-01,
 9.42382812e-02, -3.12500000e-02, 1.98974609e-02, -6.39648438e-02],
```

**Gambar 5.38** hasil olah data SICK pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data CLEAR yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [12]: genmed ['clear']
Out[12]:
array([-1.44140825e-01, -1.03959701e-01, -1.49414802e-01, -4.24884688e-01,
 2.16562759e-01, -9.76592350e-02, -2.67578125e-01, -1.29882812e-01,
 2.12562759e-01, -9.76592350e-02, -2.67578125e-01, -1.29882812e-01,
 2.08195131e-01, -6.76674219e-02, -6.83593750e-02, -1.21893750e-01,
 2.08195131e-01, -6.76674219e-02, -6.83593750e-02, -1.21893750e-01,
 -7.7587986e-02, -6.04248847e-03, -7.37394658e-02, -1.72851562e-01,
 9.66796475e-02, -4.9133088e-03, -1.78719938e-01, -1.40388539e-03,
 7.93815252e-02, 1.87918126e-02, -1.18515202e-02, -8.34960933e-02],
```

**Gambar 5.39** hasil olah data CLEAR pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data SHINE yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [13]: genmed ['shine']
Out[13]:
array([-0.12402344, 0.25976562, -0.15917969, -0.27734375, 0.30273438,
 0.09969938, 0.39257812, -0.22949219, -0.18355937, 0.3671875 ,
 0.212277344, 0.24923438, 0.5234375 , 0.12394648, -0.19359938,
 -0.05837879, 0.0512739 , -0.0540918, 0.07617188, 0.05125259,
 0.212212994, 0.25237581, 0.23925781, -0.04516602, -0.00479126,
 -0.241212994, -0.18945112, -0.15234375, -0.05493164, 0.01434326,
 0.298623 , -0.23939779, 0.14843779, -0.11328394, 0.24511779],
```

**Gambar 5.40** hasil olah data SHINE pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data BAG yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [14]: genmed ['bag']
Out[14]:
array([-0.05151525, 0.15234375, -0.12402344, 0.13378986, -0.11328125,
 -0.0133667 , -0.16113281, 0.14648438, -0.06835938, 0.140625 ,
 -0.05957031, 0.05953771, 0.10253996, 0.19842969, -0.09423829,
 -0.18081556, -0.07959064, 0.11891516, -0.07918156, 0.06835938,
 -0.18527259, 0.05953771, 0.10253996, 0.19842969, -0.09423829,
 0.01806441, 0.06689455, 0.2578125 , 0.03247097, -0.24609375,
 -0.05541992, 0.01613384, 0.24121094, -0.21871994, 0.07568359,
 -0.09814953, -0.15113281, 0.15939896, -0.09515484, -0.16601562],
```

**Gambar 5.41** hasil olah data BAG pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data CAR yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [15]: genmod ['car']
Out[15]:
array([0.13889393, 0.00842189, 0.03144727, -0.05833799, -0.04803995,
 0.14257812, 0.04093164, -0.15894531, 0.20889458, 0.11962391,
 0.18966406, -0.25, -0.18400351, -0.10742188, -0.03179883,
 0.16112121, 0.01111128, 0.03888379, 0.08447266, 0.16213895,
 0.16112121, -0.01111128, -0.03888379, 0.08447266, 0.16213895,
 0.04467773, -0.15527344, 0.25598625, 0.33984375, 0.89756856,
 0.16912109, 0.16593996, 0.06847466, -0.18945312, 0.02832031,
 -0.05346688, -0.03803965, 0.11083964, 0.24121094, -0.234375 ,])
```

**Gambar 5.42** hasil olah data CAR pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data WASH yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [16]: genmod ['wash']
Out[16]:
array([0.4662122e-03, 1.1461916e-01, -4.4467490e-02, 1.1476562e-01,
 -2.0792056e-01, 2.53986239e-01, -4.4473830e-02, -1.2848382e-02,
 -2.7954161e-02, 2.00807812e-01, -4.2724694e-02, 1.05463750e-01,
 -2.0792056e-01, 2.53986239e-01, -4.4473830e-02, -1.2848382e-02,
 2.6758125e-01, 2.12898625e-01, 1.74505847e-02, 2.02941895e-03,
 6.29828212e-02, 1.02189575e-01, 1.93359575e-01, 2.17285156e-02,
 -2.6758125e-01, -0.15893938e-02, -2.38212598e-01, 2.23533832e-01,])
```

**Gambar 5.43** hasil olah data WASH pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data MOTOR yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [17]: genmod ['motor']
Out[17]:
array([-1.37184659e-02, 1.58398625e-02, -4.11425781e-02, 1.33811580e-05,
 0.55975025e-01, -3.77733475e-01, 3.48902345e-02, 4.17909000e-01,
 2.34375080e-02, 2.57831250e-01, 1.74884686e-01, 2.41446256e-02,
 -0.83390781e-01, -0.58233125e-01, -0.58233125e-01, 1.12384668e-01,
 1.56250900e-01, -4.24884658e-02, -1.28125980e-01, 2.11934802e-01,
 1.93359575e-01, 2.17285156e-02, 1.02754375e-01, 1.53320312e-01, -1.09921875e-01, -1.01974219e-01,
 -3.26538066e-03, 2.28915625e-01, 8.98457980e-02, -7.12996254e-02,])
```

**Gambar 5.44** hasil olah data MOTOR pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data CYCLE yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [18]: genmod ['cycle']
Out[18]:
array([-0.65541016, 0.21579688, -0.02709961, 0.12335316, -0.28070312,
 -0.1328125, 0.26367188, -0.12890625, -0.125, 0.1532081,
 -0.1328125, 0.26367188, -0.12890625, -0.125, 0.1532081,
 0.02563477, -0.07568359, -0.0625, 0.04614258, -0.01054668,
 -0.13579986, -0.11669922, -0.33951755, 0.01125, 0.01057266,
 0.13579986, -0.11669922, -0.33951755, 0.01125, 0.01057266,
 0.18302734, 0.02172852, -0.10693359, 0.02409234, -0.10644531,
 -0.05541092, -0.29492186, -0.48659862, 0.06347856, -0.10644531,
 -0.17873994, 0.03165771, -0.01098777, 0.15971875, -0.16468625,])
```

**Gambar 5.45** hasil olah data CYCLE pada GoogleNews-vector

- dan pada hasil code berikut ini adalah hasil dari proses penggunaan perintah code similarity yang akan menghitung nilai value data yang dibandingkan dengan masing - masing kata seperti pada hasil dari perbandingan kata LOVE disandingkan dengan FAITH menghasilkan nilai 37 persen, sedangkan kata WASH dan SHINE menghasilkan nilai 27 persen dan kata CAR yang disandingkan dengan kata MOTOR menghasilkan 48 persen, dimana kita dapat menyimpulkan bahwa semakin data kata tersebut memiliki tingkat kesamaan yang tinggi maka

nilai hasil yang ditampilkanpun akan semakin tinggi. ilustrasi bisa dilihat pada gambar

```
In [19]: genmod.similarity('love', 'faith')
Out[19]: 0.3705347934587281

In [20]: genmod.similarity('wash', 'shine')
Out[20]: 0.2770128965426825

In [21]: genmod.similarity('car', 'motor')
Out[21]: 0.4810172832001571

In [22]: genmod.similarity('bag', 'cycle')
Out[22]: 0.040672609213443504

In [23]: genmod.similarity('shine', 'fall')
Out[23]: 0.27789493775772145
```

**Gambar 5.46** hasil olah data pada GoogleNews-vector menggunakan SIMILARITY

## 2. extract\_words dan PermutatedSentences

pada penjelasan berikut ini akan menyangkut pembersihan data yang akan digunakan untuk diproses, dimana data akan di EXTRACT dari setiap katanya agar terbebas dari data TAG HTML, APOSTROPHES, TANDA BACA, dan SPASI yang berlebih. dengan menggunakan perintah code STRIP dan SPLIT. lalu penggunaan library random yang akan dibuat untuk melakukan KOCLOK data dengan acuan datanya adalah data yang terdapat pada variable KATA. untuk ilustrasi hasil dari codenya dapat dilihat pada gambar

```
In [28]: import re
...: def extract_words(kata):
...: kata = kata.lower()
...: kata = re.sub(r'([^\w\s]+)', ' ', kata)
...: kata = re.sub(r'(\w)\.(\w)', '\1\2', kata)
...: kata = re.sub(r'\s+', ' ', kata)
...: kata = kata.strip()
...: return kata.split()
...:
...: import random
...: class PermutateSentences(object):
...: def __init__(self, lenght):
...: self.lenght = lenght
...:
...: def __iter__(self):
...: req = list(self.lenght)
...: random.shuffle(req)
...: for kata in req:
...: yield kata
```

**Gambar 5.47** hasil olah data pada GoogleNews-vector menggunakan extract\_words dan PermuteSentences

## 3. TaggedDocument dan Doc2Vec

gensim merupakan open-source model ruang vektor dan toolkit topic modeling, yang diimplementasikan dalam bahasa pemrograman Python. Untuk kinerja Gensim, digunakan NumPy, SciPy dan Cython (opsional). Gensim secara khusus ditujukan untuk menangani koleksi teks besar dengan menggunakan algoritma secara online. Gensim mengimplementasikan tf-idf, latent semantic analysis (LSA), Latent Dirichlet Analysis (LDA), dan lain-lain.

tagged document merupakan sebuah class yang terdapat pada pemrosesan data pada library gensim yang akan mengolah data teks yang ada pada dokumen - dokumen yang dipakai.

Doc2Vec merupakan algoritma doc embedding, yaitu pemetaan dari dokumen menjadi vektor, serta pemetaan data dokumen 1 dan dokumen lainnya. ilustrasi dari tagged document dan Word2Vec ada pada gambar

```
In [2]: from gensim.models.doc2vec import TaggedDocument
...: from gensim.models import Doc2Vec
```

**Gambar 5.48** TaggedDocument dan Doc2Vec

#### 4. Praktek data training

pertama buka data training yang akan diolah pada aplikasi python, import library OS dan membuat data variable unsup\_sentences dengan nilai array kosong. buatkan data direktori untuk memanggil data yang akan diolah dan buatkan juga variable data nilai fname yang akan memproses data dirname untuk diisikan pada variable unsup\_sentences. code yang digunakan dapat dilihat pada gambar

**Gambar 5.49** data code praktik data training

data pada hasil code digambar berikut ??, menghasilkan data pada gambar ?? yang akan memunculkan data variable DIRNAME, FNAME, KATA dan unsup\_sentences yang memiliki data sebanyak 55 kata dalam file yang diolah tersebut. hasil run dengan menggunakan code pada gambar ??, menghasilkan data nilai yang terdapat pada gambar ???. lalu pada code yang terdapat digambar ??, menghasilkan data ??.

```
import os
unsgd_sentences = []

for dirname in ["train/pos", "train/neg", "train/unsgd", "test/pos", "test/neg"]:
 for fname in sorted(os.listdir(dirname) + [dirname]):
 if fname == ".DS_Store": continue
 print(dirname, fname)
 with open(os.path.join(dirname, fname), encoding="UTF-8") as f:
 words = extract_words(f.read())
 words = extract_words(words)
 unsdg_sentences.append(TaggedDocument(words, [dirname + " " + fname]))
```

**Gambar 5.50** data code praktik data training

**Gambar 5.51** data code praktik data training

```
F:\D\123>
for dirname in [review_polarity/txt_sentoken/pos, review_polarity/txt_sentoken/neg]:
 for fname in os.listdir(dirname):
 if fname[4] == 't' or fname[4] == 'T':
 with open(dirname + '/' + fname, encoding='UTF-8') as f:
 for i, kata in enumerate(f):
 words = extract_words(kata)
 vasp.sentences.append(TaggedDocument(words, ['%s/%s-%s' % (dirname, fname, i)]))
```

**Gambar 5.52** data code praktik data training

| Name            | Type | Size    | Value                                                                                       |
|-----------------|------|---------|---------------------------------------------------------------------------------------------|
| dirname         | str  | 1       | review_polarity/txt_sentoken/neg                                                            |
| fname           | str  | 1       | cv999_14636.txt                                                                             |
| i               | int  | 1       | 24                                                                                          |
| kata            | str  | 1       | 'after watching _d_night_at_the_robuxy_... you'll be left with maxinu...                    |
| unsup_documents | list | 1300010 | [TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, Tagge ...] |
| words           | list | 3       | ['...', '...', '...']                                                                       |

**Gambar 5.53** data code praktik data training

```
In[23]:
with open("stanfordSentimentTreebank/original_rt_snippets.txt", encoding="UTF-8") as f:
 for i, line in enumerate(f):
 words = extract_words(line)
 unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))
```

**Gambar 5.54** data code praktik data training

| Name           | Type | Size  | Value                                                                                                                                                            |
|----------------|------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dirname        | str  | 1     | review_polarity/txt_Sentoken/neg                                                                                                                                 |
| fname          | str  | 1     | c999_14460.txt                                                                                                                                                   |
| i              | int  | 1     | 18604                                                                                                                                                            |
| para           | str  | 1     | after writing _e_late_at_the_rosbury_ , you'll be left with mostly blank lines.                                                                                  |
| kata           | str  | 1     | her fans walked out muttering words like                                                                                                                         |
| line           | str  | 1     | the first time i saw you                                                                                                                                         |
| unus_sentences | list | 10000 | [TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument, TaggedDocument] |
| words          | list | 9     | ['.', '...', '']                                                                                                                                                 |

**Gambar 5.55** data code praktik data training

## 5. Why need Shuffled and Clean memory

dilakukan shuffled adalah agar datanya lebih mudah untuk diolah dan untuk menentukan tingkat tinggi akurasi dari hasil pemrosesan. dan dilakukan pembersihan memory adalah agar chace yang disimpan tidak membuat proses pada komputer menjadi lambat dan dapat digunakan untuk memproses data lainnya agar menjadi lebih ringan dan cepat. pada gambar ?? adlah proses untuk melakukan pengoclokan data dan pada gambar ?? adalah proses untuk memasukan data unsup\_sentences kedalam variable muter untuk diproses dengan class PermuterSentences. dan pada gambar ?? adalah code yang digunakan untuk membersihkan data memory.

```
In [28]: import re
.....
....: class PermuteSentences(object):
....: def __init__(self, lenght):
....: self.lenght = lenght
....:
....: def __iter__(self):
....: req = list(self.lenght)
....: random.shuffle(req)
....: for kata in req:
....: yield kata
....:
```

**Gambar 5.56** Shuffled dan Randomisasi data

```
In [18]: muter = PermuteSentences(unsup_sentences)
...: mod = Doc2Vec(muter, dm=0, hs=1, size=50)
```

**Gambar 5.57** pembuatan variable muter untuk memuat data unsup\_sentences

```
mod.delete_temporary_training_data(keep_inference=True)
```

**Gambar 5.58** code untuk membersihkan data memory

## 6. Why model have to be saved

dalam pengolahan data dengan menggunakan proses yang panjang ditakutkan data yang sudah diproses tersebut dapat hilang jika terdapat kejadian atau emergency pada saat pengolahan dan pemrosesan data, misalnya harddisk error atau pun listrik yang padam. dan proses penyimpanan data juga dilakukan agar data yang sudah diolah data dipanggil lagi tanpa harus melakukan proses dari awal sehingga tidak memakan waktu. untuk code yang digunakan dapat dilihat pada gambar

| ④ SentimentAnalysis         | 3/18/2019 3:14 PM  | Python Source File   | 2 KB     |
|-----------------------------|--------------------|----------------------|----------|
| ④ SentimentAnalysisnew.pytb | 3/18/2019 3:22 PM  | Python Source File   | 1 KB     |
| ④ stopword.txt              | 3/18/2019 3:22 PM  | Text File            | 54KB (0) |
| ④ tagger3                   | 3/20/2019 11:41 PM | Python Source File   | 2 KB     |
| ④ word2vec                  | 3/20/2019 12:05 AM | Python Source File   | 6 KB     |
| ④ You tube01-Phi            | 3/18/2019 3:14 PM  | Microsoft Excel C... | 27 KB    |
| ④ You tube01-KeyPeru        | 3/18/2019 3:14 PM  | Microsoft Excel C... | 64 KB    |
| ④ You tube01-LM3Q           | 3/18/2019 3:14 PM  | Microsoft Excel C... | 64 KB    |
| ④ You tube04-Eminem         | 3/18/2019 3:14 PM  | Microsoft Excel C... | 32 KB    |
| ④ You tube05-Shakira        | 3/18/2019 3:14 PM  | Microsoft Excel C... | 72 KB    |

**Gambar 5.59** data code save data

berikut ini adalah hasil file dari penggunaan code save tersebut. bisa dilihat pada gambar

```
In [19]: mod.infer_vector(extract_words("This Place is not worth your time,
let alone Vegas."))
Out[19]: [-0.00374494, -0.00037316, -0.000647218, 0.00074012, 0.00060511,
-0.00058244, 0.00033789, -0.00088553, 0.00724112, 0.002009,
-0.00053485, 0.00057326, 0.00071274, 0.00074205, 0.00414126,
0.00034863, -0.00175985, -0.00996599, -0.00778317, 0.00423807,
0.0006305, 0.0077526, 0.00071274, 0.00074205, 0.00414126,
-0.00056701, 0.0064773, 0.00077156, -0.00292126, 0.00395201,
-0.0005263, 0.00077156, 0.00077156, 0.00077156, 0.00077156,
0.0005263, 0.00927048, -0.00292126, -0.00058527, 0.0073598,
0.00058527, -0.00279456, -0.00075326, 0.00646478, -0.00494211],
dtype='float32')
```

**Gambar 5.60** hasil file simpan

## 7. infer\_vector

berfungsi untuk dokumen baru, dan bisa menggunakan data vektor yang dilatih secara massal, seperti yang disimpan dalam model, untuk dokumen yang merupakan bagian dari data training. untuk percobaannya dapat dilihat pada gambar

```
In [12]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity(
...: [mod.infer_vector(extract_words("highly recommended."))],
...: [mod.infer_vector(extract_words("services sucks."))])
Out[12]: array([[0.206962]], dtype=float32)
```

**Gambar 5.61** code dan hasil infer\_vector

## 8. cosine\_similarity

merupakan sebuah algoritma yang digunakan untuk membandingkan dari dua buah data yang bukan merupakan data vector untuk menguji nilai kemiripan data satu dengan data lainnya. hasil dari percobaan pada tugas no 8 ini dapat dilihat pada gambar ?? yang menghasilkan nilai akurasi sebesar 20 persen dan gambar ?? yang menghasilkan nilai akurasi sebesar 91 persen.

```
In [13]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity(
...: [mod.infer_vector(extract_words("tolong bantuan."))],
...: [mod.infer_vector(extract_words("tolong bantuan."))])
Out[13]: array([[0.91439525]], dtype=float32)
```

**Gambar 5.62** code dan hasil penggunaan cosine\_similarity

```
In [36]: from sklearn.neighbors import KNeighborsClassifier
...: from sklearn.ensemble import RandomForestClassifier
...: from sklearn.model_selection import cross_val_score
...: import numpy as np
...: ...
...: clf = KNeighborsClassifier(n_neighbors=9)
...: clfrf = RandomForestClassifier()
```

**Gambar 5.63** code dan hasil penggunaan cosine\_similarity

## 9. Cross Validation

pertama melakukan import data dari library KNeighborsClassifier, RandomForestClassifier, cross\_val\_score dan numpy yang digunakan untuk membuat data cross validasi dapat dilihat pada gambar

```
In [40]: scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
...: np.mean(scores), np.std(scores)
Out[40]: (0.5283333333333334, 0.086411794687223791)
```

**Gambar 5.64** memasukan code import library

lalu selanjutnya membuat data variable scores yang akan memuat nilai cross\_val\_score dengan datanya diambil dari KNeighborsClassifier

yang terdiri dari sentvecs, sentiments dan clf dan mengolahnya menggunakan numpy untuk menampilkan data pada gambar yang menghasilkan nilai akurasi sebesar 53 persen.

```
[In [41]: scores2 = cross_val_score(cIrrf, sentvecs, sentiments, cv=5)
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
Out[41]: 0.5319999999999999, 0.40044635988804594]
```

**Gambar 5.65** perhitungan data KNeighborsClassifier dengan cross validasi

membuat data variable scores yang akan memuat nilai cross\_val\_score dengan datanya diambil dari RandomForestClassifier yang terdiri dari sentvecs, sentiments dan clrf dan mengolahnya menggunakan numpy untuk menampilkan data pada gambar yang menghasilkan nilai akurasi sebesar 53 persen.

```
[In [42]: from sklearn.pipeline import make_pipeline
... from sklearn.feature_extraction.text import CountVectorizer,
... RandomForestClassifier
... pipeline = make_pipeline(CountVectorizer(), TfidfTransformer(),
RandomForestClassifier())
... scores1 = cross_val_score(pipeline,sentences,sentiments, cv=5)
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
C:\Users\fatih-PC\OneDrive\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in version 0.20 to 100 in 0.22.*. FutureWarning
Out[42]: 0.7446666666666667, 0.823426979311713]
```

**Gambar 5.66** perhitungan data RandomForestClassifier dengan cross validasi

penggunaan make\_pipeline adalah untuk membuat data dari KNeighborsClassifier, RandomForestClassifier dan Vectorizer digabungkan untuk menghasilkan data nilai pada gambar menghasilkan nilai akurasi sebesar 74 persen.

| scores  | float64 (5,) | (0.53333333 0.335 0.53333333 0.51833333      |
|---------|--------------|----------------------------------------------|
| scores2 | float64 (5,) | (0.53333333 0.52333333 0.26666667 0.53333333 |
| scores3 | float64 (5,) | (0.74666667 0.77333333 0.71833333 0.71166667 |

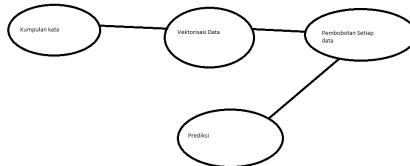
**Gambar 5.67** perhitungan data Cross Validasi untuk nilai keseluruhan dari KNeighborsClassifier, RandomForestClassifier dan Vectorizer

### 5.3 Luthfi Muhammad Nabil (1174035)

#### 5.3.1 Teori

1. Jelaskan Kenapa Kata-Kata harus dilakukan vektorisasi lengkapi dengan ilustrasi gambar.

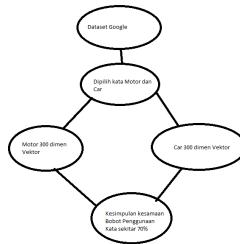
Kata kata harus dilakukan vektorisasi dikarenakan atau bertujuan untuk mengukur nilai sebuah kata yang nantinya setiap kata tersebut akan diprediksi apakah bernilai yang berarti terpakai atau tidak.



**Gambar 5.68** Ilustrasi Vektorisasi Kata-Kata

2. Jelaskan Mengapa dimensi dari vektor dataset google bisa mencapai 300 lengkapi dengan ilustrasi gambar.

Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap kata, misalkan terdapat kata dog dan cat pada dataset google tersebut setiap kata tersebut dibuat dimensi vektor 300 untuk kata dog dan 300 dimensi vektor juga untuk kata cat kemudian kata tersebut dibandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70 persen kesamaan bobot dikarenakan kata dog dan cat sama-sama digunakan untuk hewan priharaan. Untuk lebih jelasnya dapat dilihat pada gambar ??.



**Gambar 5.69** Ilustrasi Vektorisasi Kata-Kata

3. Jelaskan Konsep vektorisasi untuk kata . dilengkapi dengan ilustrasi atau gambar.

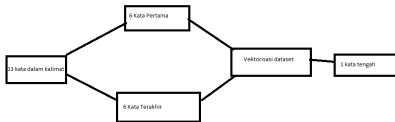
Vektorisasi untuk kata untuk mengetahui kata tengah dari suatu kalimat atau kata utama atau objek utama pada suatu kalimat kata tengah tersebut merupakan channel yang memiliki bobot sebagai kata tengah dari suatu kalimat atau bobot sebagai objek dari suatu kalimat. hal ini sangat berkaitan dengan dimensi vektor pada dataset google yang 300 tadi karena untuk mendapatkan nilai atau bobot dari kata tengah tersebut di dapatkan dari proses dimensiasi dari kata tersebut. untuk lebih jelasnya dapat dilihat pada gambar ?? berikut :



**Gambar 5.70** Ilustrasi Vektorisasi Kata-Kata

4. Jelaskan Konsep vektorisasi untuk dokumen. dilengkapi dengan ilustrasi atau gambar.

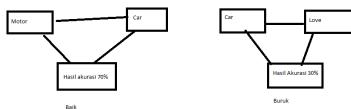
Vektorisasi untuk dokumen hampir sama seperti vektorisasi untuk kata hanya saja pemilihan kata utama atau kata tengah terdapat pada satu dokumen jadi mesin akan membuat dimensi vektor 300 untuk dokumen dan nanti kata tengahnya akan di sandingkan pada dokumen yang terdapat pada dokumen tersebut contoh dapat dilihat pada gambar ?? berikut :



**Gambar 5.71** Ilustrasi Vektorisasi Kata-Kata

5. Jelaskan apa mean dan standar deviasi, lengkapi dengan ilustrasi atau gambar.

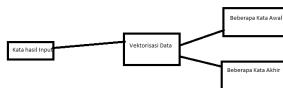
mean merupakan petunjuk terhadap kata-kata yang di olah jika kata kata itu akurasinya tinggi berarti kata tersebut sering muncul begitu juga sebaliknya untuk lebih jelasnya dapat dilihat pada gambar ?? sedangkan setandard deviation merupakan standar untuk menimbang kesalahan. sehingga kesalahan tersebut di anggap wajar misalkan kita memperkirakan kedalaman dari dataset merupakan 2 atau 3 tapi pada kenyataannya merupakan 5 itu merupakan kesalahan tapi masih bisa dianggap wajar karna masih mendekati perkiraan awal.



**Gambar 5.72** Ilustrasi Vektorisasi Kata-Kata

- Jelaskan Apa itu Skip-Gram sertakan contoh ilustrasi.

Skip-Gram adalah kebalikan dari konsep vektorisasi untuk kata dimana kata tengah menjadi acuan terhadap kata-kata pelengkap dalam suatu kalimat untuk lebih jelasnya dapat dilihat pada gambar ?? berikut :



**Gambar 5.73** Ilustrasi Vektorisasi Kata-Kata

## 5.4 1174039- Liyana Majdah Rahma

### 5.4.1 Teori

- Mengapa kata-kata dilakukan Vektorisasi

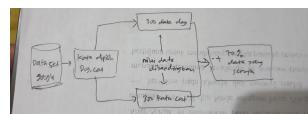
dikarenakan kata-kata yang biasanya digunakan untuk proses data supaya dapat menjadi bagian dari kumpulan beberapa data yang dapat dibaca oleh sistem, sehingga sistem tersebut dapat memproses data text secara langsung tanpa di convert. Dapat dilihat seperti gambar dibawah ini



**Gambar 5.74** Vektorisasi Kata

- Mengapa dimensi dari vektor dataset google bisa sampai 300

dikarenakan dimensi pada dataset google bisa mencapai 300 tersebut digunakan untuk membandingkan bobot dari setiap hasil data yang diproses. Dapat dilihat seperti gambar dibawah ini



**Gambar 5.75** Dataset Google

### 3. Jelaskan konsep vektorisasi untuk kata

pada vektorisasi tersebut digunakan word2vec yang memiliki keunggulan yang dapat dibedakan dengan penggunaan bag of word yang biasanya. Dapat dilihat seperti gambar dibawah ini



**Gambar 5.76** konsep untuk kata

### 4. jelaskan konsep vektorisasi untuk dokumen

vektorisasi pada Doc2Vec dimana data yang terdapat pada file document tersebut diolah dengan melakukan pemrosesan yang mengutamakan nilai filenamenya atau atribut utama dimana nilai data inputnya tidak terlalu diproses. Dapat dilihat seperti gambar dibawah ini



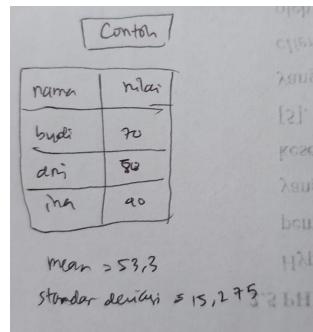
**Gambar 5.77** konsep untuk dokumen

### 5. Jelaskan apa mean dan standar deviasi

Mean adalah nilai rata-rata dari beberapa buah data. Nilai mean dapat ditentukan dengan membagi jumlah data dengan banyaknya data.

sedangkan Standar deviasi adalah nilai statistik yang digunakan untuk menentukan bagaimana sebaran data dalam sampel, dan seberapa dekat titik data individu ke mean – atau rata-rata – nilai sampel.

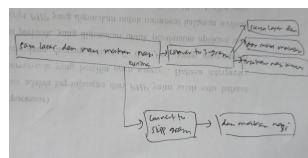
Dapat dilihat seperti gambar dibawah ini



**Gambar 5.78** Mean and Deviation Standard

## 6. Apa itu skip-gram

Skip-gram merupakan teknik yang digunakan di area speech processing, dimana n-gram yang dibentuk kemudian ditambahkan juga dengan tindakan “skip” pada token-tokennya. Dapat dilihat seperti gambar dibawah ini



**Gambar 5.79** Skip-Gram

### 5.4.2 Praktek

## 1. Try datasets GoogleNews-vectors

- berikut adalah hasil dari code yang digunakan untuk memanggil data library GENSIM dengan menggunakan perintah import, lalu dari library tersebut diambilah data yang akan digunakan untuk memproses data dari GoogleNews-vector. ilustrasi dapat dilihat pada gambar

```
In [6]: import gensim
In [7]: gennod = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)
```

**Gambar 5.80** import gensim dan olah data GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data LOVE yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [8]: genom['love']
Out[8]: array([0.10302374, -0.15234375, 0.02587891, 0.16503986,
 0.16503906, -0.06869453, 0.29296875, -0.26567188, -0.140625, 0.20117188,
 -0.02645512, -0.08208125, -0.02770996, 0.04394563, 0.23535155,
 0.16992188, 0.12809625, 0.15722565, 0.07567836, 0.06982422,
 -0.03574027, 0.07955862, 0.07955862, 0.16797891, 0.16797891,
 -0.08578951, 0.075798, 0.10933333, 0.11183333, 0.11183333,
 0.11183333, 0.13396458, 0.15516596, 0.17283961, 0.15429861])
```

**Gambar 5.81** hasil olah data LOVE pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data FAITH yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [9]: gennod['faith']
Out[9]:
array([0.26367188, -0.04150391, 0.1951325, 0.13476562, -0.14648438,
 0.11962891, 0.04345703, 0.10351562, 0.12072031, 0.13476562,
 0.06646825, 0.18945312, -0.16601562, 0.21769788, -0.27148438,
 0.03012525, 0.15332081, -0.18939844, 0.06553906, 0.03671617, 0.23144535,
 -0.05957083, 0.22949219, -0.06064248, 0.26177175, 0.10302734,
 -0.13281285, 0.21484375, 0.11352554, 0.02111816, 0.18554683])
```

**Gambar 5.82** hasil olah data FAITH pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data FALL yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [40]: gennod['fall']
Out[40]:
array([-0.04274261, 0.10742188, -0.09277344, 0.16894531, -0.13281215
 ...,
 -0.06913595, 0.04321289, 0.19594297, 0.14648438, 0.15059062,
 -0.08691466, 0.04492188, 0.1458747, 0.08691466, -0.19824219,
 -0.11895156, 0.01092529, -0.08306781, 0.07915523, -0.1955123,
 -0.10150265, 0.13671875, 0.09228516, -0.12109375, 0.12095375,
 -0.03179693, 0.21093735, 0.81977539, 0.01544199])
```

**Gambar 5.83** hasil olah data FALL pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data SICK yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [11]: genmod['sick']
Out[11]:
array([1.28617188e-01, 1.49414062e-01, -4.05273438e-02,
 1.64062500e-01, -2.59765625e-01, 3.22656250e-01, 1.73828125e-01,
 -1.47460938e-01, 1.01074219e-01, 5.46876500e-02, 1.66992188e-01,
 -1.68945121e-01, 2.24304919e-03, 9.66796757e-02, 1.66015625e-01,
 -1.12346868e-01, 1.66052500e-01, 1.79678500e-01, 5.01841016e-03,
 2.45117188e-01, 8.74023438e-02, -2.56347656e-02, 3.41796785e-02,
 4.90086750e-02])
```

**Gambar 5.84** hasil olah data SICK pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data CLEAR yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [12]: genmod['clear']
Out[12]: array([-2.44146026e-04, -1.02050781e-01, -1.49414062e-01, -4.24804688e-02,
-1.67986759e-01, -1.46484375e-01, 1.76757812e-01, 1.46448437e-01,
-1.25155121e-01, -1.25362812e-01, -6.38597188e-02, 1.30855940e-01,
2.005195312e-01, 4.76074219e-02, -6.38597188e-02, 1.30855940e-01,
-1.25155121e-01, 3.14453125e-01, -1.11816408e-01, 8.89715908e-02,
-2.73866525e-02, 8.13130000e-01, 7.67182000e-01, 1.72856600e-01,
-1.67986759e-01, 8.13130000e-01, 7.67182000e-01, 1.72856600e-01])
```

Gambar 5.85 hasil olah data CLEAR pada GoogleNews vector

- lalu pada penggunaan code berikut ini akan mengolah data SHINE yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [13]: genmod ['shine']
Out[13]: array([-0.12482344, 0.20579052, -0.15947969, -0.27734575, 0.38077458,
 -0.06090932, 0.39557512, -0.23899219, -0.13259327, 0.35715779,
 -0.18902734, 0.13571785, 0.25399625, 0.07128986, 0.02539962,
 0.22777344, 0.24023438, 0.5234735, 0.12308888, -0.19359518,
 0.20891531, 0.38805938, 0.08562896, -0.05829377, 0.14648458,
 0.20891531, 0.38805938, 0.08562896, -0.05829377, 0.14648458,
 -0.24212625, 0.02945177, 0.23947345, 0.09193193, 0.81454296,
 0.390625, -0.2109375, 0.1484375, -0.13183594, 0.24511719,
```

**Gambar 5.86** hasil olah data SHINE pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data BAG yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [14]: genmod ['bag']
Out[14]: array([-0.0515625, 0.1524875, -0.12602344, 0.13779896, -0.11328125,
 -0.0608559, -0.3046875, 0.20969694, -0.04345793, -0.2109375,
 -0.05957331, -0.05653711, 0.10253985, 0.19042969, -0.09423829,
 0.05200195, 0.02166625, 0.06045112, 0.14453123, -0.04541816,
 -0.15527344, -0.18945132, 0.11132812, 0.27539862, -0.06877109,
 0.05541445, 0.06191334, 0.24121894, 0.21874789, -0.07568359,
 -0.09814453, -0.16113281, 0.16503986, -0.09521484, -0.16601562,
```

**Gambar 5.87** hasil olah data BAG pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data CAR yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [15]: genmod ['car']
Out[15]: array([0.13005938, 0.03442285, 0.03344727, -0.05883789, 0.04003966,
 -0.18864498, -0.05132151, -0.10800391, -0.10742168, -0.01879881,
 0.05200195, -0.02166625, 0.06454512, 0.14453123, -0.04541816,
 -0.15527344, -0.15527344, 0.25399625, 0.33948475, 0.08756836,
 -0.25585938, -0.01733398, 0.02395988, 0.16308954, 0.12957656,
 -0.0534668, -0.05089365, 0.11683984, 0.24121894, -0.134375,
```

**Gambar 5.88** hasil olah data CAR pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data WASH yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [16]: genmod ['wash']
Out[16]: array([0.46044523e-05, 1.41081502e-01, -5.46875000e-02, 1.34705625e-01,
 -1.39832100e-01, 2.35532150e-01, 0.24515700e-01, -0.44735562e-01, 1.39832100e-01,
 1.87910156e-01, 2.53986250e-01, 1.13523911e-02, -1.66992105e-01,
 -2.79510761e-02, 2.0800912e-01, -4.23775000e-02, 1.85642100e-01,
 -2.87597500e-01, 2.0800912e-01, 1.13523911e-02, -1.66992105e-01,
 2.6778125e-01, 2.12898025e-01, 1.74508547e-02, 2.02841954e-03,
 0.84541445e-01, 1.08089579e-01, 0.108089579e-01, 2.02841954e-03,
 -2.67028800e-01, -0.13089579e-01, -2.38201250e-01, 2.23632812e-01],
```

**Gambar 5.89** hasil olah data WASH pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data MOTOR yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [17]: genmod['motor']
Out[17]:
array([-7.7374806e-02, 1.5839862e-02, -4.1425791e-02, -3.2381580e-01,
 3.5976525e-01, -1.77734375e-01, 3.6852354e-02, -4.7590000e-01,
 2.3475800e-01, 2.5781250e-01, 1.7480468e-01, 2.4146252e-02,
 1.5625000e-01, -1.2875000e-01, 1.2875000e-01, -1.2875000e-01,
 1.8338071e-02, 1.58203125e-01, -5.8937500e-02, 1.12384588e-01,
 1.5625000e-01, -4.2489468e-02, -1.5821250e-01, 2.11934802e-01,
 1.5625000e-01, -1.2875000e-01, 1.2875000e-01, -1.2875000e-01,
 3.02734175e-01, 1.53320312e-01, -1.69211754e-01, -1.10794219e-01,
 -2.26538006e-03, 2.28515625e-01, 8.98437500e-02, -7.12396425e-02],
```

**Gambar 5.90** hasil olah data MOTOR pada GoogleNews-vector

- lalu pada penggunaan code berikut ini akan mengolah data CYCLE yang terdapat pada file GoogleNews-vector, hasil dari pemrosesannya dapat dilihat pada gambar

```
In [18]: genmod['cycle']
Out[18]:
array([0.8545016, -0.21679688, -0.02709961, -0.12353516, -0.28970125,
 0.1328125, 0.26367188, -0.12890625, -0.125, 0.15532031,
 -0.18261719, -0.15820812, -0.06176758, 0.21972656, -0.15820312,
 -0.1328125, 0.26367188, -0.12890625, -0.125, 0.15532031,
 -0.13379986, -0.11669922, -0.3359375, 0.078125, 0.084472656,
 0.87228562, -0.06445312, 0.05517576, 0.14941495, 0.13671875,
 -0.05514992, 0.29492188, -0.40839862, 0.96347656, -0.884472656,
 0.17871094, 0.01165771, -0.01096777, 0.13671875, -0.1540625,
```

**Gambar 5.91** hasil olah data CYCLE pada GoogleNews-vector

- dan pada hasil code berikut ini adalah hasil dari proses penggunaan perintah code similarity yang akan menghitung nilai value data yang dibandingkan dengan masing - masing kata seperti pada hasil dari perbandingan kata LOVE disandingkan dengan FAITH menghasilkan nilai 37 persen, sedangkan kata WASH dan SHINE menghasilkan nilai 27 persen dan kata CAR yang disandingkan dengan kata MOTOR menghasilkan 48 persen, dimana kita dapat menyimpulkan bahwa semakin data kata tersebut memiliki tingkat kesamaan yang tinggi maka nilai hasil yang ditampilkanpun akan semakin tinggi. ilustrasi bisa dilihat pada gambar

```
In [19]: genmod.similarity('love', 'faith')
Out[19]: 0.3705347934587281

In [20]: genmod.similarity('wash', 'shine')
Out[20]: 0.2770128965426825

In [21]: genmod.similarity('car', 'motor')
Out[21]: 0.4810172832001571

In [22]: genmod.similarity('bag', 'cycle')
Out[22]: 0.0406726092113443504

In [23]: genmod.similarity('shine', 'fall')
Out[23]: 0.27789493775772145
```

**Gambar 5.92** hasil olah data pada GoogleNews-vector menggunakan SIMILARITY

## 2. extract\_words dan PermuttedSentences

pada penjelasan berikut ini akan menyangkut pembersihan data yang akan digunakan untuk diproses, dimana data akan di EXTRACT dari setiap katanya agar terbebas dari data TAG HTML, APOSTROPHES, TANDA BACA, dan SPASI yang berlebih. dengan menggunakan perintah code STRIP dan SPLIT. lalu penggunaan library random yang

akan dibuat untuk melakukan KOCLOK data dengan acuan datanya adalah data yang terdapat pada variable KATA. untuk ilustrasi hasil dari codenya dapat dilihat pada gambar

```
In [28]: import re
...: def extract_words(kata):
...: kata = kata.lower()
...: kata = re.sub(r'<[^>]+>', ' ', kata)
...: kata = re.sub(r'(\w+)\.(\w+)', '\1\2', kata)
...: kata = re.sub(r'\w+', ' ', kata)
...: kata = re.sub(r'\s+', ' ', kata)
...: kata = kata.strip()
...: return kata.split()
...:
...: import random
...: class PermuteSentences(object):
...: def __init__(self, lenght):
...: self.lenght = lenght
...:
...: def __iter__(self):
...: req = list(self.lenght)
...: random.shuffle(req)
...: for kata in req:
...: yield kata
```

**Gambar 5.93** hasil olah data pada GoogleNews-vector menggunakan extract\_words dan PermuteSentences

### 3. TaggedDocument dan Doc2Vec

gensim merupakan open-source model ruang vektor dan toolkit topic modeling, yang diimplementasikan dalam bahasa pemrograman Python. Untuk kinerja Gensim, digunakan NumPy, SciPy dan Cython (opsional). Gensim secara khusus ditujukan untuk menangani koleksi teks besar dengan menggunakan algoritma secara online. Gensim mengimplementasikan tf-idf, latent semantic analysis (LSA), Latent Dirichlet Analysis (LDA), dan lain-lain.

tagged document merupakan sebuah class yang terdapat pada penerapan data pada library gensim yang akan mengolah data teks yang ada pada dokumen - dokumen yang dipakai.

Doc2Vec merupakan algoritma doc embedding, yaitu pemetaan dari dokumen menjadi vektor, serta pemetaan data dokumen 1 dan dokumen lainnya. ilustrasi dari tagged document dan Word2Vec ada pada gambar

```
In [2]: from gensim.models.doc2vec import TaggedDocument
...: from gensim.models import Doc2Vec
```

**Gambar 5.94** TaggedDocument dan Doc2Vec

### 4. Praktek data training

pertama buka data training yang akan diolah pada aplikasi python, import library OS dan membuat data variable unsup\_sentences dengan nilai array kosong. buatkan data direktori untuk memanggil data yang akan diolah dan buatkan juga variable data nilai fname yang akan memproses data dirname untuk diisikan pada variable unsup\_sentences. code yang digunakan dapat dilihat pada gambar

**Gambar 5.95** data code praktik data training

data pada hasil code digambar berikut ??, menghasilkan data pada gambar ?? yang akan memunculkan data variable DIRNAME, FNAME, KATA dan unsup\_sentences yang memiliki data sebanyak 55 kata dalam file yang diolah tersebut. hasil run dengan menggunakan code pada gambar ??, menghasilkan data nilai yang terdapat pada gambar ???. lalu pada code yang terdapat digambar ??, menghasilkan data ??.

```

import os
unsup_sentences = []

for dirname in ["train/pos", "train/neg", "train/unsp", "test/pos", "test/neg"]:
 if dirname == "train":
 for fname in sorted(os.listdir("train/" + dirname)):
 if fname[-4:] == ".txt":
 with open("train/" + dirname + "/" + fname, encoding='UTF-8') as f:
 words = f.read().split()
 words = extract_nerds(words)
 unsup_sentences.append(TaggedDocument(words, [dirname + "/" + fname]))

```

**Gambar 5.96** data code praktik data training

**Gambar 5.97** data code praktik data training

```
[4]: [1]
for dirname in ["review_polarity/txt_sentoken/pos", "review_polarity/txt_sentoken/neg"]:
 for frame in sorted(os.listdir(dirname)):
 if frame[4:] != ".txt":
 with open(dirname + "/" + frame, encoding='UTF-8') as f:
 for i, kata in enumerate(f):
 words.append(kata)
 sentence.append(i)
 word_index.append(i)
 word_sentence.append((i, kata))
 word_sentence.append((i, frame))
```

Gambar 5.98 data code praktik data training

**Gambar 5.99** data code praktik data training

```
In[23]:
with open("stanfordSentimentTreebank/original_rt_snippets.txt", encoding='UTF-8') as f:
 for i, line in enumerate(f):
 words = extract_words(kata)
 unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))
```

**Gambar 5.100** data code praktik data training

| Name            | Type | Size | Value                                                                           |
|-----------------|------|------|---------------------------------------------------------------------------------|
| dirname         | str  | 1    | review_polarity/txt_semeval/neg                                                 |
| fnname          | str  | 1    | cw999_14630.txt                                                                 |
| i               | int  | 1    | 1866                                                                            |
| kata            | str  | 1    | after catching _a_night_at_the Roxbury_ , you'll<br>have trouble sleeping . . . |
| line            | str  | 1    | Fan fans walked out uttering words like<br>"horrible" and "terrible".           |
| usnlp_sentences | list | 1    | [{'text': 'The document', 'TaggedDocument': TaggedDocument, 'Tagger': ...},     |
| words           | list | 3    | ['.', '...', '']                                                                |

**Gambar 5.101** data code praktik data training

## 5. Mengapa dibutuhkan Shuffled Dan Clean memory

dilakukan shuffled adalah agar datanya lebih mudah untuk diolah dan untuk menentukan tingkat tinggi akurasi dari hasil pemrosesan. dan dilakukan pembersihan memory adalah agar chace yang disimpan tidak membuat proses pada komputer menjadi lambat dan dapat digunakan untuk memproses data lainnya agar menjadi lebih ringan dan cepat. pada gambar ?? adalah proses untuk melakukan pengoclokan data dan pada gambar ?? adalah proses untuk memasukan data unsup\_sentences kedalam variable muter untuk diproses dengan class PermuterSentences. dan pada gambar ?? adalah code yang digunakan untuk membersihkan data memory.

```
In [28]: import re
... def extract_words(kata):
... kata = kata.lower()
... kata = re.sub(r'<[^>]+>', ' ', kata)
... kata = re.sub(r'([^\w\s])\1+', '\1', kata)
... kata = re.sub(r'\s+', ' ', kata)
... kata = re.sub(r'^\s+', ' ', kata)
... kata = kata.strip()
... return kata.split()
...
...
...
... import random
... class PermuteSentences(object):
... def __init__(self, length):
... self.length = length
...
... def __iter__(self):
... reg = list(self.length)
... random.shuffle(reg)
... for kata in req:
... yield kata
```

**Gambar 5.102** Shuffled dan Randomisasi data

```
In [10]: muter = PermuteSentences(unsup_sentences)
 ...: mod = Doc2Vec(muter, dm=0, hs=1, size=50)
```

**Gambar 5.103** pembuatan variable muter untuk memuat data unsup\_sentences

```
mod.delete_temporary_training_data(keep_inference=True)
```

**Gambar 5.104** code untuk membersihkan data memory

## 6. Mengapa Model diperlukan Penyimpanan

dalam pengolahan data dengan menggunakan proses yang panjang ditakutkan data yang sudah diproses tersebut dapat hilang jika terdapat kejadian atau emergency pada saat pengolahan dan pemrosesan data, misalnya harddisk error atau pun listrik yang padam. dan proses penyimpanan data juga dilakukan agar data yang sudah diolah data dipanggil lagi tanpa harus melakukan proses dari awal sehingga tidak memakan waktu. untuk code yang digunakan dapat dilihat pada gambar

```
mod.save('simpanan.d2v')
```

**Gambar 5.105** data code save data

berikut ini adalah hasil file dari penggunaan code save tersebut. bisa dilihat pada gambar

|                         |                    |                      |       |
|-------------------------|--------------------|----------------------|-------|
| • SentimentAnalysis     | 3/18/2019 3:14 PM  | Python Source File   | 2 KB  |
| • SentimentAnalysis.pyw | 3/18/2019 3:14 PM  | Python Script File   | 50 KB |
| • simpanan.d2v          | 3/18/2019 3:25 AM  | DBF File             | 54 KB |
| • tugas5                | 3/20/2019 11:41 PM | Python Source File   | 2 KB  |
| • word2vec              | 3/20/2019 12:05 AM | Python Source File   | 6 KB  |
| • word2vec.py           | 3/20/2019 12:05 AM | Python Script File   | 6 KB  |
| • You tube0-KatyPerry   | 3/18/2019 3:14 PM  | Microsoft Excel C... | 2 KB  |
| • You tube0-LMFAO       | 3/18/2019 3:14 PM  | Microsoft Excel C... | 64 KB |
| • You tube0-Eminem      | 3/18/2019 3:14 PM  | Microsoft Excel C... | 32 KB |
| • You tube0-Shakira     | 3/18/2019 3:14 PM  | Microsoft Excel C... | 72 KB |

**Gambar 5.106** hasil file simpan

## 7. infer\_vector

berfungsi untuk dokumen baru, dan bisa menggunakan data vektor yang dilatih secara massal, seperti yang disimpan dalam model, untuk dokumen yang merupakan bagian dari data training. untuk percobaannya dapat dilihat pada gambar

```
In [13]: mod.infer_vector(extract_words("This Place is not worth your time,
let alone Vegas."))
Out[13]:
array([-0.00174494, -0.0017315, -0.000647218, -0.00174912, 0.009650511,
 -0.0038244, 0.00013779, 0.00008583, 0.00724112, 0.00260301,
 0.00000001, 0.00000001, 0.00000001, 0.00000001, 0.00000001,
 0.00034063, -0.0015949, -0.0009659, -0.00773017, 0.00420007,
 0.00000001, 0.00000001, 0.00000001, 0.00000001, 0.00000001,
 -0.0092356, -0.0084765, 0.00395087, 0.00035592, -0.00737975,
 -0.00567001, 0.0004775, -0.0077156, -0.00292189, 0.0001281,
 -0.00000001, 0.00000001, 0.00000001, 0.00000001, 0.00000001,
 0.0002630, 0.00927046, -0.0021578, -0.00059527, 0.0073998,
 0.00000001, -0.0027945, 0.00075359, 0.00048475, -0.00044211],
dtype='float32')
```

**Gambar 5.107** code dan hasil infer\_vector

## 8. cosine\_similarity

merupakan sebuah algoritma yang digunakan untuk membandingkan dari dua buah data yang bukan merupakan data vector untuk menguji nilai kemiripan data satu dengan data lainnya. hasil dari percobaan pada tugas no 8 ini dapat dilihat pada gambar ?? yang menghasilkan nilai akurasi sebesar 20 persen dan gambar ?? yang menghasilkan nilai akurasi sebesar 91 persen.

```
In [12]: from sklearn.metrics.pairwise import cosine_similarity
... cosine_similarity(
... ... [mod.infer_vector(extract.words("highly recommended."))],
... ... [mod.infer_vector(extract.words("service sucks. "))])
Out[12]: array([[0.208962]], dtype=float32)
```

**Gambar 5.108** code dan hasil penggunaan cosine\_similarity

```
In [13]: from sklearn.metrics.pairwise import cosine_similarity
... cosine_similarity(
... ... [mod.infer_vector(extract.words("tolong bantuan."))],
... ... [mod.infer_vector(extract.words("tolong bantuan. "))])
Out[13]: array([[0.91143525]], dtype=float32)
```

**Gambar 5.109** code dan hasil penggunaan cosine\_similarity

## 9. Cross Validation

pertama melakukan import data dari library KNeighborsClassifier, RandomForestClassifier, cross\_val\_score dan numpy yang digunakan untuk membuat data cross validasi dapat dilihat pada gambar

```
In [16]: from sklearn.neighbors import KNeighborsClassifier
... from sklearn.ensemble import RandomForestClassifier
... from sklearn.model_selection import cross_val_score
... import numpy as np
...
clf = KNeighborsClassifier(n_neighbors=9)
clf = RandomForestClassifier()
```

**Gambar 5.110** memasukan code import library

lalu selanjutnya membuat data variable scores yang akan memuat nilai cross\_val\_score dengan datanya diambil dari KNeighborsClassifier yang terdiri dari sentvecs, sentiments dan clf dan mengolahnya menggunakan numpy untuk menampilkan data pada gambar yang menghasilkan nilai akurasi sebesar 53 persen.

```
In [40]: scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
np.mean(scores), np.std(scores)
Out[40]: (0.5283333333333334, 0.086411794687223791)
```

**Gambar 5.111** perhitungan data KNeighborsClassifier dengan cross validasi

membuat data variable scores yang akan memuat nilai cross\_val\_score dengan datanya diambil dari RandomForestClassifier yang terdiri dari sentvecs, sentiments dan clfrf dan mengolahnya menggunakan numpy untuk menampilkan data pada gambar yang menghasilkan nilai akurasi sebesar 53 persen.

```
In [41]: scores2 = cross_val_score(c1Prf, sentences, sentiments, cv=5)
... np.mean(scores2), np.std(scores2)
C:\Users\Path\PC\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
... 10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Path\PC\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
... 10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Path\PC\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
... 10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Path\PC\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
... 10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Path\PC\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
... 10 in version 0.20 to 100 in 0.22.", FutureWarning)
Out[41]: (0.5119999999999999, 0.4064463596860459)
```

**Gambar 5.112** perhitungan data RandomForestClassifier dengan cross validasi

penggunaan make\_pipeline adalah untuk membuat data dari KNeighborsClassifier, RandomForestClassifier dan Vectorizer digabungkan untuk menghasilkan data nilai pada gambar menghasilkan nilai akurasi sebesar 74 persen.

```
In [42]: From sklearn.pipeline import make_pipeline
... from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer
... pipeline = make_pipeline(CountVectorizer(),
RandomForestClassifier())
... scores = cross_val_score(pipeline,sentences,sentiments, cv=5)
... np.mean(scores), np.std(scores)
C:\Users\Path\PC\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
... 10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Path\PC\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
... 10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Path\PC\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
... 10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Path\PC\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
... 10 in version 0.20 to 100 in 0.22.", FutureWarning)
Out[42]: (0.7416666666666667, 0.02345207879911713)
```

**Gambar 5.113** perhitungan data Cross Validasi untuk nilai keseluruhan dari KNeighborsClassifier, RandomForestClassifier dan Vectorizer

## 5.5 1174050 Dika Sukma Pradana

### 5.5.1 Teori

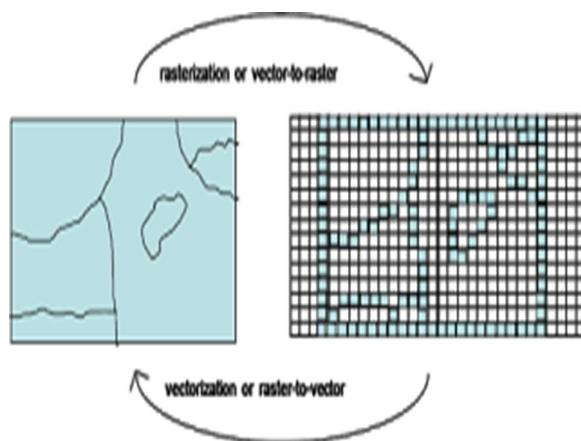
- Mengapa Kata-Kata Harus Di Lakukan Vektorisasi Dan Ilustrasi Gambar.

- Penjelasan:

Karenakan mesin hanya mampu membaca data dengan bentuk angka. Berdasarkan hal tersebut maka tentunya diperlukan vektorisasi kata atau bisa disebut dengan mengubah kata menjadi bentuk vektor agar mesin seolah-olah paham apa yang kita maksudkan dan dapat memproses aktifitas/perintah dengan benar. Selain alasan diatas, kata harus di vektorisasiuntuk mengetahui presentase kata yang sering muncul dalam setiap kalimatnya, yang berguna untuk menetukan kata kunci.

- Ilustrasi Gambar

- Mengapa Dimensi Dari Vektor Dataset Google Bisa Mencapai 300 Dan Ilustrasi Gambar.

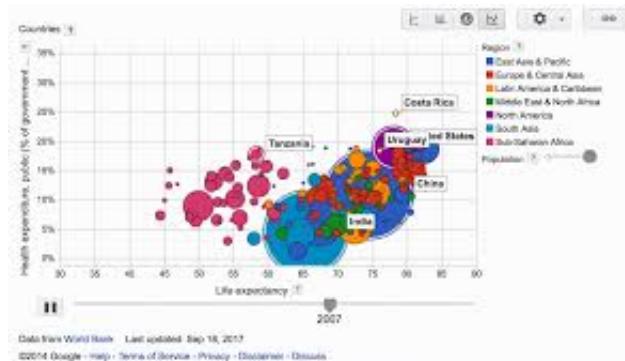


Gambar 5.114 Vektorisasi

- Penjelasan:

Karena pada masing-masing objek yang terdapat pada dataset akan memiliki identitasnya tersendiri. Apabila dicontohkan dengan penjelasan yang lebih rinci maka dilakukan perumpamaan sederhana. Misalnya untuk sebuah dataset google yang memiliki 3 buah objek yaitu berat, lebar, dan tinggi. Kemudian dari masing-masing objek tersebut dilakukan perbandingan antara berat dan lebar beserta berat dan tinggi. Hasil yang didapatkan akan memiliki presentasi yang berbeda sehingga dapat diartikan bahwa mesin dapat membedakan objek yang hampir serupa namun tak sama.

- Ilustrasi Gambar



Gambar 5.115 Dimensi Vektor Dataset

3. Konsep Vektorisasi Untuk Kata Dan Ilustrasi Gambar.

- Penjelasan:

Konsep untuk vektorisasi kata sebenarnya sama dengan ketika dilakukan input suatu kata pada mesin pencarian. Kemudian untuk hasilnya akan mengeluarkan ( berupa ) referensi mengenai kata tersebut. Jadi data kata tersebut didapatkan dari hasil pengolahan pada kalimat-kalimat sebelumnya yang telah diolah. Contoh sederhananya pada kalimat berikut ( Please click the alarm icon for more notifications about my channel ), pada kalimat tersebut terdapat konteks yakni channel, kata tersebut akan dijadikan data latih untuk mesin yang akan dipelajari dan diproses. Jadi ketika kita inputkan kta channel, maka mesin akan menampilkan keterkaitannya dengan kata tersebut sehingga akan lebih efisien dan lebih mudah.

- Ilustrasi Gambar

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

The resulting counts matrix will then be:

$$X = \begin{matrix} & \begin{matrix} I & like & enjoy & deep & learning & NLP & flying & . \end{matrix} \\ \begin{matrix} I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{matrix} & \left[ \begin{matrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{matrix} \right] \end{matrix}$$

**Gambar 5.116** Vektorisasi Untuk Kata

#### 4. Konsep Vektorisasi Untuk Dokumen Dan Ilustrasi Gambar.

- Penjelasan:

Untuk vektorisasi dokumen sebenarnya terbilang sama dengan konsep vektorisasi kata, yang membedakan hanya pada proses awalnya ( pada eksekusi awal ). Untuk vektorisasi dokumen ini, mesin akan

membaca semua kalimat yang terdapat pada dokumen tersebut, kemudian kalimat yang terdapat pada dokumen tersebut akan dipecah menjadi kata-kata. Seperti itulah konsep vektorisasi dokumen.

- Ilustrasi Gambar

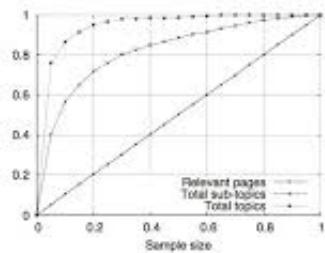


Figure 1: Impact of collection size on the fraction of relevant pages and subtopics with relevance.

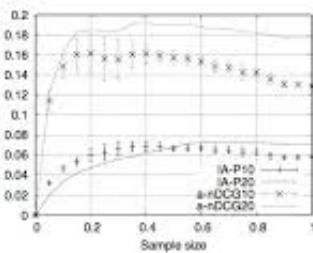


Figure 2: Impact of collection size on result diversity.

### Gambar 5.117 Vektorisasi Untuk Dokumen

## 5. Pengertian Mean Dan Standar Deviasi Beserta Ilustrasi Gambar.

- Pengertian Mean:

Mean adalah nilai rata-rata dari beberapa buah data. Nilai mean dapat ditentukan dengan membagi jumlah data dengan banyaknya data. Mean (rata-rata) merupakan suatu ukuran pemusatan data. Mean suatu data juga merupakan statistik karena mampu menggambarkan bahwa data tersebut berada pada kisaran mean data tersebut. Mean tidak dapat digunakan sebagai ukuran pemusatan untuk jenis data nominal dan ordinal.

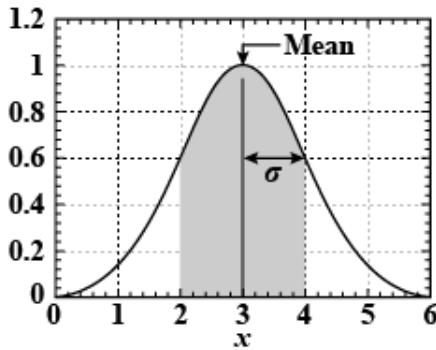
- Pengertian Standar Deviasi:

Standar Deviasi dan Varians Salah satu teknik statistik yg digunakan untuk menjelaskan homogenitas kelompok. Varians merupakan jumlah kuadrat semua deviasi nilai-nilai individual thd rata-rata kelompok. Sedangkan akar dari varians disebut dengan standar deviasi atau simpangan baku. Standar Deviasi dan Varians Simpangan baku merupakan variasi sebaran data. Semakin kecil nilai sebarannya berarti variasi nilai data makin sama Jika sebarannya bernilai 0, maka nilai semua datanya adalah sama. Semakin besar nilai sebarannya berarti data semakin bervariasi.

- Ilustrasi Gambar

## 6. Penjelasan Skip-gram Dan Ilustrasi Gambar

- Penjelasan:



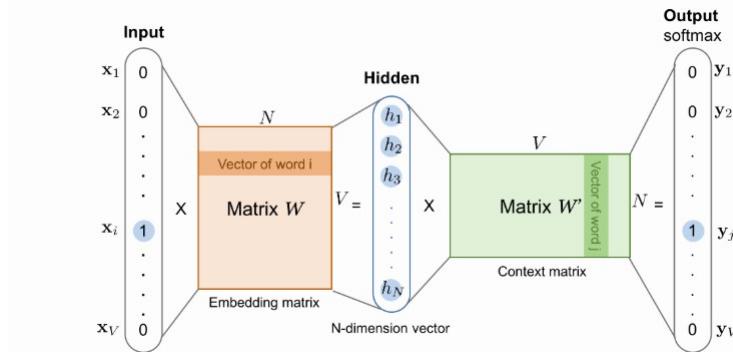
Gambar 5.118 Mean

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$$
$$s = \sqrt{\frac{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}{n(n-1)}}$$

Gambar 5.119 Standar Deviasi

Skip-Gram adalah kebalikannya, yaitu mencoba memprediksi vektor kata-kata yang ada di konteks diberikan vektor kata tertentu. Skip-Gram membuat sepasang kata target dan konteks sebagai sebuah instance sehingga Skip-Gram cenderung lebih baik ketika ukuran corpus sangat besar.

- Ilustrasi Gambar



**Gambar 5.120** Skip Gram

### 5.5.2 Praktek

1. Percobaan Google Dataset ( Perbandingan Dan Similarity ) Untuk Beberapa Data Berikut :

(a) Love

Penjelasan: Pada hasil gambar 'love' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah 0.10302734. Jika dibandingkan dengan gambar 'faith' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

(b) Faith

Penjelasan: Pada hasil gambar 'faith' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah 0.26367188. Jika dibandingkan dengan gambar 'fall' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

(c) Fall

Penjelasan: Pada hasil gambar 'fall' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah -0.04272461. Jika dibandingkan dengan gambar 'sick' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

(d) Sick

```
In [5]: gmodel['love']
Out[5]:
array([0.10302734, -0.15234375, 0.02587891, 0.16503906, -0.16503906,
 0.06689453, 0.29296875, -0.26367188, -0.140625 , 0.20117188,
 -0.02624512, -0.08203125, -0.02770996, -0.04394531, -0.23535156,
 0.16992188, 0.12890625, 0.15722656, 0.00756836, -0.06982422,
 -0.03857422, 0.07958984, 0.22949219, -0.14355469, 0.16796875,
 -0.03515625, 0.05517578, 0.10693359, 0.11181641, -0.16308594,
 -0.11181641, 0.13964844, 0.01556396, 0.12792969, 0.15429688,
 0.07714844, 0.26171875, 0.08642578, -0.02514648, 0.33398438,
 0.18652344, -0.20996094, 0.07080078, 0.02600098, -0.10644531,
 -0.10253906, 0.12384688, 0.04711914, 0.02209473, 0.05834961,
 -0.10986328, 0.14941406, -0.10693359, 0.01556396, 0.08984375,
 0.11230469, -0.04370117, -0.11376953, -0.0037384 , -0.01818848,
 0.24316406, 0.08447266, -0.07080078, 0.18066406, 0.03515625,
 -0.09667969, -0.21972656, -0.00328064, -0.03198242, 0.18457031,
 0.28515625, -0.0859375 , -0.11181641, 0.0213623 , -0.30664062,
 -0.09228516, -0.18945312, 0.01513672, 0.18554688, 0.34375 ,
 -0.31054688, 0.22558594, 0.08740234, -0.2265625 , -0.29492188,
 0.08251953, -0.38476562, 0.25390625, 0.26953125, 0.06298828,
 -0.00958252, 0.23632812, -0.17871094, -0.12451172, -0.17285156,
 -0.11767578, 0.19726562, -0.03466797, -0.10400391, -0.1640625 ,
 -0.19726562, 0.19824219, 0.09521484, 0.00561523, 0.12597656,
 0.00073624, -0.0402832 , -0.03063965, 0.01623535, -0.1640625 ,
 -0.22167969, 0.171875 , 0.12011719, -0.01965332, 0.4453125 ,
 0.06494141, 0.05932617, -0.1640625 , -0.01367188, 0.18945312,
 0.05566406, -0.05004883, -0.01422119, 0.15917969, 0.07421875,
```

Gambar 5.121 Google Dataset

```
In [6]: gmodel['faith']
Out[6]:
array([0.26367188, -0.04150391, 0.1953125 , 0.13476562, -0.14648438,
 0.11962891, 0.04345703, 0.10351562, 0.12207031, 0.13476562,
 0.06640625, 0.18945312, -0.16601562, 0.21679688, -0.27148438,
 0.3203125 , 0.10449219, 0.36132812, -0.1953125 , -0.18164062,
 0.15332031, -0.10839844, 0.10253906, -0.01367188, 0.23144531,
 -0.05957031, -0.22949219, -0.00604248, 0.26171875, 0.10302734,
 -0.1328125 , 0.21484375, 0.01135254, 0.02111816, 0.18554688,
 0.04125977, 0.12011719, 0.17480469, -0.22167969, -0.13476562,
 0.3125 , 0.06640625, -0.17675781, -0.01708984, -0.1640625 ,
 -0.02819824, 0.01257324, -0.09521484, -0.18066406, -0.140625 ,
 -0.02258301, 0.16308594, -0.13183594, -0.08007812, 0.130805938,
 0.27539062, -0.20605469, 0.10351562, -0.20214844, -0.1875 ,
 0.16992188, 0.13574219, 0.13769531, 0.16308594, -0.03881836,
 -0.11132812, 0.05688477, 0.12255859, 0.09814453, -0.04956055,
 -0.02331543, -0.04248047, -0.08203125, 0.16015625, 0.04150391,
 -0.16601562, -0.13671875, 0.09619141, 0.32617188, 0.08251953,
 -0.20800781, 0.04199219, 0.05834961, -0.27734375, 0.09130859,
 -0.17382812, -0.22460938, 0.03466797, 0.19824219, -0.08837891,
 0.18359375, 0.07324219, 0.1171875 , -0.33984375, 0.16796875,
 -0.13574219, -0.30078125, -0.00469971, 0.06005859, -0.29296875,
 0.15234375, 0.02966309, 0.33203125, 0.28320312, 0.09375 ,
 -0.20605469, -0.09982031, 0.05346668, 0.05834961, -0.03222656,
 -0.29296875, 0.25585938, 0.00430298, 0.140625 , 0.05810547,
 0.21582031, 0.0291748 , 0.02929688, 0.20019531, 0.34960938,
 0.10449219, -0.01940918, 0.04077148, 0.32226562, -0.1953125 ,
```

Gambar 5.122 Google Dataset

```
In [7]: gmodel['fall']
Out[7]:
array([-0.04272461, 0.10742188, -0.09277344, 0.16894531, -0.1328125 ,
 -0.10693359, 0.04321289, 0.01904297, 0.14648438, 0.15039062,
 -0.08691406, 0.04492188, 0.0145874 , 0.08691406, -0.19824219,
 -0.11035156, 0.01892529, -0.08300781, -0.0189209 , -0.1953125 ,
 -0.1015625 , 0.13671875, 0.09228516, -0.12109375, 0.12695312,
 0.03417969, 0.2109375 , 0.01977539, 0.125 , 0.01544189,
 -0.26953125, -0.0098877 , -0.07763672, -0.15527344, -0.03393555,
 0.04199219, -0.29882812, -0.18554688, 0.08496094, -0.02087402,
 0.13574219, -0.22558594, 0.33789062, -0.03564453, -0.10839844,
 -0.19335938, 0.0546875 , -0.04956055, 0.3671875 , -0.03295898,
 0.10205078, -0.15136719, -0.00445557, 0.04003906, 0.27539062,
 -0.06933594, 0.05834961, 0.01422119, -0.01397705, -0.05395508,
 -0.0255127 , 0.06298828, 0.07080078, -0.07617188, 0.06542969,
 -0.01672363, -0.04711914, 0.19628906, -0.08984375, 0.078125 ,
 0.2109375 , 0.0612793 , 0.08789062, 0.19628906, 0.11376953,
 0.06542969, 0.03125 , 0.12988281, 0.02270508, 0.14550781,
 -0.06225586, -0.37695312, -0.05737305, -0.06396484, 0.08984375,
 0.00448608, -0.14160156, -0.04541016, -0.0703125 , 0.06005859,
 0.26757812, 0.02001953, -0.12695312, -0.04882812, 0.18945312,
 -0.03466797, 0.04638672, 0.1484375 , 0.01708984, -0.08789062,
 -0.14941406, -0.02331543, -0.03955078, -0.10400391, -0.14160156,
 -0.18261719, -0.03076172, 0.04589844, -0.2890625 , -0.03540039,
 0.12890625, -0.10595703, 0.17578125, 0.06689453, 0.34960938,
 0.04296875, 0.09863281, -0.08056641, -0.06298828, 0.12255859,
```

Gambar 5.123 Google Dataset

Penjelasan: Pada hasil gambar 'sick' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah 1.82617188e-01. Jika dibandingkan dengan gambar 'clear' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

```
In [8]: gmodel['sick']
Out[8]:
array([1.82617188e-01, 1.49414062e-01, -4.05273438e-02, 1.64062500e-01,
 -2.59765625e-01, 3.22265625e-01, 1.73828125e-01, -1.47460938e-01,
 1.01074219e-01, 5.46875000e-02, 1.66992188e-01, -1.68945312e-01,
 2.24304199e-03, 9.66796875e-02, -1.66015625e-01, -1.12304688e-01,
 1.66015625e-01, 1.79687500e-01, 5.92041016e-03, 2.45117188e-01,
 8.74023438e-02, -2.56347656e-02, 3.41796875e-01, 4.98046875e-02,
 1.78710938e-01, -9.91821289e-04, 8.88671875e-02, -1.95312500e-01,
 1.81640625e-01, -2.65625000e-01, -1.45507812e-01, 1.00585938e-01,
 9.42382812e-02, -3.12500000e-02, 1.98974609e-02, -6.39648438e-02,
 1.18652344e-01, 1.23046875e-01, -6.03027344e-02, 4.68750000e-01,
 9.13085938e-02, -3.12500000e-01, 1.84570312e-01, -1.51367188e-01,
 5.78613281e-02, -1.04980469e-01, -1.68945312e-01, -8.00781250e-02,
 -2.01171875e-01, 1.06201172e-02, -1.29882812e-01, -1.25976562e-01,
 -5.56640625e-02, 3.14453125e-01, 5.61523438e-02, -1.20117188e-01,
 7.12890625e-02, 4.37011719e-02, 2.05078125e-01, 5.71289062e-02,
 8.44726562e-02, 2.15820312e-01, -1.26953125e-01, 8.78906250e-02,
 2.48046875e-01, -6.54296875e-02, -2.02636719e-02, 1.52343750e-01,
 -3.57421875e-01, 3.02124023e-03, -2.08007812e-01, -5.05371094e-02,
 2.81982422e-02, 1.73828125e-01, -2.08007812e-01, -5.93261719e-02,
 -6.49414062e-02, 3.63769531e-02, 1.91406250e-01, 2.77343750e-01,
 3.54003906e-02, 1.56250000e-01, -2.0387422e-02, 2.26562500e-01,
 -4.66308594e-02, -5.17578125e-02, -1.63085938e-01, 4.17480469e-02,
 2.01171875e-01, -2.01171875e-01, -1.50756836e-02, 2.61718750e-01,
 -1.10839844e-01, -4.21875000e-01, 2.22167969e-02, 1.46484375e-01,
```

Gambar 5.124 Google Dataset

## (e) Clear

Penjelasan: Pada hasil gambar 'clear' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah -2.44140625e-04. Jika dibandingkan dengan gambar 'shine' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

```
In [9]: gmodel['clear']
Out[9]:
array([-2.44140625e-04, -1.02050781e-01, -1.49414062e-01, -4.24804688e-02,
 -1.67968750e-01, -1.46484375e-01, 1.76757812e-01, 1.46484375e-01,
 2.26562500e-01, 9.76562500e-02, -2.67578125e-01, -1.29882812e-01,
 1.24511719e-01, 2.23632812e-01, -2.13867188e-01, 3.10058594e-02,
 2.00195312e-01, -4.76074219e-02, -6.83593750e-02, -1.21093750e-01,
 3.22265625e-02, 3.14453125e-01, -1.11816406e-01, 8.00781250e-02,
 -2.757878906e-02, -6.04248047e-03, -7.37304688e-02, -1.72851562e-01,
 9.66796875e-02, -4.91333008e-03, -1.78710938e-01, -1.40380859e-03,
 7.91015625e-02, 1.07910156e-01, -1.10351562e-01, -8.34960938e-02,
 1.98974609e-02, -3.14331055e-03, 1.30615234e-02, 3.34472656e-02,
 2.55859375e-01, -7.12890625e-02, 2.83203125e-01, -2.75398625e-01,
 -8.49609375e-02, -3.73535156e-02, -9.17968750e-02, -1.30859375e-01,
 3.49121094e-02, 7.32421875e-02, 2.17773438e-01, 5.00488281e-02,
 7.47070312e-02, -1.25000000e-01, -5.73730469e-02, -2.74658203e-02,
 -1.37329102e-02, -2.13867188e-01, -1.12792969e-01, -4.98046875e-02,
 -1.92382812e-01, 1.32812500e-01, -5.00488281e-02, -1.66015625e-01,
 -1.57470703e-02, -1.37695312e-01, 3.88183594e-02, 1.57226562e-01,
 -1.52343750e-01, -1.64794922e-02, -2.27539062e-01, 3.34472656e-02,
 1.55273438e-01, 1.65039062e-01, -1.66992188e-01, 3.14941406e-02,
 2.85156250e-01, 2.69531250e-01, 3.32031250e-02, 2.41210938e-01,
 1.39160156e-02, 5.12695312e-02, 9.76562500e-02, 3.14941406e-02,
 2.51464844e-02, -2.85156250e-01, -1.24023438e-01, -6.88476562e-02,
 -5.29785156e-02, 2.06054688e-01, -2.07031250e-01, -1.60156250e-01,
 -2.61230469e-02, -3.01513672e-02, 8.66699219e-03, -1.30859375e-01,
 3.88183594e-02, 8.60595703e-03, 5.31005859e-03, -6.05468750e-02,
```

**Gambar 5.125** Google Dataset

## (f) Shine

Penjelasan: Pada hasil gambar 'shine' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah -0.12402344. Jika dibandingkan dengan gambar 'bag' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

## (g) Bag

Penjelasan: Pada hasil gambar 'bag' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah -0.03515625. Jika dibandingkan dengan gambar 'car' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

## (h) Car

Penjelasan: Pada hasil gambar 'car' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah 0.13085938. Jika dibandingkan dengan gambar 'wash' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

## (i) Wash

```
In [10]: gmodel['shine']
Out[10]:
array([-0.12402344, 0.25976562, -0.15917969, -0.27734375, 0.30273438,
 0.09960938, 0.39257812, -0.22949219, -0.18359375, 0.3671875 ,
 -0.10302734, 0.13671875, 0.25390625, 0.07128906, 0.02539062,
 0.21777344, 0.24023438, 0.5234375 , 0.12304688, -0.19335938,
 -0.05883789, 0.0612793 , -0.01940918, 0.07617188, 0.05102539,
 0.20019531, 0.38085938, 0.00162506, -0.05029297, 0.14648438,
 -0.34765625, 0.02563477, -0.23925781, -0.04516602, -0.00479126,
 -0.24121094, -0.18945312, -0.15234375, -0.05493164, 0.01434326,
 0.390625 , -0.2109375 , 0.1484375 , -0.13183594, 0.24511719,
 -0.24023438, -0.36132812, -0.12792969, 0.10595703, 0.09912189,
 -0.0246582 , 0.32226562, 0.11376953, 0.18164062, 0.04931641,
 -0.10253906, -0.002283813, -0.29882812, -0.171875 , -0.18945312,
 -0.01367188, -0.20898438, -0.07861328, -0.0859375 , 0.05395508,
 -0.14257812, -0.140625 , 0.03027344, -0.14453125, 0.359375 ,
 0.16113281, 0.22265625, 0.265625 , -0.06347656, -0.02807617,
 0.04760742, 0.08837891, -0.04272461, 0.05908203, 0.07128906,
 0.01519775, -0.11621094, 0.07128906, 0.01403809, -0.10644531,
 0.08886719, 0.11523438, 0.09667969, -0.11083984, 0.16015625,
 0.3359375 , -0.1875 , 0.14550781, 0.00463867, 0.07617188,
 -0.09521484, 0.08447266, 0.20117188, 0.11230469, -0.33984375,
 -0.25390625, 0.05200195, 0.27539062, -0.08398438, -0.31054688,
 -0.22949219, 0.14941406, -0.1953125 , 0.08496094, -0.00753784,
 0.078125 , 0.05908203, 0.02355957, 0.06347656, 0.32617188,
 -0.08740234, 0.10058594, -0.11474609, -0.18164062, 0.13378906,
 0.11230469, -0.00080109, 0.08691406, 0.03808594, 0.0300293 ,
```

Gambar 5.126 Google Dataset

```
In [11]: gmodel['bag']
Out[11]:
array([-0.03515625, 0.15234375, -0.12402344, 0.13378906, -0.11328125,
 -0.0133667 , -0.16113281, 0.14648438, -0.06835938, 0.140625 ,
 -0.06005859, -0.3046875 , 0.20996094, -0.04345703, -0.2109375 ,
 -0.05957031, -0.05053711, 0.10253906, 0.19042969, -0.09423828,
 0.18847656, -0.07958984, -0.11035156, -0.07910156, 0.06347656,
 -0.15527344, -0.18945312, 0.11132812, 0.27539062, -0.06787109,
 0.01806641, 0.06689453, 0.2578125 , 0.0324707 , -0.24609375,
 -0.05541992, 0.01013184, 0.24121094, -0.21875 , 0.07568359,
 0.09814453, -0.16113281, 0.16503906, -0.09521484, -0.16601562,
 -0.41796875, 0.0300293 , 0.19433594, 0.2890625 , 0.12695312,
 -0.19242119, -0.05517578, 0.04296875, -0.10107422, 0.07324219,
 -0.13378906, 0.265625 , -0.00466919, 0.19628906, -0.10839844,
 0.14941406, 0.1484375 , 0.09619141, 0.21777344, -0.08544922,
 -0.02819824, 0.02539062, -0.03759766, 0.23242188, 0.19628906,
 0.27539062, 0.09130859, 0.23730469, 0.09033203, -0.28515625,
 0.05932617, 0.06591797, -0.01794434, -0.00055313, -0.1796875 ,
 0.05615234, -0.12207031, -0.09863281, -0.05786133, -0.09375 ,
 -0.30273438, -0.06396484, -0.00744629, -0.17871094, 0.08544922,
 -0.20410156, 0.33789062, 0.00228882, -0.39453125, -0.14453125,
 -0.328125 , -0.12695312, -0.08544922, 0.15234375, 0.03662109,
 -0.1484375 , 0.05566406, 0.02844238, 0.07519531, -0.21484375,
 -0.15722656, 0.3359375 , -0.04736328, -0.00405884, -0.19726562,
 0.27929688, 0.05566406, -0.10058594, -0.00811768, -0.20703125,
 0.03295898, -0.14550781, -0.15917969, 0.16503906, 0.234375 ,
```

Gambar 5.127 Google Dataset

```
In [12]: gmodel['car']
Out[12]:
array([-0.13085938, 0.00842285, 0.03344727, -0.05883789, 0.04003906,
 -0.14257812, 0.04931641, -0.16894531, 0.20898438, 0.11962891,
 0.18066406, -0.25 , -0.10400391, -0.10742188, -0.01879883,
 0.05200195, -0.02166675, 0.06445312, 0.14453125, -0.04541016,
 0.16113281, -0.01611328, -0.03088379, 0.08447266, 0.16210938,
 0.04467773, -0.15527344, 0.25390625, 0.33984375, 0.00756836,
 -0.25585938, -0.01733398, -0.03295898, 0.16308594, -0.12597656,
 -0.09912109, 0.16503906, 0.06884766, -0.18945312, 0.02832031,
 -0.0534668 , -0.03063965, 0.11083984, 0.24121094, -0.234375 ,
 0.12353516, -0.00294495, 0.1484375 , 0.33203125, 0.05249023,
 -0.20019531, 0.37695312, 0.12255859, 0.11425781, -0.17675781,
 0.10009766, 0.0030365 , 0.26757812, 0.20117188, 0.03710938,
 0.11083984, -0.09814453, -0.3125 , 0.03515625, 0.02832031,
 0.26171875, -0.08642578, -0.02258301, -0.05834961, -0.00787354,
 0.11767578, -0.04296875, -0.17285156, 0.04394531, -0.23046875,
 0.1640625 , -0.11474689, -0.06030273, 0.01196289, -0.24707031,
 0.32617188, -0.04492188, -0.11425781, 0.22851562, -0.01647949,
 -0.15039062, -0.13183594, 0.12597656, -0.17480469, 0.02209473,
 -0.1015625 , 0.00817871, 0.10791016, -0.24609375, -0.109375 ,
 -0.09375 , -0.01623535, -0.20214844, 0.23144531, -0.05444336,
 -0.05541992, -0.20898438, 0.26757812, 0.27929688, 0.17089844,
 -0.17578125, -0.02770996, -0.20410156, 0.02392578, 0.03125 ,
 -0.25390625, -0.125 , -0.05493164, -0.17382812, 0.28515625,
 -0.23242188, 0.0234375 , -0.20117188, -0.13476562, 0.26367188,
 0.00769043, 0.20507812, -0.01708984, -0.12988281, 0.04711914,
```

**Gambar 5.128** Google Dataset

Penjelasan: Pada hasil gambar 'wash' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah 9.46044922e-03. Jika dibandingkan dengan gambar 'motor' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

#### (j) Motor

Penjelasan: Pada hasil gambar 'motor' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah 5.73730469e-02. Jika dibandingkan dengan gambar 'cycle' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

#### (k) Cycle

Penjelasan: Pada hasil gambar 'cycle' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah 0.04541016. Jika dibandingkan dengan gambar 'love' dapat dikatakan bahwa kedua gambar tersebut tidak dapat dimasukkan pada kategori yang sama.

#### (l) Similarity

Penjelasan: Pada hasil gambar 'car' dapat dilihat bahwa nilai pada vektor baris pertamanya adalah 0.13085938. Jika dibandingkan dengan gambar 'love', 'sick', 'clear', 'motor', 'wash' dapat dikatakan bahwa semua gambar tersebut yang paling mendekati adalah 'sick'.

### 2. Penjelasan Dan Ilustrasi ExtractWords Dan PermuteSentences

- ExtractWords

```
In [13]: gmodel['wash']
Out[13]:
array([9.46044922e-03, 1.41601562e-01, -5.46875000e-02, 1.34765625e-01,
 -2.38281250e-01, 3.24218750e-01, -8.44726562e-02, -1.29882812e-01,
 1.07910156e-01, 2.53906250e-01, 1.13525391e-02, -1.66992188e-01,
 -2.79541016e-02, 2.08007812e-01, -4.27246094e-02, 1.05468750e-01,
 -7.42187500e-02, 3.04687500e-01, 2.11914062e-01, -8.88671875e-02,
 2.67578125e-01, 2.12890625e-01, 1.74560547e-02, 2.02941895e-03,
 6.29882812e-02, 1.62109375e-01, 1.93359375e-01, 2.17285156e-02,
 -2.67028809e-03, -9.13085938e-02, -2.38281250e-01, 2.23632812e-01,
 -8.00781250e-02, -3.80859375e-02, -1.00097656e-01, -1.39648438e-01,
 1.74804688e-01, 6.78710938e-02, 1.11328125e-01, 1.65039062e-01,
 -1.05468750e-01, 2.30712891e-02, 2.00195312e-01, -6.03027344e-02,
 -3.43750000e-01, -1.02050781e-01, -3.80859375e-01, -5.03571094e-02,
 5.07812500e-02, 1.45507812e-01, 2.81250000e-01, 7.03125000e-02,
 2.84423828e-02, -2.29492188e-01, -5.81054688e-02, 4.51660156e-02,
 -3.56445312e-02, 1.77734375e-01, 1.2207312e-01, 3.71093750e-02,
 -1.10839844e-01, 6.83593750e-02, -2.52685547e-02, -1.27929688e-01,
 4.21875000e-01, 5.32226562e-02, -3.92578125e-01, 1.74804688e-01,
 1.77001953e-02, -2.05078125e-02, 2.21679688e-01, 3.18359375e-01,
 1.08398438e-01, -4.30297852e-03, -2.45117188e-01, -2.08984375e-01,
 3.58886719e-02, 8.30078125e-02, 1.68945312e-01, 2.79541016e-02,
 1.04980469e-01, -3.47656250e-01, -5.20019531e-02, 2.24609375e-01,
 1.69677734e-02, 1.69921875e-01, -1.46484375e-01, 2.65625000e-01,
 2.17285156e-02, 1.12304688e-02, -1.14257812e-01, 7.22656250e-02,
 4.32128906e-02, 1.11694336e-02, 5.07354736e-04, -7.91015625e-02,
 -5.98144531e-02, -5.44433594e-02, 3.73046875e-01, 5.62500000e-01,
```

Gambar 5.129 Google Dataset

```
In [14]: gmodel['motor']
Out[14]:
array([5.73730469e-02, 1.50390625e-01, -4.61425781e-02, -1.32812500e-01,
 -2.59765625e-01, -1.77734375e-01, 3.68652344e-02, -4.37500000e-01,
 2.34375000e-02, 2.57812500e-01, 1.74804688e-01, 2.44140625e-02,
 -2.51953125e-01, -5.76171875e-02, 8.15429688e-02, 1.86767578e-02,
 -3.83300781e-02, 1.58203125e-01, -5.85937500e-02, 1.12304688e-01,
 1.56250000e-01, -4.24804688e-02, -1.32812500e-01, 2.11914062e-01,
 1.23046875e-01, 1.69921875e-01, -1.55273438e-01, 4.58984375e-01,
 3.02734375e-01, 1.53320312e-01, -1.69921875e-01, -1.01074219e-01,
 -3.26538086e-03, 2.28515625e-01, 8.98437500e-02, -7.12890625e-02,
 1.54296875e-01, -8.88671875e-02, -2.36328125e-01, 5.61523438e-03,
 -4.46777344e-02, -3.06640625e-01, 7.42187500e-02, 5.58593750e-01,
 -1.30859375e-01, 1.00585938e-01, -3.34472656e-02, 2.10937500e-01,
 3.10058594e-02, -6.50024414e-03, 6.347465625e-02, 4.02832031e-02,
 -2.78320312e-02, 1.07421875e-02, 1.47460938e-01, 2.80761719e-02,
 -1.50390625e-01, -1.37695312e-01, 9.96093750e-02, 1.28906250e-01,
 -3.34472656e-02, -1.08032227e-02, -2.14843750e-01, -9.52148438e-02,
 -6.39648438e-02, 7.51953125e-02, -3.06640625e-01, 2.17773438e-01,
 -2.21679688e-01, 2.33398438e-01, 5.05371094e-02, -3.37890625e-01,
 1.53320312e-01, -7.12890625e-02, -3.68652344e-02, 7.66601562e-02,
 -8.00781250e-02, -1.14257812e-01, -9.71679688e-02, -2.61718750e-01,
 3.84765625e-01, -1.87500000e-01, -1.10351562e-01, 1.00585938e-01,
 1.08398438e-01, 9.57031250e-02, -8.20312500e-02, 1.54296875e-01,
 -2.40234375e-01, 8.34960938e-02, 4.19921875e-02, -1.91650391e-02,
 9.71679688e-02, 2.52685547e-02, -5.46875000e-02, -5.88378906e-02,
 8.20312500e-02, -3.32031250e-01, 3.27148438e-02, 5.71289062e-02,
```

Gambar 5.130 Google Dataset

```
In [15]: gmodel['cycle']
Out[15]:
array([0.04541016, 0.21679688, -0.02709961, 0.12353516, -0.20703125,
 -0.1328125 , 0.26367188, -0.12890625, -0.125 , 0.15332031,
 -0.18261719, -0.15820312, -0.06176758, 0.21972656, -0.15820312,
 0.02563477, -0.07568359, -0.0625 , 0.04614258, -0.31054688,
 -0.13378906, -0.11669922, -0.3359375 , 0.078125 , 0.08447266,
 0.07226562, -0.06445312, 0.05517578, 0.14941406, 0.13671875,
 0.10302734, 0.02172852, -0.10693359, 0.02490234, -0.10644531,
 -0.05541992, -0.29492188, -0.40039062, 0.06347656, -0.08447266,
 0.17871894, 0.01165771, -0.01696777, 0.13671875, -0.1640625 ,
 0.11425781, 0.20800781, -0.06079102, -0.07275391, 0.15039062,
 0.18066406, -0.28515625, -0.04052734, 0.01806641, 0.00331116,
 0.00872803, 0.03564453, -0.29882812, 0.09960938, -0.1484375 ,
 -0.06787109, -0.05957031, -0.05517578, -0.19628906, 0.2265625 ,
 0.03173828, -0.07080078, 0.1484375 , -0.20214844, -0.03393555,
 0.09863281, -0.02038574, -0.08789062, -0.07226562, -0.09423828,
 -0.17089844, 0.1484375 , 0.10546875, 0.06445312, 0.01031494,
 0.02636719, -0.03686523, -0.125 , -0.06787109, 0.14257812,
 0.37109375, -0.15722656, 0.09326172, 0.34960938, -0.00091553,
 0.03613281, 0.16894531, -0.02856445, 0.10791016, -0.32421875,
 -0.14355469, 0.03173828, -0.07421875, 0.34179688, 0.140625 ,
 0.00443335, -0.12890625, -0.34960938, -0.02929688, -0.19628906,
 0.2578125 , -0.3671875 , 0.01483154, 0.20703125, 0.09667969,
 0.10351562, -0.31054688, 0.02844238, 0.10400391, 0.17773438,
 0.06689453, -0.1796875 , 0.02783203, 0.15625 , 0.02026367,
 0.0324707 , -0.13476562, 0.15527344, 0.11132812, -0.01055908,
```

Gambar 5.131 Google Dataset

```
In [130]: gmodel.similarity('car', 'love')
Out[130]: 0.0841785969147326

In [131]: gmodel.similarity('car', 'sick')
Out[131]: 0.11852219525661714

In [132]: gmodel.similarity('car', 'clear')
Out[132]: 0.02875396636498464

In [133]: gmodel.similarity('car', 'motor')
Out[133]: 0.481017283200157

In [134]: gmodel.similarity('car', 'wash')
Out[134]: 0.20769942357569984
```

Gambar 5.132 Google Dataset

```
def extract_word(sentence):
 ignore_words = ['a']
 words = re.sub("[^\\w]", " ", sentence).split()
 words_cleaned = [w.lower() for w in words if w not in ignore_words]
 return words_cleaned

import random
def __init__(self, model, color, company, speed_limit):
 self.color = color
 self.company = company
 self.speed_limit = speed_limit
 self.model = model

random.getstate()
```

Gambar 5.133 Extract Word dan PermuteSentence

Penjelasan: Pada kalimat 'This isn't really a sentence' yang akan dipisahkan perkata. Dimana library re dan library string di import terlebih dahulu. Lalu variable out mendefinisikan X untuk mengembalikan string pada objek line yang telah di split. Kemudian, X dikembalikan berdasarkan jumlah kata.

```
In [71]: import re, string
.....
....: def extract_words(line):
....: out = (x.lower() for x in line.split())
....: out = (re.sub('[%s]' % re.escape(string.punctuation), '',
x) for x in out)
....:
....: return (x for x in out if len(x) > 0)
....:
....:
....: # Test the function
....: list(extract_words("Sudah malam ikan bobok"))
Out[71]: ['sudah', 'malam', 'ikan', 'bobok']
```

**Gambar 5.134 ExtractWord**

- **PermuteSentences**

Penjelasan: Digunakan untuk melakukan pengocokan atau acak pada text yang diijinkan.

```
5b82104210,
4139547478,
1955021215,
1434985917,
2522382814,
2673039090,
2483958158,
862772382,
1482425492,
605020739,
180818081,
3731486337,
1343592513,
1717784145,
1200984563,
2681937281,
3116535912,
3956622318,
2360166062,
48175308,
1167088233,
3784066035,
4043133115,
3230778418,
2266214664,
2686485136,
624),
None)
```

**Gambar 5.135 PermuteSentences**

### 3. Library Gensim TaggedDocument Dan Doc2Vec

- Contoh atau Ilustrasi Gambar:

```
In [44]: from gensim.models.doc2vec import TaggedDocument
...: from gensim.models import Doc2Vec
```

**Gambar 5.136** Tagged Document dan Doc2Vec

Penjelasan:

Import library gensim dan meng-import modul Tagged Document dan Doc2Vec. Tagged Document adalah dokumen yang 'memisahkan informasi dan struktur dari presentasi' dengan menggunakan tag. Fungsi Doc2Vec berisi alpha dan min\_alpha parameter, tetapi itu berarti bahwa tingkat pembelajaran meluruh selama satu periode dari alpha untuk min\_alpha.

### 4. Menambahkan Data Training (Melatih Modul Doc2Vec)

- Contoh atau Ilustrasi Gambar:

```
In [8]: for dirname in ["train/pos", "train/neg", "train/unsup", "test/pos", "test/neg"]:
...: for fname in sorted(os.listdir("aclImdb/" + dirname)):
...: if fname[-4:] == '.txt':
...: with open("aclImdb/" + dirname + "/" + fname, encoding='UTF-8') as f:
...: sent = f.read()
...: words = extract_words(sent)
...: unsup_sentences.append(TaggedDocument(words, [dirname + "/" + fname]))
```

**Gambar 5.137** Model 1

```
In [9]: for dirname in ["review_polarity/txt_sentoken/pos", "review_polarity/txt_sentoken/neg"]:
...: for fname in sorted(os.listdir(dirname)):
...: if fname[-4:] == '.txt':
...: with open(dirname + "/" + fname, encoding='UTF-8') as f:
...: for i, sent in enumerate(f):
...: words = extract_words(sent)
...: unsup_sentences.append(TaggedDocument(words, ["%s/%s-%d" % (dirname, fname, i)]))
```

**Gambar 5.138** Model 2

Penjelasan:

Membaca direktori name dari data yang ada di dalam kurung, terdapat ada 3 data.

### 5. Pengocokan Dan Pembersihan Data.

- Contoh atau Ilustrasi Gambar:

```
In [11]: with open("stanfordSentimentTreebank/original_rt_snippets.txt",
encoding='UTF-8') as f:
...: for i, line in enumerate(f):
...: words = extract_words(sent)
...: unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))
```

In [12]: Traceback (most recent call last):

Gambar 5.139 Model 3

```
In [72]: import re
...: def extract_words(sent):
...: sent = sent.lower()
...: sent = re.sub(r'<[^>]+>', ' ', sent)
...: sent = re.sub(r'(\w)\\'(\w)', ' ', sent)
...: sent = re.sub(r'\W', ' ', sent)
...: sent = re.sub(r'\s+', ' ', sent)
...: return sent.split()

In [73]: import random
...: class PermuteSentences(object):
...: def __init__(self, sents):
...: self.sents=sents
...:
...: def __iter__(self):
...: shuffled = list(self.sents)
...: random.shuffle(shuffled)
...: for sent in shuffled:
...: yield sent
```

Gambar 5.140 Pengocokan dan Pembersihan Data

```
In [74]: len(unsup_sentences)
Out[74]: 31815

In [75]: unsup_sentences[0:1]
Out[75]: [TaggedDocument(words=['after', 'watching', '_a_night_at_the_roxbury_',
'you', 'll', 'be', 'left', 'with', 'exactly', 'the', 'same'], tags=['rt-0'])]

In [76]: mute=PermuteSentences(unsup_sentences)

In [77]: model = Doc2Vec(mute, dm=0, hs=1, vector_size=52)
```

Gambar 5.141 Pengocokan dan Pembersihan Data

Penjelasan:

Mengimport Library Re. Kemudian membuat fungsi untuk menghapus tag html dan perkocakan. Dimana di dalam variabel ini ada kodinan untuk menghapus tag html yaitu petik satu, tanda baca dan spasi yang berurutan. Melakukan pengacakan model terhadap data unsupervised learning. Dan kemudian baru membuat modelnya setelah dilakukan pengacakan terhadap yang pertama tadi.

## 6. Mengapa Model Harus Di Save Dan Temporari Training Harus Dihapus

- Contoh atau Ilustrasi Gambar:

```
In [56]: model.delete_temporary_training_data(keep_inference=True)
```

**Gambar 5.142** Model Disave dan Temporari Train Hapus

```
In [57]: model.save('haci.d2v')
```

**Gambar 5.143** Model Disave dan Temporari Train Hapus

Penjelasan:

Untuk mencegah ram agar tidak lemot atau lambat. Sedangkan kenapa temporari training harus dihapus mengosongkan memori agar sedikit lega atau tidak lemot.

## 7. Infercode

- Contoh atau Ilustrasi Gambar:

```
In [59]: model.infer_vector(extract_words("ahelah cape"))
Out[59]:
array([0.00220989, 0.00535457, 0.00817098, -0.0019864 , -0.00143027,
 -0.00163056, 0.00238419, -0.00621389, -0.0055491 , 0.0092905 ,
 0.00689598, 0.0016727 , -0.00941525, 0.00544497, -0.00227619,
 -0.00714622, -0.00829009, -0.00784442, -0.00561465, -0.00818796,
 -0.00663066, 0.00073265, -0.00662555, 0.00189114, -0.00272863,
 -0.00585739, 0.00160669, -0.00118292, -0.00352464, 0.00602394,
 0.00848681, 0.00831539, 0.00395764, 0.00161309, 0.0066474 ,
 -0.00597237, 0.00840245, -0.00063893, -0.00362349, -0.0005215 ,
 -0.0076603 , -0.00738083, 0.00465537, -0.00909361, -0.0062611 ,
 0.00567978, -0.00528364, 0.00788782, -0.00389069, 0.00556741,
 -0.00092625, -0.00557183], dtype=float32)
```

**Gambar 5.144** Infercode

Penjelasan:

Untuk menyimpulkan vektor yang berhubungan dengan vektor dokument baru dan output tersebut merupakan sebuah array.

## 8. Cosinesimilarity

```
In [60]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity(
...: [model.infer_vector(extract_words("nugas terus, cape deh")),
...: [model.infer_vector(extract_words("HAHAHAHA"))])
Out[60]: array([[0.24101943]], dtype=float32)
```

**Gambar 5.145** Cosinesimilarity

```
In [61]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity(
...: [model.infer_vector(extract_words("capedeh")),
...: [model.infer_vector(extract_words("capedeh"))])
Out[61]: array([[0.9999999]], dtype=float32)
```

**Gambar 5.146** Cosinesimilarity

```
In [64]: sentvecs = []
...: sentences = []
...: sentiments = []
...: for fname in ["yelp", "amazon_cells", "imdb"]:
...: with open("sentiment labelled sentences/%s_labelled.txt" % fname,
encoding='UTF-8') as f:
...: for i, line in enumerate(f):
...: line_split = line.strip().split('\t')
...: sentences.append(line_split[0])
...: words = extract_words(line_split[0])
...: sentvecs.append(model.infer_vector(words, steps=10))
...: sentiments.append(int(line_split[1]))
```

**Gambar 5.147** Cosinesimilarity

```
In [65]: combined = list(zip(sentences, sentvecs, sentiments))
...: random.shuffle(combined)
...: sentences, sentvecs, sentiments = zip(*combined)
```

**Gambar 5.148** Cosinesimilarity

- Contoh atau Ilustrasi Gambar:

Penjelasan:

Cosine Similarity dapat diimplementasikan untuk menghitung nilai kemiripan antar kalimat dan menjadi salah satu teknik untuk mengukur kemiripan teks yang popular.

## 9. Praktek Score Dari Cross Validation

- Contoh atau Ilustrasi Gambar:

```
In [66]: from sklearn.neighbors import KNeighborsClassifier
...: from sklearn.ensemble import RandomForestClassifier
...: from sklearn.model_selection import cross_val_score
...: import numpy as np
...:
...: clf = KNeighborsClassifier(n_neighbors=9)
...: clfrf = RandomForestClassifier()
```

**Gambar 5.149** Score Cross Validation

```
In [67]: scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
...: np.mean(scores), np.std(scores)
Out[67]: (0.4953333333333333, 0.013800161029656305)
```

**Gambar 5.150** Score Cross Validation

```
In [68]: scores = cross_val_score(clfrf, sentvecs, sentiments, cv=5)
...: np.mean(scores), np.std(scores)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
"10 in version 0.20 to 100 in 0.22.", FutureWarning)
Out[68]: (0.5106666666666667, 0.0212811862660165)
```

**Gambar 5.151** Score Cross Validation

Penjelasan:

Hasil dari praktek tersebut menunjukan untuk menghitung memprediksi dan mengetahui keakuratan dari suatu nilai.

```
In [69]: from sklearn.pipeline import make_pipeline
...: ...: from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer
...: pipeline = make_pipeline(CountVectorizer(), TfidfTransformer(),
RandomForestClassifier())
...:
...: scores = cross_val_score(pipeline, sentences, sentiments, cv=5)
...: np.mean(scores), np.std(scores)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
 "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
 "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
 "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
 "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246:
FutureWarning: The default value of n_estimators will change from 10 in version 0.20
to 100 in 0.22.
 "10 in version 0.20 to 100 in 0.22.", FutureWarning)
Out[69]: (0.747, 0.015362291495737231)
```

**Gambar 5.152 Score Cross Validation**

- **PENANGANAN ERROR**

- Contoh atau Ilustrasi Gambar:

```
File "C:\ProgramData\Anaconda3\lib\site-packages\smart_open\smart_open_lib.py",
line 181, in smart_open
 fobj = _shortcut_open(uri, mode, **kw)

 File "C:\ProgramData\Anaconda3\lib\site-packages\smart_open\smart_open_lib.py",
line 301, in _shortcut_open
 return open(parsed_uri.uri_path, mode, buffering=buffering, **open_kwargs)

FileNotFoundException: [Errno 2] No such file or directory: 'GoogleNews-vectors-
negative300.bin'
```

**Gambar 5.153 Error**

Solusi:

Pastikan dataset berada dalam satu folder.

## 5.6 1174057 Alit Fajar Kurniawan

### 5.6.1 Teori

1. Jelaskan kenapa kata-kata harus di lakukan vektorisasi

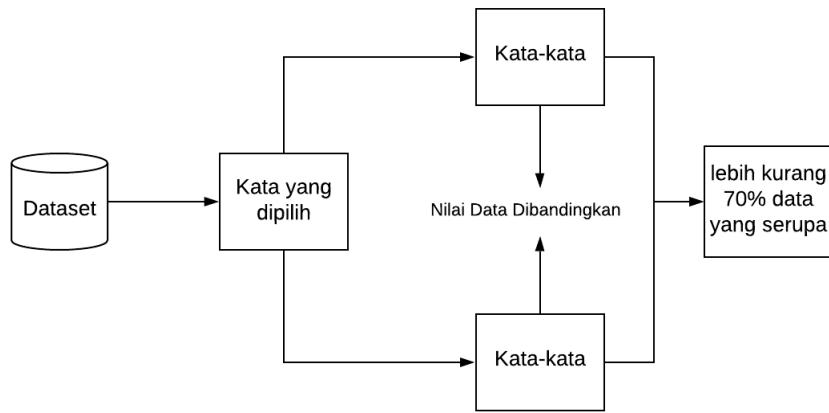
Alasan mengapa kata-kata harus dilakukan vektorisasi terlebih dahulu yaitu dikarenakan mesin hanya mampu membaca data dengan bentuk angka. Berdasarkan hal tersebut maka tentunya diperlukan vektorisasi kata atau bisa disebut dengan mengubah kata menjadi bentuk vektor agar mesin seolah-olah paham apa yang kita maksudkan dan dapat memproses aktitas/perintah dengan benar. Kata juga harus di vektorisasi untuk mengetahui presentase kata yang sering muncul dalam setiap kalimatnya, yang berguna untuk menetukan kata kunci.



**Gambar 5.154** Vektorisasi Kata

2. Jelaskan mengapa dimensi dari vektor dataset google bisa sampai 300

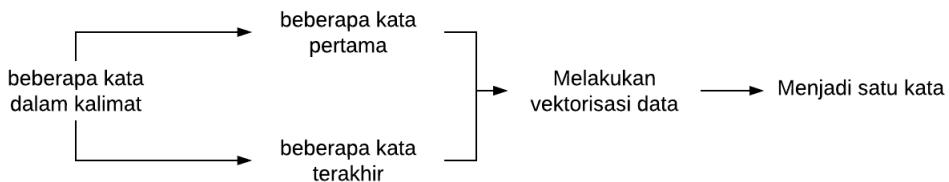
Dimensi dari Vektor Dataset Google Bisa Mencapai 300 itu dikarenakan pada masing-masing objek yang terdapat pada dataset akan memiliki identitasnya tersendiri, selain itu juga untuk nilai dalam vektor 300 dimensi yang terkait dalam sebuah kata "dioptimalkan" dalam berbagai hal untuk menangkap aspek yang berbeda dari makna dan penggunaan kata itu. Apabila dicontohkan dengan penjelasan yang lebih rinci maka dilakukan perumpamaan sederhana. Misalnya untuk sebuah dataset google yang memiliki 3 buah objek yaitu berat, lebar, dan tinggi. Kemudian dari masing-masing objek tersebut dilakukan perbandingan antara berat dan lebar beserta berat dan tinggi. Hasil yang didapatkan akan memiliki presentasi yang berbeda sehingga dapat diartikan bahwa mesin dapat membedakan objek yang hampir serupa namun tak sama.



**Gambar 5.155** Dataset Google

### 3. Jelaskan konsep vektorisasi untuk kata

Konsep untuk vektorisasi kata sebenarnya sama dengan ketika dilakukan input suatu kata pada mesin pencarian. Kemudian untuk hasilnya akan mengeluarkan ( berupa ) referensi mengenai kata tersebut. Jadi data kata tersebut didapatkan dari hasil pengolahan pada kalimat-kalimat sebelumnya yang telah diolah. misalnya pada kalimat berikut (Belajar Kecerdasan buatan Poltekpos), pada kalimat tersebut terdapat konteks yakni poltekpos, kata tersebut akan dijadikan data latih untuk mesin yang akan dipelajari dan diproses. Jadi ketika kita inputkan kata poltekpos, maka mesin akan menampilkan keterkaitannya dengan kata tersebut sehingga akan lebih esien dan lebih mudah.

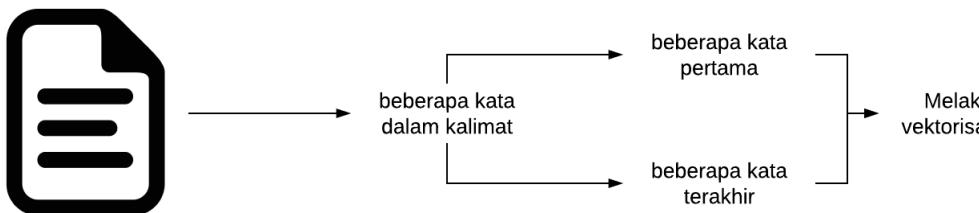


**Gambar 5.156** konsep vektorisasi

### 4. Jelaskan konsep vektorisasi untuk dokumen

Untuk vektorisasi dokumen sebenarnya terbilang sama dengan konsep vektorisasi kata, yang membedakan hanya pada proses awalnya. Untuk vektorisasi dokumen ini, mesin akan membaca semua kalimat yang ter-

dapat pada dokumen tersebut, kemudian kalimat yang terdapat pada dokumen tersebut akan di pecah menjadi kata-kata



**Gambar 5.157** konsep vektorisasi Dokumen

5. Jelaskan apa mean dan standar deviasi

Mean merupakan nilai rata-rata dari suatu data. Mean sendiri dapat dicari dengan cara membagi jumlah data dengan banyak data sehingga diperolehlah nilai rata-rata dari suatu data yang dicari sedangkan standar deviasi sendiri merupakan sebuah teknik statistik yang digunakan dalam menjelaskan homogenitas kelompok ataupun dapat diartikan dengan nilai statistik dimana dimanfaatkan untuk menentukan bagaimana sebaran data dalam sampel, serta seberapa dekat titik data individu ke mean atau rata-rata nilai sampel yang ada.

| Population Mean                         | Sample Mean                            |
|-----------------------------------------|----------------------------------------|
| $\mu = \frac{\sum_{i=1}^N x_i}{N}$      | $\bar{X} = \frac{\sum_{i=1}^n x_i}{n}$ |
| $N$ = number of items in the population | $n$ = number of items in the sample    |

**Gambar 5.158** mean

**Rumus Deviasi Standar**

$$SD = \sqrt{\frac{\sum x^2}{N}}$$

$$SD = \sqrt{\frac{\sum fx^2}{N}}$$

Keterangan:

$SD$  = Standar Deviasi

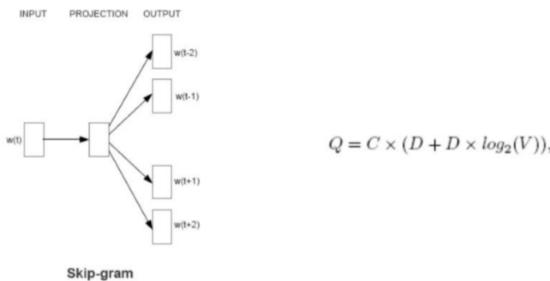
$\sum x^2$  = Jumlah semua deviasi setelah dikuadratkan

- (a) Rumus untuk frekuensi tunggal atau satu
- (b) Rumus untuk frekuensi lebih dari satu

**Gambar 5.159** standar deviasi

## 6. Jelaskan apa itu skip-gram

Sebuah teknik yang digunakan di area speech processing, dimana n-gram yang dibentuk kemudian ditambahkan juga dengan tindakan “skip” pada token-tokennya. Untuk membentuk k-skip-n-grams, ada dua nilai yang harus dikenal, dimana kedua nilai tersebut yaitu k (jumlah kata yang di-skip) dan n (banyak kata dalam n-gram, e.g. bigram (2-gram), trigram (3-gram), dll).



**Gambar 5.160** skip-gram

## 5.6.2 Praktek

1. mencoba dataset google dan penjelasan vektor dari kata love, faith, fall, sick, clear, shine, bag, car, wash, motor, dan cycle.
  - berikut merupakan code import gensim digunakan untuk membuat data model atau rangcangan data yang akan dibuat. selanjutnya dibuat variabel gmodel yang berisi data vektor negative. selanjutnya data tersebut di load agar data tersebut dapat ditampilkan dan diolah. data tersebut diambil dari GoogleNews-vectors-negative300.bin.

```
In [42]: import gensim, logging
In [43]: logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
In [44]: gmodel = gensim.models.KeyedVectors.load_word2vec_format('D:/Data Alit/Kuliah/SEMESTER VI/KB38/src/
1174057/chapters/GoogleNews-vectors-negative300.bin', binary=True)
```

**Gambar 5.161** praktek 1

- berikut merupakan hasil lpengolahan kata love pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [46]: gmodel['love']
Out[46]:
array([-0.15234375, 0.02587891, 0.16503906, -0.16503906,
 0.06689453, 0.29296875, -0.26367188, -0.140625 , 0.20171788,
 -0.02624512, -0.08203125, -0.02779996, -0.04394531, -0.23535156,
 0.16992188, 0.12896255, 0.15722656, 0.00756836, -0.06982422,
 -0.03857422, 0.07958984, 0.22949219, -0.14355469, 0.16796875,
 -0.11131641, 0.13964844, 0.01556396, 0.17792969, 0.15429988,
 0.07714844, 0.26171785, 0.08642578, -0.02514648, 0.33398438,
 0.18652344, -0.20996994, 0.07088087, 0.02600908, -0.10644531,
 -0.10253906, 0.12804688, 0.04171194, 0.02289473, 0.05834961,
 -0.10998632, 0.14941406, -0.10693399, 0.01556396, 0.08984375,
 -0.11208632, 0.13787888, 0.08595173, -0.03373772, -0.08164864,
 0.24316486, 0.80484266, -0.07088087, 0.18646406, 0.39515625,
 -0.89667989, -0.21972656, -0.00328944, -0.03198242, 0.18457931,
 0.28515625, -0.8859375, -0.11161641, 0.0213823 , -0.30646462,
 -0.89228516, -0.18945312, 0.01513672, 0.18546468, 0.34375 ,
 -0.31054688, 0.22558594, 0.08748234, -0.2265625 , -0.29492188,
 0.08251953, -0.38476562, 0.25396345, 0.26953125, 0.06238828,
 -0.13798625, 0.15722656, 0.03323262, -0.18164662, -0.11885156 ,
 -0.11757767, 0.15722656, -0.03467377, -0.10408938, 0.1646145 ,
 -0.19726562, 0.19824219, 0.09521484, 0.00561523, 0.12597656 ,
 0.00073624, -0.04082832, -0.03063965, 0.01623535, -0.1640625 ,
 -0.22167969, 0.171785 , 0.12011719, -0.01965332, 0.4453125 ,
 0.06494141, 0.05932617, -0.1640625 , -0.01367188, 0.18945312 ,
 0.00000496, -0.05984603, -0.01422119, 0.15917969, 0.07421875 ,
 -0.31686225, -0.05375689, -0.02355957, -0.01111616, 0.05083789 ,
 -0.140625 , -0.13183594, -0.12795269, -0.13000547, 0.05083789 ,
 -0.00055695, 0.05761719, -0.00447266, 0.16992188, 0.13671875,
```

**Gambar 5.162** love

- berikut merupakan hasil lpengolahan kata faith pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [47]: gmodel['faith']
Out[47]:
array([0.26367188, -0.04150391, 0.1353125 , 0.12476562, -0.14648438,
 0.11963289, 0.09445703, 0.10351562, 0.12207031, 0.13476562,
 0.06646025, 0.18945312, -0.16601562, 0.21679688, -0.27148438,
 0.3203125 , 0.10449219, 0.36132812, -0.1953125 , -0.18164662,
 0.15332031, -0.10839844, 0.10253998, -0.01367188, 0.23144531,
 -0.05970391, 0.22949219, -0.00604248, 0.26171875, 0.10302734,
 -0.1328125 , 0.02484375, 0.0152525 , 0.02111616, 0.18554688 ,
 -0.01259577, 0.12038594, -0.04809569, -0.22779894, 0.052106562 ,
 0.2126 , 0.06546025, -0.17675781, -0.01708984, -0.1546145 ,
 -0.02819824, 0.01257324, -0.09521484, -0.18066406, -0.140625 ,
 -0.022553801, 0.16308594, -0.13183594, -0.08007812, 0.130885938 ,
 0.27539662, -0.20605469, 0.10351562, -0.20214844, -0.1875 ,
 0.16992188, 0.13574219, 0.13769531, 0.16308594, -0.08381836 ,
 0.08381836, 0.09862031, 0.08381836, 0.08381835, -0.08381835 ,
 0.02321548 , -0.04248047, -0.00820845, -0.18066406, -0.04158931 ,
 -0.16601562 , 0.13671785, 0.09613141, 0.32617188, 0.08251953 ,
 -0.28080781, 0.04199219, 0.05834961, -0.27734375, 0.09130859 ,
 -0.17382812, 0.02466939, 0.03466797, 0.19824219, -0.08837891 ,
 0.18359375 , 0.07324219, 0.1171875 , -0.33984375, 0.16796875 ,
 -0.13574219 , 0.08078125, -0.00649971, 0.06005859, -0.29296875 ,
 0.15220475 , 0.08824065, 0.02346601, 0.09334961, -0.0322356 ,
 -0.02054688 , 0.09862031, 0.02346601, 0.09334961, -0.0322356 ,
 0.29236875 , 0.25565938, 0.09038298, 0.149625 , 0.05818547 ,
 0.21582031, 0.02931748, 0.02929688, 0.20019531, 0.34960938 ,
 0.10449219, -0.01940918, 0.04077148, 0.32220562 , -0.1953125 ,
 -0.0568477 , 0.10498047, -0.04785156, -0.15136719, -0.07714844 ,
 -0.45898438, 0.29492188, -0.328125 , 0.20996094, 0.38671875 ,
 0.0039978 , 0.07373847, -0.01228783, 0.22460938, 0.14550781 ,
```

**Gambar 5.163** faith

- berikut merupakan hasil lpengolahan kata fall pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [48]: gmodel['fall']
Out[48]:
array([-0.04272461, 0.10742188, -0.09277344, 0.16894531, -0.1328125 ,
 -0.10693359, 0.04321289, 0.01904297, 0.14648438, 0.15039062,
 -0.08202159, 0.08202159, 0.08202159, 0.08202159, 0.08202159,
 -0.11025156, 0.01092529, -0.08208078, -0.01932948, -0.1593125 ,
 -0.1015625 , 0.13671375, 0.09228516, -0.12108375, 0.12695312,
 0.03417969, 0.210937 , 0.01977539, 0.125 , 0.01544189,
 -0.26953125, -0.00988777, -0.07763672, -0.15527344, -0.03393555,
 0.04199219, -0.29882812, -0.18554688, 0.08496094, -0.02887492,
 0.11145119, -0.22558594, -0.37370862, -0.04445453, -0.01965444,
 -0.19315978, -0.05695609, 0.03671056, -0.02958989,
 0.10205078, -0.15136719, -0.00445557, 0.040023906, 0.27539062,
 -0.06931594, 0.05834961, 0.01422119, -0.01397705, -0.05395508,
 -0.0255127 , 0.06298828, 0.07980878, -0.07617188, 0.06542969,
 -0.01672363, -0.04711914, 0.19628986, -0.00894375, 0.078125 ,
 0.2109375 , 0.061279 , 0.08789862, 0.19628986, 0.11376953,
 0.04625569, 0.08625569, 0.12500281, 0.08394483, 0.08393531,
 -0.06225569, -0.07695312, -0.06394483, 0.06394483, -0.00543775,
 0.00448688, -0.14160156, -0.045434016, -0.0793125 , 0.06005859,
 0.26757812, 0.02001953, -0.12695312, -0.04882812, 0.18945312,
 -0.03466797, 0.04638672, 0.1484375 , 0.01708984, -0.08789062,
 -0.14941406, -0.02331543, -0.03955078, -0.10400391, -0.14160156,
 -0.18261719, -0.03076172, 0.04589844, -0.2898625 , -0.03540039,
 0.12490225, -0.08625569, 0.08625569, 0.08625569, 0.08625569,
 -0.04265075, 0.08963283, -0.00895641, 0.08298828, -0.1235509 ,
 0.02343478, -0.06494143, 0.09657969, -0.04589844, 0.04955085,
 0.00807812, -0.08482178, -0.1640625 , -0.03271484, 0.0703125 ,
 -0.07958984, -0.12898625, -0.01879883, -0.17773438, 0.01293945,
 -0.20019531, 0.08886719, -0.18867656, -0.23828125, -0.02578789,
```

Gambar 5.164 fall

- berikut merupakan hasil pengolahan kata sick pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [49]: gmodel['sick']
Out[49]:
array([1.82617188e-01, 1.49414062e-01, -4.05273438e-02, 1.64062500e-01,
 -2.59756250e-01, 3.22265625e-01, 1.73828125e-01, -1.47460938e-01,
 1.00874219e-01, 5.46875000e-01, 1.66992138e-01, -1.68945312e-01,
 2.43947312e-01, 1.25000000e-01, 1.25000000e-01, -1.25000000e-01,
 1.66015625e-01, 1.79657500e-01, 5.92041056e-02, 2.45117188e-01,
 8.74023438e-02, -2.56347656e-02, 3.41796875e-01, 4.98046875e-02,
 1.78710938e-01, -9.91821289e-04, 8.88671875e-02, -1.95312500e-01,
 1.81640625e-01, -2.65625000e-01, -1.45507812e-01, 1.00585938e-01,
 9.43282612e-02, -3.12500000e-02, 1.98974699e-02, -6.39648438e-02,
 1.18097500e-01, 1.25000000e-01, -6.17407344e-01, -4.07750000e-01,
 1.30097500e-02, -1.12500000e-01, -6.85760000e-01, -5.15120000e-01,
 5.78613381e-02, -1.04989469e-01, -1.68945312e-01, -8.00781250e-02,
 -2.01171875e-01, 1.06201172e-02, -1.29882812e-01, -1.25975562e-01,
 -5.56640625e-02, 3.14453125e-01, 5.61523438e-02, -1.20117188e-01,
 7.12890625e-02, 4.37011719e-01, 2.05078125e-01, 5.71289062e-02,
 8.44726562e-02, 2.15821126e-01, -1.26953125e-01, 8.78906250e-02,
 2.40460625e-01, -6.12500000e-01, 1.25000000e-01, -1.25000000e-01,
 -3.37421056e-01, -0.02124049e-03, -2.08007812e-01, -5.05371094e-02,
 2.81982422e-02, 1.73828125e-01, -2.08007812e-01, -5.93261719e-02,
 -6.49414062e-02, 3.63769531e-01, 1.91406250e-01, 2.77342750e-01,
 3.540083906e-02, 1.56250000e-01, -2.03857422e-02, 2.26562500e-01,
 -4.63608594e-02, -5.177578125e-02, -1.63085938e-01, 4.1748469e-02,
 2.00171875e-01, -2.01171875e-01, -1.50756836e-02, 2.61718750e-01,
 -1.18097500e-01, -4.07750000e-01, 2.22265625e-01, -1.25000000e-01,
 4.19021056e-01, -8.04756562e-01, -2.08007812e-02, -1.96289062e-01,
 -9.42382812e-02, -3.12500000e-02, 6.34765625e-02, 2.47802734e-01,
 -1.61133812e-01, -1.53328312e-01, 1.31835938e-01, -1.81640625e-01,
 -3.37421056e-02, -1.43554688e-01, 8.30078125e-03, -8.81445312e-02]
```

Gambar 5.165 sick

- berikut merupakan hasil pengolahan kata clear pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [50]: gmodel['clear']
Out[50]:
array([-2.4414625e-04, -1.02050781e-01, -1.49414602e-01, -4.24804688e-02,
 -1.7698750e-01, -1.46484375e-01, -1.76757812e-01, -1.46484375e-01,
 2.2052500e-01, -9.7655250e-02, -2.67578125e-01, -1.29038212e-01,
 1.4857250e-01, -1.46484375e-01, -1.76757812e-01, -1.46484375e-01,
 2.00619512e-01, -1.76974120e-01, -6.83593750e-02, -1.20913750e-01,
 3.22256250e-02, -1.44315235e-01, -1.11816466e-01, -8.00781250e-02,
 -2.75878906e-02, -6.04248847e-03, -7.37394688e-02, -1.78152562e-01,
 9.66796875e-02, -4.91330388e-03, -1.78718938e-01, -1.40380855e-01,
 7.91015625e-02, -1.07911051e-06, -1.18351562e-01, -3.09639860e-02,
 1.98574699e-02, -3.14331055e-01, -3.10615234e-01, -3.54265200e-01,
 2.5399750e-02, -1.76757812e-01, -1.46484375e-01, -1.7698750e-01,
 -4.96992750e-02, -1.35731556e-02, -9.17967850e-02, -1.03059375e-01,
 3.49121994e-02, -7.32421875e-02, -2.17773438e-01, -5.08482216e-01,
 7.47070312e-02, -1.25080000e-01, -5.77337480e-01, -5.27658232e-02,
 -1.37329120e-02, -2.13867299e-01, -1.12792696e-01, -4.98046875e-01,
 1.92382812e-01, -3.28125900e-01, -5.00848216e-02, -1.66156250e-01,
 -1.51798488e-02, -1.07911051e-06, -3.18318944e-01, -1.40380855e-01,
 -1.12349488e-02, -1.64749200e-02, -1.64749200e-02, -3.44748200e-02,
 -0.55273438e-01, -6.15039602e-01, -1.65991281e-01, -1.39414468e-02,
 2.85156250e-01, -6.29551250e-01, -3.32031250e-01, -2.41219938e-01,
 1.39160156e-01, -5.12695312e-02, -9.76552500e-02, -1.39414468e-02,
 2.51464844e-02, -8.25156250e-02, -1.20423438e-01, -6.88475622e-02,
 -5.29758156e-02, -1.06584688e-01, -2.07031250e-01, -1.65016250e-01,
 -2.61638444e-02, -3.01518000e-01, -8.00781250e-02, -1.65016250e-01,
 -3.68103844e-02, -3.18056880e-01, -3.18056880e-01, -6.95074560e-02,
 1.03759766e-01, -3.33798052e-01, -9.88719244e-03, -1.27244984e-02,
 -1.14746894e-01, -1.47779688e-02, -9.71679688e-02, -1.66992188e-01,
 -5.30645444e-02, -7.67160882e-03, -1.04568750e-01, -0.33797444e-01]
```

Gambar 5.166 clean

- berikut merupakan hasil pengolahan kata shine pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [51]: gmodel['shine']

Out[51]:
array([-0.12402344, -0.25976562, -0.15917969, -0.27734375,
 0.08938836, -0.35978172, -0.22930741, -0.18373575,
 0.05882734, -0.35978172, -0.22930741, -0.18373575,
 0.21773448, 0.14022458, 0.15234275, -0.12204588, -0.13935529,
 0.05883789, 0.06127939, 0.01949081, 0.07161788,
 0.20019531, 0.38085598, 0.00162596, -0.05829297,
 0.34765625, 0.26534747, -0.23925759, 0.04516602,
 0.04807912, -0.18153112, -0.15234375, -0.05451614,
 0.04931265, 0.04931265, 0.04931265, 0.04931265,
 0.04202343, 0.06132182, 0.12739969, 0.10957689,
 0.02464582, 0.32226562, 0.11367953, 0.18164662,
 0.18253906, 0.08283318, 0.08283318, -0.17178967,
 0.13671186, 0.28894838, 0.07618128, 0.08539735,
 0.14257812, 0.11625245, 0.09372384, -0.14411215,
 0.11625245, 0.09372384, 0.09372384, 0.09372384,
 0.04760742, 0.08877891, -0.00427361, 0.05908203,
 0.01515775, -0.11621299, 0.07212899, 0.01840389,
 0.08886719, 0.11522348, 0.09679569, -0.11883984,
 0.33593797, -0.1875, 0.14505781, 0.08443687,
 0.09512454, 0.08447622, 0.08447622, 0.08447622,
 0.02452525, 0.02452525, 0.02452525, 0.02452525,
 0.22049219, 0.14941406, -0.19521325, 0.08490494,
 0.078125, 0.05988203, 0.02355597, 0.06347657,
 0.08740234, 0.10858594, -0.11476489, 0.18164662,
 0.11230469, -0.00808109, 0.08619406, 0.03808954,
 0.03417969, 0.08830878, 0.14615065, -0.09619141,
 0.03417969, 0.08830878, 0.14615065, 0.03808954,
 0.00171604, 0.04589548, 0.04589548, 0.04589548,
 0.03615137, -0.11725482, 0.04646243, -0.10457943,
 0.03615137, -0.11725482, 0.04646243, -0.10457943,
 0.03615137, -0.11725482, 0.04646243, -0.10457943])
```

**Gambar 5.167** shine

- berikut merupakan hasil pengolahan kata bag pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [52]: gmodel['bag']
Out[52]:
array([-0.03515625, 0.15234375, -0.12402344, 0.13378906, -0.11328125,
 -0.0133667, -0.16113281, 0.14648438, -0.06835938, 0.140625,
 -0.0595559, -0.10825398, 0.10942309, -0.1767808, -0.09423029,
 -0.05937931, -0.05953711, 0.10253986, 0.10942309, -0.09423029,
 0.18847634, -0.07958984, 0.11835156, -0.07918156, 0.06347656,
 -0.15527344, -0.18945312, 0.11132812, 0.27539062, -0.06787189,
 0.01866641, -0.16113281, 0.2578125, 0.0324797, -0.24609375,
 -0.05541992, 0.081013184, 0.24121894, -0.21875, 0.07568359,
 0.08814453, -0.16113281, 0.16583986, -0.09521484, -0.16681562,
 -0.0474075, 0.08835156, 0.16583983, -0.09521484, -0.16681562,
 -0.1924219, -0.2551758, -0.04296375, 0.10107425, 0.17324312,
 -0.13738984, -0.265152, -0.08466591, 0.19528986, -0.1838844,
 0.14941406, -0.1484375, 0.09619141, 0.21777344, -0.08544922,
 -0.02819824, 0.02539862, -0.03759768, 0.23242188, 0.19628986,
 0.27539062, 0.09138059, 0.23730469, 0.09033283, -0.28515625,
 0.05932617, 0.06591797, -0.01794434, -0.00855311, -0.1796875,
 0.06525324, -0.1220538, -0.02744628, -0.09453125, -0.14453125,
 -0.3072338, -0.08835156, -0.02744628, -0.17671094, -0.05544922,
 -0.2041018, 0.33789862, 0.00228882, -0.39453125, -0.14453125,
 -0.328125, -0.12695312, -0.08544592, 0.15234375, 0.03662109,
 -0.1484375, -0.05566406, 0.02844238, 0.07519531, -0.21484375,
 -0.15722656, 0.3359375, -0.04736328, -0.00405884, -0.19726562,
 0.27929688, 0.05566406, -0.10695594, -0.00811768, -0.26783125,
 0.0453867, -0.14550781, -0.0217999, 0.16056166, -0.17671094,
 0.0358867, 0.0636875, -0.25, -0.08835156, -0.17671094, -0.07714844,
 0.00521851, 0.125, -0.08838719, 0.15527344, -0.02185059,
 -0.15234375, -0.12890625, -0.34765625, -0.13769531, -0.18164062,
 0.37695312, 0.14160156, -0.03051758, 0.08203125, 0.09423828,
```

Gambar 5.168 bag

- berikut merupakan hasil lpengolahan kata car pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [53]: gmodel['car']
Out[53]:
array([0.13085938, 0.00842205, 0.03344727, -0.05883789, 0.04003986,
 -0.14257813, 0.04931641, -0.16894531, 0.28898438, 0.11952891,
 -0.18866406, -0.25, -0.10490931, -0.10742188, -0.01879883,
 0.05200195, -0.00216675, 0.06445312, 0.14453125, -0.04541016,
 0.16113281, -0.01611328, -0.03880379, 0.08447266, 0.16210938,
 0.0444773, -0.15527344, 0.25398625, 0.3384343, 0.08756536,
 -0.23589867, -0.08835156, -0.02744628, -0.17671094, -0.05544922,
 -0.09912109, 0.16583986, 0.08684766, -0.18945312, 0.02323021,
 -0.0534668, -0.03063965, 0.11083984, 0.24121894, -0.2334375,
 0.12353516, -0.00294495, 0.1484375, 0.33293125, 0.05249023,
 -0.20019531, 0.37695312, 0.12255859, 0.11425781, -0.17675781,
 0.10809766, 0.0030365, 0.26757812, 0.20117188, 0.03710938,
 0.11083984, -0.08544592, -0.125, -0.08835156, 0.08283931,
 -0.2671875, -0.08544592, 0.02083801, 0.05934961, -0.04398754,
 0.11767578, -0.04296875, -0.17285156, 0.04394531, -0.23046875,
 0.16464625, -0.11474669, -0.06508273, 0.01196289, -0.24707831,
 0.32617188, -0.04492188, -0.11425781, 0.22851562, -0.01647949,
 -0.15039862, -0.13183594, 0.12597656, -0.17480469, 0.02209473,
 -0.10185625, 0.00817871, 0.10791016, -0.24609375, -0.109375 ,
 -0.09335156, 0.00816233, -0.02121044, 0.2314453, -0.0860336,
 -0.05419925, -0.02084329, 0.0783788, -0.17689844,
 -0.17578125, -0.02770996, -0.26438156, 0.02302578, 0.83125 ,
 -0.25239625, -0.125, -0.05493164, -0.17382812, 0.28515625,
 -0.23242188, 0.0234375, -0.20117188, -0.13476562, 0.26367188,
 0.00769043, 0.20507812, -0.01708984, -0.12988281, 0.04711914,
 0.22670312, 0.02899608, -0.29101562, -0.02893066, 0.17285156,
 0.04272461, -0.19824219, -0.04083966, -0.16992188, 0.10658594,
```

Gambar 5.169 car

- berikut merupakan hasil lpengolahan kata wash pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [54]: gmodel['wash']
Out[54]:
array([9.4604492e-03, 1.41601562e-01, -5.4687500e-02, 1.34765625e-01,
 -2.38281250e-01, 3.24218750e-01, -6.44726562e-02, -1.29882182e-01,
 1.76190156e-01, 2.53986250e-01, 1.13953391e-02, -1.68162188e-01,
 -2.79507450e-02, 1.30885750e-01, 2.11914096e-01, -5.54840562e-01,
 -2.42157500e-02, 3.04685750e-01, 2.11914096e-01, -5.88671075e-02,
 2.67578125e-01, 2.12898625e-01, 1.74569547e-02, 2.02941895e-03,
 6.29882182e-01, 1.62109375e-01, 1.93359375e-01, 2.172385156e-02,
 -2.67928809e-03, -9.13865938e-02, -2.38281250e-01, 2.23632812e-01,
 -8.00781250e-02, -3.08859375e-02, -1.008097656e-01, -1.39648438e-01,
 1.76190156e-01, 6.78710625e-02, 1.11301250e-01, 1.68162188e-01,
 -1.69467950e-01, -2.03885750e-01, 2.02941895e-01, -3.30973444e-02,
 -3.43750000e-01, 1.030659781e-01, 3.08859375e-01, -5.85371099e-02,
 5.07812500e-02, 1.45507812e-01, 2.81250000e-01, 7.81312500e-02,
 2.84423828e-02, -2.29492188e-01, -5.81054688e-02, 4.15660156e-02,
 -3.56445312e-02, 1.77734375e-01, 1.22070312e-01, 3.710937350e-02,
 -1.10839844e-01, 6.83593750e-02, -2.52685547e-02, -1.27929688e-01,
 4.21089384e-01, 5.32228562e-02, -3.92537500e-01, 3.00000000e-01,
 1.77985939e-02, -2.03885750e-02, 1.30885750e-01, 3.10359375e-01,
 1.08398443e-01, -4.30297852e-03, -2.45117188e-01, -2.08984734e-01,
 3.58886719e-02, 8.30878125e-02, 1.68945312e-01, 2.79541616e-02,
 1.04988469e-01, -3.47556250e-01, -5.20019531e-02, 2.24668575e-01,
 1.69677734e-02, 1.69921875e-01, -1.46484375e-01, 2.26525000e-01,
 2.17285156e-02, 1.12304688e-02, -1.14257812e-01, 7.22656250e-02,
 4.32309375e-02, -1.57804468e-01, 5.07339375e-04, -7.91073444e-02,
 -5.90144531e-02, -5.44433594e-02, 2.02941895e-02, -2.25900000e-01,
 2.26562500e-01, 5.395509781e-02, 1.13769531e-01, -5.83496954e-02,
 -1.533208312e-01, -4.375509000e-01, 2.59765625e-01, -1.49414662e-01,
 6.64047500e-02, 7.13867188e-01, -7.86666733e-02, -7.70808473e-01]
```

Gambar 5.170 wash

- berikut merupakan hasil pengolahan kata motor pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [56]: gmodel['motor']
Out[56]:
array([5.73730469e-02, 1.50390625e-01, -4.61425781e-02, -1.32812500e-01,
 -2.59565625e-01, -1.77734375e-01, 3.08859375e-02, -4.37126250e-01,
 -2.43578125e-02, -1.57804468e-01, 7.74804468e-01, 2.02941895e-02,
 -2.51953125e-02, -1.58263125e-01, -5.16271875e-02, 1.12304688e-01,
 -3.83390781e-02, 1.58263125e-01, -5.85937500e-02, 1.12304688e-01,
 1.56250000e-01, -4.24804688e-02, -1.32812500e-01, 2.11194062e-01,
 1.23046875e-01, 1.69921875e-01, -1.55273438e-01, 4.59894375e-01,
 3.02734375e-01, 1.53320312e-01, -1.69921875e-01, -1.01074219e-01,
 -3.20238675e-03, 2.02941895e-02, 8.00437500e-02, -1.27929688e-02,
 1.54296875e-02, -1.38671062e-01, 2.11914096e-01, -5.16271875e-01,
 -4.46777344e-02, -3.06649625e-01, 7.42187500e-02, 5.50896375e-01,
 -1.30859375e-01, 1.09585938e-01, -3.34472656e-02, 2.10937500e-01,
 3.10058594e-02, -6.50924414e-03, 6.34765625e-02, 4.02832031e-02,
 -2.78320312e-02, 1.07421875e-02, 1.47460938e-01, 2.80761719e-02,
 -1.50390625e-01, -1.37695312e-01, 9.66937500e-02, 1.28906250e-01,
 -3.13726250e-02, -1.008097656e-01, -2.11914096e-01, -5.16271875e-02,
 -3.59640625e-02, -1.15933125e-01, -1.06640625e-01, -5.16271875e-01,
 -2.21679688e-01, 2.33398438e-01, -5.085710944e-02, -3.37896350e-01,
 1.53320312e-01, -7.12898625e-02, -3.68652344e-02, 7.66601562e-02,
 -8.00781250e-02, -1.14257812e-01, -9.17679688e-02, -2.61178750e-01,
 3.84755625e-01, -1.87500000e-01, -1.10351562e-01, 1.00585938e-01,
 1.08383438e-01, 9.57031250e-02, -2.82312500e-02, 1.54296875e-01,
 -2.24804688e-02, -1.38671062e-01, 2.11914096e-01, -5.16271875e-01,
 9.71679688e-02, -2.52685547e-02, -5.46687500e-02, -5.88671075e-02,
 2.80312500e-02, -3.32031250e-01, 3.27148438e-02, 5.71289862e-02,
 1.77734375e-01, -9.57031250e-02, 2.45117188e-01, 6.88476562e-02,
 2.63671875e-01, -8.15429688e-02, 1.25976562e-01, 1.00585938e-01,
```

Gambar 5.171 motor

- berikut merupakan hasil pengolahan kata cycle pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [57]: gmodel['cycle']
Out[57]:
array([-0.04540106, 0.21679688, -0.02709961, 0.12353516, -0.20703125,
 -0.1328125, 0.26367188, -0.12890625, -0.125, 0.15332031,
 -0.12890625, 0.12353516, -0.02709961, 0.21679688, -0.04540106,
 -0.02552477, -0.07562359, -0.0635, 0.04614258, -0.21054588,
 -0.13378906, -0.11669922, -0.3359375, 0.078125, 0.08847266,
 0.07226562, -0.06445512, 0.05517578, 0.14941406, 0.13671785,
 0.10382734, 0.02172852, -0.10693359, 0.02498234, -0.10644531,
 -0.05541992, -0.29492188, -0.40803962, 0.06347656, -0.08847266,
 0.07371081, 0.18157771, -0.08770777, 0.13671785, -0.1640625,
 0.11457791, 0.08806181, -0.08979146, 0.09033982, 0.18066496,
 0.18066496, -0.28515625, -0.04952734, 0.01806641, 0.00331116,
 0.00872803, 0.03564453, -0.29882612, 0.0999389, -0.1448375,
 -0.06787109, 0.05957031, -0.05517578, -0.19628906, 0.2265625,
 0.03173828, -0.07080078, 0.1484375, -0.20214844, -0.03393555,
 0.09963201, -0.02038574, -0.00789662, -0.07226562, -0.09423288,
 -0.17940375, 0.080546875, 0.080546875, 0.080546875, 0.080546875,
 -0.02636719, -0.03608023, 0.12890625, -0.0677109, 0.14257012,
 0.37109375, -0.15722656, -0.009326172, 0.34969038, -0.00891553,
 0.03613281, 0.16894531, -0.02856445, 0.10791016, 0.32421875,
 -0.14355469, 0.03173828, -0.07421875, 0.34179688, 0.146625,
 0.0043335, -0.12896625, -0.34960938, -0.02929688, -0.19628906,
 -0.2578125, -0.3671875, 0.01483154, 0.20783125, 0.09667969,
 -0.04540106, 0.21679688, -0.02709961, 0.12353516, -0.20703125,
 -0.06590453, -0.1796075, 0.0278208, 0.15625, 0.03633637,
 0.83247897, -0.13476562, 0.15527344, 0.11132812, -0.81655988,
 0.07958984, 0.01989746, 0.25585938, 0.13379806, 0.02539962,
 0.10986328, -0.20685469, 0.07275391, -0.35546875, -0.02746582,
 -0.20800781, 0.10498847, -0.0625, 0.01177979, 0.17382812,
```

Gambar 5.172 cycle

- berikut merupakan hasil dari similaritas kata kata yang diolah menjadi matrix tadi adapun persentase untuk perbandingan setiap katanya yaitu 9 persen untuk kata wash dan clear 7 persen untuk kata bag dan love 48 persen untuk kata motor dan car 12 persen untuk kata sick dan faith dan terakhir yaitu 6 persen untuk kata cycle dan shine.

```
In [60]: gmodel.similarity('wash', 'clear')
Out[60]: 0.09019179

In [61]: gmodel.similarity('bag', 'love')
Out[61]: 0.075360954

In [62]: gmodel.similarity('motor', 'car')
Out[62]: 0.4810173

In [63]: gmodel.similarity('sick', 'faith')
Out[63]: 0.12307322

In [64]: gmodel.similarity('cycle', 'shine')
Out[64]: 0.00161793
```

Gambar 5.173 similarity

## 2. jelaskan dengan kata dan ilustrasi fungsi dari extract words dan Permute-Sentences

Untuk Extract Words berfungsi untuk memecahkan kata menjadi beberapa komponen atau lebih mudahnya kata yang dieksekusi dipecah kemudian dikelompokkan sesuai dengan keinginan

Pada hasil yang didapatkan untuk contoh ini yaitu terdapat satu kalimat untuk yang terbentuk dari 5 kata. Kata tersebut kemudian dipisahkan menggunakan perintah ( test string.split) kemudian hasil keluarannya akan di print dengan parameter tambahan kata sehingga memberikan penjelasan ataupun tanda yang lebih jelas terhadap perbedaan eksekusi kata ketika masih terangkai dan ketika telah terpecahkan.

```
In [65]: test_string = "Alit Fajar Kurniawan Merita Qadery"
.....
.... #printing original string
.... print("String Originalnya adalah : " + test_string)
....
.... #using split()
.... #to extract words from string
.... res = test_string.split()
....
.... print ("Daftar Kata-katanya adalah : " + str(res))
String Originalnya adalah : Alit Fajar Kurniawan Merita Qadery
Daftar Kata-katanya adalah : ['Alit', 'Fajar', 'Kurniawan', 'Merita', 'Qadery']
```

Gambar 5.174 praktek2

Gambar tersebut didalamnya telah mengilustrasikan sebuah kalimat 'Alit Fajar Kurniawan Qadery' yang akan dipisahkan perkata. Dimana library re (regex module) dan library string di import terlebih dahulu. Kemudian untuk variable out mendenisikan X untuk mengembalikan string pada objek line yang telah di split yang eksekusi perintah split itu diartikan sebagai (dibagi atau dipisahkan). Kemudian, X dikembalikan berdasarkan jumlah kata yang ada, sehingga muncullah hasil seperti pada gambar yang dijadikan sebagai keluaran.

```
In [67]: import re, string
.....
.... def extract_words(line):
.... out = (x.lower() for x in line.split())
.... out = (re.sub('[\s+]%', re.escape(string.punctuation), ' ', x) for x in out)
....
.... return (x for x in out if len(x) > 0)
....
....
.... #test the function
.... list(extract_words("Alit Fajar Kurniawan Qadery"))
Out[67]: ['alit', 'fajar', 'kurniawan', 'qadery']
```

Gambar 5.175 praktek2

PermuteSentences digunakan untuk melakukan pengocokan ( shue ataupun random ) pada text yang diiginkan. Pada gambar tersebut mengeluarkan sebuah hasil ( keluaran ) yang telah berupa pengacakan/pengocokan kata/huruf/kalimat yang telah didenisikan pada perintah untuk dieksekusi. Kata ALIT di random dengan beberapa kali sehingga memberikan hasil yang beragam dimana semuanya berbentuk string dan dieksekusi berupa len. Untuk return dari inputan yang dieksekusi tersebut di denisikan dengan pemanggilan variabel next list.

### 3. Library Gensim TaggedDocument Dan Doc2Vec

- Gensim TaggedDocument : Tagged Document merupakan class yang digunakan dalam library gensim yang dicontohkan dimana tagged dokument yang 'memisahkan informasi dan struktur dari presentasi' dengan menggunakan tag .
- Doc2Vec : Untuk membuat representasi numerik dari suatu dokumen, terlepas dari panjangnya. Model doc2vec dapat digunakan dengan cara berikut: untuk pelatihan, diperlukan seperangkat dokumen.

```
In [69]: from gensim.models.doc2vec import TaggedDocument
...: from gensim.models import Doc2Vec
```

**Gambar 5.176 praktek3**

4. menambahkan data training dari le yang dimasukkan kepada variabel dalam rangka melatih model doc2vac.

Penjelasan Untuk Penambahan Data Training : Ada beberapa point penting yang bisa diperhatikan dan dipertimbangkan pada penambahan data yaitu

- Disintegrasi dan Komposisi: Langkah ini melibatkan pemecahan tür tertentu untuk membangun data pelatihan yang lebih baik untuk model yang Anda pahami lebih komprehensif sehingga menghasilkan hasil yang lebih tepat dan esien.
- Penskalaan dimana dataset diempatkan sambil mempertimbangkan kumpulan data linier seperti data bank ataupun data lainnya.
- Kemudian untuk Komposisi: merupakan proses terakhir yang melibatkan penggabungan berbagai tür menjadi satu tür untuk mendapatkan data yang lebih akurat atau bermakna.

```
In [70]: for dirname in ["train/pos", "train/neg", "train/unsup","train/
pos", "train/neg"]:
 ...
 for fname in sorted(os.listdir("aclImdb/" + dirname)):
 ...
 if fname[-4:] == '.txt':
 ...
 with open("aclImdb/" + dirname + "/" + fname,
encoding="UTF-8") as f:
 ...
 sent = f.read()
 ...
 words = extract_words(sent)
 ...
 unsup_sentences.append(TaggedDocument(words,
[dirname + "/" + fname]))
```

**Gambar 5.177 praktek4**

penjelasan : direalisasikan pemanggilan directory name ( dirname ) dari le yang akan dieksekusi . Kemudian didenisikan fname untuk pemberian nama terhadap le tersebut yang akan di urutkan sesuai dengan list dir dengan parameter dirnamanya. Setiap contoh yang diberikan semuanya akan direalisasikan dalam inputan variabel words dengan extract word yang dihubungkan dengan unsup sentences yang mengeksekusi class tagged document. Ketiga gambar tersebut ketika dijalankan tidak terjadi error, maka pengujian atau praktekpun berhasil dilakukan.

5. pengocokan dan pembersihan data

Pengocokan Data yaitu Melakukan pengacakan terhadap data kemudian nantinya bisa dieksekusi. Kemudian pada setiap eksekusinya akan menghasilkan hasil yang berbeda berdasarkan/berkaitan dengan penerapan mode ( shufe atau random ) dalam pengeksekusian data. sedangkan Pembersihan Data yaitu Melakukan pengecekan dan pemulihan data.

```
In [72]: import re
...: def extract_words(sent):
...: sent = sent.lower()
...: sent = re.sub(r'<[^>]+>', ' ', sent) #hapus tag html
...: sent = re.sub(r'(\w)\|(\w)', ' ', sent) #hapus petik satu
...: sent = re.sub(r'\w', ' ', sent) #hapus tanda baca
...: sent = re.sub(r'\s+', ' ', sent) #hapus spasi yang berurutan
...: return sent.split()

In [73]: import random
...: class PermuteSentences(object):
...: def __init__(self, sents):
...: self.sents = sents
...: def __iter__(self):
...: shuffled = list(self.sents)
...: random.shuffle(shuffled)
...: for sent in shuffled:
...: yield sent
```

**Gambar 5.178** praktek5

penjelasan terjadi pengimportan Library Re. Selanjutnya dilakukan pembuatan fungsi untuk menghapus tag html dan perkocokan ( pengocokan ) . Berkaitan dengan penghapusan tersebut, pada variabel ini terdapat kodingan petik satu yang bisa direalisasikan, tanda baca dan spasi yang berurutan pun ada Selanjutnya yaitu melakukan pengacakan model terhadap data unsupervised learning yang ada, kemudian baru dibuatkan (membuat) modelnya setelah dilakukan pengacakan data yang telah ada sebelumnya.

6. Mengapa Model Harus Di Save Dan Temporari Training Harus Dihapus suapaya dalam pengolahan data tidak perlu menjalankan kembali data vektorisasi serta untuk meringankan beban ram. kemudian temporary harus dihapus guna meningkatkan peforma komputer. Model harus di save dikarenakan atau difungsikan untuk mencegah ram pada komputer/laptop tidak terjadi malfunction ataupun lemot ( loading yang lama ). Kemudian untuk mengapa temporari training harus dihapus dimana digunakan untuk mengosongkan memori sehingga terdapat ruang lebih ataupun lapang sehingga tidak terjadi malfunction pada komputer dll.

```
In [13]: from gensim.models.doc2vec import TaggedDocument
...: from gensim.models import Doc2Vec
...:
...:
model.delete_temporary_training_data(keep_inference=True)
...: model.save('haci.d2v')
```

**Gambar 5.179** praktek6

Untuk model gambar pertama diperlihatkan praktek untuk percobaan penghapusan temporary training data dimana apabila parameternya "keep inference" true maka akan dilakukan penghapusan sesuai dengan perintah yang ada. Kemudian untuk gambar kedua memperlihatkan praktek dari perintah save terhadap sebuah parameter kata yaitu (alit.d2v) yang disimpan berupa INFO berbarengan / disertakan dengan tanggal dan waktunya.

7. Maksud Dari Infer code

Difungsikan untuk menyimpulkan vector yang mana berhubungan dengan vektor dokumen baru (pada subjek pengeksekusian). Berdasarkan dari hasil tersebut, kalimat yang dieksekusi yaitu " saya kamu dia mereka " dipecah kemudian disimpulkan dimana keluarannya menghasilkan array dengan dtype=oat32.

8. Maksud Dari Cosine similarity pengeksekusian dan juga pengujian terhadap model cosine similarity dengan 2 objek kata yaitu "services sucks 2" memberikan keluaran berupa array sebagai tingkatan kesamaan ataupun perbandingan terhadap kedua kata yang sama tersebut. Hasilnya 0.9 sehingga membuktikan tingkat kesamaan yang signikan diantara keduanya.
9. Praktek Score Dari Cross Validation

Untuk Praktek dari score cross validation ini akan berpatokan pada perhitungan model clrf, sentvecs, setiments kemudian akan menghitung keakuratan dari data ataupun parameter yang dieksekusi tersebut. Telah dilakukan pengeksekusian untuk pengujian score dari cross validation dengan perhitungan model clrf, sentvecs, setiments sebagai inputan kemudian menghasilkan keluaran ( output ) yang berupa angka dimana akan tersebut diartikan sebagai tingkat keakuratan perhitungan yang terjadi pada inputan yang ada.

### 5.6.3 Penanganan Error

## BAB 6

---

# CHAPTER 6

---

### 6.1 Faisal Najib ABdullah / 1174042

#### 6.1.1 Teori

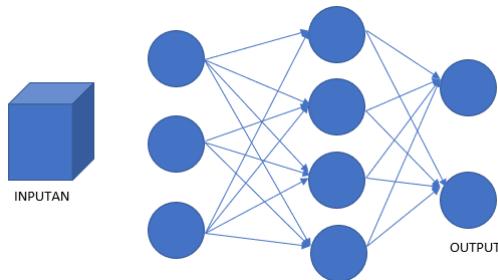
1. Jelaskan kenapa file suara harus dilakukan MFCC dilengkapi dengan ilustrasi atau gambar.  
digunakan untuk mengidentifikasi jenis suara misalkan jenis suara gendre lagu jes pop metal dan klasikal atau suara ultra sonic.



**Gambar 6.1** Ilustrasi gambar metode MFCC

2. Jelaskan konsep dasar neural network. dilengkapi dengan ilustrasi gambar.

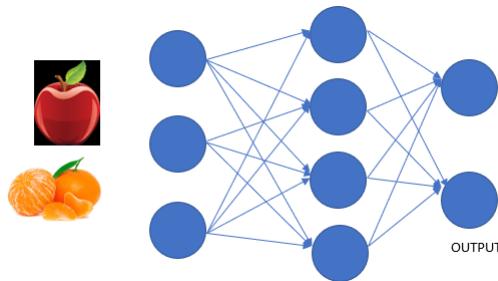
konsep neural network dilakukan ada inputan pasti ada outputan sesuai dengan kategori inputan dan fungsi di dalamnya.



**Gambar 6.2** Ilustrasi Konsep dasar neural network

3. Jelaskan konsep pembobotan dalam neural network. dilengkapi dengan ilustrasi gambar.

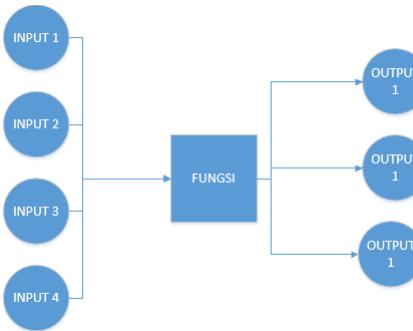
pembobotan dalam neural network yaitu digunakan untuk membedakan objek inputan atau variabel inputan untuk AI misalkan apel dan jeruk digunakan untuk variabel inputan maka dibuat pembobotan antara kedua benda tersebut untuk menentukan output yang pasti dari inputan yang dilakukan.



**Gambar 6.3** Ilustrasi Konsep pembobotan pada neural network

4. Jelaskan konsep aktifitas dalam neural network. dilengkapi dengan ilustrasi gambar.

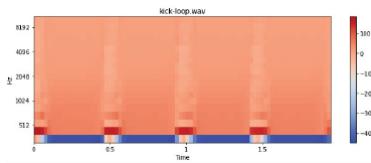
cara aktifitas dalam neural network dilakukan terhadap input pada neural network inputan tersebut dimasukan kepada fungsi pada mesin misalkan fungsi  $\tanh(x)$  sehingga dihasilkanlah output yang sesuai dengan fungsi tersebut.



**Gambar 6.4** Gambar yang dibaca hasil plotnya

5. Jelaskan cara membaca hasil plot dari MFCC dilengkapi dengan ilustrasi gambar.

cara membaca hasil plotting dari MFCC yaitu tentukan terlebih dahulu batas minimal Hz dari gelombang suara dan batas maksimal dari suara tersebut. kemudian warna yang paling pekat merupakan hasil dari pengolahan data tersebut misalkan muncul warna orange pekat di bagian bawah dan orange muda di bagian atas yang berarti suara tersebut kuat bagian basnya dan biasanya juga antara warna yang pekat tersebut ada jarak.



**Gambar 6.5** Ilustrasi Cara Membaca Hasil Plot

6. Jelaskan apa itu one-hot encoding, dilengkapi dengan ilustrasi kode atau gambar.

one-hot encoding merupakan pemberian nilai pada suatu variabel jika nilai itu iya maka nilainya satu dan jika tidak maka nilainya nol.

7. Jelaskan apa dari np.unique dan to\_categorical dalam kode program, dilengkapi dengan ilustrasi atau gambar.

digunakan untuk membuat array sedangkan to\_categorical digunakan untuk membuat matrix bauititu 64 bit atau 32 bit.

8. Jelaskan apa fungsi dari Sequential dalam kode program, dilengkapi dengan ilustrasi atau gambar.

|        | pop | rock | blues | rege | clasical |
|--------|-----|------|-------|------|----------|
| Lagu 1 | 1   | 0    | 0     | 0    | 0        |
| Lagu 2 | 1   | 0    | 0     | 0    | 0        |
| Lagu 3 | 0   | 1    | 0     | 0    | 0        |
| Lagu 4 | 0   | 0    | 1     | 0    | 0        |
| Lagu 5 | 0   | 0    | 0     | 1    | 0        |
| Lagu 6 | 0   | 0    | 0     | 0    | 1        |
| Lagu 7 | 0   | 0    | 0     | 0    | 1        |

**Gambar 6.6** Ilustrasi Konsep one-hot encoding

```
>>> np . unique ([1 , 1 , 2 , 2 , 3 , 3])
array([1, 2, 3])
>>> a = np . array ([[1 , 1], [2 , 3]])
>>> np . unique (a)
array([1, 2, 3])
```

**Gambar 6.7** Ilustrasi np.unique

`to_categorical`

```
keras.utils.to_categorical(y, num_classes=None, dtype='float32')
```

**Gambar 6.8** Ilustrasi to\_categorical

sequential adalah proses perbandingan setiap elemen satu persatu mulai dari dari objek pertama hingga yang di tuju atau jika mencari angka 100 maka sequential akan membagi bagian misalnya dari satu sampai 20 dan seterusnya sampai mendapat nilai seratus.

| Bobot 1 | Bobot 2 | Bobot 3 | Bobot 4 | Bobot 5 |
|---------|---------|---------|---------|---------|
| 1-20    | 21-40   | 41-60   | 61-80   | 81-100  |

**Gambar 6.9** Ilustrasi Konsep pembobotan pada neural network

### 6.1.2 Praktikum

1. Jelaskan isi dari data GTZAN Genre Collection dan data dari freesound. Buat kode program untuk meload data tersebut untuk digunakan pada MFCC. Jelaskan arti dari perbaris kode yang dibuat (harus beda dengan teman satukelas).

Isi data data merupakan datasets lagu atau suara yang tersirri dari 10 gendre yang di simpan kedalam 10 folder yaitu folder blues, clasical, country, disco, hiphop, jazz, metal, pop, reggae, dan rock ke sepuh folder tersebut masing-masing berisi 100 data suara sedangkan data freesound merupakan contoh data suara yang akan di gunakan untuk menguji hasil pengolahan data tersebut dengan menggunakan metode

mfcc. apakah suara dari freesound termasuk kategori jazz pop atau sebagainya ?.

```

1 import librosa
2 import librosa.feature
3 import librosa.display
4 import glob
5 import numpy as np
6 import matplotlib.pyplot as plt
7 from keras.layers import Dense, Activation
8 from keras.models import Sequential
9 from keras.utils.np_utils import to_categorical
10
11 # In[1]: buat fungsi mfcc untuk ngetest ajah
12 def display_mfcc(song):
13 y, _ = librosa.load(song)
14 mfcc = librosa.feature.mfcc(y)
15
16 plt.figure(figsize=(10, 4))
17 librosa.display.specshow(mfcc, x_axis='time', y_axis='mel',
18)
19 plt.colorbar()
20 plt.title(song)
21 plt.tight_layout()
22 plt.show()
```

dapat dilihat pada kode diatas pada baris kesatu dilakukan import librosa tang digunakan untuk fungsi mfcc pada suara. pada baris kedua dilakukan import librosa feature dan pada baris ke tiga dilakukan librosa display selanjutnya pada baris ke empat dilakukan import glob kemudian insert numpy untuk pengolahan data menjadi vektor setelah itu dilakukan import matplotlib untuk melakukan plotting setelah itu dilakukan import librari keras.

Selanjutnya yaitu membuat fungsi mfcc dengan nama display\_mfcc yang didalamnya terdapat variabel y yang berisi method librosa load kemudian variabel mfcc yang berisi method librosa featurea mfcc. Setelah itu membuat float figure dengan ukuran 10 banding 4 kemudian di isi oleh data librosa display dengan variabel x nya yaitu waktu dan y yaitu mel atau Hz kemudian melakukan plot warna setelah itu melakukan plot judul dan terakhir float di tampilkan.

2. Jelaskan perbaris kode program dengan kata-kata dan lengkapi ilustrasi gambar fungsi dari display\_mfcc().

```

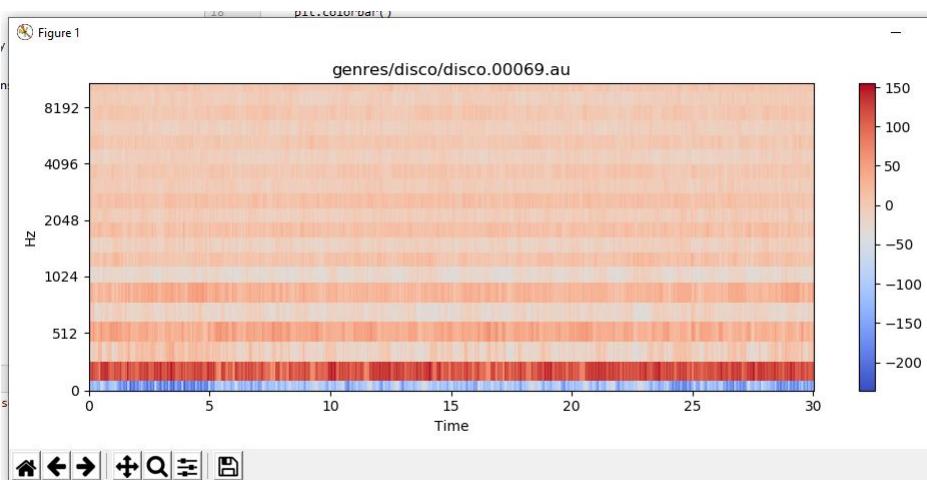
1 # In [2]: cek fungsi
2 display_mfcc('genres/disco/disco.00069.au')
3 # In [2]: cek fungsi
4 display_mfcc('genres/blues/blues.00069.au')
5 # In [2]: cek fungsi
6 display_mfcc('genres/classical/classical.00069.au')
7 # In [2]: cek fungsi
8 display_mfcc('genres/country/country.00069.au')
9 # In [2]: cek fungsi
```

```

10 display_mfcc('genres/hiphop/hiphop.00069.au')
11 # In [2]: cek fungsi
12 display_mfcc('genres/jazz/jazz.00069.au')
13 # In [2]: cek fungsi
14 display_mfcc('genres/pop/pop.00069.au')
15 # In [2]: cek fungsi
16 display_mfcc('genres/reggae/reggae.00069.au')
17 # In [2]: cek fungsi
18 display_mfcc('genres/rock/rock.00069.au')

```

pada baris ke dua program diatas digunakan untuk mendisplay tampilan glombang suara dari file 266093\_stereo-surgeon\_kick-loop-5.wav menggunakan metode mfcc dengan menggunakan fungsi display\_mfcc yang telah tadi di buat pada nomer dua begitu juga pada baris ke 4 6 8 sampai ke 22 secara teksis sama menggunakan fungsi display\_mfcc hanyasaja beda penyimpanan data yang akan di tampilkan atau di eksekusi. untuk contoh hasilnya dapat dilihat pada gambar ?? berikut:



**Gambar 6.10** Ilustrasi gambar fungsi dari display\_mfcc()

Gambar tersebut merupakan hasil dari mfcc dari salah satu gender lagu pop yang ada pada datasets yang 1000 atau terdapat dalam sepuluh folder tadi.

3. Jelaskan perbaris dengan kata-kata dan dilengkapi dengan ilustrasi gambar fungsi dari extract\_features\_song jelaskan kenapa data yangdiambil merupakan data 25.000 baris pertama ?

```

1 def extract_features_song(f):
2 y, _ = librosa.load(f)
3
4 # get Mel-frequency cepstral coefficients

```

```

5 mfcc = librosa.feature.mfcc(y)
6 # normalize values between -1,1 (divide by max)
7 mfcc /= np.abs(np.absolute(mfcc))
8
9 return np.ndarray.flatten(mfcc)[:25000]

```

pada baris ke tiga di definisikan nama extract\_features\_song yang nantinya akan di gunakan pada fungsi yang lainnya kemudian dibuat variabel y dengan method librosa load setelah itu dibuat variabel baru mfcc dengan isi librosa features mfcc dengan isi variabel y tadi kemudian dibuat variabel mfcc dengan isian np.max dan variabel mfcc tadi terakhir di buat array dari data tersebut merupakan data 25000 data pertama. kenapa data 25000 pertama yang digunakan dikarenakan data tersebut digunakan sebagai data testing semakin besar data testing yang di gunakan maka semakin akurat hasil AI. tapi sebenarnya data tersebut relatif bisa lebih besar atau lebih kecil tergantung pada komputer masing masing.

4. Jelaskan Perbaris kode program dengan kata-kata dan di lengkapi ilustrasi gambar fungsi dari generate features and labels.

```

1 def generate_features_and_labels():
2 all_features = []
3 all_labels = []
4
5 genres = ['blues' , 'classical' , 'country' , 'disco' , ,
6 'hiphop' , 'jazz' , 'metal' , 'pop' , 'reggae' , 'rock']
7 for genre in genres:
8 sound_files = glob.glob('genres/' + genre + '/*.au')
9 print('Processing %d songs in %s genre...' % (len(
10 sound_files), genre))
11 for f in sound_files:
12 features = extract_features_song(f)
13 all_features.append(features)
14 all_labels.append(genre)
15
16 # convert labels to one-hot encoding cth blues :
17 # 1000000000 classic 0100000000
18 label_uniq_ids, label_row_ids = np.unique(all_labels,
19 return_inverse=True) #ke integer
20 label_row_ids = label_row_ids.astype(np.int32, copy=False)
21 onehot_labels = to_categorical(label_row_ids, len(
22 label_uniq_ids)) #ke one hot
23 return np.stack(all_features), onehot_labels

```

pada baris ke tiga merupakan pendefinisian nama fungsi yaitu generate features and labels kemudian membuat variabel baru dengan array kosong yaitu all\_features dan all\_labels kemudian mendefinisikan isian label untuk gendre dengan cara membuat variabel genres kemudian di isi dengan 10 gendre yang tadi setelah itu dilakukan fungsi if else dengan code for dan in setelah itu akan di buat encoding untuk data tiap tiap label contoh untuk blues 1000000000 dan untuk clasical 0100000000.

5. Jelaskan dengan kata dan praktek kenapa penggunaan fungsi generate features and labels sangat lama saat meload dataset gendre tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

halnini menjadi lama dikarenakan mesin membaca satupersatu file yang ada pada folder dan dalam foldertersebut terdapat 100 file sehingga wajar menjadi lama ditambah lagi mengolah data yang tadinya suara menjadi bentuk vektor. berikut merupakan codenya.

```
1 # In [3]: passing parameter dari fitur ekstraksi menggunakan
 mfcc
2 features , labels = generate_features_and_labels()
```

```
57 features, labels = generate_features_and_labels()
 generate_features_and_labels()
Run: 1 ×
▶ Using TensorFlow backend.
Processing 100 songs in blues genre...
Processing 100 songs in classical genre...
Processing 100 songs in country genre...
Processing 100 songs in disco genre...
Processing 100 songs in hiphop genre...
Processing 100 songs in jazz genre...
Processing 100 songs in metal genre...
Processing 100 songs in pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in rock genre...

Process finished with exit code 0
```

**Gambar 6.11** Hasil dari fungsi generate features and labels

6. jelaskan kenapa harus dilakukan pemisahan data training dan data testing sebesar 80 persen praktekan dengan kode dan tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut. untuk code nya adalah sebagai berikut yang merupakan code untuk membagi data sebanyak 80 persen untuk data training maka data musik tadi yang total jumlahnya 1000 akan di bagi dua untuk data training sebanyak 800 dan 200 untuk data testing.

```
1 # In [3]: fitur ekstraksi
2 training_split = 0.8
```

7. praktekan dan jelaskan masing masing parameter dari fungsi Sequential(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

fungsi sequential digunakan untuk mengolah data inputan sesuai dengan fungsi yang ada pada fungsi sequential pada fungsi sequential kali ini menggunakan dua fungsi sequential mengkompile data dari 100 neuron atau dari 1 folder file dengan menggunakan fungsi relu dan softmax untuk menghasilkan outputan yang sesuai dengan keriteria.

```

1 # In [3]: membuat seq NN, layer pertama dense dari 100 neurons
2 model = Sequential([
3 Dense(100, input_dim=np.shape(train_input)[1]),
4 Activation('relu'),
5 Dense(10),
6 Activation('softmax'),
7])

```

8. praktekan dan jelaskan masing masing parameter dari fungsi compile(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

yaitu fungsi kompile yang digunakan untuk mengetahui parameter yang digunakan dari data yang telah diolah untuk caranya dapat menggunakan codingan sebagai berikut, pada gambar tersebut memunculkan parameternya berapasaja dan total parameter yang digunakan.

```

1 # In [3]: fitur ekstraksi
2 model.compile(optimizer='adam',
3 loss='categorical_crossentropy',
4 metrics=['accuracy'])
5 print(model.summary())

```

| Layer (type)              | Output Shape | Param # |
|---------------------------|--------------|---------|
| dense_1 (Dense)           | (None, 100)  | 2500100 |
| activation_1 (Activation) | (None, 100)  | 0       |
| dense_2 (Dense)           | (None, 10)   | 1010    |
| activation_2 (Activation) | (None, 10)   | 0       |

Total params: 2,501,110  
Trainable params: 2,501,110  
Non-trainable params: 0

Process finished with exit code 0

**Gambar 6.12** Hasil fungsi compile

9. praktekan dan jelaskan masing masing parameter dari fungsi fit(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

pada fungsi ini dilakukan pengolahan data dari 10 label tadi atau 10 file data sets tadi kemudian di hitung tingkat akurasi masing masing dan tingkat kegagalan atau loss data darisetiap file tersebut caranya dengan melakukan codingan berikut. pada gambar tersebut menunjukkan 10 pengolahan data untuk menentukan nilai akurasi dan loss dari data tersebut dan selanjutnya dilakukan fingsi evaluasi.

```

1 # In [3]: fitur_ekstraksi
2 model.fit(train_input, train_labels, epochs=10, batch_size
3 =32,
4 validation_split=0.2)

```

Run: 1 ×

```

32/640 [>.....] - ETA: 1s - loss: 0.2331 - accuracy: 1.0000
64/640 [==>.....] - ETA: 1s - loss: 0.2113 - accuracy: 1.0000
96/640 [==>.....] - ETA: 1s - loss: 0.2268 - accuracy: 0.9896
128/640 [=====]>.....] - ETA: 1s - loss: 0.2172 - accuracy: 0.9922
160/640 [=====]>.....] - ETA: 1s - loss: 0.2371 - accuracy: 0.9812
192/640 [=====]>.....] - ETA: 0s - loss: 0.2391 - accuracy: 0.9844
224/640 [=====]>.....] - ETA: 0s - loss: 0.2484 - accuracy: 0.9821
256/640 [=====]>.....] - ETA: 0s - loss: 0.2529 - accuracy: 0.9727
288/640 [=====]>.....] - ETA: 0s - loss: 0.2549 - accuracy: 0.9722
320/640 [=====]>.....] - ETA: 0s - loss: 0.2469 - accuracy: 0.9719
352/640 [=====]>.....] - ETA: 0s - loss: 0.2512 - accuracy: 0.9688
384/640 [=====]>.....] - ETA: 0s - loss: 0.2525 - accuracy: 0.9661
416/640 [=====]>.....] - ETA: 0s - loss: 0.2508 - accuracy: 0.9688
448/640 [=====]>.....] - ETA: 0s - loss: 0.2511 - accuracy: 0.9710
480/640 [=====]>.....] - ETA: 0s - loss: 0.2503 - accuracy: 0.9708
512/640 [=====]>.....] - ETA: 0s - loss: 0.2519 - accuracy: 0.9727
544/640 [=====]>.....] - ETA: 0s - loss: 0.2543 - accuracy: 0.9724
576/640 [=====]>.....] - ETA: 0s - loss: 0.2503 - accuracy: 0.9740
608/640 [=====]>.....] - ETA: 0s - loss: 0.2602 - accuracy: 0.9720
640/640 [=====] - 1s 2ms/step - loss: 0.2544 - accuracy: 0.9734 - val_loss: 1.2366 - val_accuracy: 0.5625

```

Process finished with exit code 0

**Gambar 6.13** Hasil fungsi fit

- praktekan dan jelaskan masing masing parameter dari fungsi evaluate(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

pada fungsi ini dilakukan evaluasi terhadap datayang telah di runing sebelumnya untuk lebih jelasnya dapat di lihat codingan tersebut pada codingan tersebut dilakukan evaluasi pada tingkat kegagalan dan akurasi kebenaran maka hasilnya munculkan hasil evaluasi dari 10 proses dari setiap gendre yaitu akurasi sebesar 51 persen dan loss data sebesar 1.4105 data.

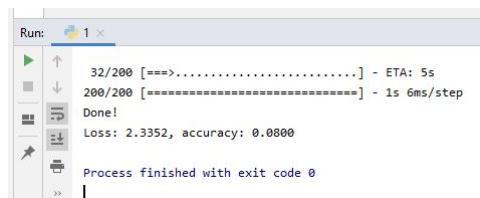
```

1 # In [3]: fitur_ekstraksi
2 loss, acc = model.evaluate(test_input, test_labels,
3 batch_size=32)
4 # In [3]: fitur_ekstraksi
5 print("Done!")
6 print("Loss: %.4f, accuracy: %.4f" % (loss, acc))

```

- praktekan dan jelaskan masing masing parameter dari fungsi predict tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

fungsi predict merupakan fungsi untuk membandingkan tingkat akurasi pada setiap label yang sepuluh tadi maka data akan di sandingkan ke



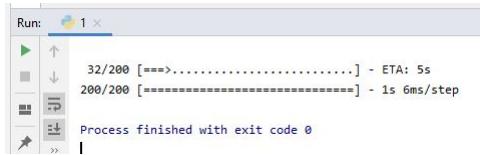
**Gambar 6.14** Hasil fungsi evaluasi

masing masing tingkat akurasinya, yang akurasinya paling tinggi maka itulah jawaban untuk setiap inputan yang dilakukan.

```

1 # In[3]: fitur_ekstraksi
2 model.predict(test_input [:1])

```



**Gambar 6.15** Hasil fungsi prediksi

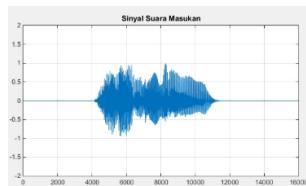
Alhamduluillah

## 6.2 1174039- Liyana Majdah Rahma

### 6.2.1 Teori

1. jelaskan kenapa file suara harus dilakukan MFCC dilengkapi dengan ilustrasi atau gambar.

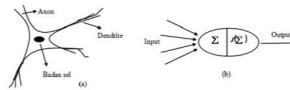
MFCC (Mel Frequency Cepstrum Coefficients) merupakan proses ekstraksi ciri dari sinyal wicara. Proses MFCC akan mengkonversikan sinyal suara menjadi beberapa vektor yang berguna untuk proses pengenalan. MFCC merupakan salah satu metode yang digunakan dalam bidang speech teknologi seperti speaker recognition serta speech recognition. Selain itu juga speakearnya mampu Mampu untuk menangkap karakteristik suara yang sangat penting bagi pengenalan suara, serta dapat menangkap informasi-informasi penting yang terkandung dalam signal suara. Dapat dilihat seperti gambar dibawah ini



**Gambar 6.16** Ilustrasi gambar metode MFCC

2. Jelaskan konsep dasar neural network. dilengkapi dengan ilustrasi gambar.

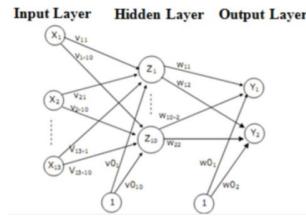
Konsep dasar neural network sendiri di mulai dari ide dasar neural network dari otak manusia, dimana otak terdiri dari 1011 neuron. sehingga konsep dasar ini membangun neural network buatan yang disebut (Artificial Neural Network). Dapat dilihat seperti gambar dibawah ini



**Gambar 6.17** Ilustrasi Konsep dasar neural network

3. Jelaskan konsep pembobotan dalam neural network. dilengkapi dengan ilustrasi gambar.

konsep pembobotan dalam neural network sendiri menggunakan algoritma backpropagation untuk memperkecil tingkat eror dengan menyesuaikan nilai bobot berdasarkan perbedaan output serta target yang dicapai. Dapat dilihat seperti gambar dibawah ini



**Gambar 6.18** Ilustrasi Konsep pembobotan pada neural network

4. jelaskan konsep aktifitas dalam neural network. dilengkapi dengan ilustrasi gambar.

fungsi aktivasi sendiri menggunakan nilai threshold untuk membatasi nilai keluaran agar selalu dalam batas nilai yang ditetapkan. Dapat dilihat seperti gambar dibawah ini

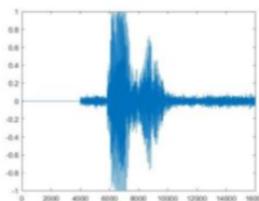
$$f(x) = \frac{1}{1+e^{-x}} \quad \text{logistic function}$$

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad \text{hyperbolic tangent function}$$

**Gambar 6.19** Gambar yang dibaca hasil plotnya

5. Jelaskan cara membaca hasil plot dari MFCC dilengkapi dengan ilustrasi gambar.

Mcara membaca plot hasil mfcc dapat dilakukan dengan cara suara pengguna dijadikan file dalam bentuk \*.wav yang digunakan sebagai inputan kemudian direpresentasikan menjadi sinyal suara dalam bentuk matriks dengan perintah audioread di Matlab R2017a.Dapat dilihat seperti gambar dibawah ini



**Gambar 6.20** Ilustrasi Cara Membaca Hasil Plot

6. Jelaskan apa itu one-hot encoding, dilengkapi dengan ilustrasi kode atau gambar.

One Hot Encoding mengkategorikan variabel yang mengandung nilai label dan bukan nilai numerik.Dapat dilihat seperti gambar dibawah ini

| Country | France | Germany | Spain |
|---------|--------|---------|-------|
| France  | 1      | 0       | 0     |
| Spain   | 0      | 0       | 1     |
| Germany | 0      | 1       | 0     |
| Spain   | 0      | 0       | 1     |
| Germany | 0      | 1       | 0     |
| France  | 1      | 0       | 0     |
| Spain   | 0      | 0       | 1     |
| France  | 1      | 0       | 0     |
| Germany | 0      | 1       | 0     |
| France  | 1      | 0       | 0     |



**Gambar 6.21** Ilustrasi Konsep one-hot encoding

7. jelaskan apa itu fungsi np.unique dan to.categorical, dilengkapi dengan ilustrasi kode atau gambar.

Kegunaan np array untuk keperluan analisis data, seperti operasi vektor dan matriks.Dapat dilihat seperti gambar dibawah ini

```
In [3]: 1 import numpy as np
In [4]: 1 # membuat array
2 a = np.array([1, 2, 3])
3 a
Out[4]: array([1, 2, 3])
```

**Gambar 6.22** Ilustrasi np.unique

- Jelaskan apa fungsi dari sequential dalam kode program, dilengkapi dengan ilustrasi kode atau gambar.

Metode Sequential merupakan proses membandingkan setiap elemen larik satu per satu secara beruntun, mulai dari elemen pertama, sampai dengan elemen terakhir atau elemen yang dicari sudah ditemukan. Dapat dilihat seperti gambar dibawah ini

```
Program pencarian beruntun

a = [10, 4, 2, 3, 7, 1, 6, 8]

cari = input("Masukkan nilai yang dicari: ")
ketemu = False
for i in range(0, len(a)):
 if cari == a[i]:
 ketemu = True
 break

if ketemu:
 print "Nilai: ", cari, "berhasil ditemukan"
else:
 print "Nilai: ", cari, "tidak ditemukan"
```

**Gambar 6.23** Ilustrasi sequential encoding

### 6.2.2 Praktek

- Jelaskan isi dari data GTZAN Genre Collection dan data dari freesound. Buat kode program untuk meload data tersebut untuk digunakan pada MFCC. Jelaskan arti dari perbaris kode yang dibuat (harus beda dengan teman satukelas).
- berikut adalah hasil dari Isi data yang merupakan datasets lagu atau suara yang terdiri dari 10 gendre yang di simpan kedalam 10 folder yaitu folder blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, dan rock ke sepuluh folder tersebut masing-masing berisi 100 data suara sedangkan data freesound merupakan contoh data suara yang akan digunakan untuk menguji hasil pengolahan data tersebut dengan menggunakan metode mfcc.ilustrasi dapat dilihat pada gambar

```
1 import librosa
2 import librosa.feature
3 import librosa.display
4 import numpy as np
5 import os
6 import librosa.util as lutil
7 from keras.layers import Input, Dense, Activation
8 from keras.models import Sequential
9 from keras.utils import np_utils, to_categorical
10
11 # Set the path to your dataset
12 # mfc = Mel Frequency Cepstral Coefficients
13 # y = labels (load song)
14 # mfc = librosa.feature.mfcc(y)
15 # y = np_utils.to_categorical(y)
16
17 plt.figure(figsize=(10, 4))
18 librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
19 plt.colorbar()
20 plt.title(song)
21 plt.tight_layout()
22 plt.show()
```

**Gambar 6.24** Data GTZAN

3. Pada code tersebut digunakan untuk mendisplay tampilan gelombang suara yang menggunakan metode mfcc.dapat dilihat pada gambar

```
[1]: In[2]: cee fungsi
display_mfcc('genres/disco/disco.00069.au')
In[2]: cee fungsi
display_mfcc('genres/blues.blues.00069.au')
display_mfcc('genres/classical/classical.00069.au')
In[2]: cee fungsi
display_mfcc('genres/country/country.00069.au')
display_mfcc('genres/hiphop/hiphop.00069.au')
In[2]: cee fungsi
display_mfcc('genres/jazz/jazz.00069.au')
```

**Gambar 6.25** hasil olah data dengan mendisplay

4. pada baris ke tiga nama extract features digunakan pada fungsi lain yang akan dibuat menjadi variabel y dengan metode librosa kemudian variabel tersebut disii dengan np.max daan variabel terakhir dibuat array dari data 250000 data pertama.dapat dilihat pada gambar

```
def extract_features_song(f):
 y, sr = librosa.load(f)

 # get Mel-frequency cepstral coefficients
 mfcc = librosa.feature.mfcc(y)
 # normalize values between -1,1 (divide by max)
 mfcc /= np.amax(np.absolute(mfcc))

 return np.ndarray.flatten(mfcc[:25000])
```

**Gambar 6.26** hasil olah data extract features

5. Pada baris ke tiga sebagai pendefinisian fungsi generate features dan label setelah itu buat variabel baru dengan array, kemudian isian label untuk gendre dengan cara membuat variabel genres, setiap datanya isi dan in setelah itu akan di buat encoding untuk data tiap tiap label contoh untuk blues 1000000000 dan untuk clasical 0100000000.hasil dari pemrosesan-nya dapat dilihat pada gambar

```

def generate_features_and_labels():
 all_features = []
 all_labels = []

 genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop']

 for genre in genres:
 sound_files = glob.glob('genres/' + genre + '/*.ogg')
 genre_label = 1 if genre == 'blues' else 0
 for sound_file in sound_files:
 features = extract_features(sound_file)
 labels = np.array([genre_label] * len(sound_files), dtype='int')
 all_features.append(features)
 all_labels.append(labels)

 # converts labels to one-hot encoding
 class_size = len(all_labels[0])
 label_matrix = np.zeros((len(all_labels), class_size))
 for i in range(len(all_labels)):
 label_matrix[i][all_labels[i]] = 1

 return np.array(all_features), label_matrix

```

**Gambar 6.27** hasil olah data generate features label

- hal ini menjadi lama dikarenakan mesin membaca satupersatu file yang ada pada folder dan dalam foldertersbut terdapat 100 file sehingga wajar menjadi lama ditambah lagi mengolah data yang tadinya suara menjadi bentuk vektor. dapat dilihat pada gambar

```

In [1]: features, labels = generate_features_and_labels()
 generate_features_and_labels()

Run: 1 Using TensorFlow backend.
 Processing 100 songs in blues genre...
 Processing 100 songs in classical genre...
 Processing 100 songs in country genre...
 Processing 100 songs in disco genre...
 Processing 100 songs in hip hop genre...
 Processing 100 songs in jazz genre...
 Processing 100 songs in metal genre...
 Processing 100 songs in pop genre...
 Processing 100 songs in reggae genre...
 Processing 100 songs in rock genre...

Process finished with exit code 0

```

**Gambar 6.28** hasil olah data genre

- untuk code nya adalah sebagai berikut yang merupakan code untuk membagi data sebanyak 80 persen untuk data training maka data musik tadi yang total jumlahnya 1000 akan di bagi dua untuk data training sebanyak 800 dan 200 untuk data testing.dapat dilihat pada gambar

```
In[3]: fitur ekstraksi
training_split = 0.8
```

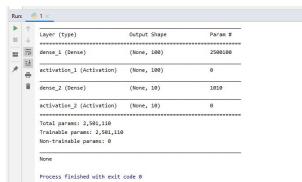
**Gambar 6.29** hasil olah data pemisahan data training

- fungsi sequential digunakan untuk mengolah data inputan sesuai dengan fungsi yang ada pada fungsi sequential pada fungsi sequential kali ini menggunakan dua fungsi sequential mengkompile data dari 100 neuron atau dari 1 folder file dapat dilihat pada gambar

```
In [4]: pembuatan NN, layer pertama dense dari 100 neurons
model = Sequential([
 Dense(100, input_dim=np.shape(train_input)[1]),
 Activation('relu'),
 Dense(10),
 Activation('softmax'),
])
```

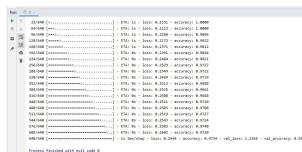
**Gambar 6.30** hasil olah data fungsi sequential

- fungsi kompile yang digunakan untuk mengetahui parameter yang digunakan dari data yang telah diolah untuk caranya dapat menggunakan codingan sebagai berikut, dapat dilihat pada gambar



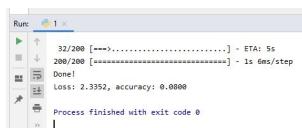
**Gambar 6.31** hasil fungsi compile

10. pada fungsi ini dilakukan pengolahan data dari 10 label tadi atau 10 file data sets tadi kemudian di hitung tingkat akurasi masing masing dan tingkat kegagalan atau loss data darisetiap file tersebut caranya dengan melakukan codingan berikut,dapat dilihat pada gambar



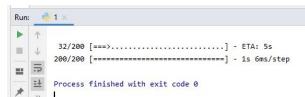
**Gambar 6.32** hasil olah data 10 label

11. pada fungsi ini dilakukan evaluasi terhadap datayang telah di runing sebelumnya untuk lebih jelasnya dapat di lihat codingan tersebut pada codingan tersebut dilakukan evaluasi pada tingkat kegagalan dan akurasi kebenaran maka hasilnya munculkan hasil evaluasi dari 10 proses dari setiap gendre yaitu akurasi sebesar 51 persen dan loss data sebesar 1.4105 data.dapat dilihat pada gambar



**Gambar 6.33** hasil fungsi evaluasi

12. fungsi predic merupakan fungsi untuk membandingkan tingkat akurasi pada setiap label yang sepuluh tadi maka data akan di sandingkan ke masing masing tingkat akurasinya, yang akurasinya paling tinggi maka itulah jawaban untuk setiap inputan yang dilakukan.dapat dilihat pada gambar



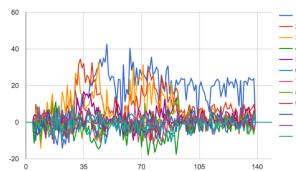
**Gambar 6.34** hasil fungsi predict

## 6.3 1174040 - Hagan Rowlenstino A. S

### 6.3.1 Teori

- Kenapa file suara harus dilakukan MFCC

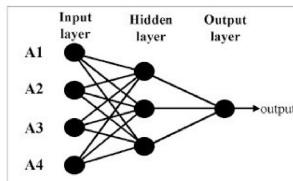
MFCC digunakan untuk mengidentifikasi jenis suara yang dimasukan seperti suara lagu, hujan atau pun suara yang tidak dapat didengar oleh telinga manusia yaitu Ultrasonik, sehingga dibutuhkan penggunaan MFCC untuk memproses data tersebut agar dapat dibaca oleh manusia. ilustrasi untuk MFCC dapat dilihat pada gambar



**Gambar 6.35** Ilustrasi MFCC

- Konsep dasar Neural Network

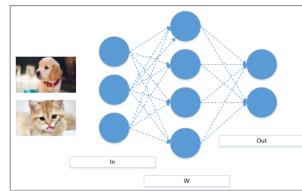
Neural Network sebenarnya mengadopsi dari kemampuan otak manusia yang mampu memberikan stimulasi/rangsangan, melakukan proses, dan memberikan output. Output diperoleh dari variasi stimulasi dan proses yang terjadi di dalam otak manusia. ilustrasi Neural Network dapat dilihat pada gambar



**Gambar 6.36** Ilustrasi Neural Network

- Konsep Pembobotan dalam Neural Network

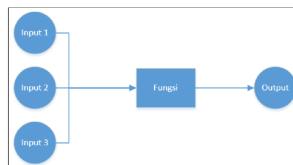
konsep pembobotan dalam neural network digunakan untuk membedakan data satu dengan data lainnya, sebagai contoh dapat dilihat pada gambar. dimana data inputan yang masuk adalah 2 data yaitu "anjing" dan "kucing" yang diolah dengan proses membandingkan data dan diolah melalui pembobotan sehingga menampilkan hasil output.



**Gambar 6.37** Ilustrasi pembobotan Neural Network

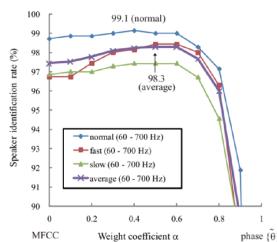
#### 4. Konsep Aktifasi dalam Neural Network

dalam Neural Network fungsi aktifasi meliputi inputan dan diproses sehingga menghasilkan Output nilai yang diharapkan, dengan menggunakan persepsi alur cara penyampaian informasi pada jaringan syaraf otak. ilustrasi dapat dilihat pada gambar



**Gambar 6.38** Ilustrasi fungsi aktifasi Neural Network

#### 5. cara membaca hasil PLOT dari MFCC



**Gambar 6.39** Ilustrasi Membaca nilai Plot dari MFCC

#### 6. One-hot Encoding

penggunaan One-hot Encoding adalah dengan membaca data yang memiliki nilai 1 sebagai nilai tinggi atau bisa disebut juga sebagai nilai positif dan 0 sebagai nilai rendah yang bermakna juga sebagai nilai negatif. ilustrasi dapat dilihat pada gambar

| X       | PUBG | FORTNITE | APEX | BFS |
|---------|------|----------|------|-----|
| Gamer 1 | 1    | 0        | 0    | 0   |
| Gamer 2 | 0    | 1        | 0    | 0   |
| Gamer 3 | 0    | 0        | 1    | 0   |
| Gamer 4 | 0    | 0        | 0    | 1   |

**Gambar 6.40** Ilustrasi One-hot Encoding

## 7. fungsi dari np.unique dan to\_categorical dalam Code Program

fungsi dari NP.UNIQUE adalah untuk membuat data elemen menjadi nilai yang bersifat unik dalam artian (Array), ilustrasi dapat dilihat pada gambar

```
>>> np.unique([1, 1, 2, 2, 3])
array([1, 2, 3])
>>> a = np.array([[1, 1], [2, 3]])
>>> np.unique(a)
array([1, 2, 3])
```

**Gambar 6.41** Ilustrasi np.unique

sedangkan perintah to\_categorical adalah untuk membuat data integer yang terdeteksi untuk diubah menjadi data matrix biner. ilustrasi dapat dilihat pada gambar

```
Consider an array of 5 labels out of a set of 3 classes {0, 1, 2}:
>>> labels
array([0, 1, 1, 2, 0])
np_utils.to_categorical converts this into a matrix with as many
columns as there are classes. The number of rows
stays the same.
>>> np_utils.to_categorical(labels)
array([[1., 0., 0.],
 [0., 1., 0.],
 [0., 1., 0.],
 [0., 0., 1.],
 [1., 0., 0.]], dtype=float32)
```

**Gambar 6.42** Ilustrasi to\_categorical

## 8. fungsi dari Sequential

fungsi dari Sequential dari code program adalah untuk membagi data - data agar dapat dianalisis oleh sistem lebih mudah, misalkan dari data 100 dibagi prosesnya menjadi 4 yaitu 25 perproses, seperti yang terdapat pada gambar ?? sehingga hasil yang didapatkan akan lebih baik lagi.

| X  | B1   | B2    | B3    | B4     |
|----|------|-------|-------|--------|
| N1 | 1>25 | 26>50 | 56>75 | 76>100 |

**Gambar 6.43** Ilustrasi fungsi Sequential

### 6.3.2 Praktek

- Penjelasan data GTZAN Genre Collection dan Freesound, Buat Code Program untuk Load data Tersebut.

GTZAN Genre Collection merupakan datasets yang berisi data lagu yang terdiri dari 10 genre yaitu :

- Blues
- Classical
- Country
- Disco
- Hip-Hop
- Jazz
- Metal
- Pop
- Reggae
- Rock

10 genre tersebut memiliki data sebesar 100 data suara.

sedangkan Freesound merupakan sebuah contoh suara yang digunakan untuk menguji hasil dari pengolahannya dengan menggunakan metode MFCC, untuk mencari genre yang pas bagi contoh suara tersebut.

```

1 #%%
2 import librosa
3 import librosa.feature
4 import librosa.display
5 import glob
6 import numpy as np
7 import matplotlib.pyplot as plt
8 from keras.models import Sequential
9 from keras.layers import Dense, Activation
10 from keras.utils import np_utils

```

Code tersebut digunakan untuk memanggil library Librosa yang memuat metode Feature dan Display yang akan digunakan untuk memproses data suara tersebut dengan MFCC. lalu Library Glob yang digunakan untuk mencocokan pattern yang spesifik dari data tersebut, Library Numpe yang digunakan untuk membuat data Vector, Library matplotlib yang

digunakan untuk membuat data grafik dan Library Keras adalah open-source yang bekerja untuk memproses TensorFlow, CNTK dan Theano, yang didesain untuk melakukan penelitian dengan menggunakan Deep Neural Network.

## 2. Penjelasan Code Program display\_mfcc

```

1 #%%
2 def display_mfcc(song):
3 y, _ = librosa.load(song)
4 mfcc = librosa.feature.mfcc(y)
5
6 plt.figure(figsize=(10, 4))
7 librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
8 plt.colorbar()
9 plt.title(song)
10 plt.tight_layout()
11 plt.show()
```

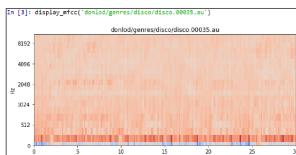
Code tersebut meliputi penggunaan MFCC dengan proses Display yang memiliki penjelasan sebagai berikut, Variable Y berisi Library Librosa dengan method LOAD dan berisi nilai SONG sebagai penggunaan classnya. dan Variable MFCC yang berisi method feature.mfcc dan memiliki nilai dari Variable Y. penggunaan plt sebagai pemanggilan matplotlib dengan method FIGURE yang akan menampilkan data gambar. dan Librosa yang akan menampilkan method display.specshow dengan data dari Variable mfcc dan pengaturan Axis X dan Y. lalu mengisikan data dengan nilai dari COLORBAR, TITLE yang berisi data dari class SONG dan tight\_layout serta show yang akan menampilkan hasil RUN yang dilakukan.

dengan menggunakan code ini akan menampilkan hasil dari penggunaan display mfcc

```

1 #%%
2 display_mfcc('donlod/genres/disco/disco.00035.au')
```

hasilnya adalah sebagai berikut, dengan menampilkan data dari file disco.00035.au dapat dilihat pada gambar



**Gambar 6.44** Hasil dari Code Program display\_mfcc

## 3. Penjelasan Code Program extract\_feature\_song

```

1 #%%
2 def extract_features_song(f):
3 y, _ = librosa.load(f)
4
5 # get Mel-frequency cepstral coefficients
6 mfcc = librosa.feature.mfcc(y)
7 # normalize values between -1,1 (divide by max)
8 mfcc /= np.amax(np.absolute(mfcc))
9
10 return np.ndarray.flatten(mfcc)[:25000]

```

Pada Code Program diatas akan menjelaskan tentang ekstrasi data feature dari lagu yang akan diolah dengan menggunakan Variable Y yang berisi method LOAD dengan record datanya dari class F pada extract\_feature\_song lalu membuat Variable mfcc dengan nilai yang akan meload data mfcc dari class Y, lalu melakukan normalisasi pada data Variable mfcc dengan penggunaan np.amax dengan nilai np.absolute(mfcc) lalu melakukan return dengan method np.ndarray.flatten(mfcc)[:25000] dimana data yang akan dibaca adalah sebanyak 25000 data dengan nilainya adalah array.

```

In [4]: def extract_features_song(f):
...: y, _ = librosa.load(f)
...: ...
...: # get Mel-frequency cepstral coefficients
...: mfcc = librosa.feature.mfcc(y)
...: # normalize values between -1,1 (divide by max)
...: mfcc /= np.amax(np.absolute(mfcc))
...: ...
...: return np.ndarray.flatten(mfcc)[:25000]

```

**Gambar 6.45** Code Program extract\_feature\_song

#### 4. Penjelasan Code program generate\_features\_and\_labels

```

1 #%%
2 def generate_features_and_labels():
3 all_features = []
4 all_labels = []
5
6 genres = ['blues' , 'classical' , 'country' , 'disco' , 'hiphop' , 'jazz' , 'metal' , 'pop' , 'reggae' , 'rock']
7 for genre in genres:
8 sound_files = glob.glob('donlod/genres/*'+genre+'/*.au')
9 print('Processing %d songs in %s genre...' % (len(sound_files), genre))
10 for f in sound_files:
11 features = extract_features_song(f)
12 all_features.append(features)
13 all_labels.append(genre)
14
15 # convert labels to one-hot encoding
16 label_uniq_ids, label_row_ids = np.unique(all_labels,
17 return_inverse=True)
18 label_row_ids = label_row_ids.astype(np.int32, copy=False)
19

```

```

18 onehot_labels = to_categorical(label_row_ids, len(
19 label_uniq_ids))
 return np.stack(all_features), onehot_labels

```

pada code ini dimulai dengan membuat 3 data Variable array dengan format 2 nilai kosong dan 1 memiliki nilai. dimana data tersebut akan diolah dengan perintah FOR untuk membagi datasetnya sesuai dengan data GENRE yang telah ada. dengan menggunakan NUMPY pada perintah selanjutnya untuk melakukan pembagian data dan dikonversikan menjadi data one-hot encoding.

hasil dari penggunaan perintah generate\_features\_and\_labels dapat dilihat pada gambar

```

In [9]: features, labels = generate_features_and_labels()
Processing 100 songs in blues genre...
Processing 100 songs in classical genre...
Processing 100 songs in country genre...
Processing 100 songs in folk genre...
Processing 100 songs in hiphop genre...
Processing 100 songs in jazz genre...
Processing 100 songs in metal genre...
Processing 100 songs in pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in rock genre...

```

**Gambar 6.46** Code Program generate\_features\_and\_labels

5. Penjelasan tentang Kenapa proses pada generate\_features\_and\_labels lama

```

1 #%%
2 features, labels = generate_features_and_labels()

```

pada bagian penampilan hasil tersebut bisa lama dikarenakan data yang diolah itu tidaklah sedikit yang artinya memiliki data yang banyak, dimana masing - masing data yang diolah akan dilakukan cek terlebih dahulu untuk menentukan GENRE yang sesuai dan data tersebut akan dibagi menjadi 100 data yang telah dikonversikan menjadi data one-hot encoding.

6. Penjelasan tentang pembagian data training sebesar 80 persen.

```

1 #%%
2 training_split = 0.8

```

pemisahan data tersebut adalah untuk memastikan bahwa sistem telah siap untuk dilakukan uji coba dengan benar, dimana sistem tersebut telah dilakukan test dengan menggunakan testing data sehingga data yang akan didapatkan nanti dari penggunaan training data tidaklah terlalu jauh untuk nilai akhirnya. hasil dari penggunaan training\_split dapat dilihat pada gambar

|                |       |   |     |
|----------------|-------|---|-----|
| training_split | float | 1 | 0.8 |
|----------------|-------|---|-----|

**Gambar 6.47** Hasil Code Program training\_split

dan berikut ini adalah hasil akhir dari data yang telah dipisahkan menjadi data TEST dan TRAIN dengan data tersebut sudah dilakukan SHUFFLE (acak) terdapat pada gambar.

|              |         |             |                                                                  |
|--------------|---------|-------------|------------------------------------------------------------------|
| test         | Flatten | (200, 2000) | [1.0 -0.0338907 -0.6471001 -0.7162555 ... 0. 0.]                 |
| test_input   | Flatten | (200, 2000) | [1.0 -0.0338907 -0.6471001 -0.7162555 ... -0.4902487 -0.8140887] |
| test_labels  | Flatten | (200, 10)   | [1.0 0. 0. 0. 0. 0. 0. 0. 0. 0.]                                 |
| train        | Flatten | (200, 2000) | [1.0 -0.0338907 -0.3289002 -0.3881331 ... 0. 0.]                 |
| train_input  | Flatten | (200, 2000) | [1.0 -0.0338907 -0.3289002 -0.3881331 ... 0.21508992 0.8587448]  |
| train_labels | Flatten | (200, 10)   | [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]                                  |

Gambar 6.48 Hasil Code Program training\_split

## 7. Penjelasan parameter fungsi Sequential

```

1 #%%
2 model = Sequential([
3 Dense(100, input_dim=np.shape(train_input)[1]),
4 Activation('relu'),
5 Dense(10),
6 Activation('softmax'),
7])

```

fungsi Sequential pada code program ini adalah untuk mengolah data inputan agar sesuai dengan fungsi - fungsi yang ada pada code tersebut, dimana data yang akan diolah akan dilakukan pembentukan terlebih dahulu dengan menggunakan perintah pada Library NUMPY yaitu SHAPE dengan data yang digunakan adalah train\_input.

pada gambar dibawah ini adalah keluaran dari RUNNING code tersebut

```

In [47]: model = Sequential([
...: Dense(100, input_dim=np.shape(train_input)[1]),
...: Activation('relu'),
...: Dense(10),
...: Activation('softmax'),
...:])

```

Gambar 6.49 Hasil Code fungsi Sequential

## 8. Penjelasan parameter fungsi Compile

```

1 #%%
2 model.compile(optimizer='adam',
3 loss='categorical_crossentropy',
4 metrics=['accuracy'])

```

fungsi Compile digunakan untuk melakukan proses yang akan mengecek data parameter yang akan digunakan dari data yang telah diolah dari proses Sequential. hasil dari RUNNING pada fungsi Compile dapat dilihat pada gambar

```

In [18]: model.compile(optimizer='adam',
...: loss='categorical_crossentropy',
...: metrics=['accuracy'])

```

Gambar 6.50 Hasil Code fungsi Compile

pada gambar dibawah ini adalah hasil dari penjelasan tentang data Sequential dan Compile

```
In [19]: print(model.summary())
Layer (type) Output Shape Param #
dense_1 (Dense) (None, 100) 2500100
activation_1 (Activation) (None, 100) 0
dense_2 (Dense) (None, 10) 1010
activation_2 (Activation) (None, 10) 0
==
Total params: 2,501,110
Trainable params: 2,501,110
Non-trainable params: 0
None
```

**Gambar 6.51** Hasil Code Program Summary

### 9. Penjelasan parameter fungsi Fit

```
1 #%%
2 model.fit(train_input, train_labels, epochs=10, batch_size
3 =32,
4 validation_split=0.2)
```

fungsi Fit digunakan untuk mengolah data dari 10 label data menjadi 10 File datasets, yang kemudian akan dihitung untuk mengukur tingkat akurasi penilaianya untuk keakuratan data yang diolah serta tingkat Loss (gagal) data yang terjadi. hasil dari RUNNING fungsi Fit dapat dilihat pada gambar.

```
In [20]: model.fit(train_input, train_labels, epochs=10, batch_size=32)
/usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/resource_loader.py:243: ResourceWarning: Using a non-absolute file path in the TensorFlow runtime! Please see https://www.tensorflow.org/api_docs/python/tf/compat/v1/resource_loader/resource_path for more details.
 warnings.warn("Using a non-absolute file path in the TensorFlow runtime! Please see %s for more details." % https://www.tensorflow.org/api_docs/python/tf/compat/v1/resource_loader/resource_path)
Instructions for updating:
 Use os.path.abspath(path) to get an absolute path.
/usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/resource_loader.py:243: ResourceWarning: Using a non-absolute file path in the TensorFlow runtime! Please see https://www.tensorflow.org/api_docs/python/tf/compat/v1/resource_loader/resource_path for more details.
 warnings.warn("Using a non-absolute file path in the TensorFlow runtime! Please see %s for more details." % https://www.tensorflow.org/api_docs/python/tf/compat/v1/resource_loader/resource_path)
Epoch 1/10
Train on 480 samples, validate on 108 samples
 50/50 [=====] - 0s - loss: 2.2412 - acc: 0.2625 - val_loss: 2.7679 -
 50/50 [=====] - 0s - acc: 0.3555
Epoch 2/10
 50/50 [=====] - 0s - loss: 2.4655 - acc: 0.4794 - val_loss: 1.5051 -
 50/50 [=====] - 0s - acc: 0.5877
Epoch 3/10
 50/50 [=====] - 0s - loss: 1.1455 - acc: 0.8536 - val_loss: 0.4998 -
 50/50 [=====] - 0s - acc: 0.9101
Epoch 4/10
 50/50 [=====] - 0s - loss: 0.8775 - acc: 0.7154 - val_loss: 1.5995 -
 50/50 [=====] - 0s - acc: 0.6775
Epoch 5/10
 50/50 [=====] - 0s - loss: 0.7528 - acc: 0.8899 - val_loss: 1.4058 -
 50/50 [=====] - 0s - acc: 0.8446
Epoch 6/10
 50/50 [=====] - 0s - loss: 0.5844 - acc: 0.9179 - val_loss: 1.5387 -
 50/50 [=====] - 0s - acc: 0.8868
Epoch 7/10
 50/50 [=====] - 0s - loss: 0.4848 - acc: 0.9265 - val_loss: 1.4789 -
 50/50 [=====] - 0s - acc: 0.9113
Epoch 8/10
 50/50 [=====] - 0s - loss: 0.3889 - acc: 0.9464 - val_loss: 1.4472 -
 50/50 [=====] - 0s - acc: 0.9388
Epoch 9/10
 50/50 [=====] - 0s - loss: 0.2946 - acc: 0.9796 - val_loss: 1.4259 -
 50/50 [=====] - 0s - acc: 0.9546
Epoch 10/10
 50/50 [=====] - 0s - loss: 0.2108 - acc: 0.9823 - val_loss: 1.5164 -
 50/50 [=====] - 0s - acc: 0.9623
/usr/local/lib/python3.6/dist-packages/callbacks/Histroy at 0x7f94d27430
```

**Gambar 6.52** Hasil Code fungsi Fit

### 10. Penjelasan parameter fungsi Evaluate

```
1 #%%
2 loss, acc = model.evaluate(test_input, test_labels,
3 batch_size=32)
```

fungsi Evaluate digunakan untuk mengevaluasi data yang telah diolah dengan menggunakan perintah fungsi Sequential, Compile dan juga Fit. untuk hasilnya dapat dilihat pada gambar.

```
[In [21]: loss, acc = model.evaluate(test_input, test_labels, batch_size=32)
280/280 [=====] - 0s - 440ms/step
```

**Gambar 6.53** Hasil Code fungsi Evaluate

```

1 #%%
2 print("Done!")
3 print("Loss: %.4f, accuracy: %.4f" % (loss, acc))

```

untuk data yang ditampilkan pada gambar dibawah ini adalah data yang telah disusun untuk membandingkan tingkat Akurasi dan Loss yang terjadi.

```

In [22]: print("Done!")
 print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
Done!
Loss: 1.4620, accuracy: 0.5350

```

**Gambar 6.54** Hasil Code fungsi Evaluate

## 11. Penjelasan parameter fungsi Predict

```

1 #%%
2 model.predict(train_input [:1])

```

fungsi Predict adalah untuk membandingkan tingkat Akurasi dari data pada setiap label yang ada pada dataset GENRE dimana data akurasi nilainya yang tertinggi maka akan diambil sebagai hasil akhir untuk nilai akurasi Predict. hasil RUNNING dapat dilihat pada gambar

```

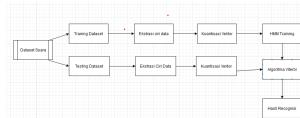
In [21]: model.predict(test_input [:1])
Out[21]:
array([3.4487510e-01, 0.3595807e-01, 0.3846527e-07, 1.2933597e-05,
 3.989064e-05, 4.0385259e-02, 1.0100836e-06, 7.5218845e-07,
 2.2499446e-05, 0.6817698e-07]], dtype=float32)

```

**Gambar 6.55** Hasil Code fungsi Predict

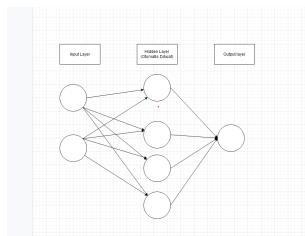
## 6.4 Luthfi Muhammad Nabil (1174035)

1. Jelaskan kenapa file suara harus dilakukan MFCC. Dilengkapi dengan ilustrasi atau gambar Karena MFCC dibutuhkan untuk mengidentifikasi jenis suara. Karena perlunya pengambilan ciri dari suara yang didapat seperti sinyal suara, frekuensi dan gelombang suara untuk dapat diubah menjadi beberapa input lain (Konversi). Karena perlunya komputer untuk mendapat inputan tertentu, maka metode MFCC diperlukan untuk mengkonversi menjadi sesuatu yang dibutuhkan.



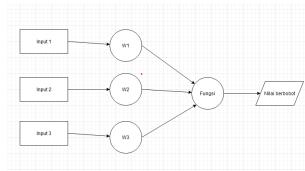
**Gambar 6.56** Illustrasi MFCC

2. Jelaskan konsep dasar neural network. Dilengkapi dengan ilustrasi atau gambar. Neural Network merupakan kumpulan algoritma yang terinspirasi konsepnya dari otak manusia. Konsep tersebut diantaranya mempelajari pola - pola yang diberikan oleh inputan lain. Untuk menerima data, biasanya Neural Network mendapatkannya dari inputan sensor atau komputasi otomatis yang sudah disediakan oleh programmer untuk melakukan input otomatis ke media Neural Network. Data yang diterima berupa numerik yang terdapat pada vektor yang sesuai dengan data berwujud aslinya. Neural network mengkluster dan menklasifikasikan data untuk dapat disesuaikan dengan parameter - parameter yang sudah ada.



**Gambar 6.57** Konsep Dasar Neural Network

3. Jelaskan konsep pembobotan dalam neural network. Dilengkapi dengan ilustrasi atau gambar. Pembobotan dalam neural network yaitu digunakan untuk membedakan objek inputan atau variabel inputan untuk AI misalkan apel dan jeruk digunakan untuk variabel inputan maka dibuat pembobotan antara kedua benda tersebut untuk menentukan output yang pasti dari inputan yang dilakukan untuk lebih jelasnya dapat dilihat pada gambar.



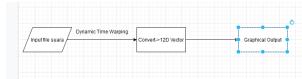
**Gambar 6.58** Pembobotan Neural Network

4. Jelaskan konsep fungsi aktifasi dalam neural network. dilengkapi dengan ilustrasi atau gambar. Fungsi aktivasi merupakan fungsi yang digunakan untuk mendapatkan output neuron dari nilai inputnya. Fungsi aktivasi akan melakukan penilaian jika output yang dikeluarkan telah mencapai hasil yang telah ditentukan.



**Gambar 6.59** Fungsi Aktivasi

- Jelaskan cara membaca hasil plot dari MFCC,dilengkapi dengan ilustrasi atau gambar. Untuk membaca hasil plotting dari MFCC diantaranya menentukan terlebih dahulu batas minimal gelombang suara (Hz) dan batas maksimal dari suara tersebut. Lalu warna yang paling pekat merupakan hasil dari pengolahan data tersebut. Saat membaca file, gelombang akan diklasifikasikan dengan cara mengkalkulasikan jarak diantara vektor 12 Dimensi.



**Gambar 6.60** Metode pembacaan plot MFCC

- Jelaskan apa itu one-hot encoding,dilengkapi dengan ilustrasi kode dan atau gambar. One-hot encoding merupakan numerisasi nilai untuk dapat mengindikasikan sebuah status. Karena butuhnya mesin untuk membaca nilai binary, biasanya mesin akan mengkonversi (Decode) terlebih dahulu untuk dapat membaca nilai. Namun saat nilai dilakukan Encoding dengan metode ini, maka nilai tidak perlu di encode lagi karena nilai yang ada sudah merupakan nilai binary.

| Label Encoding |          |           | One-hot Encoding |           |      |            |
|----------------|----------|-----------|------------------|-----------|------|------------|
| Type Itemname  | Kategori | Kode Gata | Daftar Item      | Coca Cola | Coke | Kalbe Gula |
| Ex. 1st Name   |          | 10        | >                | 1         | 0    | 0          |
| COKE           | 2        | 10        |                  | 0         | 1    | 0          |
| Coca           | 3        | 00        |                  | 0         | 0    | 1          |

**Gambar 6.61** One Hot Encoding

- Jelaskan apa fungsi dari np.unique dan to categorical dalam kode program,dilengkapi dengan ilustrasi atau gambar. Fungsi ini akan mengembalikan array dengan elemen yang berbeda - beda dalam input array. Fungsi ini akan memangkas nilai sehingga hanya beberapa nilai saja yang ada dan nilai tersebut harus berbeda dengan yang lainnya. Biasanya fungsi ini untuk mengidentifikasi jenis nilai atau nilai apa saja yang terdapat pada array yang bersangkutan. to categorical merupakan fungsi untuk mengkonversikan dataset menjadi sebuah beberapa kategori kelas yang ditentukan. Nilai itu sendiri akan diubah menjadi bentuk 'One-hot vector' untuk dapat dibaca oleh komputer.

Contoh Program :

```
import numpy as np
import keras.utils as ku
arr = [1,5,2,3,3,2,3,3,3,1,2,3,4,5,1,2,5,3,7,8,2,3,7,5,7,2]
print("Awalnya Gini")
print(arr)
#NP Unique akan melepas nilai yang sama
print("Jadinya gini kalau pake np.unique")
print(np.unique(arr))
print("Pakai To Categorical (From keras.utils)")
print(ku.to_categorical(np.unique(arr), num_classes=None))
```

```
hasIngs Gini
[[1, 5, 2, 3, 3, 2, 3, 3, 3, 1, 2, 3, 4, 5, 1, 2, 3, 5, 3, 7, 6, 2, 3, 7, 5, 1, 2,
Jadinya gini kalau pakai rgn.unique
[1 2 3 4 5 6 7]]
```

**Gambar 6.62** Output Contoh program to categorical dan np unique

8. Jelaskan apa fungsi dari Sequential dalam kode program,dilengkapi dengan ilustrasi atau gambar. Sequential berfungsi untuk mencari nilai yang dicari berdasarkan pencarian berurutan sesuai numeric terendah ke tertinggi atau sebaliknya. Sequential dapat digunakan dalam pencarian data vektor untuk mengetahui apakah nilai ada atau tidak. Karena vektor isinya berupa nilai numeric, maka pencarian sequential dapat dijadikan sebagai salah satu metode untuk mencari data tersebut.

## Contoh Program :

```
target = 7
arr = [1,5,2,3,3,2,3,3,3,1,2,3,4,5,1,2,5,3,7,8,2,3,7,5,7,2]
for x in range(0, len(arr)):
if target == arr[x]:
 print("\nNilai "+str(target)+" ada")
 break
```

## Nilai 7 ada

**Gambar 6.63** Contoh Sequential programming

#### 6.4.1 Praktek

1. Jelaskan isi dari data GTZAN Genre Collection dan data dari freesound. Buat kode program untuk meload data tersebut untuk digunakan.

nakan pada MFCC. Jelaskan arti dari perbaris kode yang dibuat (harus beda dengan teman satukelas).

Isi data data merupakan datasets lagu atau suara yang terdiri dari 10 genre yang di simpan kedalam 10 folder yaitu folder blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, dan rock ke sepuluh folder. Masing - masing dari folder terdapat 100 data suara sedangkan data freesound merupakan contoh data suara yang akan di gunakan untuk menguji hasil pengolahan data tersebut dengan menggunakan metode mfcc. hal tersebut untuk mengetahui apakah suara dari freesound termasuk kategori jazz pop atau sebagainya ?.

```

1 import librosa
2 import librosa.feature
3 import librosa.display
4 import glob
5 import numpy as np
6 import matplotlib.pyplot as plt
7 from keras.layers import Dense, Activation
8 from keras.models import Sequential
9 from keras.utils.np_utils import to_categorical
10
11 # In [1]: buat fungsi mfcc untuk testomg
12 def display_mfcc(song):
13 y, _ = librosa.load(song)
14 mfcc = librosa.feature.mfcc(y)
15
16 plt.figure(figsize=(10, 4))
17 librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
18 plt.colorbar()
19 plt.title(song)
20 plt.tight_layout()
21 plt.show()
```

dapat dilihat pada kode diatas pada baris kesatu dilakukan import librosa tang digunakan untuk fungsi mfcc pada suara. pada baris kedua dilakukan import librosa featuse dan pada baris ke tiga dilakukan librosa display selanjutnya pada baris ke empat dilakukan import glob kemudian insert numpy untuk pengolahan data menjadi vektor setelah itu dilakukan import matplotlib untuk melakukan ploting setelah itu dilakukan import librari keras.

Selanjutnya yaitu membuat fungsi mfcc dengan nama display mfcc yang didalamnya terdapat variabel y yang berisi method librosa load kemudian variabel mfcc yang berisi method librosa featurea mfcc. Setelah itu membuat plot figure dengan ukuran 10 banding 4 kemudian di isi oleh data librosa display dengan variabel x nya yaitu waktu dan y yaitu mel atau Hz kemudian melakukan plot warna setelah itu melakukan plot judul dan terakhir flot di tampilkan.

2. Jelaskan perbaris kode program dengan kata-kata dan di lengkapi ilustrasi gambar fungsi dari display mfcc().

```

1 # In [2]: cek fungsi
2 display_mfcc('disco.00068.au')
3 # In [2]: cek fungsi
4 display_mfcc('disco.00068.au')
5 # In [2]: cek fungsi
6 display_mfcc('jazz.00068.au')
7 # In [2]: cek fungsi
8 display_mfcc('blues.00068.au')
9 # In [2]: cek fungsi
10 display_mfcc('classical.00068.au')
11 # In [2]: cek fungsi
12 display_mfcc('country.00068.au')
13 # In [2]: cek fungsi
14 display_mfcc('hiphop.00068.au')
15 # In [2]: cek fungsi
16 display_mfcc('jazz.00068.au')
17 # In [2]: cek fungsi
18 display_mfcc('reggae.00068.au')
19 # In [2]: cek fungsi
20 display_mfcc('reggae.00068.au')
21 # In [2]: cek fungsi
22 display_mfcc('rock.00068.au')
```

pada baris ke dua program diatas digunakan untuk mendisplay tampilan glombang suara dari file disco.00067.wav menggunakan metode mfcc dengan menggunakan fungsi display mfcc yang telah tadi di buat pada nomer dua begitu juga pada baris ke 4 6 8 sampai ke 22 secara teksis sama menggunakan fungsi display mfcc hanyasaja beda penyimpanan data yang akan di tampilkan atau di eksekusi.

```

In [2]: def display_mfcc(song):
...: y, _ = librosa.load(song)
...: mfcc = librosa.feature.mfcc(y)
...: plt.figure(figsize=(10, 4))
...: librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
...: plt.colorbar()
...: plt.title(song)
...: plt.tight_layout()
...: plt.show()
```

**Gambar 6.64** Ilustrasi gambar fungsi dari displaymfcc()

3. Jelaskan perbaris dengan kata-kata dan dilengkapi dengan ilustrasi gambar fungsi dari extract features song jelaskan kenapa data yang diambil merupakan data 25.000 baris pertama ?

```

1 # In [3]: fitur ekstraksi
2
3 def extract_features_song(f):
4 y, _ = librosa.load(f)
5
6 # get Mel-frequency cepstral coefficients
7 mfcc = librosa.feature.mfcc(y)
8 # normalize values between -1,1 (divide by max)
9 mfcc /= np.absolute(np.amax(mfcc))
```

```

10
11 return np.ndarray.flatten(mfcc)[:25000]

```

pada baris ke tiga di definisikan nama extract features song yang nantinya akan di gunakan pada fungsi yang lainnya kemudian dibuat variabel y dengan method librosa load setelah itu dibuat variabel baru mfcc dengan isi librosa features mfcc dengan isi variabel y tadi kemudian dibuat variabel mfcc dengan isian np.max dan variabel mfcc tadi terakhir di buat array dari data tersebut merupakan data 25000 data pertama. kenapa data 25000 pertama yang digunakan dikarenakan data tersebut digunakan sebagai data testing semakin besar data testing yang di gunakan maka semakin akurat hasil AI. tapi sebenarnya data tersebut relatif bisa lebih besar atau lebih kecil tergantung pada komputer masing masing.

4. Jelaskan Perbaris kode program dengan kata-kata dan di lengkapi ilustrasi gambar fungsi dari generate features and labels.

```

1 # In [3]: fitur_ekstraksi
2
3 def generate_features_and_labels():
4 all_features = []
5 all_labels = []
6
7 genres = ['blues' , 'classical' , 'country' , 'disco' , 'hiphop' ,
8 'jazz' , 'metal' , 'pop' , 'reggae' , 'rock']
9 for genre in genres:
10 sound_files = glob.glob(genre+'/*.au')
11 print('Processing %d songs in %s genre...' % (len(
12 sound_files), genre))
13 for f in sound_files:
14 features = extract_features_song(f)
15 all_features.append(features)
16 all_labels.append(genre)
17
18 # convert labels to one-hot encoding cth blues :
19 # 1000000000 classic 0100000000
20 label_uniq_ids, label_row_ids = np.unique(all_labels,
21 return_inverse=True) #ke integer
22 label_row_ids = label_row_ids.astype(np.int32, copy=False)
23
24 onehot_labels = to_categorical(label_row_ids, len(
25 label_uniq_ids)) #ke one hot
26
27 return np.stack(all_features), onehot_labels

```

pada baris ke tiga merupakan pendefinisian nama fungsi yaitu generate features and labels kemudian membuat variabel baru dengan array kosong yaitu all features dan all labels kemudian mendefinisikan isian label untuk gendre dengan cara membuat variabel genres kemudian di isi dengan 10 gendre yang tadi setelah itu dilakukan fungsi if else dengan code for dan in setelah itu akan di buat encoding untuk data tiap tiap label contoh untuk blues 1000000000 dan untuk clasical 0100000000.

5. Jelaskan dengan kata dan praktek kenapa penggunaan fungsi generate features and labels sangat lama saat meload dataset gendre tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

halnini menjadi lama dikarenakan mesin membaca satupersatu file yang ada pada folder dan dalam foldertersebut terdapat 100 file sehingga wajar menjadi lama ditambah lagi mengolah data yang tadinya suara menjadi bentuk vektor. berikut merupakan codenya.

```
1 # In [3]: passing parameter dari fitur ekstraksi menggunakan
 mfc
2
3 features , labels = generate_features_and_labels()
```

```
In [24]: def generate_features_and_labels():
...: all_features = []
...: all_labels = []
...: genres = ['blues', 'classical', 'country', 'disco', 'hiphop',
...: 'jazz', 'metal', 'pop', 'reggae', 'rock']
...: for genre in genres:
...: print("Processing %d songs in %s genre..." % (len(sound_file[genre]), genre))
...: for sound_file in sound_file[genre]:
...: features = extract_features(songF)
...: all_features.append(features)
...: all_labels.append(genre)
...: # convert labels to one-hot encoding (thi blaaa : 2000000000
...: class_label = np.eye(len(genres))
...: label_uniq_ids, label_row_ids = np.unique(all_labels,
...: return_inverse=True)
...: onehot_labels = np.zeros((len(label_uniq_ids), len(class_label)))
...: onehot_labels[0] = class_label[all_labels[0]]
...: for i in range(1, len(label_uniq_ids)):
...: onehot_labels[i] = class_label[all_labels[i]]
...: onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids))
...: return np.stack(all_features), onehot_labels

In [25]: features, labels = generate_features_and_labels()
Processsing 1 songs in classical genre...
Processsing 1 songs in country genre...
Processsing 1 songs in disco genre...
Processsing 1 songs in hiphop genre...
Processsing 1 songs in jazz genre...
Processsing 8 songs in metal genre...
Processsing 1 songs in pop genre...
Processsing 1 songs in reggae genre...
Processsing 1 songs in rock genre...
```

**Gambar 6.65** Hasil dari fungsi generate features and labels

6. jelaskan kenapa harus dilakukan pemisahan data training dan data testing sebesar 80 persen praktekan dengan kode dan tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

untuk code nya adalah sebagai berikut yang merupakan code untuk membagi data sebanyak 80 persen untuk data training maka data musik tadi yang total jumlahnya 1000 akan di bagi dua untuk data training sebanyak 800 dan 200 untuk data testing.

```
1 # In [3]: fitur ekstraksi
2 training_split = 0.8
```

```
In [76]: training_split = 0.8
```

**Gambar 6.66** Hasi eksekusi code

|       |                    |                                      |
|-------|--------------------|--------------------------------------|
| test  | float32 (2, 25000) | [[-0.5687581 -0.6273657 -0.7698...   |
| train | float32 (6, 25000) | [[-0.18076201 -0.21068698 -0.2808... |

**Gambar 6.67** Hasil pembagian 80 persen

7. praktekan dan jelaskan masing masing parameter dari fungsi Sequential(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

fungsi sequential digunakan untuk mengolah data inputan sesuai dengan fungsi yang ada pada fungsi sequential pada fungsi sequential kali ini menggunakan dua fungsi. fungsi sequential mengkompile data dari 100 neuron atau dari 1 folder file dengan menggunakan fungsi relu dan softmax untuk menghasilkan outputan yang sesuai dengan keriteria.

```
1 # In [3]: membuat seq NN, layer pertama dense dari 100 neurons
2 model = Sequential([
3 Dense(100, input_dim=np.shape(train_input)[1]),
4 Activation('relu'),
5 Dense(10),
6 Activation('softmax'),
7])
```

```
In [84]: print(np.shape(train_input))
...: print(np.shape(train_labels))
(6, 24998)
(6, 10)
```

**Gambar 6.68** Hasil fungsi sequential

8. praktekan dan jelaskan masing masing parameter dari fungsi compile(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

yaitu fungsi kompile yang digunakan untuk mengetahui parameter yang digunakan dari data yang telah diolah untuk caranya dapat menggunakan codingan sebagai berikut dan untuk hasilnya akan memunculkan parameternya berpasaja dan total parameter yang digunakan.

```
1 # In [3]: fitur ekstraksi
2 model.compile(optimizer='adam',
3 loss='categorical_crossentropy',
4 metrics=['accuracy'])
5 print(model.summary())
```

```
In [48]: model.compile(optimizer='adam',
...: ...
...: loss='categorical_crossentropy',
...: metrics=['accuracy'])
Model: "sequential_1"
Layer (Type) Output Shape Param #
dense_1 (Dense) (None, 100) 2499900
activation_1 (Activation) (None, 100) 0
dense_2 (Dense) (None, 10) 1010
activation_2 (Activation) (None, 10) 0
=====
Total params: 2,500,910
Trainable params: 2,500,910
Non-trainable params: 0
None
```

**Gambar 6.69** Hasil fungsi compile

9. praktekan dan jelaskan masing masing parameter dari fungsi fit(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

pada fungsi ini dilakukan pengolahan data dari 10 label tadi atau 10 file data sets tadi kemudian di hitung tingkat akurasi masing masing dan tingkat kegagalan atau loss data dari setiap file tersebut caranya dengan melakukan codingan berikut.data menunjukan 10 pengolahan data untuk menentukan nilai akurasi dan loss dari data tersebut dan selanjutnya dilakukan fingsi evaluasi.

```
1 # In [3]: fitur_ekstraksi
2 model.fit(train_input, train_labels, epochs=10, batch_size
3 =32,
4 validation_split=0.2)
```

```
In [89]: model.fit(train_input, train_labels, epochs=10, batch_size=32,
... validation_split=0.2)
Train on 4 samples, validate on 2 samples
Epoch 1/10
4/4 [=====] - 0s 47ms/step - loss: 2.4867 -
accuracy: 0.0000e+00 - val_loss: 4.8689 - val_accuracy: 0.0000e+00
Epoch 2/10
4/4 [=====] - 0s 47ms/step - loss: 2.4867 -
accuracy: 0.0000e+00 - val_loss: 4.8689 - val_accuracy: 0.0000e+00
Epoch 3/10
4/4 [=====] - 0s 47ms/step - loss: 2.4867 -
accuracy: 0.0000e+00 - val_loss: 4.9378 - val_accuracy: 0.0000e+00
Epoch 4/10
4/4 [=====] - 0s 47ms/step - loss: 2.4867 -
accuracy: 0.0000e+00 - val_loss: 5.0282 - val_accuracy: 0.0000e+00
Epoch 5/10
4/4 [=====] - 0s 47ms/step - loss: 0.4626 -
accuracy: 0.0000e+00 - val_loss: 6.2392 - val_accuracy: 0.0000e+00
Epoch 6/10
4/4 [=====] - 0s 47ms/step - loss: 0.3341 -
accuracy: 1.0000 - val_loss: 6.7464 - val_accuracy: 0.0000e+00
Epoch 7/10
4/4 [=====] - 0s 47ms/step - loss: 0.3341 -
accuracy: 1.0000 - val_loss: 7.1442 - val_accuracy: 0.0000e+00
Epoch 8/10
4/4 [=====] - 0s 47ms/step - loss: 0.3045 -
accuracy: 1.0000 - val_loss: 7.5665 - val_accuracy: 0.0000e+00
Epoch 9/10
4/4 [=====] - 0s 47ms/step - loss: 0.2021 -
accuracy: 1.0000 - val_loss: 7.9958 - val_accuracy: 0.0000e+00
Epoch 10/10
4/4 [=====] - 0s 47ms/step - loss: 0.2977 -
accuracy: 1.0000 - val_loss: 8.4281 - val_accuracy: 0.0000e+00
Epochs 10/10
4/4 [=====] - 0s 47ms/step - loss: 0.2914 -
accuracy: 1.0000 - val_loss: 8.8349 - val_accuracy: 0.0000e+00
Out[89]: <keras.callbacks.History at 0x1d2d52b0c8c>
```

**Gambar 6.70** Hasil fungsi fit

10. praktekan dan jelaskan masing masing parameter dari fungsi evaluate(). tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

pada fungsi ini dilakukan evaluasi terhadap datayang telah di runing sebelumnya untuk lebih jelasnya dapat di lihat codingan tersebut pada codingan tersebut dilakukan evaluasi pada tingkat kegagalan dan akurasi kebenaran maka hasilnya akan memunculkan hasil evaluasi dari 10 proses dari setiap gendre yaitu akurasi sebesar 51 persen dan loss data sebesar 1.4105 data.

```
1 # In [3]: fitur_ekstraksi
2 loss, acc = model.evaluate(test_input, test_labels,
3 batch_size=32)
4 # In [3]: fitur_ekstraksi
5 print("Done!")
6 print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
```

```
In [90]: loss, acc = model.evaluate(test_input, test_labels,
batch_size=32)
2/2 =====
In [91]: print("Done!")
... print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
Done!
Loss: 9.2173, accuracy: 0.0000
```

**Gambar 6.71** Hasil fungsi evaluasi

11. praktekan dan jelaskan masing masing parameter dari fungsi predic tunjukan keluarannya dari komputer sendiri dan artikan maksud dari luaran tersebut.

fungsi predic merupakan fungsi untuk membandingkan tingkat akurasi pada setiap label yang sepuluh tadi maka data akan di sandingkan ke masing masing tingkat akurasinya, yang akurasinya paling tinggi maka itulah jawaban untuk setiap inputan yang dilakukan.

```
1 # In [3]: fitur_ekstraksi
2 model.predict(test_input [:1])
```

```
In [92]: model.predict(test_input[:1])
Out[92]:
array([13.6887997e-04, 1.966062e-03, 7.4016742e-02, 6.5938862e-02,
2.6221142e-09, 5.2641005e-02, 8.0541098e-01, 6.1763829e-05,
1.1412354e-04, 5.92360095e-05]), dtype=float32)
```

**Gambar 6.72** Hasil fungsi prediksi

#### 6.4.2 Skrinsut error

1. Tipe Error : ValueError
2. Cara Penanganan : Pastikan array sudah tersedia, karena kasus yang ada disini merupakan kasus pengambilan file, maka cek filenya apakah ada atau tidak

```
#Before
sound_files = glob.glob(genre+'/*.sau')
#after
sound_files = glob.glob(genre+'/*.au')
#Note : Tidak ada file .sau di folder, maka diganti eksten
```

3. Skrinsut Error

```
onehot_labels = to_categorical(label_row_ids,
len(label_row_ids))#ke one hot
file "D:\Software\Anaconda\lib\site-packages\keras\utils\ng_utils.py",
line 49, in to_categorical
num_classes = np.max(y) + 1
file "D:\Software\Anaconda\lib\site-packages\numpy\core\fromnumeric.py",
line 100, in mean
initial=initial)
file "D:\Software\Anaconda\lib\site-packages\numpy\core\fromnumeric.py",
line 86, in _pre_reduction
return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
ValueError: zero-size array to reduction operation maximum which has no
identity
```

**Gambar 6.73** Skrinsut error

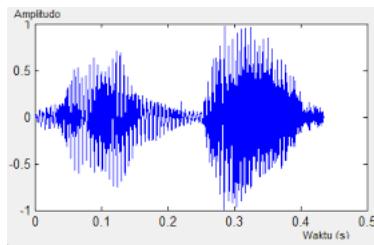
## 6.5 1174050 Dika Sukma Pradana

### 6.5.1 Teori

1. Kenapa file suara harus di lakukan MFCC. dilengkapi dengan ilustrasi atau gambar.

Nilai-nilai MFCC meniru pendengaran manusia dan mereka biasanya digunakan dalam aplikasi pengenalan suara serta genre musik deteksi. Nilai-nilai MFCC ini akan dimasukkan langsung ke jaringan saraf. Agar dapat diubah menjadi bentuk vektor, dan dapat digunakan pada machine learning. Disebabkan machine learning hanya mengerti bilangan vektor saja.

Ilustrasinya, Ketika ingin menggunakan file suara dalam machine learn-



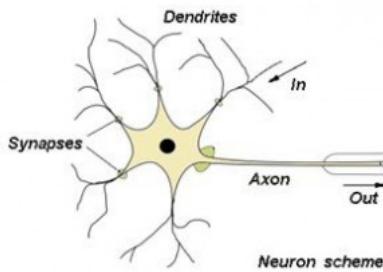
**Gambar 6.74** Contoh MFCC

ing, misalnya untuk melihat jam. Machine learning tidak memahami rekaman suara melainkan vektor. Maka rekaman tersebut akan diubah kedalam bentuk vektor kemudian vektor akan menyesuaikan dengan kata kata yang sudah disediakan. Jika cocok maka akan mengembalikan waktu yang diinginkan

2. Konsep dasar neural network dilengkapi dengan ilustrasi atau gambar

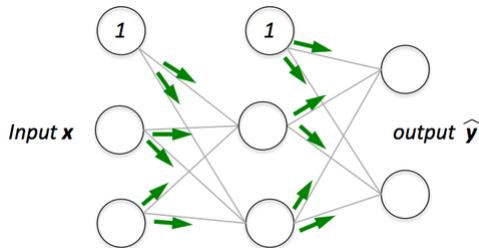
Neural Network ini terinspirasi dari jaringan saraf otak manusia. Dimana setiap neuron terhubung ke setiap neuron di lapisan berikutnya. Lapisan pertama menerima input dan lapisan terakhir memberikan keluaran. Struktur jaringan, yang berarti jumlah neuron dan koneksi, diputuskan sebelumnya dan tidak dapat berubah, setidaknya tidak selama training. Juga, setiap input harus memiliki jumlah nilai yang sama. Ini berarti bahwa gambar, misalnya, mungkin perlu diubah ukurannya agar sesuai dengan jumlah neuron input.

3. Konsep pembobotan dalam neural network.dilengkapi dengan ilustrasi atau gambar



**Gambar 6.75** Contoh Pembobotan Neural Network

Bobot mewakili kekuatan koneksi antar unit. Jika bobot dari node 1 ke node 2 memiliki besaran lebih besar, itu berarti bahwa neuron 1 memiliki pengaruh lebih besar terhadap neuron 2. Bobot penting untuk nilai input. Bobot mendekati nol berarti mengubah input ini tidak akan mengubah output. Bobot negatif berarti meningkatkan input ini akan mengurangi output. Bobot menentukan seberapa besar pengaruh input terhadap output. Seperti contoh berikut :



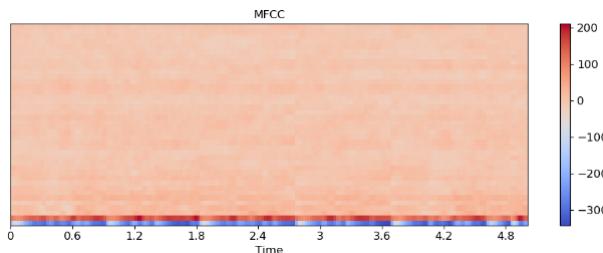
**Gambar 6.76** Contoh Pembobotan Neural Network

4. Konsep fungsi aktifasi dalam neural network. dilengkapi dengan ilustrasi atau gambar

Fungsi aktivasi digunakan untuk memperkenalkan non-linearitas ke jaringan saraf. Ini menekan nilai dalam rentang yang lebih kecil yaitu. fungsi aktivasi Sigmoid memeras nilai antara rentang 0 hingga 1. Ada banyak fungsi aktivasi yang digunakan dalam industri pembelajaran yang dalam dan ReLU, SELU dan TanH lebih disukai daripada fungsi aktivasi sigmoid. Ilustrasinya, ketika fungsi aktivasi linier, jaringan saraf dua lapis mampu mendekati hampir semua fungsi. Namun, jika fungsi aktivasi identik dengan fungsi aktivasi  $F(X) = X$ , properti ini tidak puas, dan jika MLP menggunakan fungsi aktivasi yang sama, seluruh jaringan setara dengan jaringan saraf lapis tunggal.

5. Cara membaca hasil plot dari MFCC,dilengkapi dengan ilustrasi atau gambar

Berikut merupakan hasil plot dari rekaman suara : Dari gambar tersebut



**Gambar 6.77** Cara Membaca Hasil Plot MFCC

dapat diketahui :

- Terdapat 2 dimensi yaitu x sebagai waktu, dan y sebagai power atau desibel.
- Dapat dilihat bahwa jika berwarna biru maka power dari suara tersebut rendah, dan jika merah power dari suara tersebut tinggi
- Dibagian atas terdapat warna pudar yang menandakan bahwa tidak ada suara sama sekali dalam jangkauan tersebut.

6. Jelaskan apa itu one-hot encoding,dilengkapi dengan ilustrasi kode dan atau gambar.

One-hot encoding adalah representasi variabel kategorikal sebagai vektor biner. Mengharuskan nilai kategorikal dipetakan ke nilai integer. Kemudian, setiap nilai integer direpresentasikan sebagai vektor biner yang semuanya bernilai nol kecuali indeks integer, yang ditandai dengan 1.

| Label Encoding   |               |          | One Hot Encoding |   |   |     |
|------------------|---------------|----------|------------------|---|---|-----|
| <b>Food Name</b> |               |          | <b>Apple</b>     |   |   |     |
| Apple            | Categorical # | Calories | 1                | 0 | 0 | 95  |
| Chicken          |               |          | 0                | 1 | 0 | 231 |
| Broccoli         |               |          | 0                | 0 | 1 | 50  |

→

|       |         |          |          |
|-------|---------|----------|----------|
| Apple | Chicken | Broccoli | Calories |
| 1     | 0       | 0        | 95       |
| 0     | 1       | 0        | 231      |
| 0     | 0       | 1        | 50       |

**Gambar 6.78** One Hot Encoding

7. fungsi dari np/.unique dan to categorical dalam kode program,dilengkapi dengan ilustrasi atau gambar.

Untuk np.unique fungsinya yaitu menemukan elemen unik array. Mengembalikan elemen unik array yang diurutkan. Ada tiga output opsional selain elemen unik:

- Indeks array input yang memberikan nilai unik
- Indeks array unik yang merekonstruksi array input
- Berapa kali setiap nilai unik muncul dalam array input.

```
>>> np.unique([1, 1, 2, 2, 3, 3])
array([1, 2, 3])
>>> a = np.array([[1, 1], [2, 3]])
>>> np.unique(a)
array([1, 2, 3])
```

**Gambar 6.79** Numpy Unique

Untuk To Categorical fungsinya untuk mengubah vektor kelas (integer) ke matriks kelas biner.

```
Consider an array of 5 labels out of a set of 3 classes {0, 1, 2}:
> labels
array([0, 2, 1, 2, 0])
`to_categorical` converts this into a matrix with as many
columns as there are classes. The number of rows
stays the same.
> to_categorical(labels)
array([[1., 0., 0.],
 [0., 0., 1.],
 [0., 1., 0.],
 [0., 0., 1.],
 [1., 0., 0.]], dtype=float32)
```

**Gambar 6.80** To Categorical

8. Fungsi dari Sequential dalam kode program,dilengkapi dengan ilustrasi atau gambar.

Sequential berfungsi sebagai tumpukan linear lapisan. COntohnya sebagai berikut :

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(2, input_dim=1))
model.add(Dense(1))
```

**Gambar 6.81** Sequential

## 6.5.2 Praktek

1. Penjelasan isi data GTZAN Genre Collection dan data dari Freesound. Isi data data merupakan datasets lagu atau suara yang terdiri dari 10 genre

yang di simpan kedalam 10 folder yaitu folder blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, dan rock ke sepuluh folder. Masing - masing dari folder terdapat 100 data suara sedangkan data freesound merupakan contoh data suara yang akan di gunakan untuk menguji hasil pengolahan data tersebut dengan menggunakan metode mfcc. Code Yang Digunakan :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:21:58 2020
4
5 @author: User
6 """
7
8 import librosa
9 import librosa.feature
10 import librosa.display
11 import glob
12 import numpy as np
13 import matplotlib.pyplot as plt
14 from keras.layers import Dense, Activation
15 from keras.models import Sequential
16 from keras.utils.np_utils import to_categorical

```

## 2. Penjelasan perbaris kode program dari display MFCC.

- Code Yang Digunakan :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:22:17 2020
4
5 @author: User
6 """
7
8 def display_mfcc(song):
9 y, _ = librosa.load(song)
10 mfcc = librosa.feature.mfcc(y)
11
12 plt.figure(figsize=(10, 4))
13 librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
14 plt.colorbar()
15 plt.title(song)
16 plt.tight_layout()
17 plt.show()

```

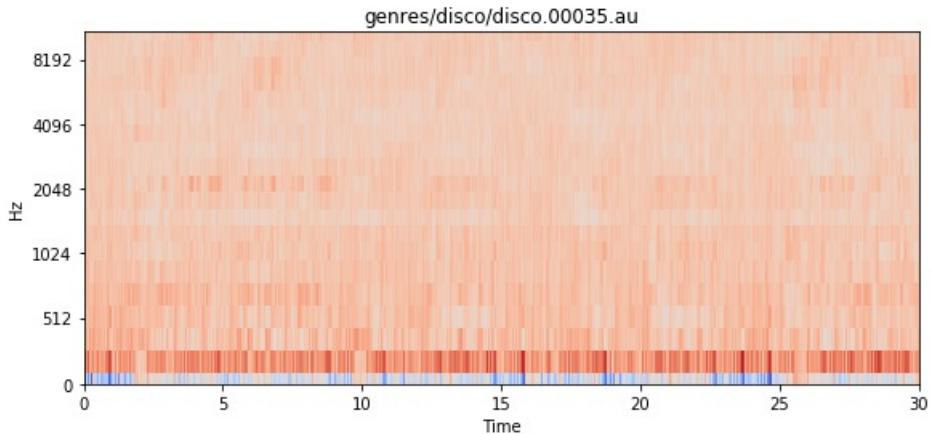
- Penjelasan:

- Baris Code 1: Membuat fungsi display MFCC untuk menampilkan vektorisasi dari sebuah suara dimana variabel parameter.
- Baris Code 2: Membuat variabel Y dimana untuk membaca variable parameter song dari perintah librosa load.

- (c) Baris Code 3: Membuat variabel MFCC untuk memanggil variabel Y dan mengubah suara menjadi vektor.
- (d) Baris Code 4: Melakukan plotting gambar dengan ukuran 10x4 dari figsize.
- (e) Baris Code 5: Menampilkan spektrogram dari library librosa dimana untuk x\_axis didefinisikan dengan time kemudian y\_axis di definisikan dengan mel.
- (f) Baris Code 6: Menambahkan colorbar pada plot yang dijalankan.
- (g) Baris Code 7: Menetapkan atau memberikan judul untuk suara yang dieksekusi.
- (h) Baris Code 8: Untuk menyesuaikan subplot params sehingga subplot cocok dengan area gambar.
- (i) Baris Code 9: Fungsi untuk menampilkan hasil plot dari inputan yang telah dieksekusi.

- Ilustrasi Gambar:

```
In [5]: display_mfcc('genres/disco/disco.00035.au')
```



**Gambar 6.82** Display MFCC

### 3. Penjelasan perbaris code dari Extract Feature Song.

- Code yang digunakan:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:25:02 2020
4
5 @author: User
6 """
```

```

7 def extract_features_song(f):
8 y, _ = librosa.load(f)
9
10
11 # get Mel-frequency cepstral coefficients
12 mfcc = librosa.feature.mfcc(y)
13 # normalize values between -1,1 (divide by max)
14 mfcc /= np.amax(np.absolute(mfcc))
15
16 return np.ndarray.flatten(mfcc)[:25000]

```

▪ Penjelasan Code:

- Baris Code 1 : Membuat fungsi extract feature song dengan inputan parameter f
- Baris Code 2 : Membuat variabel y dimana untuk meload atau membaca inputan parameter f dari perintah librosa load song
- Baris Code 3 : Membuat variabel mfcc yang difungsikan untuk membuat feature dari variabel y berdasarkan library librosa
- Baris Code 4 : Membuat normalisasi nilai antara -1 sampai 1 yang didapatkan dari eksekusi np.absolute
- Baris Code 5 : Didefinisikan untuk mengambil 25000 data pertama berdasarkan durasi suara atau musik lalu dikembalikan salinan arraynya dan dikecilkan menjadi satu.

▪ Ilustrasi Gambar:

```

In [57]: def extract_features_song(f):
...: y, _ = librosa.load(f)
...:
...: # get Mel-frequency cepstral coefficients
...: mfcc = librosa.feature.mfcc(y)
...: # normalize values between -1,1 (divide by max)
...: mfcc /= np.amax(np.absolute(mfcc))
...:
...: return np.ndarray.flatten(mfcc)[:25000]

```

**Gambar 6.83** Extract Features Song

- Mengapa yang diambil merupakan 25.000 baris data pertama?  
Biar supaya tidak terjadi overhead pada komputer atau laptop atau proses eksekusi tidak terlalu lama.

4. Penjelasan perbaris code dari Generate Features and Labels.  
Code yang digunakan:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:25:14 2020
4

```

```

5 @author: User
6 """
7
8 def generate_features_and_labels():
9 all_features = []
10 all_labels = []
11
12 genres = ['blues' , 'classical' , 'country' , 'disco' , 'hiphop' , 'jazz' , 'metal' , 'pop' , 'reggae' , 'rock']
13 for genre in genres:
14 sound_files = glob.glob('genres/' +genre+ '/* au')
15 print('Processing %d songs in %s genre...' % (len(sound_files), genre))
16 for f in sound_files:
17 features = extract_features_song(f)
18 all_features.append(features)
19 all_labels.append(genre)
20
21 # convert labels to one-hot encoding cth blues :
22 1000000000 classic 0100000000
23 label_uniq_ids, label_row_ids = np.unique(all_labels, return_inverse=True)#ke integer
24 label_row_ids = label_row_ids.astype(np.int32, copy=False)
25 onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids))#ke one hot
26 return np.stack(all_features), onehot_labels

```

▪ Penjelasan:

- Baris Code 1: Membuat perintah untuk fungsi generate features and labels.
- Baris Code 2: Pembuatan variabel all features dengan array atau parameter kosong.
- Baris Code 3: Pembuatan variabel all labels dengan array atau parameter kosong.
- Baris Code 4: Mendefinisikan variable genres yang didalamnya berisi nama folder-folder pada variabel genres tersebut.
- Baris Code 5: Membuat perintah fungsi looping.
- Baris Code 6: Membuat atribut sound files yang berisi perintah looping perfolder dari folder genres dan mengambil semua file berekstensi au.
- Baris Code 7: Memunculkan jumlah song yang dieksekusi.
- Baris Code 8: Membuat perintah fungsi dari sound files.
- Baris Code 9: Membuat variabel features untuk memanggil fungsi extract features song (f) sebagai inputan. Setiap satu file array sound files dilakukan ekstrak fitur.
- Baris Code 10: Memasukkan semua features menggunakan perintah append kedalam all features.

- (k) Baris Code 11: Memasukkan semua genres menggunakan perintah append ke dalam all labels.
- (l) Baris Code 12: Mendefinisikan label uniq ids dan label row ids sebagai variabel dimana mengeksekusi perintah np.unique dengan parameter variabelnya all labels dan return inverse=True.
- (m) Baris Code 13: Membuat variabel label row ids untuk menentukan type dari variabel tersebut dengan type bit yang sesuai dengan yang digunakan.
- (n) Baris Code 14: Membuat variabel onehot labels dimana mengeksekusi to\_categorical dengan variabel parameter low row ids dan len.
- (o) Baris Code 15: Mengembalikan dan menampilkan hasil eksekusi dari variabel parameter all features dan onehot labels perintah dari np.stack.

- Ilustrasi Gambar:

```
In [58]: def generate_features_and_labels():
...: all_features = []
...: all_labels = []
...:
...: genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz',
...: 'metal', 'pop', 'reggae', 'rock']
...: for genre in genres:
...: sound_files = glob.glob('genres/'+genre+'/*.au')
...: print('Processing %d songs in %s genre...' % (len(sound_files),
...: genre))
...: for f in sound_files:
...: features = extract_features_song(f)
...: all_features.append(features)
...: all_labels.append(genre)
...:
...: # convert labels to one-hot encoding cth blues : 1000000000 classic
...: # 0100000000
...: label_uniq_ids, label_row_ids = np.unique(all_labels,
...: return_inverse=True)#ke integer
...: label_row_ids = label_row_ids.astype(np.int32, copy=False)
...: onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids))#ke
...: one hot
...: return np.stack(all_features), onehot_labels
```

**Gambar 6.84** Generate Features and Label

5. Penjelasan penggunaan fungsi Generate Features and Labels sangat lama ketika Meload Dataset Genre.

- Code yang digunakan:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:26:15 2020
4
5 @author: User
6 """
7
8 features , labels = generate_features_and_labels()
```

- Penjelasan:

- Baris 1: Variabel features and label akan mengeksekusi isi dari features and label
- Baris 2: Memproses 100 lagu di genre blues
- Baris 3: Memproses 100 lagu di genre classical
- Baris 4: Memproses 100 lagu di genre country
- Baris 5: Memproses 100 lagu di genre disco
- Baris 6: Memproses 100 lagu di genre hip hop
- Baris 7: Memproses 100 lagu di genre jazz
- Baris 8: Memproses 100 lagu di genre metal
- Baris 9: Memproses 100 lagu di genre pop
- Baris 10: Memproses 100 lagu di genre reggae
- Baris 11: Memproses 100 lagu di genre rock

- Ilustrasi Gambar:

```

Processing 100 songs in blues genre...
Processing 100 songs in classical genre...
Processing 100 songs in country genre...
Processing 100 songs in disco genre...
Processing 100 songs in hiphop genre...
Processing 100 songs in jazz genre...
Processing 100 songs in metal genre...
Processing 100 songs in pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in rock genre...

```

**Gambar 6.85** Fungsi Generate Features and Label Load Dataset Genre

6. Kenapa harus dilakukan pemisahan data training dan dataset sebesar 80%

Code Yang Digunakan :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:26:39 2020
4
5 @author: User
6 """
7
8 training_split = 0.8
9
10 alldata = np.column_stack((features, labels))
11
12 np.random.shuffle(alldata)
13 splitidx = int(len(alldata) * training_split)
14 train, test = alldata[:splitidx, :], alldata[splitidx:, :]

```

```

15
16 print(np.shape(train))
17 print(np.shape(test))
18
19 train_input = train[:, :-10]
20 train_labels = train[:, -10:]
21
22 test_input = test[:, :-10]
23 test_labels = test[:, -10:]
24
25 print(np.shape(train_input))
26 print(np.shape(train_labels))

```

- Penjelasan:

Untuk kemudahan dalam melakukan pengacakan, sebesar 80% untuk data training dan 20% untuk data test. Sehingga memperlihatkan bahwa data dipisah dan dipecah berpatokan dengan ketentuan 80%. Untuk hasil pertama data trainingnya ada 800 baris dengan 25000 kolom dan data set sebanyak 200 baris dengan 10 kolom, sedangkan untuk hasil kedua yang telah digabungkan dengan one-hot encoding maka data training terdapat 800 baris dan data set dengan 200 baris namun keduanya memiliki jumlah kolom yang sama yaitu 25010.

- Ilustrasi Gambar:

```

In [7]: training_split = 0.8

In [8]: alldata = np.column_stack((features, labels))

In [9]: np.random.shuffle(alldata)
...: splitidx = int(len(alldata) * training_split)
...: train, test = alldata[:splitidx,:], alldata[splitidx:,:]

In [10]: print(np.shape(train))
...: print(np.shape(test))
(800, 25010)
(200, 25010)

In [11]: train_input = train[:, :-10]
...: train_labels = train[:, -10:]

In [12]: test_input = test[:, :-10]
...: test_labels = test[:, -10:]

In [13]: print(np.shape(train_input))
...: print(np.shape(train_labels))
(800, 25000)
(800, 10)

```

**Gambar 6.86** Pemisahan Data Training dan Dataset

## 7. Parameter dari fungsi Sequensial()

Code Yang Digunakan :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:27:00 2020
4
5 @author: User
6 """
7
8 model = Sequential([
9 Dense(100, input_dim=np.shape(train_input)[1]),
10 Activation('relu'),
11 Dense(10),
12 Activation('softmax'),
13])

```

- Penjelasan:

Untuk layer pertama densenya dari 100 neuron kemudian untuk inputan activationnya menggunakan fungsi relu. Dense 10 mengkategorikan 10 neuron untuk jenis genrenya untuk keluarnnya menggunakan aktivasi yaitu fungsi Softmax.

- Ilustrasi Gambar:

```

In [14]: model = Sequential([
...: Dense(100, input_dim=np.shape(train_input)[1]),
...: Activation('relu'),
...: Dense(10),
...: Activation('softmax'),
...:])
WARNING:tensorflow:From E:\Anaconda3\lib\site-packages\tensorflow\python\framework
\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is deprecated and
will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.

```

**Gambar 6.87** Parameter Fungsi Sequensial

## 8. Parameter dari fungsi Compile().

Code Yang Digunakan :

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:27:20 2020
4
5 @author: User
6 """
7
8 model.compile(optimizer='adam',
9 loss='categorical_crossentropy',
10 metrics=['accuracy'])
11 print(model.summary())

```

- Penjelasan:

Untuk dilakukan pemrosesan menggunakan algortima adam sebagai optimizer yang sudah didefinisikan. Kemudian adam tersebut merupakan algoritma pengoptimalan dan untuk memperbarui bobot jaringan yang berulang berdasarkan data training sebelumnya. Untuk loss sendiri menggunakan categorical crossentropy yang difungsikan sebagai optimasi skor atau accuracy. Dan model tersebut digabungkan serta disimpulkan kemudian dicetak.

- Ilustrasi Gambar:

```
In [15]: model.compile(optimizer='adam',
...: loss='categorical_crossentropy',
...: metrics=['accuracy'])
...: print(model.summary())

```

| Layer (type)              | Output Shape | Param # |
|---------------------------|--------------|---------|
| dense_1 (Dense)           | (None, 100)  | 2500100 |
| activation_1 (Activation) | (None, 100)  | 0       |
| dense_2 (Dense)           | (None, 10)   | 1010    |
| activation_2 (Activation) | (None, 10)   | 0       |

```
Total params: 2,501,110
Trainable params: 2,501,110
Non-trainable params: 0
```

---

```
None
```

**Gambar 6.88** Parameter Fungsi Compile

## 9. Parameter dari fungsi Fit().

Code Yang Digunakan :

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:27:36 2020
4
5 @author: User
6 """
7
8 model.fit(train_input, train_labels, epochs=10, batch_size
9 =32,
10 validation_split=0.2)
```

- Penjelasan:

Untuk dilakukan pelatihan dengan epoch dengan rambatan balik sebanyak 10, kemudian dalam sekali epochs dilakukan 32 sampel yang diproses sebelum model diperbarui. Dilakukan validation split sebesar 20% untuk melakukan pengecekan pada cross score validation yang telah dilakukan.

```
In [47]: model.fit(train_input, train_labels, epochs=10, batch_size=32,
 ...: validation_split=0.2)
Train on 640 samples, validate on 160 samples
Epoch 1/10
640/640 [=====] - 5s 8ms/step - loss: 2.1933 - acc: 0.2500 - val_loss:
1.8094 - val_acc: 0.3187
Epoch 2/10
640/640 [=====] - 2s 4ms/step - loss: 1.4572 - acc: 0.4844 - val_loss:
1.6058 - val_acc: 0.4125
Epoch 3/10
640/640 [=====] - 2s 3ms/step - loss: 1.0529 - acc: 0.6844 - val_loss:
1.5243 - val_acc: 0.4437
Epoch 4/10
640/640 [=====] - 2s 3ms/step - loss: 0.8013 - acc: 0.8125 - val_loss:
1.4311 - val_acc: 0.5375
Epoch 5/10
640/640 [=====] - 2s 3ms/step - loss: 0.6804 - acc: 0.8172 - val_loss:
1.5196 - val_acc: 0.4125
Epoch 6/10
640/640 [=====] - 2s 3ms/step - loss: 0.5424 - acc: 0.8797 - val_loss:
1.4561 - val_acc: 0.5437
Epoch 7/10
640/640 [=====] - 2s 3ms/step - loss: 0.4272 - acc: 0.9172 - val_loss:
1.5381 - val_acc: 0.4938
Epoch 8/10
640/640 [=====] - 2s 3ms/step - loss: 0.3408 - acc: 0.9531 - val_loss:
1.4135 - val_acc: 0.5437
Epoch 9/10
640/640 [=====] - 2s 3ms/step - loss: 0.2694 - acc: 0.9734 - val_loss:
1.5703 - val_acc: 0.4750
```

**Gambar 6.89** Parameter Fungsi Fit

- Ilustrasi Gambar:

## 10. Parameter dari fungsi Evaluate()

Code Yang Digunakan :

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:27:57 2020
4
5 @author: User
6 """
7
8 loss , acc = model.evaluate(test_input , test_labels ,
9 batch_size=32)
10 print("Done!")
11 print("Loss: %.4f , accuracy: %.4f" % (loss , acc))
```

- Penjelasan:

Untuk menggunakan test input dan test label dilakukan evaluasi atau proses menemukan model terbaik yang mewakili data dan seberapa baik model yang dipilih akan dijalankan kedepannya. Kemudian pada hasilnya sendiri dapat dilihat bahwa Loss merupakan hasil prediksi yang salah sebanyak 1,7985 dan keakurasiannya sebesar 0,4200.

```
In [48]: loss, acc = model.evaluate(test_input, test_labels, batch_size=32)
200/200 [=====] - 0s 1ms/step

In [49]: print("Done!")
...: print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
Done!
Loss: 1.3712, accuracy: 0.5300
```

**Gambar 6.90** Parameter Fungsi Evaluate

- Ilustrasi Gambar:

## 11. Parameter dari fungsi Predict()

Code Yang Digunakan :

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Mon Apr 6 23:28:16 2020
4
5 @author: User
6 """
7
8 model.predict(test_input[:1])
```

- Penjelasan:

Untuk melakukan prediksi diambil dari satu baris berdasarkan test\_input . Nilai yang tertinggi terdapat pada label kedua yang dipilih prediksi yang tepat kemudian akan dikelompokkan.

- Ilustrasi Gambar:

```
In [50]: model.predict(test_input[:1])
Out[50]:
array([[1.14100585e-02, 5.04052186e-05, 9.51231807e-04, 2.89004028e-01,
 1.89829752e-01, 7.78380111e-02, 3.21492068e-02, 8.59113061e-06,
 3.85905296e-01, 1.28533337e-02]], dtype=float32)
```

**Gambar 6.91** Parameter Fungsi Predict

## 12. PENANGANAN ERROR

- Ilustrasi Gambar:

- Penjelasan:

Install library tersebut pada Anaconda Prompt, jika proses penginstalan telah berhasil atau selesai dilakukan maka error akan teratas.

```

...: import glob
...: import numpy as np
...: import matplotlib.pyplot as plt
...: from keras.models import Sequential
...: from keras.layers import Dense, Activation
...: from keras.utils.np_utils import to_categorical
Traceback (most recent call last):

File "<ipython-input-1-3bc11cc81b24>", line 1, in <module>
 import librosa

ModuleNotFoundError: No module named 'librosa'

```

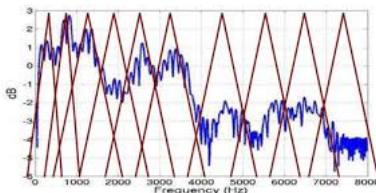
**Gambar 6.92** Error

## 6.6 1174057 Alit Fajar Kurniawan

### 6.6.1 Teori

1. Jelaskan kenapa le suara harus di lakukan MFCC

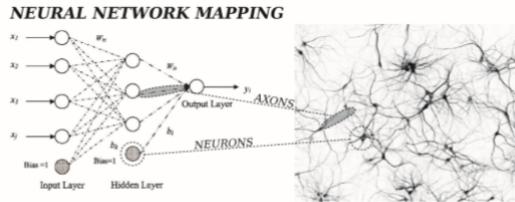
MFCC ( Mel Frequency Cepstral Coefcients ) merupakan koefisien yang merepresentasikan audio. Ekstraksi ciri dalam proses ini ditandai dengan pengubahan data suara menjadi data citra berupa spektrum gelombang. Diharuskan melakukan MFCC kepada objek suara agar suara dapat diubah ke dalam bentuk data matrix dimana telah dilakukan ekstraksi oleh MFCC kemudian direalisasikan sebagai data matrix. Suara tersebut menjadi vektor yang nantinya akan diolah jadi outputan.

**Gambar 6.93** Ilustrasi MFCC

Penjelasan dari gambar diatas, digambarkan sebuah bingkai dari klip suara bernyanyi untuk pengujian yang sama. Dengan menggunakan jendela Hamming, harmonik dalam respons frekuensi jauh lebih tajam.Untuk bingkai input terdiri dari 3 periode fundamental yang identik, maka respons frekuensi magnitudo akan dimasukkan 2 nol antara setiap dua titik tetangga dari respons frekuensi dari periode fundamental tunggal. Dengan kata lain, harmonik dari respons frekuensi umumnya disebabkan oleh periode fundamental berulang dalam bingkai.

2. Jelaskan konsep dasar neural network

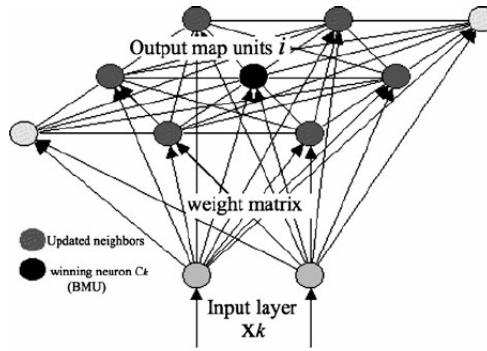
Neural Network merupakan kategori ilmu Soft Computing. Neural Network mengambil kemampuan dari otak manusia yang mampu memberikan stimulasi, melakukan proses, dan memberikan output. Output atau hasil didapatkan dari variasi stimulasi dan proses yang terjadi pada otak manusia. Kemampuan manusia dalam memproses informasi adalah hasil kompleksitas proses di dalam otak manusia. Misalnya, yang terjadi pada anak-anak, mereka mampu belajar untuk melakukan pengenalan meskipun mereka tidak mengetahui algoritma apa yang digunakan.



**Gambar 6.94** Konsep Dasar Neural Network

### 3. Jelaskan konsep pembobotan dalam neural network

Sebuah Neural Network dikonfigurasi untuk aplikasi tertentu, seperti pengetahuan pola atau klasifikasi data. Terjadi penglibatan dalam penyesuaian koneksi sinaptik yang ada antara neuron ketika melakukan penyempurnaan dengan proses pembelajaran. Penyesuaian nilai bobot yang ada pada tiap koneksi baik dari input, neuron maupun output disinkronkan dengan penyesuaian koneksi sinaptik antar neuron itu sendiri.

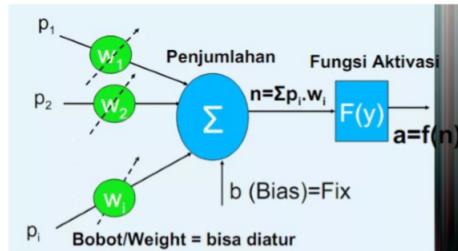


**Gambar 6.95** Pembobotan Neural Network

### 4. Jelaskan konsep fungsi aktifasi dalam neural network

Fungsi aktivasi pada neural network berfungsi layaknya sinapsis pada neuron manusia. Dimana pada neural network, fungsi aktivasi sebagai

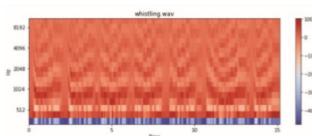
penentu aktivasi dari sebuah nilai input-an yang sebelumnya telah dihitung pada hidden layer.



**Gambar 6.96** fungsi aktifasi dalam neural network

#### 5. Jelaskan cara membaca hasil plot dari MFCC

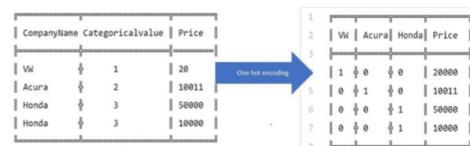
Pada sebuah grak hasil plot dari MFCC gambar ??, terdapat 2 sumbu yaitu x dan y. Sumbu Y merupakan waktu atau durasi dari sebuah suara/musik/lagu, sedangkan sumbu X merupakan frekuensi dari suara yang dihasilkan, dan hasilnya merupakan desibel/power. Desibel yang dihasilkan memiliki range warna yang berbeda-beda. Range warna dari desibel adalah mulai dari biru tua hingga coklat tua. Range warna biru merupakan suara yang tidak dapat didengar manusia dan coklat merupakan suara yang dapat didengar manusia.



**Gambar 6.97** Membaca hasil plot

#### 6. Jelaskan apa itu one-hot encoding

Proses di mana variabel kategorikal dikonversi menjadi bentuk yang dapat disediakan untuk algoritma ML untuk melakukan pekerjaan yang lebih baik dalam prediksi.



**Gambar 6.98** One-Hot Encoding

Nilai kategoris mewakili nilai numerik dari entri dalam dataset. Diconthokan apabila ada perusahaan lain dalam dataset, Ketika jumlah entri unik meningkat, nilai kategoris juga meningkat secara proporsional. Tabel sebelumnya hanyalah representasi. Pada kenyataannya, nilai-nilai kategorikal mulai dari 0 sampai dengan kategori N-1. Inimerupakan bentuk organisasi yang didenisikan dengan VW, Acura, Honda berdasarkan pada nilai-nilai kategorikal. Rata-rata VW dan Honda adalah Acura, dengan menggunakan satu one-hot encoding untuk melakukan "binarisasi" kategori dan memasukkannya sebagai tur untuk melatih model sehingga memberikan hasil yang sesuai.

- Jelaskan apa fungsi dari np.unique dan to categorical dalam kode program np.unique berfungsi Untuk mengekstaksi elemen-elemen unik (tertentu) dalam array.

```
>>> a = np.array([1,1,1,2,2,3,4,4,4,5,5,5,5])
>>> np.unique(a)
array([1., 2., 3., 4., 5.])
```

**Gambar 6.99** np.unique

To categorical Berfungsi untuk mengubah vektor kelas yang berupa integer ( number ) menjadi matriks kelas biner.

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD

Generate dummy data
import numpy as np
x_train = data.data
y_train = keras.utils.to_categorical(data.target, num_classes=3)
x_test = data.data
y_test = keras.utils.to_categorical(data.target, num_classes=3)
```

**Gambar 6.100** To categorical

- Jelaskan apa fungsi dari Sequential dalam kode program

Sebuah jenis model yang digunakan dalam perhitungan ataupun code program yang direalisasikan. Neural Networks Sequential membangun tur tingkat tinggi melalui lapisannya yang berurutan. Sequential juga

suatu proses dimana membandingkan setiap per elemen larik satu per satu secara berurutan, mulai dari elemen pertama, sampai dengan elemen terakhir atau elemen yang dicari sudah ditemukan.

Search for 47

|   |   |   |    |    |    |    |    |    |
|---|---|---|----|----|----|----|----|----|
| 0 | 4 | 7 | 10 | 14 | 23 | 45 | 47 | 53 |
|---|---|---|----|----|----|----|----|----|

Gambar 6.101 fungsi dari Sequential

Pada gambar diatas mendenisikan pencarian terhadap angka 47 dimana hasilnya tetap diurutkan sesuai dengan elemennya.

### 6.6.2 Praktek

1. Jelaskan isi dari data GTZAN Genre Collection dan data dari freesound Sebelumnya teman-teman bisa mendownload terlebih dahulu data nya pada google untuk melakukan pengujian agar dapat berjalan. Pada GTZAN Genre Collection merupakan berupa data musik yang sudah di folderkan berdasarkan genre lagunya. Datasets lagu atau suara yang tersimpan dari 10 gendre musik yang tersimpan dalam 10 folder yaitu folder yaitu classical, disco, pop, country, hiphop, blues, jazz, rock, metal, dan reggea. untuk menguji hasil pengolahan data tersebut dengan menggunakan metode mfcc.

```
import librosa #import librosa tang digunakan untuk fungsi mfcc
import librosa.feature #import librosa feature dan pada baris
import librosa.display #import librosa display
import glob #import glob
import numpy as np #insert numpy untuk pengolahan data menjadi vektor
import matplotlib.pyplot as plt #import matplotlib untuk melakukan plot
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.utils.np_utils import to_categorical

In[1]: membuat fungsi mfcc untuk melakukan pengujian
def display_mfcc(song): #membuat fungsi mfcc (display_mfcc) dan terdapat parameter song
 y, _ = librosa.load(song) #method librosa load
 mfcc = librosa.feature.mfcc(y) # method librosa featurea mfcc

 plt.figure(figsize=(10, 4)) #membuat ot gure dengan ukuran 10 x 4
```

```
librosa.display.specshow(mfcc, x_axis='time', y_axis='mel') #dat
plt.colorbar() #plot warna
plt.title(song) #plot judul
plt.tight_layout()
plt.show() #plot di tampilkan
```

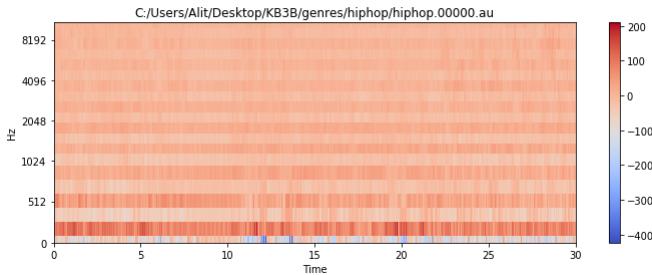
```
In [24]: def display_mfcc(song):
...: y, _ = librosa.load(song)
...: mfcc = librosa.feature.mfcc(y)
...: plt.figure(figsize=(10, 4))
...: librosa.display.specshow(mfcc, x_axis='time',
y_axis='mel')
...: plt.colorbar()
...: plt.title(song)
...: plt.tight_layout()
...: plt.show()
```

**Gambar 6.102** Praktek 1

2. Jelaskan perbaris kode program dengan kata kata dan dilengkapi ilustrasi gambar fungsi dari display mfcc

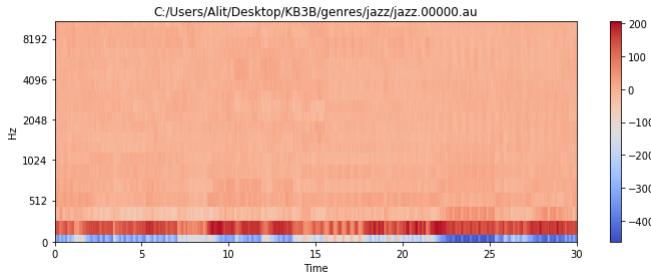
```
In[2]: cek fungsi
display_mfcc('C:/Users/Alit/Desktop/KB3B/genres/hiphop/hiphop.00050.
#untuk mendisplay tampilan glombang suara dari le 266093 stereo-surg
In[2]: cek fungsi
display_mfcc('C:/Users/Alit/Desktop/KB3B/genres/jazz/jazz.00050.au')
In[2]: cek fungsi
display_mfcc('C:/Users/Alit/Desktop/KB3B/genres/pop/pop.00050.au')
In[2]: cek fungsi
display_mfcc('C:/Users/Alit/Desktop/KB3B/genres/reggae/reggae.00050.
In[2]: cek fungsi
display_mfcc('C:/Users/Alit/Desktop/KB3B/genres/rock/rock.00050.au')
```

```
In [30]: display_mfcc('C:/Users/Alit/Desktop/KB3B/genres/hiphop/hiphop.00000.au')
```



**Gambar 6.103** Praktek

```
In [31]: display_mfcc('C:/Users/Alit/Desktop/KB3B/genres/jazz/jazz.00000.au')
```



**Gambar 6.104** Praktek

Untuk hasil contoh selanjutnya bisa melakukan pengujian sendiri

- Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari extract features song(). Jelaskan juga mengapa yang diambil 25.000 baris pertama?

```
In[3]:
```

```
def extract_features_song(f): #nama extract features song yang nanti
y, _ = librosa.load(f) #membuat variabel y dengan method librosa

mfcc = librosa.feature.mfcc(y) #membuat variabel baru mfcc dengan

mfcc /= np.amax(np.absolute(mfcc)) #variabel mfcc dengan isian n

return np.ndarray.flatten(mfcc)[:25000] #membuat array dari data
```

kenapa data 25000 pertama yang digunakan karena data tersebut digunakan sebagai data testing semakin besar data testing yang di gunakan maka semakin akurat hasil kecerdasan buatan yang dihasilkan. tapi sebenarnya data tersebut relatif bisa lebih besar atau lebih kecil tergantung pada komputer masing masing.

- Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari generate features and labels().

Lihatlah pada praktek kode program berikut, dan sudah ada penjelasan pada kode program

```
In[4]:
```

```
def generate_features_and_labels(): #pendekripsi nama fungsi yaitu g
all_features = [] #variabel baru dengan array all features
all_labels = [] #variabel baru dengan array all labels
```

```

genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'j
for genre in genres:
 sound_files = glob.glob('genres/' + genre + '/*.au')
 print('Processing %d songs in %s genre...' % (len(sound_file
 for f in sound_files:
 features = extract_features_song(f)
 all_features.append(features)
 all_labels.append(genre)

 # convert labels to one-hot encoding cth blues : 10000000000 clas
label_uniq_ids, label_row_ids = np.unique(all_labels, return_inv
label_row_ids = label_row_ids.astype(np.int32, copy=False)
onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids)
return np.stack(all_features), onehot_labels

```

5. Jelaskan dengan kata dan praktek kenapa penggunaan fungsi generate features and labels() sangat lama ketika meload dataset genre. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

dalam melakukan penggunaan fungsi generate features and labels() membutuhkan waktu yang lama dikarenakan mesin membaca file secara satu persatu dari total jumlah 100 file yang dibaca ditambah lagi dengan proses perubahan data yang dilakukan oleh mesin dari bentuk suara menjadi bentuk vektor.

```
In[5]: passing parameter dari fitur ekstraksi menggunakan mfcc
features, labels = generate_features_and_labels()
```

```

Processing 100 songs in blues genre...
Processing 100 songs in classical genre...
Processing 100 songs in country genre...
Processing 100 songs in disco genre...
Processing 100 songs in hiphop genre...
Processing 100 songs in jazz genre...
Processing 100 songs in metal genre...
Processing 100 songs in pop genre...
Processing 100 songs in reggae genre...
Processing 100 songs in rock genre...

Process finished with exit code 0

```

**Gambar 6.105** Praktek 2 Contoh 5

6. Jelaskan kenapa harus dilakukan pemisahan data training dan data set sebesar 80 persen? Praktekkan dengan kode dan Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

berikut kode program yang merupakan kode untuk membagi data sebanyak 80 persen untuk data training maka data musik tadi yang total jumlahnya 1000 akan dibagi dua untuk data training sebanyak 800 dan 200 untuk data testing.

```
In[6] fitur ekstraksi
training_split = 0.8
```

7. Praktekkan dan jelaskan masing-masing parameter dari fungsi Sequential(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

Fungsi sequential itu sendiri digunakan untuk melakukan pengolahan data inputan sesuai dengan fungsi yang ada pada fungsi sequential pada fungsi sequential kali ini menggunakan dua fungsi se

quential meng-compile data dari 100 neuron atau dari 1 folder le dengan menggunakan fungsi relu dan softmax untuk menghasilkan outputan yang sesuai dengan keriteria.

```
In[7]: membuat seq NN, layer pertama dense dari 100 neurons
model = Sequential([
 Dense(100, input_dim=np.shape(train_input)[1]),
 Activation('relu'),
 Dense(10),
 Activation('softmax'),
])
```

8. Praktekkan dan jelaskan masing-masing parameter dari fungsi compile(). Tunjukkan keluarannya dengan fungsi summary dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

yaitu fungsi compile yang digunakan untuk mengetahui parameter yang digunakan dari data yang telah diolah untuk caranya dapat menggunakan codingan sebagai berikut, pada gambar tersebut memunculkan parameternya berpasaja dan total parameter yang digunakan.

```
In[8]:
model.compile(optimizer='adam',
 loss='categorical_crossentropy',
 metrics=['accuracy'])
print(model.summary())
```

| Layer (type)                | Output Shape | Param # |
|-----------------------------|--------------|---------|
| dense_1 (Dense)             | (None, 100)  | 2500100 |
| activation_1 (Activation)   | (None, 100)  | 0       |
| dense_2 (Dense)             | (None, 10)   | 1010    |
| activation_2 (Activation)   | (None, 10)   | 0       |
| <hr/>                       |              |         |
| Total params: 2,501,110     |              |         |
| Trainable params: 2,501,110 |              |         |
| Non-trainable params: 0     |              |         |
| <hr/>                       |              |         |
| None                        |              |         |

**Gambar 6.106** Praktek 2 Contoh 8

9. Praktekkan dan jelaskan masing-masing parameter dari fungsi `t()`. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

Pada fungsi ini dilakukan pengolahan data dari 10 label tadi atau 10 le dataset tadi kemudian dihitung tingkat akurasi masing masing dan tingkat kegagalan atau loss data darisetiap le tersebut caranya dengan melakukan codingan berikut. pada gambar tersebut menunjukan 10 pengolahan data untuk menentukan nilai akurasi dan loss dari data tersebut dan selanjutnya dilakukan ngsi evaluasi.

```
In[9]:
model.fit(train_input, train_labels, epochs=10, batch_size=32,
 validation_split=0.2)
```

```

32/640 [>.....] - ETA: 1s - loss: 0.2331 - accuracy: 1.0000
64/640 [==>.....] - ETA: 1s - loss: 0.2113 - accuracy: 1.0000
96/640 [==>.....] - ETA: 1s - loss: 0.2268 - accuracy: 0.9896
128/640 [=====>.....] - ETA: 1s - loss: 0.2172 - accuracy: 0.9922
160/640 [=====>.....] - ETA: 1s - loss: 0.2371 - accuracy: 0.9812
192/640 [=====>.....] - ETA: 0s - loss: 0.2391 - accuracy: 0.9844
224/640 [=====>.....] - ETA: 0s - loss: 0.2484 - accuracy: 0.9821
256/640 [=====>.....] - ETA: 0s - loss: 0.2529 - accuracy: 0.9727
288/640 [=====>.....] - ETA: 0s - loss: 0.2549 - accuracy: 0.9722
320/640 [=====>.....] - ETA: 0s - loss: 0.2469 - accuracy: 0.9719
352/640 [=====>.....] - ETA: 0s - loss: 0.2512 - accuracy: 0.9688
384/640 [=====>.....] - ETA: 0s - loss: 0.2525 - accuracy: 0.9661
416/640 [=====>.....] - ETA: 0s - loss: 0.2508 - accuracy: 0.9688
448/640 [=====>.....] - ETA: 0s - loss: 0.2511 - accuracy: 0.9710
480/640 [=====>.....] - ETA: 0s - loss: 0.2503 - accuracy: 0.9708
512/640 [=====>.....] - ETA: 0s - loss: 0.2519 - accuracy: 0.9727
544/640 [=====>.....] - ETA: 0s - loss: 0.2543 - accuracy: 0.9724
576/640 [=====>.....] - ETA: 0s - loss: 0.2503 - accuracy: 0.9740

```

**Gambar 6.107** Praktek 2 Contoh 9

10. Praktekkan dan jelaskan masing-masing parameter dari fungsi evaluate(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

pada fungsi ini dilakukan evaluasi terhadap datayang telah di runing sebelumnya untuk lebih jelasnya dapat di lihat codingan tersebut pada codingan tersebut dilakukan evaluasi pada tingkat kegagalan dan akurasi kebenaran maka hasilnya munculkan hasil evaluasi dari 10 proses dari setiap gendre yaitu akurasi sebesar 51 persen dan loss data sebesar 1.4105 data.

```

32/200 [==>.....] - ETA: 5s
200/200 [=====] - 1s 6ms/step
Done!
Loss: 2.3352, accuracy: 0.0800

```

**Process finished with exit code 0**

**Gambar 6.108** Praktek 2 Contoh 10

11. Praktekkan dan jelaskan masing-masing parameter dari fungsi predict(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

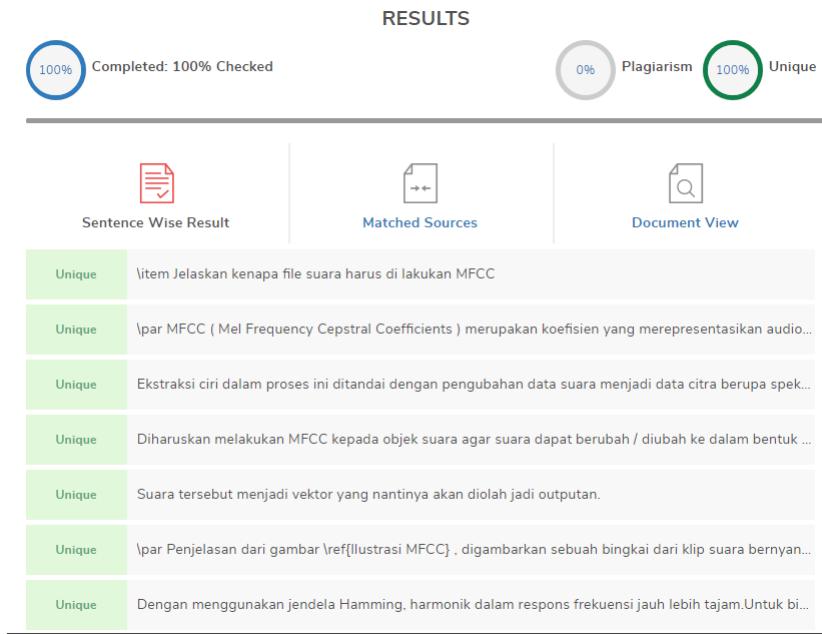
fungsi predict merupakan fungsi untuk membandingkan tingkat akurasi pada setiap label yang sepuluh tadi maka data akan di sandingkan ke masing masing tingkat akurasinya, yang akurasinya paling tinggi maka itulah jawaban untuk setiap inputan yang dilakukan.

```
32/200 [====>.....] - ETA: 5s
200/200 [=====] - 1s 6ms/step

Process finished with exit code 0
```

Gambar 6.109 Praktek 2 Contoh 11

### 6.6.3 Plagiarisme



Gambar 6.110 Plagiarisme

## BAB 7

---

# CHAPTER 7

---

### 7.1 Faisal Najib Abdullah / 1174042

#### 7.1.1 Teori

1. Jelaskan Kenapa file teks harus dilakukan tokenizer. Dilengkap dengan ilustrasi atau gambar.

Tokenizer adalah proses untuk membagi kalimat menjadi beberapa teks, hal ini sangat di perlukan dalam AI karena nanti setiap teks akan di hitung bobotnya dan akan memunculkan nilai vektor sehingga teks tersebut bisa di gunakan sebagai data untuk memprediksi teks yang muncul dalam satu kalimat sedangkan proses tkenizer merupakan caramembagi bagi teks dari suatu kalimat biasanya pembagi kalimat tersebut merupakan spasi dalam suatu kalimat.

2. jelaskan konsep dasar K Fold Cross VAlidation dan dataset komentar youtube pada code berikut dilengkapi dengan ilustrasi gambar

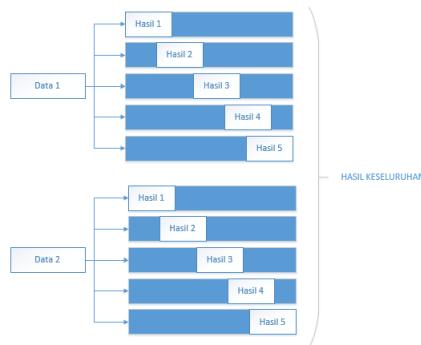
```
kfold = StratifiedKFold(n_splits=5)
```



**Gambar 7.1** Ilustrasi Tokenisasi

```
splits = kfold.split(d, d['CLASS'])
```

pada codingan tersebut terdapat kfold sebagai variabel yang didalamnya terdiri dari split 5 yang berarti pengulangan terhadap pengolahan masing masing lima kali pada kasus ini terdapat data sebanyak 5 berarti ke lima data tersebut di ulang sebanyak lima kali dengan atribut class sebagai acuan pengolahan datanya kemudian akan di hasilkan akurasi dari pengulangan data tersebut sebesar sekian persen tergantung datanya.



**Gambar 7.2** Ilustrasi K Fold Cross Validation

3. Jelaskan apa maksudnya kode program for train, test in split dilengkapi di lengkapi dengan ilustrasi atau gambar.  
for train di gunakan melakukan training atau pelatihan terhadap data yang telah di deklarasikan sebelumnya. sedangkan test in split di gunakan untuk digunakan untuk membatasi jumlah data yang akan di inputkan atau data yang akan di gunakan.
4. Jelaskan apa maksud kode program train\_content =d['CONTENT'].iloc[train\_idx] dan test\_content =d['CONTENT'].iloc[test\_idx]. dilengkapi dengan ilustrasi atau gambar.  
maksud dari kode program tersebut adalah membaca isian kolom pada field yang bernama CONTENT sebagai data training dan data testing untuk program tersebut.

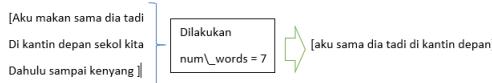
```
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> X, y = np.arange(6).reshape((3, 2)), range(3)
>>> X
array([[0, 1],
 [2, 3],
 [4, 5]])
>>> list(y)
[0, 1, 2]
>>> X_train, X_test, y_train, y_test = train_test_split(
... X, y, test_size=0.33, random_state=42)
>>> X_train
array([[2, 3],
 [4, 5]])
>>> X_test
array([[0, 1]])
>>>
```

**Gambar 7.3** Ilustrasi For Train dan Test in split

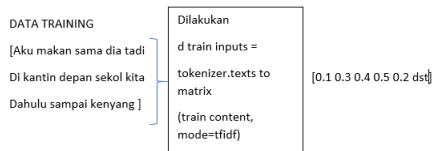
| No | Nama    | KONTENT                                    |
|----|---------|--------------------------------------------|
| 1  | Cokro   | Seseorang yang berusaha                    |
| 2  | Edi     | Anak yang patuh pada ibunya                |
| 3  | Prawiro | Seorang prajurit negara indonesia          |
| 4  | Ali     | Orang yang berlapang dada                  |
| 5  | Hibibi  | Dia yang di berkati akan semua tindakannya |

**Gambar 7.4** Ilustrasi Penggunaan kolom CONTENT

5. Jelaskan maksud dari fungsi `tokenizer = Tokenizer(num_words=2000)` dan `tokenizer.fit_on_texts(train_konten)` di lengkapi dengan ilustrasi gambar.
- fungsi `tokenizer = Tokenizer(num_words=2000)` digunakan untuk membaca kalimat yang telah di buat menjadi token sebanyak 2000 kata dan fungsi `fit_on_texts(train_konten)` digunakan untuk membuat membaca data token teks yang telah di masukan kedalam fungsi yaitu fungsi `train_konten`.

**Gambar 7.5** Ilustrasi fit tokenizer dan num\_word=2000

6. Jelaskan apa maksud dari fungsi `d_train_inputs = tokenizer.texts_to_matrix(train_content, mode='tfidf')` dan `d_test_inputs = tokenizer.texts_to_matrix(test_content, mode='tfidf')`, dilengkapi dengan ilustrasi kode dan atau gambar.
- yaitu di gunakan untuk mengubah urutan teks yang tadi telah di lakukan tokenizer menjadi matriks yang berutu=rutan seperti tf idf
7. Jelaskan apa maksud dari fungsi `d_train_inputs = d_train_inputs/np.amax(np.absolut`
- fungsi tersebut digunakan untuk membagi matriks tfidf dengan dengan penentuan maksimum aray sepanjang sumbu sehingga akan menimbulkan garis ke bawah dan keatas yang membentuk gambar v kemudian hasil

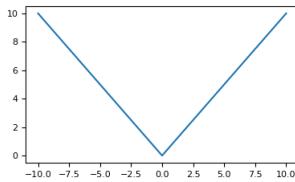


**Gambar 7.6** Ilustrasi  $d\_train\_inputs = \text{tokenizer}.\text{texts\_to\_matrix}(\text{train content}, \text{mode}=\text{tfidf})$

tersebut akan di masukan ke dalam variabel  $d\_train\_input$  dan  $d\_test\_input$  dengan methode absolute. yang berarti tanpa bilangan negatif.

```
>>> import matplotlib.pyplot as plt
```

```
>>> x = np.linspace(start=-10, stop=10, num=101)
>>> plt.plot(x, np.absolute(x))
>>> plt.show()
```



**Gambar 7.7** Ilustrasi  $d\_train\_inputs = d\_train\_inputs / \text{np}.\text{amax}(\text{np}.\text{absolute}(d\_train\_inputs))$

8. Jelaskan apa maksud fungsi dari  $d\_train\_outputs = \text{np}.\text{utils}.\text{to\_categorical}(d[\text{'CLASS'}].\text{iloc}[\text{train}])$  dan  $d\_test\_outputs = \text{np}.\text{utils}.\text{to\_categorical}(d[\text{'CLASS'}].\text{iloc}[\text{test idx}])$  dalam kode program, dilengkapi dengan ilustrasi atau gambar.

maksud dari fungsi tersebut yaitu untuk merubah nilai vektor yang ada pada atribut class menjadi bentuk matrix dengan pengurutan berdasarkan data index training dan testing.

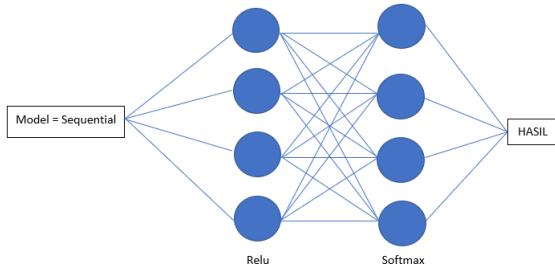
9. model = sequential berarti variabel model berisi method sequential yang berguna untuk searching data dengan menerima parameter atau argumen kunci dengan langkah tertentu untuk mencari data yang telah di olah. kemudian model di tambahkan metod add dengan Dense yang berarti data data yang di inputkan akan terhubung, dengan data 612 dan 2000 data kata atau word kemudian model tersebut di masukan fungsi aktivitation dengan rumusa atau methode relu setelah itu data tersebut di dropout 0.5 atau di pangkas sebanyak 50 persen di karenakan pada pohon bobot terlalu akurat terhadap data. sehingga data dilakukan pemangkasan 50 persen. kemudian data tersebut di hubungkan. setelah data tersebut

```
Consider an array of 5 Labels out of a set of 3 classes {0, 1, 2}:
> labels
array([0, 2, 1, 2, 0])
to_categorical converts this into a matrix with as many
columns as there are classes. The number of rows
stays the same.
> to_categorical(labels)
array([[1., 0., 0.],
 [0., 0., 1.],
 [0., 1., 0.],
 [0., 0., 1.],
 [1., 0., 0.]], dtype=float32)

>>> type(df.iloc[0])
<class 'pandas.core.series.Series'>
>>> df.iloc[0]
a 1
b 2
c 3
d 4
Name: 0, dtype: int64
```

**Gambar 7.8** Ilustrasi train outputs = np utils.to categorical(d['CLASS'].iloc[train]

di hubungkan maka di lakukan perhitungan dengan menggunakan fungsi softmax.



**Gambar 7.9** Ilustrasi Neural Network

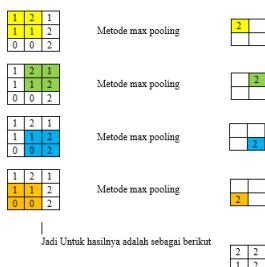
10. model tersebut kemudian di compile atau di kembalikan kembali fungsi nilainya yang mana akan mengembalikan fungsi nilai loss nya berapa yang diambil dari fungsi adamax yang berberguna untuk mengetahui nilai lossnya kemudian metrics = accuracy merupakan akurasi dari nilai matrixnya.
11. Jelaskan apa itu Deep Learning  
merupakan salah satu algoritma jaringan saraf tiruan yang menggunakan meta data sebagai inputan dan mengolahnya menggunakan layer layer yang tersembunyi. deep learning memiliki suatu keunikan yaitu fitur yang dapat mengekstraksi secara otomatis.
12. Jelaskan Apa itu Deep Neural Network dan apa perbedaanya dengan Deep Learning.  
merupakan algoritma jaringan saraf juga yang mena akan melakukan pembobotan terhadap data yang sudah ada sebagai acuan untuk data inputan selanjutnya. kemudian terdiri atas beberapa layer atau hiden

layer. perbedaan antara deep learning dan DNN atau Deep Neural Network yaitu deep learning merupakan pemakai algoritma dari DNN dan DNN merupakan algoritma yang ada pada deep learning.

13. Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride (NPM mod3+1) x (NPM mod3+1) yang terdapat max pooling.(nilai 30)

sebelum membuat ilustrasi perlu di ketahui apa itu stride, stride adalah acuan atau parameter yang menentukan pergeseran pada filter fixcel. sebagai contoh nilai stride 1 yang berarti filter akan bergeser sebanyak satu fixcel secara vertikal dan horizontal. selanjutnya apa itu max pooling contoh pada suatu gambar di tentukan Max Pooling dari  $3 \times 3$  dengan stride 1 yang berarti setiap pergeseran 1 pixcel akan diambil nilai terbesar dari pixcel  $3 \times 3$  tersebut.

selanjutnya jawaban dari soal ini yaitu menggunakan stride 1 dengan ketentuan max pooling.



**Gambar 7.10** Hasil perhitungan stride 1 max pooling

### 7.1.2 Pemrograman

Penjelasan setiap baris code dari nomer 1 sampai 20 bisa di lihat pada bagian cooment coding yang berwarna hijau.

- Jelaskan kode program pada blok In[1].

```

1 # In [1]: import lib
2 # menimport librari CSV untuk mengolah data ber ekstensi csv
3 import csv
4 #kemudian mengimport librari Image yang berguna untuk dari
 PIL atau Python Imaging Library yang berguna untuk
 mengolah data berupa gambar
5 from PIL import Image as pil_image
6 # kemudian mengimport librari keras yang menggunakan method
 preprocessing yang digunakan untuk membuat neutal network
7 import keras.preprocessing.image

```

```
In [30]: # In[1]:import lib
menimport librari CSV untuk mengolah data ber ekstensi csv
import csv
#kemudian mengimport librari Image yang berguna untuk dari PIL atau Python Image
from PIL import Image as pil_image
Kemudian mengimport librari keras yang menggunakan method preprocessing yang
import keras.preprocessing.image
```

**Gambar 7.11** hasil kode program

Untuk hasilnya dapat di lihat pada gambar berikut.

2. Jelaskan kode program pada blok In[2].

```
1 # In[2]:load all images (as numpy arrays) and save their
 classes
2 #membuat variabel imgs dengan variabel kosong
3 imgs = []
4 #membuat variabel classes dengan variabel kosong
5 classes = []
6 #membuka file hasy-data-labels.csv yang berada di foleder
 HASYv2 yang di inisialisasi menjadi csvfile
7 with open('HASYv2/hasy-data-labels.csv') as csvfile:
8 #membuat variabel csvreader yang berisi method csv.reader
 yang membaca variabel csvfile
9 csvreader = csv.reader(csvfile)
10 # membuat variabel i dengan isi 0
11 i = 0
12 # membuat looping pada variabel csvreader
13 for row in csvreader:
14 # dengan ketentuan jika i lebihkecil daripada 0
15 if i > 0:
16 # dibuat variabel img dengan isi keras untuk
 aktivasi neural network fungsi yang membaca data yang
 berada dalam folder HASYv2 dengan input nilai -1.0 dan 1.0
17 img = keras.preprocessing.image.img_to_array(
18 pil_image.open("HASYv2/" + row[0]))
19 # neuron activation functions behave best when
 input values are between 0.0 and 1.0 (or -1.0 and 1.0),
20 # so we rescale each pixel value to be in the
 range 0.0 to 1.0 instead of 0-255
21 #membagi data yang ada pada fungsi img sebanyak
 255.0
22 img /= 255.0
23 # menambah nilai baru pada imgs pada row ke 1 2
 dan dilanjutkan dengan variabel img
24 imgs.append((row[0], row[2], img))
25 # menambahkan nilai pada row ke 2 pada variabel
 classes
26 classes.append(row[2])
27 # penambahan nilai satu pada variabel i
 i += 1
```

Untuk hasilnya dapat di lihat pada gambar berikut.

3. Jelaskan kode program pada blok In[3].

```
In [30]: # In[1]:import lib
menimport librari CSV untuk mengolah data ber ekstensi csv
import csv
#Kemudian mengimport librari Image yang berguna untuk dari PIL atau Python Image
from PIL import Image as pil_image
Kemudian mengimport librari keras yang menggunakan method preprocessing yang
import keras.preprocessing.image
```

**Gambar 7.12** hasil kode program

```
1 # In [3]:shuffle the data, split into 80% train , 20% test
2 # mengimport library random
3 import random
4 # melakukan random pada vungsi imgs
5 random.shuffle(imgs)
6 # membuat variabel split_idx dengan nilai integer 80 persen
 dikali dari pengembalian jumlah dari variabel imgs
7 split_idx = int(0.8*len(imgs))
8 # membuat variabel train dengan isi lebih besar split idx
9 train = imgs[:split_idx]
10 # membuat variabel test dengan isi lebih kecil split idx
11 test = imgs[split_idx:]
```

Untuk hasilnya dapat di lihat pada gambar berikut.

```
In [32]: import random
random.shuffle(imgs)
split_idx = int(0.8*len(imgs))
train = imgs[:split_idx]
test = imgs[split_idx:]
```

**Gambar 7.13** hasil kode program

4. Jelaskan kode program pada blok In[4].

```
1 # In [4]:
2 # mengimport librari numpy dengan inisial np
3 import numpy as np
4 # membuat variabel train_input dengan np method asarray yang
 mana membuat array dengan isi row 2 dari data train
5 train_input = np.asarray(list(map(lambda row: row[2] , train)))
6 # membuat test_input input dengan np method asarray yang mana
 membuat array dengan isi row 2 dari data test
7 test_input = np.asarray(list(map(lambda row: row[2] , test)))
8 # membuat variabel train_output dengan np method asarray yang
 mana membuat array dengan isi row 1 dari data train
9 train_output = np.asarray(list(map(lambda row: row[1] , train)))
10 # membuat variabel test_output dengan np method asarray yang
 mana membuat array dengan isi row 1 dari data test
11 test_output = np.asarray(list(map(lambda row: row[1] , test)))
```

Untuk hasilnya dapat di lihat pada gambar berikut.

5. Jelaskan kode program pada blok In[5].

```
In [33]: import numpy as np
train_input = np.asarray(list(map(lambda row: row[2], train)))
test_input = np.asarray(list(map(lambda row: row[2], test)))
train_output = np.asarray(list(map(lambda row: row[1], train)))
test_output = np.asarray(list(map(lambda row: row[1], test)))
```

**Gambar 7.14** hasil kode program

```
1 # In [5]: import encoder and one hot
2 # mengimport librari LabelEncode dari sklearn
3 from sklearn.preprocessing import LabelEncoder
4 # mengimport librari OneHotEncoder dari sklearn
5 from sklearn.preprocessing import OneHotEncoder
```

Untuk hasilnya dapat di lihat pada gambar berikut.

```
In [34]: from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

**Gambar 7.15** hasil kode program

## 6. Jelaskan kode program pada blok In[6].

```
1 # In [6]:convert class names into one-hot encoding
2 # membuat variabel label_encoder dengan isi LabelEncoder
3 label_encoder = LabelEncoder()
4 # membuat variabel integer_encoded yang berfungsi untuk
 mengkonvert variabel classes kedalam bentuk integer
5 integer_encoded = label_encoder.fit_transform(classes)
```

Untuk hasilnya dapat di lihat pada gambar berikut.

```
In [35]: # first, convert class names into integers
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(classes)
```

**Gambar 7.16** hasil kode program

## 7. Jelaskan kode program pada blok In[7].

```
1 # In [7]:then convert integers into one-hot encoding
2 # membuat variabel onehot_encoder dengan isi OneHotEncoder
3 onehot_encoder = OneHotEncoder(sparse=False)
4 # mengisi variabel integer_encoded dengan isi integer_encoded
 yang telah di convert pada fungsi sebelumnya
5 integer_encoded = integer_encoded.reshape(len(integer_encoded),
 1)
6 # mengkonvert variabel integer_encoded kedalam onehot_encoder
7 onehot_encoder.fit(integer_encoded)
```

Untuk hasilnya dapat di lihat pada gambar berikut.

## 8. Jelaskan kode program pada blok In[8].

```
In [36]: # then convert integers into one-hot encoding
onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoder.fit(integer_encoded)

Out[36]: OneHotEncoder(categories='auto', drop=None, dtype=<class 'numpy.float64'>,
handle_unknown='error', sparse=False)
```

**Gambar 7.17** hasil kode program

```
1 # In [8]: convert train and test output to one-hot
2 # mengkonvert data train output menggunakan variabel
label_encoder kedalam variabel train_output_int
3 train_output_int = label_encoder.transform(train_output)
4 # mengkonvert variabel train_output_int kedalam fungsi
onehot_encoder
5 train_output = onehot_encoder.transform(train_output_int.
 reshape(len(train_output_int), 1))
6 # mengkonvert data test_output menggunakan variabel
label_encoder kedalam variabel test_output_int
7 test_output_int = label_encoder.transform(test_output)
8 # mengkonvert variabel test_output_int kedalam fungsi
onehot_encoder
9 test_output = onehot_encoder.transform(test_output_int.
 reshape(len(test_output_int), 1))
10 # membuat variabel num_classes dengan isi variabel
label_encoder dan classes
11 num_classes = len(label_encoder.classes_)
12 # mencetak hasil dari nomer Class berupa persen
13 print("Number of classes: %d" % num_classes)
```

Untuk hasilnya dapat di lihat pada gambar berikut.

```
In [37]: # convert train and test output to one-hot
train_output_int = label_encoder.transform(train_output)
train_output = onehot_encoder.transform(train_output_int.reshape(len(train_output_int), 1))
test_output_int = label_encoder.transform(test_output)
test_output = onehot_encoder.transform(test_output_int.reshape(len(test_output_int), 1))

num_classes = len(label_encoder.classes_)
print("Number of classes: %d" % num_classes)

Number of classes: 369
```

**Gambar 7.18** hasil kode program

9. Jelaskan kode program pada blok In[9].

```
1 # In [9]: import sequential
2 # mengimport librari Sequential dari Keras
3 from keras.models import Sequential
4 # mengimport librari Dense, Dropout, Flatten dari Keras
5 from keras.layers import Dense, Dropout, Flatten
6 # mengimport librari Conv2D, MaxPooling2D dari Keras
7 from keras.layers import Conv2D, MaxPooling2D
```

Untuk hasilnya dapat di lihat pada gambar berikut.

10. Jelaskan kode program pada blok In[10].

```
In [38]: from keras.models import Sequential
 from keras.layers import Dense, Dropout, Flatten
 from keras.layers import Conv2D, MaxPooling2D
```

**Gambar 7.19** hasil kode program

```
1 # In[10]: desain jaringan
2 # membuat variabel model dengan isian librari Sequential
3 model = Sequential()
4 # variabel model di tambahkan librari Conv2D tigapuluhan dua
 bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
 relu dang menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
6 input_shape=np.shape(train_input[0])))
7 # variabel model di tambahkan dengan lib MaxPooling2D dengan
 ketentuan ukuran 2 x 2 pixcel
8 model.add(MaxPooling2D(pool_size=(2, 2)))
9 # variabel model di tambahkan dengan librari Conv2D 32 bit
 dengan kernel 3 x 3
10 model.add(Conv2D(32, (3, 3), activation='relu'))
11 # variabel model di tambahkan dengan lib MaxPooling2D dengan
 ketentuan ukuran 2 x 2 pixcel
12 model.add(MaxPooling2D(pool_size=(2, 2)))
13 # variabel model di tambahkan librari Flatten
14 model.add(Flatten())
15 # variabel model di tambahkan librari Dense dengan fungsi
 tanh
16 model.add(Dense(1024, activation='tanh'))
17 # variabel model di tambahkan librari dropout untuk memangkas
 data tree sebesar 50 persen
18 model.add(Dropout(0.5))
19 # variabel model di tambahkan librari Dense dengan data dari
 num_classes dan fungsi softmax
20 model.add(Dense(num_classes, activation='softmax'))
21 # mengkompile data model untuk mendapatkan data loss akurasi
 dan optimasi
22 model.compile(loss='categorical_crossentropy', optimizer='adam',
23 metrics=['accuracy'])
24 # mencetak variabel model kemudian memunculkan kesimpulan
 berupa data total parameter, trainable paremeter dan bukan
 trainable parameter
25 print(model.summary())
```

Untuk hasilnya dapat di lihat pada gambar berikut.

11. Jelaskan kode program pada blok In[11].

```
1 # In[11]: import sequential
2 # mengimport librari keras callbacks
3 import keras.callbacks
4 # membuat variabel tensorboard dengan isi lib keras
5 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/
 mnist-style')
```

Untuk hasilnya dapat di lihat pada gambar berikut.

| Model: "sequential_5"          |                    |         |
|--------------------------------|--------------------|---------|
| Layer (type)                   | Output Shape       | Param # |
| conv2d_7 (Conv2D)              | (None, 30, 30, 32) | 896     |
| max_pooling2d_7 (MaxPooling2D) | (None, 15, 15, 32) | 0       |
| conv2d_8 (Conv2D)              | (None, 13, 13, 32) | 9248    |
| max_pooling2d_8 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| flatten_5 (Flatten)            | (None, 1152)       | 0       |
| dense_9 (Dense)                | (None, 1024)       | 1180672 |
| dropout_3 (Dropout)            | (None, 1024)       | 0       |
| dense_10 (Dense)               | (None, 369)        | 378225  |

Total params: 1,569,041  
Trainable params: 1,569,041  
Non-trainable params: 0

None

**Gambar 7.20** hasil kode program

```
In [40]: import keras.callbacks
tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-style')
```

**Gambar 7.21** hasil kode program

12. Jelaskan kode program pada blok In[12].

```
1 # In[12]: 5menit kali 10 epoch = 50 menit
2 # fungsi model titambahkan metod fit untuk mengetahui
 perhitungan dari train_input train_output
3 model.fit(train_input, train_output,
4 # dengan batch size 32 bit
5 batch_size=32,
6 epochs=10,
7 verbose=2,
8 validation_split=0.2,
9 callbacks=[tensorboard])
10
11 score = model.evaluate(test_input, test_output, verbose=2)
12 print('Test loss:', score[0])
13 print('Test accuracy:', score[1])
```

Untuk hasilnya dapat di lihat pada gambar berikut.

```
In [15]: model.fit(train_input, train_output,
batch_size=32,
epochs=10,
verbose=2,
validation_split=0.2,
callbacks=[tensorboard])

score = model.evaluate(test_input, test_output, verbose=2)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

**Gambar 7.22** hasil kode program

13. Jelaskan kode program pada blok In[13].

```
1 # In[13]: try various model configurations and parameters to
 find the best
```

```
2 # mengimport librari time
3 import time
4 #membuat variabel result dengan array kosong
5 results = []
6 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
7 for conv2d_count in [1, 2]:
8 # menentukan ukuran besaran fixcel dari data atau konvert
9 1 fixcel mnjadi data yang berada pada codigan dibawah.
10 for dense_size in [128, 256, 512, 1024, 2048]:
11 # membuat looping untuk memangkas masing-masing data
12 dengan ketentuan 0 persen 25 persen 50 persen dan 75
13 persen .
14 for dropout in [0.0, 0.25, 0.50, 0.75]:
15 # membuat variabel model Sequential()
16 model = Sequential()
17 #membuat looping untuk variabel i dengan jarak
18 dari hasil konvolusi .
19 for i in range(conv2d_count):
20 # syarat jika i samadengan bobotnya 0
21 if i == 0:
22 # menambahkan method add pada variabel
23 model dengan konvolusi 2 dimensi 32 bit didalamnya dan
24 membuat kernel dengan ukuran 3 x 3 dan rumus aktifasi relu
25 dan data shape yang di hitung dari data train.
26 model.add(Conv2D(32, kernel_size=(3, 3),
27 activation='relu', input_shape=np.shape(train_input[0])))
28 # jika tidak
29 else:
30 # menambahkan method add pada variabel
31 model dengan konvolusi 2 dimensi 32 bit dengan ukuran
32 kernel 3 x3 dan fungsi aktivasi relu
33 model.add(Conv2D(32, kernel_size=(3, 3),
34 activation='relu'))
35 # menambahkan method add pada variabel model
36 dengan isian method Max pooling berdimensi 2 dengan
37 ukuran fixcel 2 x 2.
38 model.add(MaxPooling2D(pool_size=(2, 2)))
39 # merubah feature gambar menjadi 1 dimensi vektor
40 model.add(Flatten())
41 # menambahkan method dense untuk pematatan data
42 dengan ukuran dense di tentukan dengan rumus fungsi tanh.
43 model.add(Dense(dense_size, activation='tanh'))
44 # membuat ketentuan jika pemangkasan lebih besar
45 dari 0 persen
46 if dropout > 0.0:
47 # menambahkan method dropout pada model
48 dengan nilai dari dropout
49 model.add(Dropout(dropout))
50 # menambahkan method dense dengan fungsi num
51 classs dan rumus softmax
52 model.add(Dense(num_classes, activation='softmax',
53))
54 # mongkompile variabel model dengan hasi loss
55 optimasi dan akurasi matrix
56 model.compile(loss='categorical_crossentropy',
57 optimizer='adam', metrics=['accuracy'])
```

```

38 # melakukan log pada dir
39 log_dir = './logs/conv2d_%d-dense_%d-dropout_.%2f'
40 , % (conv2d_count, dense_size, dropout)
41 # membuat variabel tensorboard dengan isian dari
42 # librari keras dan nilai dari lig dir
43 # tensorboard = keras.callbacks.TensorBoard(log_dir
44 =log_dir)
45 # membuat variabel start dengan isian dari
46 # librari time menggunakan method time
47
48 start = time.time()
49 # menambahkan method fit pada model dengan data
50 # dari train input train output nilai batch nilai epoch
51 # verbose nilai 20 persen validation split dan callback
52 # dengan nilai tnsorboard.
53 # model.fit(train_input, train_output, batch_size
54 =32, epochs=10,
55 verbose=0, validation_split=0.2,
56 callbacks=[tensorboard])
57 # membuat variabel score dengan nilai evaluasi
58 # dari model menggunakan data tes input dan tes output
59 score = model.evaluate(test_input, test_output,
60 verbose=2)
61 # membuat variabel end
62 end = time.time()
63 # membuat variabel elapsed
64 elapsed = end - start
65 # mencetak hasil perhitungan
66 print("Conv2D count: %d, Dense size: %d, Dropout: %2f - Loss: %2f, Accuracy: %2f, Time: %d sec" % (conv2d_count, dense_size, dropout, score[0], score[1], elapsed))
67 results.append((conv2d_count, dense_size, dropout,
68 , score[0], score[1], elapsed))

```

Untuk hasilnya dapat di lihat pada gambar berikut.

```

In [18]: import time
results = []
for conv2d_count in [1, 2]:
 for dense_size in [128, 256, 512, 1024, 2048]:
 for dropout in [0.0, 0.25, 0.50, 0.75]:
 model = Sequential()
 for i in range(conv2d_count):
 if i == 0:
 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=np.shape(train_input[0])))
 else:
 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
 model.add(Flatten())
 model.add(Dense(dense_size, activation='tanh'))
 if dropout > 0.0:
 model.add(Dropout(dropout))
 model.add(Dense(num_classes, activation='softmax'))

 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

 log_dir = './logs/conv2d_%d-dense_%d-dropout_.%2f' % (conv2d_count, dense_size, dropout)
 tensorboard = keras.callbacks.TensorBoard(log_dir=log_dir)

 start = time.time()
 model.fit(train_input, train_output, batch_size=32, epochs=10,
 verbose=0, validation_split=0.2, callbacks=[tensorboard])
 score = model.evaluate(test_input, test_output, verbose=2)
 end = time.time()
 elapsed = end - start
 print("Conv2D count: %d, Dense size: %d, Dropout: %2f - Loss: %2f, Accuracy: %2f, Time: %d sec" % (conv2d_count, dense_size, dropout, score[0], score[1], elapsed))
 results.append((conv2d_count, dense_size, dropout, score[0], score[1], elapsed))

```

Gambar 7.23 hasil kode program

14. Jelaskan kode program pada blok In[14].

```

1 # In[14]: rebuild/retrain a model with the best parameters (
 from the search) and use all data
2 # membuat variabel model dengan isian librari Sequential
3 model = Sequential()
4 # variabel model di tambahkan librari Conv2D tigapuluhan dua
 bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
 relu dang menggunakan data train_input
5 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
 input_shape=np.shape(train_input[0])))
6 # variabel model di tambahkan dengan lib MaxPooling2D dengan
 ketentuan ukuran 2 x 2 pixcel
7 model.add(MaxPooling2D(pool_size=(2, 2)))
8 # variabel model di tambahkan dengan librari Conv2D 32bit
 dengan kernel 3 x 3
9 model.add(Conv2D(32, (3, 3), activation='relu'))
10 # variabel model di tambahkan dengan lib MaxPooling2D dengan
 ketentuan ukuran 2 x 2 pixcel
11 model.add(MaxPooling2D(pool_size=(2, 2)))
12 # variabel model di tambahkan librari Flatten
13 model.add(Flatten())
14 # variabel model di tambahkan librari Dense dengan fungsi
 tanh
15 model.add(Dense(128, activation='tanh'))
16 # variabel model di tambahkan librari dropout untuk memangkas
 data tree sebesar 50 persen
17 model.add(Dropout(0.5))
18 # variabel model di tambahkan librari Dense dengan data dari
 num_classes dan fungsi softmax
19 model.add(Dense(num_classes, activation='softmax'))
20 # mengkompile data model untuk mendapatkan data loss akurasi
 dan optimasi
21 model.compile(loss='categorical_crossentropy', optimizer='adam',
 metrics=['accuracy'])
22 # mencetak variabel model kemudian memunculkan kesimpulan
 berupa data total parameter, trainable paremeter dan bukan
 trainable parameter
23 print(model.summary())

```

Untuk hasilnya dapat di lihat pada gambar berikut.

| Model: "sequential_2"          |                    |         |
|--------------------------------|--------------------|---------|
| Layer (Type)                   | Output Shape       | Param # |
| conv2d_3 (Conv2D)              | (None, 30, 30, 32) | 896     |
| max_pooling2d_3 (MaxPooling2D) | (None, 15, 15, 32) | 0       |
| conv2d_4 (Conv2D)              | (None, 13, 13, 32) | 9248    |
| max_pooling2d_4 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| flatten_2 (Flatten)            | (None, 1152)       | 0       |
| dense_3 (Dense)                | (None, 128)        | 147584  |
| dropout_2 (Dropout)            | (None, 128)        | 0       |
| dense_4 (Dense)                | (None, 369)        | 47501   |
| Total params: 205,329          |                    |         |
| Trainable params: 205,329      |                    |         |
| Non-trainable params: 0        |                    |         |
| None                           |                    |         |

Gambar 7.24 hasil kode program

15. Jelaskan kode program pada blok In[15].

```

1 # In[15]:join train and test data so we train the network on
 all data we have available to us
2 # melakukan join numpy menggunakan data train_input
 test_input
3 model.fit(np.concatenate((train_input, test_input)),
4 # kelanjutan data yang di gunakan pada join
 train_output test_output
5 np.concatenate((train_output, test_output)),
6 #menggunakan ukuran 32 bit dan epoch 10
7 batch_size=32, epochs=10, verbose=2)

```

Untuk hasilnya dapat di lihat pada gambar berikut.

```

Epoch 1/10
- 472s - loss: 1.7953 - accuracy: 0.5831
Epoch 2/10
- 244s - loss: 1.0884 - accuracy: 0.7067
Epoch 3/10
- 198s - loss: 0.9776 - accuracy: 0.7291
Epoch 4/10
- 213s - loss: 0.9183 - accuracy: 0.7402
Epoch 5/10
- 200s - loss: 0.8796 - accuracy: 0.7504
Epoch 6/10
- 297s - loss: 0.8495 - accuracy: 0.7549
Epoch 7/10
- 716s - loss: 0.8304 - accuracy: 0.7590
Epoch 8/10

```

**Gambar 7.25** hasil kode program

16. Jelaskan kode program pada blok In[16].

```

1 # In[16]:save the trained model
2 #menyimpan model atau mengekspor model yang telah di
 jalantadi
3 model.save("mathsymbols.model")

```

Untuk hasilnya dapat di lihat pada gambar berikut.

```
In [22]: # save the trained model
model.save("mathsymbols.model")
```

**Gambar 7.26** hasil kode program

17. Jelaskan kode program pada blok In[17].

```

1 # In[17]:save label encoder (to reverse one-hot encoding)
2 # menyimpan label encoder dengan nama classes.npy
3 np.save('classes.npy', label_encoder.classes_)

```

Untuk hasilnya dapat di lihat pada gambar berikut.

18. Jelaskan kode program pada blok In[18].

```

1 # In[18]:load the pre-trained model and predict the math
 symbol for an arbitrary image;
2 # the code below could be placed in a separate file
3 # mengimpport librari keras model

```

```

import keras.models
model2 = keras.models.load_model("mathsymbols.model")
print(model2.summary())
Model: "sequential_2"
Layer (type) Output Shape Param #
conv2d_3 (Conv2D) (None, 30, 30, 32) 896
max_pooling2d_3 (MaxPooling2D) (None, 15, 15, 32) 0
conv2d_4 (Conv2D) (None, 13, 13, 32) 9248
max_pooling2d_4 (MaxPooling2D) (None, 6, 6, 32) 0
flatten_2 (Flatten) (None, 1152) 0
dense_3 (Dense) (None, 128) 147584
dropout_2 (Dropout) (None, 128) 0
dense_4 (Dense) (None, 369) 47691
=====
Total params: 205,329
Trainable params: 205,329
Non-trainable params: 0
None

```

**Gambar 7.27** hasil kode program

```

4 import keras.models
5 # membuat variabel model2 untuk meload model yang telah di
 simpan tadi
6 model2 = keras.models.load_model("mathsymbols.model")
7 # mencetak hasil model2
8 print(model2.summary())

```

Untuk hasilnya dapat dilihat pada gambar berikut.

```

In [24]: # save Label encoder (to reverse one-hot encoding)
np.save('classes.npy', label_encoder.classes_)

```

**Gambar 7.28** hasil kode program

19. Jelaskan kode program pada blok In[19].

```

1 # In[19]: restore the class name to integer encoder
2 # membuat variabel label encoder ke 2 dengan isian fungsi
 label encoder.
3 label_encoder2 = LabelEncoder()
4 # menambahkan method classess dengan data classess yang di
 eksport tadi
5 label_encoder2.classes_ = np.load('classes.npy')
6 # membuat fungsi predict dengan path img
7 def predict(img_path):
8 # membuat variabel newimg dengan membuat image menjadi
 array dan membuka data berdasarkan img path
9 newimg = keras.preprocessing.image.img_to_array(pil_image
 .open(img_path))
10 # membagi data yang terdapat pada variabel newimg
 sebanyak 255
11 newimg /= 255.0
12
13 # do the prediction
14 # membuat variabel prediciton dengan isian variabel
 model2 menggunakan fungsi predict dengan syarat variabel
 newimg dengan data reshape

```

```

15 prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
16
17 # figure out which output neuron had the highest score,
18 # and reverse the one-hot encoding
19 # membuat variabel inverted dengan label encoder2 dan
20 menggunakan argmax untuk mencari skor luaran tertinggi
21 inverted = label_encoder2.inverse_transform([np.argmax(
22 prediction)])
23 # mencetak prediksi gambar dan confidence dari gambar.
24 print("Prediction: %s, confidence: %.2f" % (inverted[0],
25 np.max(prediction)))

```

Untuk hasilnya dapat di lihat pada gambar berikut.

```

In [25]: # restore the class name to integer encoder
label_encoder2 = LabelEncoder()
label_encoder2.classes_ = np.load('classes.npy')

def predict(img_path):
 newimg = keras.preprocessing.image.img_to_array(pil_image.open(img_path))
 newimg /= 255.0

 # do the prediction
 prediction = model2.predict(newimg.reshape(1, 32, 32, 3))

 # figure out which output neuron had the highest score, and reverse the one-hot encoding
 inverted = label_encoder2.inverse_transform([np.argmax(prediction)]) # argmax finds highest-scoring output
 print("Prediction: %s, confidence: %.2f" % (inverted[0], np.max(prediction)))

```

**Gambar 7.29** hasil kode program

20. Jelaskan kode program pada blok In[20].

```

1 # In [20]: grab an image (we'll just use a random training
2 # image for demonstration purposes)
3 # mencari prediksi menggunakan fungsi prediksi yang di buat
4 # tadi dari data di HASYv2/hasy-data/v2-00010.png
5 predict("HASYv2/hasy-data/v2-00010.png")
6 # mencari prediksi menggunakan fungsi prediksi yang di buat
7 # tadi dari data di HASYv2/hasy-data/v2-00500.png
8 predict("HASYv2/hasy-data/v2-00500.png")
9 # mencari prediksi menggunakan fungsi prediksi yang di buat
10 # tadi dari data di HASYv2/hasy-data/v2-00700.png
11 predict("hasy-data/v2-00700.png")

```

Untuk hasilnya dapat di lihat pada gambar berikut.

```

In [26]: # grab an image (we'll just use a random training image for demonstration purposes)
predict("HASYv2/hasy-data/v2-00010.png")
predict("HASYv2/hasy-data/v2-00500.png")
predict("HASYv2/hasy-data/v2-00700.png")

Prediction: A, confidence: 0.62
Prediction: \pi, confidence: 0.57
Prediction: \alpha, confidence: 0.94

```

**Gambar 7.30** hasil kode program

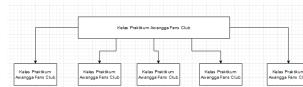
## 7.2 Luthfi Muhammad Nabil (1174035)

### 7.2.1 Soal Teori

Praktek teori penunjang yang dikerjakan total nilai 100 sebagai nilai terpisah dari praktek pada modul ini(nilai 5 per nomor kecuali nomor terakhir 30 sisanya merupakan penanganan error, untuk hari pertama) :

1. Jelaskan kenapa file teks harus di lakukan tokenizer. dilengkapi dengan ilustrasi atau gambar.

Tokenizer diperlukan untuk perhitungan bobot pada setiap teks karena Tokenizer akan membagi kalimat menjadi beberapa teks sehingga nantinya kalimat yang dimasukkan otomatis langsung diubah menjadi kata - kata dan akan dinilai sehingga akan memunculkan nilai vektor untuk digunakan saat prediksi teks yang muncul pada satu kalimat tersebut. Biasanya untuk memisahkan sebuah kata, tokenizer akan menggunakan spasi sebagai pemisah antar kata.



**Gambar 7.31** Illustrasi Tokenizer

2. Jelaskan konsep dasar K Fold Cross Validation pada dataset komentar Youtube pada kode listing ??.

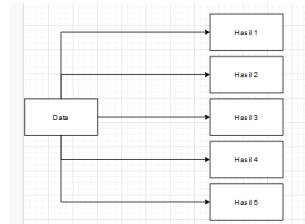
```

1 kfolds = StratifiedKFold(n_splits=5)
2 splits = kfolds.split(d, d['CLASS'])

```

**Listing 7.1** K Fold Cross Validation

Pada koding diatas terdapat variabel kfolds yang didalamnya berisi parameter split yang diisikan nilai 5. hal tersebut dimaksudkan untuk membuat pengolahan data akan diulang setiap datanya sebanyak lima kali dengan atribut class sebagai acuan pengolahan datanya. Lalu kemudian akan di hasilkan akurasi dari pengulangan data tersebut sebesar sekian persen tergantung datanya



**Gambar 7.32** Illustrasi K Fold Cross Validation

3. Jelaskan apa maksudnya kode program *for train, test in splits*.dilengkapi dengan ilustrasi atau gambar. For train digunakan untuk melakukan training atau pelatihan pada data yang sudah dideklarasikan sebelumnya. Sedangkan test in split digunakan untuk membatasi jumlah data yang akan diinputkan atau data yang akan digunakan.
4. Jelaskan apa maksudnya kode program *train\_content = d['CONTENT'].iloc[train\_idx]* dan *test\_content = d['CONTENT'].iloc[test\_idx]*. dilengkapi dengan ilustrasi atau gambar. Maksud dari kode program tersebut adalah membaca isian kolom pada df yang bernama CONTENT sebagai data training dan data testing untuk program

```
In [3]: import numpy as np
..... from sklearn.model_selection import train_test_split
..... x = np.array([1,2,3,4,5])
..... y = np.array([1,2,3,4,5])
..... print(x)
[1 2 3 4 5]
In [4]: train, test = train_test_split(x,y,test_size=0.3, random_state=42)
..... print(train)
[1 2 3]
In [5]: print(test)
[4 5]
```

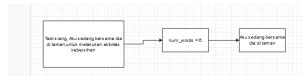
**Gambar 7.33** Illustrasi For train dan test in split

| No | Nama       | Content                                                   |
|----|------------|-----------------------------------------------------------|
| 1  | Mobil      | Kendaraan darat yang biasanya memiliki roda 4             |
| 2  | Motor      | Kendaraan darat yang biasanya memiliki roda 2             |
| 3  | Traktor    | Kendaraan darat yang dipakai untuk membajak sawah         |
| 4  | Helikopter | Kendaraan udara yang memiliki sayap - sayap untuk terbang |

**Gambar 7.34** Illustrasi penggunaan kolom Content

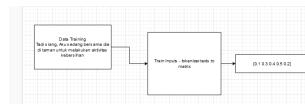
5. Jelaskan apa maksud dari fungsi *tokenizer = Tokenizer(num\_words=2000)* dan *tokenizer.fit\_on\_texts(train\_content)*, dilengkapi dengan ilustrasi atau gambar.

- *tokenizer = Tokenizer(num\_words=2000)* digunakan untuk membaca kalimat yang telah dibuat menjadi token sebanyak 2000 kata
- *fit\_on\_texts* digunakan untuk membuat membaca data token teks yang telah dimasukan kedalam fungsi yaitu fungsi *train\_content*



**Gambar 7.35** Illustrasi fit tokenizer dan num\_word=2000

6. Jelaskan apa maksud dari fungsi *d.train\_inputs = tokenizer.texts\_to\_matrix(train\_content, mode='tfidf')* dan *d.test\_inputs = tokenizer.texts\_to\_matrix(test\_content, mode='tfidf')*, dilengkapi dengan ilustrasi kode dan atau gambar. Untuk digunakan sebagai pengubah urutan teks yang tadi telah dilakukan tokenizer menjadi matriks yang berurutan seperti tf idf



**Gambar 7.36** Illustrasi d train inputs = tokenizer.texts to matrix

7. Jelaskan apa maksud dari fungsi  $d\_train\_inputs = d\_train\_inputs / np.amax(np.abs(d\_train\_inputs))$  dan  $d\_test\_inputs = d\_test\_inputs / np.amax(np.absolute(d\_test\_inputs))$ , dilengkapi dengan ilustrasi atau gambar. Fungsi tersebut digunakan untuk membagi matriks tfidf dengan penentuan maksimum array sepanjang sumbu sehingga akan menimbulkan garis ke bawah dan ke atas yang membentuk gambar v. Lalu hasil tersebut akan dimasukkan ke variabel d train input dan d test input dengan metode absolute. Yang berarti tanpa bilangan negatif.
8. Jelaskan apa maksud fungsi dari  $d\_train\_outputs = np_utils.to_categorical(d['CLASS'])$  dan  $d\_test\_outputs = np_utils.to_categorical(d['CLASS'].iloc[test_idx])$  dalam kode program, dilengkapi dengan ilustrasi atau gambar. Maksud dari fungsi tersebut yaitu untuk merubah nilai vektor yang ada pada atribut class menjadi bentuk matrix dengan pengurutan berdasarkan data index training dan testing.
9. Jelaskan apa maksud dari fungsi di listing ?.?. Gambarkan ilustrasi Neural Network nya dari model kode tersebut.

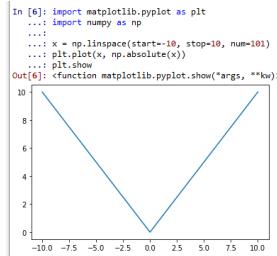
```

1 model = Sequential()
2 model.add(Dense(512, input_shape=(2000,)))
3 model.add(Activation('relu'))
4 model.add(Dropout(0.5))
5 model.add(Dense(2))
6 model.add(Activation('softmax'))

```

**Listing 7.2** Membuat model Neural Network

model = sequential berarti variabel model berisi method sequential yang berguna untuk searching data dengan menerima parameter atau argumen kunci dengan langkah tertentu untuk mencari data yang telah diolah. Kemudian model akan ditambahkan method add dengan dense yang berarti data - data yang diinputkan akan terhubung, dengan data 612 dan 2000 data kata atau word kemudian model tersebut di masukan fungsi activation dengan rumus atau metode relu. setelah itu data akan di dropout 0.5atau dipangkas sebanyak 50 persen dikarenakan pada pohon bobot terlalu akurat terhadap data.



**Gambar 7.37** Illustrasi d train inputs

```
In [10]: labels = [0,2,1,2,0,1]
...: keras.utils.to_categorical(labels)Out[11]:
array([[1., 0., 0.],
 [0., 1., 0.],
 [0., 0., 1.],
 [0., 0., 1.],
 [1., 0., 0.],
 [0., 1., 0.]], dtype=float32)
```

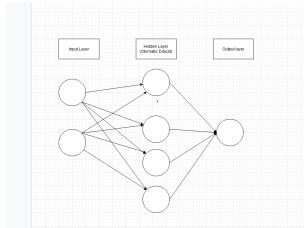
**Gambar 7.38** Illustrasi train outputs = np utils.to categorical

10. Jelaskan apa maksud dari fungsi di listing ?? dengan parameter tersebut.

```
1 model.compile(loss='categorical_crossentropy', optimizer='
2 adamax',
3 metrics=['accuracy'])
```

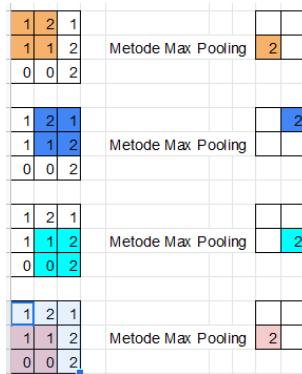
**Listing 7.3** Compile model

model tersebut kemudian di compile atau di kembalikan kembali fungsi nilainya yang mana akan mengembalikan fungsi nilai loss nya berapa yang diambil dari fungsi adamax yang berberguna untuk mengetahui nilai loss-nya kemudian metrics = accuracy merupakan akurasi dari nilai matrixnya. kemudian terdiri atas beberapa layer atau hidden layer. perbedaan antara deep learning dan DNN atau Deep Neural Network yaitu deep learning merupakan pemakai algoritma dari DNN dan DNN merupakan algoritma yang ada pada deep learning.



**Gambar 7.39** Illustrasi Neural Network

11. Jelaskan apa itu Deep Learning Deep learning merupakan salah satu algoritma yang seperti Neural Network yang menggunakan meta data sebagai inputan dan mengolahnya menggunakan layer layer yang tersembunyi.
12. Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning Deep Neural Network merupakan algoritma jaringan syaraf yang melakukan pembobotan terhadap data yang sudah ada sebagai acuan untuk data inputan selanjutnya.
13. Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride (NPM mod3+1) x (NPM mod3+1) yang terdapat max pooling.(nilai 30) sebelum membuat ilustrasi perlu di ketahui apa itu stride, stride adalah acuan atau parameter yang menentukan pergeseran pada lter xcel. sebagai contoh nilai stride 1 yang berarti lter akan bergeser sebanyak satu xcel secara vertikal dan horizontal. selanjutnya apa itu max pooling contoh pada suatu gambar di tentukan Max Pooling dari 3 x 3 dengan stride 1 yang berarti setiap pergeseran 1 pixcel akan diambil nilai terbesar dari pixcel 3 x 3 tersebut.



**Gambar 7.40** Illustrasi perhitungan stride 1 max pooling

### 7.2.2 Praktek Program

Tugas nilai terpisah dari teori maksimal 100. Praktekkan dan jelaskan dengan menggunakan bahasa yang mudah dimengerti dan bebas plagiat dan wajib skrinsut dari komputer sendiri masing masing nomor di bawah ini(nilai 5 masing masing pada hari kedua). Buka kode program pada repo Python-Artificial-Intelligence-Projects-for-Beginners pada github awangga. Buka folder Chapter04 file MathSymbols.py

1. Jelaskan kode program pada blok # In[1]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In [1]:import lib
2 # menimpor libtari CSV untuk mengolah data ber ekstensi csv
3 import csv
4 #kemudian Melakukan import library Image yang berguna untuk
 dari PIL atau Python Imaging Library yang berguna untuk
 mengolah data berupa gambar
5 from PIL import Image as pil_image
6 # kemudian Melakukan import library keras yang menggunakan
 method preprocessing yang digunakan untuk membuat neural
 network
7 import keras.preprocessing.image

```

```

In [14]: import csv
...# Memindai file hasy-data-labels.csv dan melakukan import library Image yang berguna untuk
...# mengolah data berupa gambar
...# Menggunakan library Image yang berguna untuk mengolah data berupa
...# gambar
...# Kemudian melakukan import library keras yang menggunakan
...# method preprocessing yang digunakan untuk membuat neural network
...# Import keras.preprocessing.image

```

**Gambar 7.41** Illustrasi perhitungan stride 1 max pooling

2. Jelaskan kode program pada blok # In[2]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # In [2]:load all images (as numpy arrays) and save their
 classes
2 #Menginisiasi variabel imgs dan classes dengan variabel array
 kosong
3 imgs = []
4 classes = []
5 #membuka file hasy-data-labels.csv yang berada di foleder
 HASYv2 yang di inisialisasi menjadi csvfile
6 with open('HASYv2/hasy-data-labels.csv') as csvfile:
7 #Menginisiasi variabel csvreader yang berisi method csv.
 reader yang membaca variabel csvfile
8 csvreader = csv.reader(csvfile)
9 # Menginisiasi variabel i dengan isi 0
10 i = 0
11 # membuat looping pada variabel csvreader
12 for row in csvreader:
13 # dengan ketentuan jika i lebihkecil daripada 0
14 if i > 0:
15 # dibuat variabel img dengan isi keras untuk
 aktivasi neural network fungsi yang membaca data yang
 berada dalam folder HASYv2 dengan input nilai -1.0 dan 1.0
16 img = keras.preprocessing.image.img_to_array(
 pil_image.open("HASYv2/" + row[0]))
17 #Pembagian data yang ada pada fungsi img sebanyak
 255.0
18 img /= 255.0
19 # Penambahan nilai baru pada imgs pada row ke 1 2
 dan dilanjutkan dengan variabel img
20 imgs.append((row[0], row[2], img))
21 # Penambahan nilai pada row ke 2 pada variabel
 classes
22 classes.append(row[2])

```

```

23 # penambahan nilai satu pada variabel i
24 i += 1
25
26 # In [3]: shuffle the data, split into 80% train , 20% test
27 # Melakukan import library random
28 import random

```

```

In [34]: imgs = []
...: classes = []
...: #membuka file hasil-data-labels.csv yang berada di folder HASV2
...: with open('hasil-data-labels.csv') as csvfile:
...: ...: menginisiasi variabel csvreader yang berisi method
...: csvreader = csv.reader(csvfile)
...: ...: csvreader = csv.reader(csvfile)
...: i = 0
...: # membuat looping pada variabel csvreader
...: for row in csvreader:
...: ...: dengan ketentuan jika i lebih besar dari pada o
...: if i > 0:
...: ...: membuat variabel img dengan isi keras untuk
...: #membaca neural network fungsi yang membaca data yang berada dalam folder
...: #membaca data yang ada pada fungsi img sebanyak
...: img =
...: keras.preprocessing.image.img_to_array(pil_image.open("HASV2/" +
...: row[0]))
...: #Penambahan data yang ada pada fungsi img sebanyak
235.0
...: img /= 255.0
...: # Penambahan nilai baru pada imgs pada row ke 1
...: imgs.append((row[0], row[1], img))
...: # Penambahan nilai pada row ke 2 pada variabel
...: classes.append(row[1])
...: # penambahan nilai satu pada variabel i
...: i += 1

```

**Gambar 7.42** Illustrasi perhitungan stride 1 max pooling

3. Jelaskan kode program pada blok **# In[3]**. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # melakukan random pada vungsi imgs
2 random.shuffle(imgs)
3 # Menginisiasi variabel split_idx dengan nilai integer 80
 persen dikali dari pengembalian jumlah dari variabel imgs
4 split_idx = int(0.8*len(imgs))
5 # Menginisiasi variabel train dengan isi lebih besar split
 idx
6 train = imgs[:split_idx]
7 # Menginisiasi variabel test dengan isi lebih kecil split idx
8 test = imgs[split_idx:]
9
10 # In [4]:
11 # Melakukan import library numpy dengan inisial np
12 import numpy as np

```

```

In [3]: import random
...: # melakukan random pada vungsi imgs
...: random.shuffle(imgs)
...: # membuat variabel split_idx dengan nilai integer 80 persen
dibagi dari pengembalian jumlah dari variabel imgs
...: split_idx = int(0.8*len(imgs))
...: # menginisiasi variabel train dengan isi lebih besar split_idx
...: train = imgs[:split_idx]
...: # menginisiasi variabel test dengan isi lebih kecil split_idx
...: test = imgs[split_idx]

```

**Gambar 7.43** Illustrasi perhitungan stride 1 max pooling

4. Jelaskan kode program pada blok **# In[4]**. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # Menginisiasi variabel train_input dengan np method asarray
 yang mana membuat array dengan isi row 2 dari data train
2 train_input = np.asarray(list(map(lambda row: row[2], train)))
)
3 # membuat test_input dengan np method asarray yang mana
 membuat array dengan isi row 2 dari data test
4 test_input = np.asarray(list(map(lambda row: row[2], test)))
5 # Menginisiasi variabel train_output dengan np method asarray
 yang mana membuat array dengan isi row 1 dari data train
6 train_output = np.asarray(list(map(lambda row: row[1], train)))
)
7 # Menginisiasi variabel test_output dengan np method asarray
 yang mana membuat array dengan isi row 1 dari data test
8 test_output = np.asarray(list(map(lambda row: row[1], test)))
9
10 # In [5]: import encoder and one hot
11 # Melakukan import library LabelEncode dari sklearn
12 from sklearn.preprocessing import LabelEncoder

```

```

In [8]: Import numpy as np
...: # Menginisiasi variabel train_input dengan np method asarray yang
 mana membuat array dengan isi row 2 dari data train
...: train_input = np.asarray(list(map(lambda row: row[2], train)))
...: # Menginisiasi variabel test_input dengan np method asarray yang
 mana membuat array dengan isi row 2 dari data test
...: test_input = np.asarray(list(map(lambda row: row[2], test)))
...: # Menginisiasi variabel train_output dengan np method asarray yang
 mana membuat array dengan isi row 1 dari data train
...: train_output = np.asarray(list(map(lambda row: row[1], train)))
...: # Menginisiasi variabel test_output dengan np method asarray yang
 mana membuat array dengan isi row 1 dari data test
...: test_output = np.asarray(list(map(lambda row: row[1], test)))

```

**Gambar 7.44** Illustrasi perhitungan stride 1 max pooling

5. Jelaskan kode program pada blok # In[5]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # Melakukan import library OneHotEncoder dari sklearn
2 from sklearn.preprocessing import OneHotEncoder
3
4 # In [6]: convert class names into one-hot encoding
5
6 # Menginisiasi variabel label_encoder dengan isi LabelEncoder

```

```

In [9]: from sklearn.preprocessing import LabelEncoder
...: # Melakukan import library OneHotEncoder dari sklearn
...: from sklearn.preprocessing import OneHotEncoder

```

**Gambar 7.45** Illustrasi perhitungan stride 1 max pooling

6. Jelaskan kode program pada blok # In[6]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 label_encoder = LabelEncoder()
2 # Menginisiasi variabel integer_encoded yang berfungsi untuk
 Menconvert variabel classes kedalam bentuk integer
3 integer_encoded = label_encoder.fit_transform(classes)
4

```

```
5 # In [7]: then convert integers into one-hot encoding
6 # Menginisiasi variabel onehot_encoder dengan isi
 OneHotEncoder
7 onehot_encoder = OneHotEncoder(sparse=False)
```

```
In [10]: label_encoder = LabelEncoder()
...: # Menginisiasi variabel integer_encoded yang berfungsi untuk
...: # Mengconvert variabel classes kedalam kategori integer
...: integer_encoded = label_encoder.fit_transform(classes)
```

**Gambar 7.46** Illustrasi perhitungan stride 1 max pooling

7. Jelaskan kode program pada blok # In[7]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # mengisi variabel integer_encoded dengan isi integer-encoded
2 # yang telah di convert pada fungsi sebelumnya
3 integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
4 # Menconvert variabel integer_encoded kedalam onehot_encoder
5 onehot_encoder.fit(integer_encoded)
6
7 # In [8]: convert train and test output to one-hot
8 # Menconvert data train output menggunakan variabel
label_encoder kedalam variabel train_output_int
9 train_output_int = label_encoder.transform(train_output)
```

**Gambar 7.47** Illustrasi perhitungan stride 1 max pooling

8. Jelaskan kode program pada blok # In[8]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # Menconvert variabel train_output_int kedalam fungsi
 onehot_encoder
2 train_output = onehot_encoder.transform(train_output_int.
 reshape(len(train_output_int), 1))
3 # Menconvert data test_output menggunakan variabel
 label_encoder kedalam variabel test_output_int
4 test_output_int = label_encoder.transform(test_output)
5 # Menconvert variabel test_output_int kedalam fungsi
 onehot_encoder
6 test_output = onehot_encoder.transform(test_output_int.
 reshape(len(test_output_int), 1))
```

```

7 # Menginisiasi variabel num_classes dengan isi variabel
 label_encoder dan classes
8 num_classes = len(label_encoder.classes_)
9 # mencetak hasil dari nomer Class berupa persen
10 print("Number of classes: %d" % num_classes)
11
12 # In [9]: import sequential
13 # Melakukan import library Sequential dari Keras
14 from keras.models import Sequential

```

```

In [12]: train_output_int = label_encoder.transform(train_output)
...# Mengonvert variabel train_output_int kedalam fungsi
...# reshape
...# train_output =
...# cekpoint_encoder.transform(train_output_int.reshape(len(train_output_int),
13))
...# Mengonvert data test_output menggunakan variabel label_encoder
...# reshape
...# test_output_int = label_encoder.transform(test_output)
...# Mengonvert variabel test_output_int kedalam fungsi
...# reshape
...# test_output =
...# cekpoint_encoder.transform(test_output_int.reshape(len(test_output_int),
13))
...# Mengonvert variabel num_classes dengan isi variabel
...# label_encoder.classes_
...# num_classes = len(label_encoder.classes_)
...# mencetak hasil dari nomer Class berupa persen
...# print("Number of classes: %d" % num_classes)
Number of classes: 369

```

**Gambar 7.48** Illustrasi perhitungan stride 1 max pooling

9. Jelaskan kode program pada blok # In[9]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # Melakukan import library Dense, Dropout, Flatten dari Keras
2 from keras.layers import Dense, Dropout, Flatten
3 # Melakukan import library Conv2D, MaxPooling2D dari Keras
4 from keras.layers import Conv2D, MaxPooling2D
5
6 # In [10]: desain jaringan
7 # Menginisiasi variabel model dengan isian library Sequential
8 model = Sequential()

```

```

In [13]: from keras.models import Sequential
...# Melakukan import library Dense, Dropout, Flatten dari Keras
...# from keras.layers import Dense, Dropout, Flatten
...# ...# Menginisiasi variabel model dengan isian library Sequential
...# ...# from keras.layers import Conv2D, MaxPooling2D

```

**Gambar 7.49** Illustrasi perhitungan stride 1 max pooling

10. Jelaskan kode program pada blok # In[10]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # variabel model di tambahkan library Conv2D tigapuluhan dua
 bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
 relu dang menggunakan data train_input
2 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
3 input_shape=np.shape(train_input[0])))
4 # variabel model di tambahkan dengan lib MaxPooling2D dengan
 ketentuan ukuran 2 x 2 pixcel
5 model.add(MaxPooling2D(pool_size=(2, 2)))
6 # variabel model di tambahkan dengan library Conv2D 32bit
 dengan kernel 3 x 3

```

```

7 model.add(Conv2D(32, (3, 3), activation='relu'))
8 # variabel model di tambahkan dengan lib MaxPooling2D dengan
9 # ketentuan ukuran 2 x 2 pixcel
10 model.add(MaxPooling2D(pool_size=(2, 2)))
11 # variabel model di tambahkan library Flatten
12 model.add(Flatten())
13 # variabel model di tambahkan library Dense dengan fungsi
14 # tanh
15 model.add(Dense(1024, activation='tanh'))
16 # variabel model di tambahkan library dropout untuk memangkas
17 # data tree sebesar 50 persen
18 model.add(Dropout(0.5))
19 # variabel model di tambahkan library Dense dengan data dari
20 # num_classes dan fungsi softmax
21 model.add(Dense(num_classes, activation='softmax'))
22 # mengkompile data model untuk mendapatkan data loss akurasi
23 # dan optimasi
24 model.compile(loss='categorical_crossentropy', optimizer='adam',
25 metrics=['accuracy'])
26 # mencetak variabel model kemudian memunculkan kesimpulan
berupa data total parameter, trainable paremeter dan bukan
trainable parameter
27 print(model.summary())
28
29 # In[11]: import sequential
30 # Melakukan import library keras callbacks
31 import keras.callbacks

```

| Model: sequential_1            |                    |         |
|--------------------------------|--------------------|---------|
| Layer (Type)                   | Output Shape       | Param # |
| conv2d_1 (Conv2D)              | (None, 30, 30, 32) | 896     |
| max_pooling2d_1 (MaxPooling2D) | (None, 15, 15, 32) | 0       |
| conv2d_2 (Conv2D)              | (None, 15, 15, 32) | 9248    |
| max_pooling2d_2 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| flatten_1 (Flatten)            | (None, 1152)       | 0       |
| dense_1 (Dense)                | (None, 1024)       | 1180672 |
| dropout_1 (Dropout)            | (None, 1024)       | 0       |
| dense_2 (Dense)                | (None, 360)        | 378225  |
| Total params: 1,569,041        |                    |         |
| Trainable params: 1,569,041    |                    |         |
| Non-trainable params: 0        |                    |         |
| None                           |                    |         |

Gambar 7.50 Illustrasi perhitungan stride 1 max pooling

11. Jelaskan kode program pada blok # In[11]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # Menginisiasi variabel tensorboard dengan isi lib keras
2 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/
mnist-style')
3
4 # In[12]: 5menit kali 10 epoch = 50 menit
5 # fungsi model titambahkan metod fit untuk mengetahui
perhitungan dari train_input train_output
6 model.fit(train_input, train_output,

```

```
In [19]: import keras.callbacks
...# Fungsi yang memanggil tensorboard dengan isi lib keras
...# tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/mnist-
style')
```

**Gambar 7.51** Illustrasi perhitungan stride 1 max pooling

12. Jelaskan kode program pada blok # In[12]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 # dengan batch size 32 bit
2 batch_size=32,
3 epochs=10,
4 verbose=2,
5 validation_split=0.2,
6 callbacks=[tensorboard])
7
8 score = model.evaluate(test_input , test_output , verbose=2)
9 print('Test loss:', score[0])
10 print('Test accuracy:', score[1])
```

```
187668/187668 - 88s - loss: 1.5529 - accuracy: 0.6278 - val_loss: 0.9929
- val_accuracy: 0.7288
Epoch 1/10
187668/187668 - 81s - loss: 0.9793 - accuracy: 0.7387 - val_loss: 0.8903
- val_accuracy: 0.7510
Epoch 2/10
187668/187668 - 78s - loss: 0.8683 - accuracy: 0.7533 - val_loss: 0.8577
- val_accuracy: 0.7545
Epoch 3/10
187668/187668 - 82s - loss: 0.7983 - accuracy: 0.7676 - val_loss: 0.8427
- val_accuracy: 0.7629
Epoch 4/10
187668/187668 - 81s - loss: 0.7511 - accuracy: 0.7771 - val_loss: 0.8347
- val_accuracy: 0.7700
Epoch 5/10
187668/187668 - 81s - loss: 0.7691 - accuracy: 0.7872 - val_loss: 0.8329
- val_accuracy: 0.7767
Epoch 6/10
187668/187668 - 85s - loss: 0.6712 - accuracy: 0.7951 - val_loss: 0.8389
- val_accuracy: 0.7657
Epoch 7/10
187668/187668 - 85s - loss: 0.6437 - accuracy: 0.8014 - val_loss: 0.8562
- val_accuracy: 0.7735
Epoch 8/10
187668/187668 - 86s - loss: 0.6216 - accuracy: 0.8098 - val_loss: 0.8719
- val_accuracy: 0.7672
Epoch 9/10
187668/187668 - 86s - loss: 0.5980 - accuracy: 0.8188 - val_loss: 0.8638
- val_accuracy: 0.7652
33449/33449 - loss: 0.8659 - accuracy: 0.7634
Test loss: 0.865981729218628
Test accuracy: 0.76339644
```

**Gambar 7.52** Illustrasi perhitungan stride 1 max pooling

13. Jelaskan kode program pada blok # In[13]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```
1 #Menginisiasi variabel result dengan array kosong
2 results = []
3 # melakukan looping dengan ketentuan konvolusi 2 dimensi 1 2
4 for conv2d_count in [1, 2]:
5 # menentukan ukuran besaran fixcel dari data atau konvert
6 1 fixcel mnjadi data yang berada pada codigan dibawah.
7 for dense_size in [128, 256, 512, 1024, 2048]:
8 # membuat looping untuk memangkas masing-masing data
9 dengan ketentuan 0 persen 25 persen 50 persen dan 75
10 persen.
11 for dropout in [0.0, 0.25, 0.50, 0.75]:
12 # Menginisiasi variabel model Sequential
13 model = Sequential()
```

```
11 #membuat looping untuk variabel i dengan jarak
12 dari hasil konvolusi.
13 for i in range(conv2d_count):
14 # syarat jika i samadengan bobotnya 0
15 if i == 0:
16 # Penambahan method add pada variabel
17 model dengan konvolusi 2 dimensi 32 bit didalamnya dan
18 membuat kernel dengan ukuran 3 x 3 dan rumus aktifasi relu
19 dan data shape yang di hitung dari data train.
20 model.add(Conv2D(32, kernel_size=(3, 3),
21 activation='relu', input_shape=np.shape(train_input[0])))
22 # jika tidak
23 else:
24 # Penambahan method add pada variabel
25 model dengan konvolusi 2 dimensi 32 bit dengan ukuran
26 kernel 3 x3 dan fungsi aktivasi relu
27 model.add(Conv2D(32, kernel_size=(3, 3),
28 activation='relu'))
29 # Penambahan method add pada variabel model
30 dengan isian method Max pooling berdimensi 2 dengan
31 ukuran fixcel 2 x 2.
32 model.add(MaxPooling2D(pool_size=(2, 2)))
33 # merubah feature gambar menjadi 1 dimensi vektor
34 model.add(Flatten())
35 # Penambahan method dense untuk pematatan data
36 dengan ukuran dense di tentukan dengan rumus fungsi tanh.
37 model.add(Dense(dense_size, activation='tanh'))
38 # membuat ketentuan jika pemangkasan lebih besar
39 dari 0 persen
40 if dropout > 0.0:
41 # Penambahan method dropout pada model dengan
42 nilai dari dropout
43 model.add(Dropout(dropout))
44 # Penambahan method dense dengan fungsi num
45 classss dan rumus softmax
46 model.add(Dense(num_classes, activation='softmax',
47))
48 # mongkompile variabel model dengan hasi loss
49 optimasi dan akurasi matrix
50 model.compile(loss='categorical_crossentropy',
51 optimizer='adam', metrics=['accuracy'])
52 # melakukan log pada dir
53 log_dir = './logs/conv2d_%d-dense_%d-dropout_.%2f'
54 % (conv2d_count, dense_size, dropout)
55 # Menginisiasi variabel tensorboard dengan isian
56 dari library keras dan nilai dari lig dir
57 tensorboard = keras.callbacks.TensorBoard(log_dir
58 =log_dir)
59 # Menginisiasi variabel start dengan isian dari
60 library time menggunakan method time
61
62 start = time.time()
63 # Penambahan method fit pada model dengan data
64 dari train input train output nilai batch nilai epoch
65 verbose nilai 20 persen validation split dan callback
66 dengan nilai tnsorboard.
```

```

43 model.fit(train_input, train_output, batch_size
44 =32, epochs=10,
45 verbose=0, validation_split=0.2,
46 callbacks=[tensorboard])
 # Menginisiasi variabel score dengan nilai
47 evaluasi dari model menggunakan data tes input dan tes
48 output
49 score = model.evaluate(test_input, test_output,
50 verbose=2)
 # Menginisiasi variabel end
51 end = time.time()
 # Menginisiasi variabel elapsed
52 elapsed = end - start
 # mencetak hasil perhitungan
53 print("Conv2D count: %d, Dense size: %d, Dropout:
54 %.2f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" %
55 (conv2d_count, dense_size, dropout, score[0], score[1],
56 elapsed))
57 results.append((conv2d_count, dense_size, dropout,
58 score[0], score[1], elapsed))
59
60
61 # In [14]: rebuild/retrain a model with the best parameters (
62 from the search) and use all data
63 # Menginisiasi variabel model dengan isian library Sequential
64 model = Sequential()

```

```

65
66 % (conv2d_count, dense_size, dropout)
67 # Menginisiasi variabel tensorboard dengan isian
68 dari library keras dan nilai dari tig dir
69 tensorboard =
70 keras.callbacks.TensorBoard(log_dir=log_dir)
71 # Menginisiasi variabel start dengan nilai dari
72 library time menggunakan method time
73 ...
74 start = time.time()
75 # Menginisiasi variabel fit pada model dengan data dor
76 trien input train output nilai batch nilai epoch verbose nilai 20 persi
77 validation split dan callbacks dengan nilai tensorboard.
78 ...
79 model.fit(train_input, train_output, batch_size=32
80 epochs=10,
81 verbose=0, validation_split=0.2,
82 callbacks=[tensorboard])
 # Menginisiasi variabel score dengan nilai evaluasi
83 dari model menggunakan data tes input dan tes output
84 score = model.evaluate(test_input, test_output,
85 verbose=2)
 # Menginisiasi variabel end
86 end = time.time()
 # Menginisiasi variabel elapsed
87 elapsed = end - start
 # mencetak hasil perhitungan
88 print("Conv2D count: %d, Dense size: %d, Dropout:
89 %.2f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" %
90 (conv2d_count,
91 dense_size, dropout, score[0], score[1], elapsed))
 ...
92 results.append((conv2d_count, dense_size, dropout,
93 score[0], score[1], elapsed))
94

```

**Gambar 7.53** Illustrasi perhitungan stride 1 max pooling

14. Jelaskan kode program pada blok **# In[14]**. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # variabel model di tambahkan library Conv2D tigapuluhan dua
2 bit dengan ukuran kernel 3 x 3 dan fungsi penghitungan
3 relu dang menggunakan data train_input
4 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
5 input_shape=np.shape(train_input[0])))
6 # variabel model di tambahkan dengan lib MaxPooling2D dengan
7 ketentuan ukuran 2 x 2 pixel
8 model.add(MaxPooling2D(pool_size=(2, 2)))

```

```

5 # variabel model di tambahkan dengan library Conv2D 32 bit
 dengan kernel 3 x 3
6 model.add(Conv2D(32, (3, 3), activation='relu'))
7 # variabel model di tambahkan dengan lib MaxPooling2D dengan
 ketentuan ukuran 2 x 2 pixcel
8 model.add(MaxPooling2D(pool_size=(2, 2)))
9 # variabel model di tambahkan library Flatten
10 model.add(Flatten())
11 # variabel model di tambahkan library Dense dengan fungsi
 tanh
12 model.add(Dense(128, activation='tanh'))
13 # variabel model di tambahkan library dropout untuk memangkas
 data tree sebesar 50 persen
14 model.add(Dropout(0.5))
15 # variabel model di tambahkan library Dense dengan data dari
 num_classes dan fungsi softmax
16 model.add(Dense(num_classes, activation='softmax'))
17 # mengkompile data model untuk mendapatkan data loss akurasi
 dan optimasi
18 model.compile(loss='categorical_crossentropy', optimizer='adam',
 metrics=['accuracy'])
19 # mencetak variabel model kemudian memunculkan kesimpulan
 berupa data total parameter, trainable paremeter dan bukan
 trainable parameter
20 print(model.summary())
21 # In[15]:join train and test data so we train the network on
 all data we have available to us
22 # melakukan join numpy menggunakan data train_input
 test_input
23 model.fit(np.concatenate((train_input, test_input)),

```

| Layer (Type)                    | Output Shape       | Param # |
|---------------------------------|--------------------|---------|
| conv2d_10 (Conv2D)              | (None, 10, 10, 32) | 896     |
| max_pooling2d_10 (MaxPooling2D) | (None, 5, 5, 32)   | 0       |
| conv2d_11 (Conv2D)              | (None, 13, 13, 32) | 9248    |
| max_pooling2d_11 (MaxPooling2D) | (None, 6, 6, 32)   | 0       |
| flatten_8 (Flatten)             | (None, 1152)       | 0       |
| dense_16 (Dense)                | (None, 128)        | 147554  |
| dropout_2 (Dropout)             | (None, 128)        | 0       |
| dense_17 (Dense)                | (None, 369)        | 47682   |
| Total params: 295,329           |                    |         |
| Trainable params: 205,329       |                    |         |
| Non-trainable params: 0         |                    |         |
| None                            |                    |         |

Gambar 7.54 Illustrasi perhitungan stride 1 max pooling

15. Jelaskan kode program pada blok # In[15]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # kelanjutan data yang di gunakan pada join
 train_output test_output
2 np.concatenate((train_output, test_output)),
3 #menggunakan ukuran 32 bit dan epoch 10
4 batch_size=32, epochs=10, verbose=2)
5
6 # In[16]:save the trained model

```

```

7 #menyimpan model atau mengekspor model yang telah di
 jalantadi
8 model.save("mathsymbols.model")

```

```

Epoch 1/10
168233/168233 - 83s - loss: 1.8214 - accuracy: 0.5784
Epoch 2/10
168233/168233 - 81s - loss: 1.0979 - accuracy: 0.7021
Epoch 3/10
168233/168233 - 77s - loss: 0.9910 - accuracy: 0.7246
Epoch 4/10
168233/168233 - 76s - loss: 0.9328 - accuracy: 0.7376
Epoch 5/10
168233/168233 - 79s - loss: 0.8945 - accuracy: 0.7453
Epoch 6/10
168233/168233 - 81s - loss: 0.8649 - accuracy: 0.7511
Epoch 7/10
168233/168233 - 78s - loss: 0.8446 - accuracy: 0.7553
Epoch 8/10
168233/168233 - 75s - loss: 0.8256 - accuracy: 0.7598
Epoch 9/10
168233/168233 - 74s - loss: 0.8111 - accuracy: 0.7616
Epoch 10/10
168233/168233 - 79s - loss: 0.8015 - accuracy: 0.7650

```

**Gambar 7.55** Illustrasi perhitungan stride 1 max pooling

16. Jelaskan kode program pada blok # In[16]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1
2 # In[17]: save label encoder (to reverse one-hot encoding)
3 # menyimpan label encoder dengan nama classes.npy
4 np.save('classes.npy', label_encoder.classes_)

```

```

In [80]: model.save("mathsymbols.model")
WARNING:tensorflow:From C:\Users\TITIKOYIK_01\AppData\Roaming\Python
\python37\site-packages\tensorflow\python\framework\python_op.py:126:
ResourceVariable.__init__(self, name, shape, dtype, initializer,
 constraint, validate_shape, use_resource=True,
 trainable=True, collections=None, **kwargs):
 warnings.warn("ResourceVariable.__init__(name, shape, dtype, "
 "initializer, constraint, validate_shape, "
 "use_resource=True, trainable=True, collections=None, "
 "**kwargs) is deprecated and will be removed in a future version.
 Instead, use ResourceVariable(name, shape, dtype, "
 "initializer, constraint, validate_shape, "
 "trainable=True, collections=None, "
 "constraint_args=constraint)."
 "If using Keras, pass `constraint` arguments to layers."
In [80]: tensorflow\Assets written to: mathsymbols.model.assets

```

**Gambar 7.56** Illustrasi perhitungan stride 1 max pooling

17. Jelaskan kode program pada blok # In[17]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1
2
3 # In[18]: load the pre-trained model and predict the math
 symbol for an arbitrary image;
4 # the code below could be placed in a separate file
5 # mengimpport library keras model

```

```

In [81]: np.save('classes.npy', label_encoder.classes_)

```

**Gambar 7.57** Illustrasi perhitungan stride 1 max pooling

18. Jelaskan kode program pada blok # In[18]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 import keras.models
2 # Menginisiasi variabel model2 untuk meload model yang telah
 di simpan tadi
3 model2 = keras.models.load_model("mathsymbols.model")
4 # mencetak hasil model2
5 print(model2.summary())
6
7 # In[19]: restore the class name to integer encoder
8 # Menginisiasi variabel label encoder ke 2 dengan isian
 fungsi label encoder.
9 labelencoder2 = LabelEncoder()

```

```

In [19]: model2.load("mathsymbols.model")
Using TensorFlow from C:\Users\luthfi\AppData\Roaming\Python
\Python37\site-packages\tensorflow_core\python\ops
\resource_variable_ops.py:1178: calling BaseResourceVariable.__init__
() from tensorflow.python.ops.resource_variable_ops: ResourceVariable with constraint is
deprecated and will be removed in a future version.
Instructions for updating:
If using keras, pass _constraint arguments to layers.
If not, remove /fwd/ assets written to: mathsymbols.model.assets

```

**Gambar 7.58** Illustrasi perhitungan stride 1 max pooling

19. Jelaskan kode program pada blok # In[19]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # Penambahan method classes dengan data classess yang di
 eksport tadi
2 labelencoder2.classes_ = np.load('classes.npy')
3 # membuat fumgsi predict dengan path img
4 def predict(img_path):
5 # Menginisiasi variabel newimg dengan membuay immage
 menjadi array dan membuka data berdasarkan img path
6 newimg = keras.preprocessing.image.img_to_array(pil_image
 .open(img_path))
7 # membagi data yang terdapat pada variabel newimg
 sebanyak 255
8 newimg /= 255.0
9
10 # do the prediction
11 # Menginisiasi variabel predivtion dengan isian variabel
 model2 menggunakan fungsi predic dengan syarat variabel
 newimg dengan data reshape
12 prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
13
14 # figure out which output neuron had the highest score ,
 and reverse the one-hot encoding
15 # Menginisiasi variabel inverted denagan label encoder2
 dan menggunakan argmax untuk mencari skor luaran
 tertinggi
16 inverted = labelencoder2.inverse_transform([np.argmax(
 prediction)])
17 # mencetak prediksi gambar dan confidence dari gambar.
18 print("Prediction: %s, confidence: %.2f" % (inverted[0],
 np.max(prediction)))
19
20 # In[20]: grab an image (we'll just use a random training
 image for demonstration purposes)

```

```

21 # mencari prediksi menggunakan fungsi prediksi yang di buat
 tadi dari data di HASYv2/hasy-data/v2-00010.png
22 predict("HASYv2/hasy-data/v2-00010.png")

```

```

In [89]: label_encoder2 = LabelEncoder()
...# Pendekripsi method classes dengan data classes yang di
...# export
...# label_encoder2.classes_ = np.load('classes.npy')
...# ... def predict(img_path):
...# ... newimg = np.array(label_neveling dengan membaca image menjadi
...# array dan menulis data berdasarkan img path
...# ... newimg =
...# ... newimg = keras.preprocessing.image.img_to_array(pil_image.open(img_path))
...# ... keras.preprocessing.image.array(data yang terdapat pada variabel newimg sebanyak
...# 256
...# ... newimg /= 255.0
...# ...
...# ... # do the prediction
...# ... # Menginisiasi variabel prediksi dengan isian variabel
...# model2 menggunakan fungsi predict dengan syarat variabel newimg dengan
...# data yang diminta
...# ... prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
...# ...
...# ... # figure out which output neuron had the highest score, ans
...# reverse the one-hot encoding
...# ... # Menginisiasi variabel inverted dengan Label encoder2 di
...# mengambil argmax dari variabel prediction. Lalu run berturut-turut
...# ... inverted =
...# label_encoder2.inverse_transform(np.argmax(prediction))
...# ... print("Prediction: %s, confidence: %.2f % (inverted[0]),
...# np.max(prediction))")
...# ... print("Prediction: %s, confidence: %.2f % (inverted[0]),
...# np.max(prediction))")

```

**Gambar 7.59** Illustrasi perhitungan stride 1 max pooling

20. Jelaskan kode program pada blok # In[20]. Jelaskan arti dari setiap baris kode yang dibuat(harus beda dengan teman sekelas) dan hasil luarannya dari komputer sendiri.

```

1 # mencari prediksi menggunakan fungsi prediksi yang di buat
 tadi dari data di HASYv2/hasy-data/v2-00500.png
2 predict("HASYv2/hasy-data/v2-00500.png")
3 # mencari prediksi menggunakan fungsi prediksi yang di buat
 tadi dari data di HASYv2/hasy-data/v2-00700.png
4 predict("HASYv2/hasy-data/v2-00700.png")

```

```

In [86]: predict("D:/v2-00010.png")
...# ... # mencari prediksi menggunakan fungsi prediksi yang di
...# ... # ... predict("D:/v2-00500.png")
...# ... # ... # mencari prediksi menggunakan fungsi prediksi yang di
...# ... # ... # ... predict("D:/v2-00700.png")
...# ... # ... # ... predict("D:/v2-00700.png")

```

**Gambar 7.60** Illustrasi perhitungan stride 1 max pooling

### 7.3 1174040 - Hagan Rowlenstino A. S

#### 7.3.1 Teori

1. Kenapa file Text harus di Tokenizer

Tokenizer adalah langkah pertama yang diperlukan dalam tugas pemrosesan bahasa, seperti penghitungan kata, penguraian, pemeriksaan ejaan, pembuatan corpus, dan analisis statistik teks. Itu mengkonversi string teks Python menjadi aliran objek token, di mana setiap objek token adalah kata yang terpisah, tanda baca, nomor / jumlah, tanggal, email, URL / URI, dll. Ini juga mengelompokkan aliran token menjadi kalimat, dengan mempertimbangkan sudut kasus-kasus seperti singkatan dan tanggal di tengah kalimat. ilustrasi dapat dilihat pada gambar



**Gambar 7.61** Ilustrasi Tokenisasi

## 2. Konsep dasar K Fold Cross Validation pada dataset Komentar Youtube

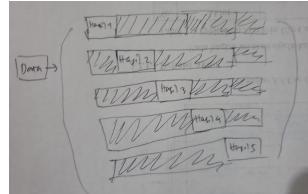
```

1 kfold = StratifiedKFold(n_splits=5)
2 splits = kfold.split(d, d['CLASS'])

```

**Listing 7.4** K Fold Cross Validation

pada Code pada Listing ?? terdapat penggunaan K Fold dengan metode yang akan membagi data dalam 5 hasil pengulangan dari pembagian data yang akan dilakukan pada Attribute CLASS. ilustrasinya dapat dilihat pada Gambar



**Gambar 7.62** Ilustrasi K Fold Cross Validation pada dataset Komentar Youtube

## 3. Jelaskan apa maksudnya kode program *for train, test in splits*

Membagi susunan atau matriks menjadi rangkaian acak, dimana data yang dimasukan akan dibagi dengan porsi yang sesuai dengan Train dan Test datanya. untuk lebih jelas tentang bagaimana penggunaannya dapat dilihat pada Gambar

| Examples                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> &gt;&gt;&gt; import numpy as np &gt;&gt;&gt; from sklearn.model_selection import train_test_split &gt;&gt;&gt; X, y = np.arange(10).reshape(5, 2), range(5) &gt;&gt;&gt; array([[0, 1],    [2, 3],    [4, 5],    [6, 7],    [8, 9]]) &gt;&gt;&gt; list(y) [0, 1, 2, 3, 4]  &gt;&gt;&gt; X_train, X_test, y_train, y_test = train_test_split( ...     X, y, test_size=0.33, random_state=42) ... &gt;&gt;&gt; X_train array([[4, 5],        [0, 1],        [2, 3]]) &gt;&gt;&gt; y_train [2, 0, 3] &gt;&gt;&gt; X_test array([[1, 3],        [8, 9]]) &gt;&gt;&gt; y_test [1, 4]  &gt;&gt;&gt; train_test_split(y, shuffle=False) [[0, 1, 2], [3, 4]]   </pre> |

**Gambar 7.63** Ilustrasi penggunaan Train dan Test Split

4. Maksud Code program  $train\_content = d['CONTENT'].iloc[train\_idx]$  dan  $test\_content = d['CONTENT'].iloc[train\_idx]$

pada code tersebut dimaksudkan untuk membaca data isian yang terdapat pada kolom yang bernama attributnya CONTENT sebagai data TRAIN dan data TEST, untuk ilustrasinya dapat dilihat pada Gambar



```

Examples
>>> import numpy as np
>>> from sklearn.model_selection import train_test_split
>>> y = np.arange(10).reshape(5, 2), range(5)
>>> X
array([[0, 1],
 [2, 3],
 [4, 5],
 [6, 7],
 [8, 9]])
>>> list(y)
[0, 1, 2, 3, 4]
>>> X_train, X_test, y_train, y_test = train_test_split(
... X, y, test_size=0.33, random_state=42)
>>> X_train
array([[4, 5],
 [0, 1],
 [2, 3]])
>>> y_train
[2, 0, 3]
>>> X_test
array([[1, 2],
 [6, 7],
 [8, 9]])
>>> y_test
[1, 4]
>>> train_test_split(y, shuffle=False)
[[0, 1, 2], [3, 4]]

```

**Gambar 7.64** Ilustrasi penggunaan Train dan Test Split

5. maksud dari fungsi  $tokenizer = Tokenizer(num\_words=2000)$  dan  $tokenizer.fit\_on\_texts(train\_content)$

pada code tersebut dimana penggunaan Tokenizer berguna untuk membaca data teks yang berformat kalimat untuk disusun sebanyak 2000 kata, fungsi fit\_on\_texts(train\_content) untuk membaca data Teks Tokenizer tadi untuk dimasukan kedalam kolom CONTENT pada dataset. untuk contoh ilustrasi data yang digunakan dapat dilihat pada Gambar

| content_id | actual   | text     | CONTENT  | category |
|------------|----------|----------|----------|----------|
| 1          | positive | positive | positive | -1       |
| 2          | positive | positive | positive | -1       |
| 3          | positive | positive | positive | -1       |
| 4          | positive | positive | positive | -1       |
| 5          | positive | positive | positive | -1       |
| 6          | positive | positive | positive | -1       |
| 7          | positive | positive | positive | -1       |
| 8          | positive | positive | positive | -1       |
| 9          | positive | positive | positive | -1       |
| 10         | positive | positive | positive | -1       |

**Gambar 7.65** data Content yang dimaksud

6. maksud dari fungsi  $d\_train\_inputs = tokenizer.texts\_to\_matrix(train\_content, mode='tfidf')$  dan  $d\_test\_inputs = tokenizer.texts\_to\_matrix(test\_content, mode='tfidf')$

pada code yang pertama terdapat fungsi yang digunakan untuk mengolah data variable d\_train.inputs dan d\_test.inputs dengan menggunakan tokernizer dengan fungsi yang merubah data teks menjadi data matrix dengan pemrosesan datanya sesuai dengan data Variable train\_content dan test\_content dengan metodenya adalah TF-IDF. untuk contoh ilustrasi dari penjelasan tersebut dapat dilihat pada Gambar

```
In [34]: from keras.preprocessing.text import Tokenizer
.....
.... # define 2 documents
.... docs = ["Well done!",
.... "Good work"]
.... t = Tokenizer()
.... # fit the tokenizer
.... t.fit_on_texts(docs)
.... encoded_docs = t.texts_to_matrix(docs, mode='tfidf')
.... print(encoded_docs)
[[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]]
```

**Gambar 7.66** Contoh code tentang penggunaan texts\_to\_matrix

7. maksud dari fungsi  $d\_train\_inputs = d\_train\_inputs / np.amax(np.absolute(d\_train\_inputs))$  dan  $d\_test\_inputs = d\_test\_inputs / np.amax(np.absolute(d\_test\_inputs))$

pada code tersebut berfungsi sebagai pengelolaan data Variable d\_train\_inputs dan d\_test\_inputs dengan menggunakan perintah metode yang terdapat pada Library Numpy metode yang digunakan adalah AMAX yang berguna untuk mengambil nilai maksimum yang terdapat pada data arraynya atau mencari nilai maksimum dari panjang sumbu yang terproses dan ABSOLUTE yang berguna untuk untuk mengkalkulasi nilai pasti dari setiap data elemennya. untuk contoh dari penggunaan AMAX dan ABSOLUTE dapat dilihat pada Gambar

berikut ini adalah contoh Code yang digunakan untuk menjelaskan penggunaan AMAX :

```
In [24]: import numpy as np
.....
.... arr1 = [1, -3, 15, -466]
.... print ("Absolute Value of arr1 : \n",
.... np.absolute(arr1))
.....
.... arr2 = [23 , -56]
.... print ("\nAbsolute Value of arr2 : \n",
.... np.absolute(arr2))
Absolute Value of arr1 :
[1 3 15 466]

Absolute Value of arr2 :
[23 56]
```

**Gambar 7.67** penggunaan Numpy AMAX

berikut ini adalah contoh Code yang digunakan untuk menjelaskan penggunaan ABSOLUTE :

```
In [29]: import numpy as geek
.....
.... arr = geek.arange(10)
.... print("arr : ", arr)
.... print("Max of arr : ", geek.max(arr))
.... print("Min of arr : ", geek.min(arr))
.... print("Mean of arr : ", geek.mean(arr))
.... print("Median of arr : ", geek.median(arr))
.... print("Variance of arr, axis = None : ", geek.var(arr))
.... print("Variance of arr, axis = 0 : ", geek.var(arr, axis = 0))
.... print("Variance of arr, axis = 1 : ", geek.var(arr, axis = 1))
arr : [0 1 2 3 4 5 6 7 8 9]
Max of arr :
[9 6 7 8 9]

Max of arr, axis = None : 9
Max of arr, axis = 0 : [9 6 7 8 9]
Max of arr, axis = 1 : [4 9]
```

**Gambar 7.68** penggunaan Numpy ABSOLUTE

8. maksud fungsi dari `d_train_outputs = np_utils.to_categorical(d['CLASS'].iloc[train_idx])` dan `d_test_outputs = np_utils.to_categorical(d['CLASS'].iloc[test_idx])` dalam kode program

pada code tersebut menjelaskan penggunaan Library Numpy yang akan mengelola data d\_train\_outputs dan d\_test\_outputs dengan menggunakan perintah atau metode TO\_CATEGORICAL dimana data yang diambil adalah data CLASS dengan index data yang dituju adalah train\_idx dan test\_idx, dimana to\_categorical berguna untuk mengkonversikan data kelas Vektor(Integer) menjadi data kelas Binary Matrix. contoh penggunaan to categorical dapat dilihat pada Gambar

```

> labels
array([0, 2, 1, 2, 0])
> to_categorical(labels)
array([[1., 0., 0.],
 [0., 0., 1.],
 [0., 1., 0.],
 [0., 0., 1.],
 [1., 0., 0.]], dtype=float32)
> mydict = [{"a": 1, "b": 2, "c": 3, "d": 4},
 {"a": 100, "b": 200, "c": 300, "d": 400},
 {"a": 1000, "b": 2000, "c": 3000, "d": 4000}]
> df.iloc[[0, 1]]
 a b c d
0 1 2 3 4
1 100 200 300 400

```

**Gambar 7.69** Contoh Code dan Hasil dari penggunaan to\_categorical dan iloc

9. Penjelasan pada fungsi code dengan Ilustrasi Neural Networknya

```

1 model = Sequential()
2 model.add(Dense(512, input_shape=(2000,)))
3 model.add(Activation('relu'))
4 model.add(Dropout(0.5))
5 model.add(Dense(2))
6 model.add(Activation('softmax'))

```

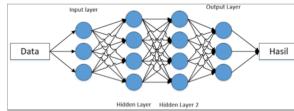
**Listing 7.5** Membuat model Neural Network

pada code yang terdapat pada Listing ??, dijelaskan bahwa dibuatkannya Variable 'model' untuk menampung model Sequential() atau metode yang berguna untuk menumpuk data Linear. berikut ini adalah penjelasan dari masing - masing fungsi yang terdapat pada Code tersebut :

- Dense (512, input\_shape = (2000,)) dan (2)
- dimana data yang diolah akan berjumlah 512 dengan shape yang diolah adalah 2000 begitu juga dengan yang 2
- Activation ('relu') dan ('softmax')
- Activation adalah untuk melakukan penggunaan fungsi matematika, relu yang berarti fungsi dari metode 'Rectified Linear Unit' dan softmax yang berguna untuk menghasilkan data Vector yang merepresentasikan nilai distibusi probabilitas dari daftar nl=ilai output yang memiliki potensi.
- Dropout(0.5)

melakukan set sebanyak 50 persen pada output atau nilai hasil yang dikeluarkan dari pengelolaan data pada visible layer dan hidden layer.

berikut ini adalah contoh dari struktur Neural Network yang terjadi :



**Gambar 7.70** Contoh dari struktur Neural Network yang terjadi

10. Maksud dari fungsi pada Code berikut :

```

1 model.compile(loss='categorical_crossentropy', optimizer='
2 adamax',
 metrics=['accuracy'])

```

**Listing 7.6** Compile model

pada code di Listing ?? menjelaskan tentang proses yang berlangsung untuk menampilkan nilai LOSS dan ACCURACY yang dihasilkan dari Compile pada program diatas.

11. Jelaskan apa itu Deep Learning

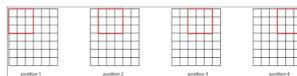
Deep Learning adalah rangkaian metode untuk bisa melatih jaringan saraf buatan multi lapisan atau mempunyai banyak lapisan. Metode ini sangat efektif dan lebih mudah dalam mengidentifikasi pola dari data yang dimasukkan. Deep Learning juga dapat meningkatkan bagian AI, mulai dari memproses bahasa alami yang kita katakana sampai dengan mengambil gambar. Oleh karena itu Deep Learning merupakan otak yang lebih baik dan unggul untuk meningkatkan cara kerja pada sistem komputer

12. Jelaskan Apa itu Deep Neural Network dan apa perbedaanya dengan Deep Learning

Deep Neural Network adalah Neural Network dengan tingkat kompleksitas tertentu, Neural Network dengan lebih dari dua lapisan. Neural Network dalam menggunakan pemodelan matematika yang canggih untuk memproses data dengan cara yang kompleks. perbedaan antara Deep Learning dan DNN adalah DNN merupakan sebuah Algoritma yang terdapat dalam Deep Learning itu sendiri sedangkan Deep Learning adalah sebuah pengguna dari algoritma pada DNN.

13. Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride (NPM mod3+1) x (NPM mod3+1) yang terdapat max pooling

Konvolusi terdapat pada operasi pengolahan citra yang mengalikan sebuah citra dengan sebuah mask atau kernel. Stride adalah parameter yang berfungsi untuk menentukan pergeseran pada filter data pixel yang terjadi. Untuk contoh penggunaannya dapat dilihat pada Gambar



**Gambar 7.71** Penggunaan Konvolusi dengan Stridennya adalah  $3 \times 3$

### 7.3.2 Pemrograman

#### 1. No. 1 Kode Program Blok # In 1

```
1 # In [1]: import lib
2 import csv
3 from PIL import Image as pil_image
4 import keras.preprocessing.image
```

Keterangannya sebagai berikut :

- Melakukan Import Library CSV
- Melakukan Import Library PIL (Python Imaging Library) yang berguna untuk melakukan pemrosesan data yang bersifat Image. dengan fungsi yang diimport adalah pil\_image
- Melakukan Import data preprocessing dari Library Keras yang akan memproses data Image.
- hasilnya dapat dilihat pada Gambar

```
In [1]: import csv
...: from PIL import Image as pil_image
...: import keras.preprocessing.image
Using TensorFlow backend.
```

**Gambar 7.72** kode program pada blok In[1].

#### 2. No. 2 Kode Program Blok # In 2

```
1 # In [2]: load all images (as numpy arrays) and save their
 classes
2 imgs = []
3 classes = []
4 with open('HASYv2/hasy-data-labels.csv') as csvfile:
5 csvreader = csv.reader(csvfile)
6 i = 0
7 for row in csvreader:
8 if i > 0:
9 img = keras.preprocessing.image.img_to_array(
 pil_image.open("HASYv2/" + row[0]))
```

```

10 # neuron activation functions behave best when
11 input values are between 0.0 and 1.0 (or -1.0 and 1.0),
12 # so we rescale each pixel value to be in the
13 range 0.0 to 1.0 instead of 0-255
14 img /= 255.0
15 imgs.append((row[0], row[2], img))
 classes.append(row[2])
 i += 1

```

Keterangannya sebagai berikut :

- Membuat Variable 'imgs' dan 'classes' yang memiliki nilai array kosong agar dapat diisi oleh data nantinya.
- Membuka file csv dari Folder HSYv2 dengan nama file hasy-data-labels.csv yang dialiaskan sebagai Variable csvfile.
- Variabel csvreader akan menggunakan fungsi reader pada library csv untuk membaca file pada variable csvfile.
- Membuat Variable i yang bernilai NOL.
- Membuat pemrosesan data yang akan membaca datan dari csvreader dengan membuat Variable 'row', dimasukan dengan fungsi IF jika i lebih besar dari NOL.
- membuat Variable 'img' yang berisi data pemrosesan Library keras dengan fungsi image.im\_to\_array yang akan mmebuat data yang dibuka menjadi format array pada Folder HASYv2 yang dimulai dari NOL.
- Hasil dari variabel img akan dibagi dengan 255.0
- .append akan membuat list array baru untuk baris 0 baris 2 pada img.
- Menyimpan setiap class nya pada baris 2
- Penambahan i sebanyak 1.
- hasilnya dapat dilihat pada Gambar

```

In [4]:
In [4]: imgs = []
In [4]: classes = []
In [4]: with open("hsy-data-labels.csv") as csvfile:
... csvreader = csv.reader(csvfile)
... i = 0
... for row in csvreader:
... if i > 0:
... img = keras.preprocessing.image.img_to_array(pil_image.open("HSYv2/" +
... row[0])) # neuron activation functions behave best when input values are between
... # 0.0 and 1.0 (or -1.0 and 1.0), so we rescale each pixel value to be in the
... # range 0.0 to 1.0 instead of 0-255
... img /= 255.0
... imgs.append((row[0], row[2], img))
... classes.append(row[2])
... i += 1

```

**Gambar 7.73** kode program pada blok In[2].

### 3. No. 3 Kode Program Blok # In 3

```

1 # In [3]: shuffle the data, split into 80% train , 20% test
2 import random
3 random.shuffle(imgs)
4 split_idx = int(0.8*len(imgs))
5 train = imgs[:split_idx]
6 test = imgs[split_idx:]

```

Keterangannya sebagai berikut :

- Melakukan Import Library random.
- Membuat parameter dari penggunaan library random yang akan mengacak data dengan fungsi 'shuffle' pada data Variable 'imgs'.
- Membagi index data dari imgs dengan cara mengalikan 80% dengan jumlah data dari imgs.
- Untuk data train mengambil hasil dari perhitungan sebelumnya.
- Untuk data test mengambil sisa dari jumlah yang telah dijadikan data train.
- hasil dapat dilihat pada Gambar'

```
In [5]: import random
...: random.shuffle(imgs)
...: split_idx = int(0.8*len(imgs))
...: train = imgs[:split_idx]
...: test = imgs[split_idx:]
```

**Gambar 7.74** kode program pada blok In[3].

#### 4. No. 4 Kode Program Blok # In 4

```
1 # In [4]:
2 import numpy as np
3
4 train_input = np.asarray(list(map(lambda row: row[2], train)))
5 test_input = np.asarray(list(map(lambda row: row[2], test)))
6
7 train_output = np.asarray(list(map(lambda row: row[1], train)))
8 test_output = np.asarray(list(map(lambda row: row[1], test)))
```

Keterangannya sebagai berikut :

- Melakukan Import dari Library Numpy dialiaskan menjadi np.
- Variabel train\_input mengubah input menjadi sebuah array yang diambil dari baris 2, dari data train.
- Variabel test\_input mengubah input menjadi sebuah array yang diambil dari baris 2, dari data test.
- Variabel train\_output mengubah input menjadi sebuah array yang diambil dari baris 1, dari data train.
- Variabel train\_output mengubah input menjadi sebuah array yang diambil dari baris 1, dari data test.
- hasil dapat dilihat pada Gambar

```
In [4]: import numpy as np
.....
.... train_input = np.asarray(list(map(lambda row: row[2], train)))
.... test_input = np.asarray(list(map(lambda row: row[2], test)))
.....
.... train_output = np.asarray(list(map(lambda row: row[1], train)))
.... test_output = np.asarray(list(map(lambda row: row[1], test)))
```

**Gambar 7.75** kode program pada blok In[4].

## 5. No. 5 Kode Program Blok # In 5

```
1 # In [5]: import encoder and one hot
2 from sklearn.preprocessing import LabelEncoder
3 from sklearn.preprocessing import OneHotEncoder
```

Keterangannya sebagai berikut :

- Melakukan Import fungsi LabelEncoder dari Library sklearn
- Melakukan Import fungsi OneHotEncoder dari Library sklearn
- hasil dapat dilihat pada Gambar

```
In [7]: from sklearn.preprocessing import LabelEncoder
.... from sklearn.preprocessing import OneHotEncoder
```

**Gambar 7.76** kode program pada blok In[5].

## 6. No. 6 Kode Program Blok # In 6

```
1 # In [6]: convert class names into one-hot encoding
2 # first, convert class names into integers
3 label_encoder = LabelEncoder()
4 integer_encoded = label_encoder.fit_transform(classes)
```

Keterangannya sebagai berikut :

- Membuat Variabel label\_encoder yang akan memanggil fungsi LabelEncoder tadi.
- Membuat Variabel integer\_encoded yang akan menggunakan labelencoder untuk melakukan fit pada classes agar berubah datanya menjadi integer.
- hasil dapat dilihat pada Gambar

```
In [8]: label_encoder = LabelEncoder()
.... integer_encoded = label_encoder.fit_transform(classes)
```

**Gambar 7.77** kode program pada blok In[6].

## 7. No. 7 Kode Program Blok # In 7

```
1 # In [7]: then convert integers into one-hot encoding
2 onehot_encoder = OneHotEncoder(sparse=False)
3 integer_encoded = integer_encoded.reshape(len(integer_encoded),
 1)
4 onehot_encoder.fit(integer_encoded)
```

Keterangannya sebagai berikut :

- Membuat Variabel onehot\_encoder akan memanggil fungsi OneHotEncoder dimana tidak berisikan matriks sparse.
- Pada variabel integer\_encoded akan diubah bentuknya dimana setiap nilai integer akan direpresentasikan sebagai vektor binari dengan nilai 0 kecuali index dari integer tersebut ditandai dengan 1.
- Melakukan fit untuk one hot encoder kedalam integer\_encoder.
- hasil dapat dilihat pada Gambar

```
In [8]: onehot_encoder = OneHotEncoder(sparse=False)
... integer_encoder = IntegerEncoder()
... integer_encoded = integer_encoder.transform(integer_encoded)
... onehot_encoder.fit(integer_encoded)
C:\Users\Herman\PycharmProjects\Klasifikasi\prpprocessead_encoder.py:13:
FutureWarning: The handling of integer data will change in version 0.22.
Currently, the categories are determined based on the range [0, max(values)].
This behaviour will change in version 0.22. Categories will be based on the
values themselves. If you want the future behaviour and silence this warning, you can specify
categories='auto'.
In case you used a LabelEncoder before this OneHotEncoder to convert the
integer data, you should now use the OneHotEncoder directly.
warnings.warn("The handling of integer data will change in version 0.22.
Currently, the categories are determined based on the range [0, max(values)].
This behaviour will change in version 0.22. Categories will be based on the
values themselves.", FutureWarning)
labelencoder = LabelEncoder()
onehotencoder = OneHotEncoder(categories='auto',
 drop='first', handle_unknown='error',
 sparse=False)
```

Gambar 7.78 kode program pada blok In[7].

## 8. No. 8 Kode Program Blok # In 8

```
1 # In[8]: convert train and test output to one-hot
2 train_output_int = label_encoder.transform(train_output)
3 train_output = onehot_encoder.transform(train_output_int.
 reshape(len(train_output_int), 1))
4 test_output_int = label_encoder.transform(test_output)
5 test_output = onehot_encoder.transform(test_output_int.
 reshape(len(test_output_int), 1))
6
7 num_classes = len(label_encoder.classes_)
8 print("Number of classes: %d" % num_classes)
```

Keterangannya sebagai berikut :

- Membuat Variabel train\_output\_int akan mengubah data dari train\_output menjadi LabelEncoder
- Membuat Variable train\_output dimana setelah diubah labelnya menjadi integer dilakukan onehot\_encoding diambil dari train\_output\_int dan menggunakan .reshape untuk memberikan bentuk baru ke array tanpa mengubah datanya dengan keterangan jika index dari integer tersebut ditandai dengan 1 dan sisanya yang bukan nol.
- Membuat Variabel test\_output\_int akan mengubah data dari test\_output menjadi LabelEncoder
- Membuat Variable test\_output dimana setelah diubah labelnya menjadi integer dilakukan one hot encoding diambil dari test\_output\_int dan menggunakan .reshape untuk memberikan bentuk baru ke array tanpa mengubah datanya dengan keterangan jika index dari integer tersebut ditandai dengan 1 dan sisanya yang bukan nol.

- Membuat Variabel num\_classes akan menampilkan jumlah data dari classes yang telah dilakukan label encoder
- Melakukan print dengan Format tulisannya adalah ”Number of classes : %d” dimana mengembalikan nilai integer dari num\_classes.
- hasil dapat dilihat pada Gambar

```
In [8]: train_output_int = label_encoder.transform(train_output)
...::: test_output_int = label_encoder.transform(test_output)
...::: train_output_int = np.reshape(train_output_int,(train_output_int.shape[0], 1))
...::: test_output_int = np.reshape(test_output_int,(test_output_int.shape[0], 1))
...::: num_classes = len(label_encoder.classes_)
...::: print("Number of classes: ", num_classes)
Number of classes: 8
```

**Gambar 7.79** kode program pada blok In[8].

### 9. No. 9 Kode Program Blok # In 9

```
1 # In [9]: import sequential
2 from keras.models import Sequential
3 from keras.layers import Dense, Dropout, Flatten
4 from keras.layers import Conv2D, MaxPooling2D
```

Keterangannya sebagai berikut :

- Melakukan Import Sequential dari model pada Library Keras.
- Melakukan Import Dense, Dropout, Flatten dari modul Layers pada Library Keras.
- Melakukan Import Conv2D, MaxPooling2D dari modul Layers pada Library Keras.
- hasil dapat dilihat pada Gambar

```
In [10]: from keras.models import Sequential
...::: from keras.layers import Dense, Dropout, Flatten
...::: from keras.layers import Conv2D, MaxPooling2D
```

**Gambar 7.80** kode program pada blok In[9].

### 10. No. 10 Kode Program Blok # In 10

```
1 # In [10]: desain jaringan
2 model = Sequential()
3 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
4 input_shape=np.shape(train_input[0])))
5 model.add(MaxPooling2D(pool_size=(2, 2)))
6 model.add(Conv2D(32, (3, 3), activation='relu'))
7 model.add(MaxPooling2D(pool_size=(2, 2)))
8 model.add(Flatten())
9 model.add(Dense(1024, activation='tanh'))
10 model.add(Dropout(0.5))
11 model.add(Dense(num_classes, activation='softmax'))
12
13 model.compile(loss='categorical_crossentropy', optimizer='adam',
```

```
14 metrics=['accuracy'])
15
16 print(model.summary())
```

Keterangannya sebagai berikut :

- Melakukan pemodelan Sequential.
  - Menambahkan Konvolusi 2D dengan 32 filter konvolusi masing-masing berukuran  $3 \times 3$  dengan algoritam activation relu dengan data dari train\_input mulai dari baris nol.
  - Menambahkan Max Pooling dengan matriks  $2 \times 2$ .
  - Dilakukan lagi penambahan Konvolusi 2D dengan 32 filter konvolusi masing-masing berukuran  $3 \times 3$  dengan algoritam activation relu.
  - Menambahkan lagi Max Pooling dengan matriks  $2 \times 2$ .
  - Mendefinisikan inputan dengan 1024 neuron dan menggunakan algoritma tanh untuk activationnya.
  - Dropout terdiri dari pengaturan secara acak tingkat pecahan unit input ke 0 pada setiap pembaruan selama waktu pelatihan, yang membantu mencegah overfitting sebesar 50% .
  - Untuk output layer menggunakan data dari variabel num\_classes dengan fungsi activationnya softmax.
  - Mengonfigurasi proses pembelajaran, yang dilakukan melalui metode compile, sebelum melatih suatu model.
  - Menampilkan atau mencetak representasi ringkasan model yang telah dibuat.

**Gambar 7.81** kode program pada blok In[10].

11. No. 11 Kode Program Blok # In 11

```
1 # In [11]: import sequential
2 import keras.callbacks
3 tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/
 mnist-style')
```

Keterangannya sebagai berikut :

- Impor Modul Callbacks dari Librari Keras.

- Variabel callback mendefinisikan Callback ini untuk menulis log untuk TensorBoard, yang memungkinkan Anda untuk memvisualisasikan grafik dinamis dari pelatihan dan metrik pengujian Anda, serta histogram aktivasi untuk berbagai lapisan dalam model Anda.

```
In [11]: import keras.callbacks
...: tensorboard = keras.callbacks.TensorBoard(log_dir='./logs/',
...: histogram_freq=1)
```

**Gambar 7.82** kode program pada blok In[11].

## 12. No. 12 Kode Program Blok # In 12

```
1 # In [12]: 5menit kali 10 epoch = 50 menit
2 model.fit(train_input, train_output,
3 batch_size=32,
4 epochs=10,
5 verbose=2,
6 validation_split=0.2,
7 callbacks=[tensorboard])
8
9 score = model.evaluate(test_input, test_output, verbose=2)
10 print('Test loss:', score[0])
11 print('Test accuracy:', score[1])
```

Keterangannya sebagai berikut :

- Melakukan fit model dengan 32 ukuran subset dari sampel pelatihan Anda
- Epoch sebanyak 10 kali
- Vebrose=2 maksudnya menampilkan nomor dari epoch yang sedang berjalan atau yang sudah dijalankan.
- Validasi plit sebanayk 20% sebagai fraksi data pelatihan untuk digunakan sebagai data validasi.
- Menggunakan TensorBoard sebagai callback untuk diterapkan selama pelatihan dan validasi.
- Variabel score mengembalikan nilai evaluate untuk menampilkan data lost dan data accuracy dari test
- Menampilkan data loss dengan menghitung jumlah kemunculan nol .
- Menampilkan data accuracy dengan menghitung jumlah kemunculan 1.

**Gambar 7.83** kode program pada blok In[12].

13. No. 13 Kode Program Blok # In 13

```

1 # In[13]:try various model configurations and parameters to
2 # find the best
3
4 import time
5
6 results = []
7 for conv2d_count in [1, 2]:
8 for dense_size in [128, 256, 512, 1024, 2048]:
9 for dropout in [0.0, 0.25, 0.50, 0.75]:
10 model = Sequential()
11 for i in range(conv2d_count):
12 if i == 0:
13 model.add(Conv2D(32, kernel_size=(3, 3),
14 activation='relu', input_shape=np.shape(train_input[0])))
15 else:
16 model.add(Conv2D(32, kernel_size=(3, 3),
17 activation='relu'))
18 model.add(MaxPooling2D(pool_size=(2, 2)))
19 model.add(Flatten())
20 model.add(Dense(dense_size, activation='tanh'))
21 if dropout > 0.0:
22 model.add(Dropout(dropout))
23 model.add(Dense(num_classes, activation='softmax',
24))
25
26 model.compile(loss='categorical_crossentropy',
27 optimizer='adam', metrics=['accuracy'])
28
29 log_dir = './logs/conv2d_%d-dense_%d-dropout_%d-%f'
30 % (conv2d_count, dense_size, dropout)
31 tensorboard = keras.callbacks.TensorBoard(log_dir,
32 log_dir)
33
34 start = time.time()
35 model.fit(train_input, train_output, batch_size
36 =32, epochs=10,
37 verbose=0, validation_split=0.2,
38 callbacks=[tensorboard])
39 score = model.evaluate(test_input, test_output,
40 verbose=2)
41 end = time.time()
42 elapsed = end - start
43 print("Conv2D count: %d, Dense size: %d, Dropout:
44 %.2f - Loss: %.2f, Accuracy: %.2f, Time: %d sec" %
45 (

```

```
33 conv2d_count , dense_size , dropout , score [0] , score [1] ,
 elapsed))
 results . append ((conv2d_count , dense_size , dropout
 , score [0] , score [1] , elapsed))
```

Keterangannya sebagai berikut :

- impor modul time dari python anaconda
- Variabel result berisikan array kosong.
- Menggunakan convolution 2D yang dimana akan memiliki 1 atau 2 layer.
- Mendefinisikan dense\_size dengan ukuran 128, 256, 512, 1024, 2048
- Mendefinisikan drop\_out dengan 0, 25%, 50%, dan 75%
- Melakukan pemodelan Sequential
- Jika ini adalah layer pertama, kita perlu memasukkan bentuk input.
- Kalau tidak kita hanya akan menambahkan layer.
- Kemudian, setelah menambahkan layer konvolusi, kita akan melakukan hal yang sama dengan max pooling.
- Lalu, kita akan meratakan atau flatten dan menambahkandense size ukuran apa pun yang berasal dari dense\_size. Dimana akan selalu menggunakan algoritma tanh
- Jika dropout digunakan, kita akan menambahkan layer dropout. Menyebut dropout ini berarti, katakanlah 50%, bahwa setiap kali ia memperbarui bobot setelah setiap batch, ada peluang 50% untuk setiap bobot yang tidak akan diperbarui
- menempatkan ini di antara dua lapisan padat untuk dihidupkan dari melindunginya dari overfitting.
- Lapisan terakhir akan selalu menjadi jumlah kelas karena itu harus, dan menggunakan softmax. Itu dikompilasi dengan cara yang sama.
- Atur direktori log yang berbeda untuk TensorBoard sehingga dapat membedakan konfigurasi yang berbeda.
- Variabel start akan memanggil modul time atau waktu
- Melakukan fit atau compile
- MELakukan scoring dengan .evaluate yang akan menampilkan data loss dan accuracy dari model
- end merupakan variabel untuk melihat waktu akhir pada saat pemodelan berhasil dilakukan.
- Menampilkan hasil dari run skrip diatas

```

Conv2D (input=1, output=16, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=16, output=32, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=32, output=64, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=64, output=128, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=128, output=256, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=256, output=512, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=512, output=1024, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=1024, output=2048, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
MaxPooling2D (pool_size=(2, 2))
Conv2D (input=2048, output=1024, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
MaxPooling2D (pool_size=(2, 2))
Conv2D (input=1024, output=512, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=512, output=256, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=256, output=128, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=128, output=64, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=64, output=32, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=32, output=16, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=16, output=8, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=8, output=4, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=4, output=2, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Conv2D (input=2, output=1, kernel_size=(3, 3), activation='relu', padding='same', data_format='channels_last')
Dropout (0.5)
Dense (128, activation='tanh')
Dropout (0.5)
Dense (num_classes, activation='softmax')
compile (loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

```

Gambar 7.84 kode program pada blok In[13].

#### 14. No. 14 Kode Program Blok # In 14

```

1 # In[14]: rebuild/retrain a model with the best parameters (
2 from the search) and use all data
3 model = Sequential()
4 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
5 input_shape=np.shape(train_input[0])))
6 model.add(MaxPooling2D(pool_size=(2, 2)))
7 model.add(Conv2D(32, (3, 3), activation='relu'))
8 model.add(MaxPooling2D(pool_size=(2, 2)))
9 model.add(Flatten())
10 model.add(Dense(128, activation='tanh'))
11 model.add(Dropout(0.5))
12 model.compile(loss='categorical_crossentropy', optimizer='adam',
13 metrics=['accuracy'])
14 print(model.summary())

```

Keterangannya sebagai berikut :

- Melakukan pemodelan Sequential
- Untuk layer pertama, Menambahkan Convolutio 2D dengan dmensi 32, dan ukuran matriks 3x3 dengan function aktivasi yang digunakan yaitu relu dan menampilkan input\_shape
- Dilakukan Max Pooling 2D dengan ukuran matriks 2x2
- Untuk layer kedua, melakukan Convolusi lagi dengan kriteria yang sama tanpa menambahkan input, ini dilakukan untuk mendapatkan data yang terbaik
- Flatten digunakan ntuk meratakan inputan
- Menambahkan dense input sebanyak 128 neuron dengan menggunakan function aktivasi tanh.
- Dropout sebanyak 50% untuk menghindari overfitting
- Menambahkan dense pada model untuk output dimana layer ini akan menjadi jumlah dari class yang ada.
- Mengcompile model yang didefinisikan diatas
- Menampilkan ringkasan dari pemodelan yang dilakukan

```
[In [15]: model = Sequential()
...: .add(Conv2D(16, kernel_size=(3, 3), activation='relu',
...: .add(MaxPooling2D(pool_size=(2, 2)))
...: .add(Flatten())
...: .add(Dense(128, activation='tanh'))
...: .add(Dense(10, activation='softmax'))
...: .compile(optimizer='adam', metrics=['accuracy'])
...: .print_summary()

Layer (type) Output Shape Param #
conv2d_31 (Conv2D) (None, 15, 15, 32) 896
max_pooling2d_31 (MaxPooling2D) (None, 7, 7, 32) 0
conv2d_32 (Conv2D) (None, 15, 15, 32) 1024
max_pooling2d_32 (MaxPooling2D) (None, 3, 3, 32) 0
dense_33 (Dense) (None, 128) 4096
dense_34 (Dense) (None, 10) 147504
dropout_35 (Dropout) (None, 128) 0
dense_36 (Dense) (None, 100) 47600 *****
total params: 285,229
trainable params: 285,229
non-trainable params: 0
None
```

**Gambar 7.85** kode program pada blok In[14].

## 15. No. 15 Kode Program Blok # In 15

```
1 # In [15]: join train and test data so we train the network on
 all data we have available to us
2 model.fit(np.concatenate((train_input, test_input)),
3 np.concatenate((train_output, test_output)),
4 batch_size=32, epochs=10, verbose=2)
```

Keterangannya sebagai berikut :

- Melakukan fit dengan join data train dan test agar dapat dilakukan pelatihan untuk jaringan pada semua data yang dimiliki.

```
In [16]: model.fit(np.concatenate((train_input, test_input)),
...: np.concatenate((train_output, test_output)),
...: batch_size=32, epochs=10, verbose=2)
Epoch 0/10
- 387s - loss: 1.7827 - acc: 0.5058
Epoch 1/10
- 387s - loss: 1.0765 - acc: 0.7067
Epoch 2/10
- 387s - loss: 0.9664 - acc: 0.7301
Epoch 3/10
- 365s - loss: 0.9603 - acc: 0.7426
Epoch 4/10
- 367s - loss: 0.8671 - acc: 0.7508
Epoch 5/10
- 367s - loss: 0.8394 - acc: 0.7567
Epoch 6/10
- 358s - loss: 0.8163 - acc: 0.7617
Epoch 7/10
- 367s - loss: 0.7972 - acc: 0.7659
Epoch 8/10
- 370s - loss: 0.7819 - acc: 0.7692
Epoch 9/10
- 473s - loss: 0.7680 - acc: 0.7725
Out[16]: <keras.callbacks.History at 0x22e989c5438>
```

**Gambar 7.86** kode program pada blok In[15].

## 16. No. 16 Kode Program Blok # In 16

```
1 # In [16]: save the trained model
2 model.save("mathsymbol.model")
```

Keterangannya sebagai berikut :

- Menyimpan atau save model yang telah di latih dengan nama mathsymbol.model

|                      |                   |                    |          |
|----------------------|-------------------|--------------------|----------|
| mathsymbol.model     | 4/1/2019 8:25 PM  | MODEL File         | 2,443 KB |
| mathsymbol.py        | 4/1/2019 4:55 PM  | Python Source File | 0 KB     |
| README.txt           | 1/20/2019 6:44 PM | Text Document      | 3 KB     |
| SpanDetectorNumpy    | 4/9/2019 8:41 PM  | IPYNB File         | 22 KB    |
| SpanDetectorNumpy.py | 4/9/2019 8:41 PM  | Python Source File | 3 KB     |

**Gambar 7.87** kode program pada blok In[16].

### 17. No. 17 Kode Program Blok # In 17

```
1 # In[17]: save label encoder (to reverse one-hot encoding)
2 np.save('classes.npy', label_encoder.classes_)
```

Keterangannya sebagai berikut :

- Simpan label enkoder (untuk membalikkan one-hot encoder) dengan nama classes.npy

```
In [18]: np.save('classes.npy', label_encoder.classes_)
```

**Gambar 7.88** kode program pada blok In[17].

### 18. No. 18 Kode Program Blok # In 18

```
1 # In[18]: load the pre-trained model and predict the math
 symbol for an arbitrary image;
2 # the code below could be placed in a separate file
3 import keras.models
4 model2 = keras.models.load_model("mathsymbols.model")
5 print(model2.summary())
```

Keterangannya sebagai berikut :

- Impor models dari librari Keras
- Variabel model2 akan memanggil model yang telah disave tadi
- Menampilkan ringkasan dari hasil pemodelan

```
In [19]: import keras.models
...: model2 = keras.models.load_model("mathsymbols.model")
...: print(model2.summary())
Layer (type): Output Shape Param # ...
conv2d_63 (Conv2D): (None, 38, 38, 32) 896
max_pooling2d_63 (MaxPooling): (None, 15, 15, 32) 0
conv2d_64 (Conv2D): (None, 13, 13, 32) 9248
max_pooling2d_64 (MaxPooling): (None, 6, 6, 32) 0
flatten_42 (Flatten): (None, 1152) 0
dense_83 (Dense): (None, 128) 147584
dropout_32 (Dropout): (None, 128) 0
dense_84 (Dense): (None, 389) 47591
...
Total params: 205,329
Trainable params: 205,329
Non-trainable params: 0
None
```

**Gambar 7.89** kode program pada blok In[18].

### 19. No. 19 Kode Program Blok # In 19

```
1 # In[19]: restore the class name to integer encoder
2 label_encoder2 = LabelEncoder()
3 label_encoder2.classes_ = np.load('classes.npy')
4
5 def predict(img_path):
6 newimg = keras.preprocessing.image.img_to_array(pil_image
 .open(img_path))
```

```

7 newimg /= 255.0
8
9 # do the prediction
10 prediction = model2.predict(newimg.reshape(1, 32, 32, 3))
11
12 # figure out which output neuron had the highest score,
13 # and reverse the one-hot encoding
14 inverted = label_encoder2.inverse_transform([np.argmax(
 prediction)]) # argmax finds highest-scoring output
 print("Prediction: %s, confidence: %.2f" % (inverted[0],
 np.max(prediction)))

```

Keterangannya sebagai berikut :

- Memanggil fungsi LabelEncoder
- Variabel label\_encoder akan memanggil class yang disave sebelumnya.
- Function Predict akan mengubah gambar kedalam bentuk array
- Variabel prediction akan melakukan prediksi untuk model2 dengan reshape variabel newimg dengan bentukarray 4D.
- Variabel inverted akan mencari nilai tertinggi output dari hasil prediksi tadi
- Menampilkan hasil dari variabel prediction dan inverted

```

In [20]: label_encoder2 = LabelEncoder()
... label_encoder2.classes_ = np.load('classes.npy')
... def predict(img_path):
... ...
... img = Image.open(img_path)
... img = np.array(img)
... img = img / 255.0
... ...
... # do the prediction
... prediction = model2.predict(img.reshape(1, 32, 32, 3))
... ...
... # figure out which output neuron had the highest score, and reverse
... # the one-hot encoding
... inverted = label_encoder2.inverse_transform([np.argmax(prediction)]) # argmax finds
... highest_scoring_output = ...
... print("Prediction: %s, confidence: %.2f" % (inverted[0],
... np.max(prediction)))

```

**Gambar 7.90** kode program pada blok In[19].

## 20. No. 20 Kode Program Blok # In 20

```

1 # In[20]: grab an image (we'll just use a random training
 # image for demonstration purposes)
2 predict("HASYv2/hasy-data/v2-00010.png")
3
4 predict("HASYv2/hasy-data/v2-00500.png")
5
6 predict("HASYv2/hasy-data/v2-00700.png")

```

Keterangannya sebagai berikut :

- Melakukan prediksi dari pelatihan dari gambar v2-00010.png
- Melakukan prediksi dari pelatihan dari gambar v2-00500.png
- Melakukan prediksi dari pelatihan dari gambar v2-00700.png

```
In [23]: predict("HASYv2/hasy-data/v2-00010.png")
...
...: predict("HASYv2/hasy-data/v2-00500.png")
...
...: predict("HASYv2/hasy-data/v2-00700.png")
Prediction: A, confidence: 0.83
Prediction: \pi, confidence: 0.48
Prediction: \alpha, confidence: 0.92

In [24]: predict("HASYv2/hasy-data/v2-00010.png")
...
...: predict("HASYv2/hasy-data/v2-00500.png")
...
...: predict("HASYv2/hasy-data/v2-01000.png")
Prediction: A, confidence: 0.83
Prediction: \pi, confidence: 0.48
Prediction: \sum, confidence: 0.56

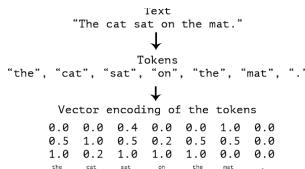
In [25]: predict("HASYv2/hasy-data/v2-00010.png")
...
...: predict("HASYv2/hasy-data/v2-00900.png")
...
...: predict("HASYv2/hasy-data/v2-01000.png")
Prediction: A, confidence: 0.83
Prediction: \beta, confidence: 0.84
Prediction: \sum, confidence: 0.56
```

**Gambar 7.91** kode program pada blok In[20].

## 7.4 1174039- Liyana Majdah Rahma

### 7.4.1 Teori

1. Teks Tokenizer Untuk memudahkan mesin memahami maksud dari apa yang kita inginkan dalam machine learning, kata pada teks disebut token, dan proses vektorisasi dari bentuk kata ke dalam token tersebut disebut tokenizer dan tokenizer akan merubah sebuah teks menjadi simbol, kata, ataupun biner dan bentuk lainnya kedalam token. Dapat dilihat seperti gambar dibawah ini



**Gambar 7.92** Ilustrasi gambar Teks Tokenizer

2. konsep dasar K Fold Cross Validation pada dataset komentar Youtube

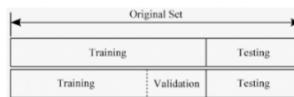
KStartifiedKFold berisikan presentasi sampel untuk setiap kelas. Di mana dalam ilustrasi ini sampel dibagi menjadi 5 dalam setiap class nya. Kemudian sampel tadi akan dimasukan kedalam class dari dataset youtube tadi. Dapat dilihat seperti gambar dibawah ini

```
>>> from sklearn.model_selection import StratifiedKFold
>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
>>> y = np.array([1, 1, 0, 0])
>>> skf = StratifiedKFold(n_splits=2)
>>> skf.get_n_splits(X, y)
2
>>> print(skf)
StratifiedKFold(n_splits=2, random_state=None, shuffle=False)
>>> for train_index, test_index in skf.split(X, y):
... print("TRAIN:", train_index, "TEST:", test_index)
...
... X_train, X_test = X[train_index], X[test_index]
... y_train, y_test = y[train_index], y[test_index]
TRAIN: [1 3] TEST: [0 2]
TRAIN: [0 2] TEST: [1 3]
```

**Gambar 7.93** Ilustrasi konsep dasar K Fold Cross Validation

### 3. kode program for train, test in splits

Maksudnya yaitu untuk menguji apakah setiap data pada dataset sudah di split dan tidak terjadi penumpukan. Yang dimana maksudnya di setiap class tidak akan muncul id yang sama. Dapat dilihat seperti gambar dibawah ini



**Gambar 7.94** Ilustrasi kode program for train, test in splits

### 4. Jelaskan apa maksudnya kode program *train\_content = df['CONTENT'].iloc[train\_idx]* dan *test\_content = df['CONTENT'].iloc[test\_idx]*. dilengkapi dengan ilustrasi atau gambar.

Maksudnya yaitu mengambil data pada kolom atau index CONTENT yang merupakan bagian dari train\_idx dan test\_idx. Ilustrasinya, ketika data telah diubah menjadi train dan test maka kita dapat memilihnya untuk ditampilkan pada kolom yang diinginkan.

```
>>> x = np.array([-1.2, 1.2])
>>> np.absolute(x)
array([1.2, 1.2])
```

**Gambar 7.95** Gambar hasil train

### 5. Jelaskan apa maksud dari fungsi *tokenizer = Tokenizer(num\_words=2000)* dan *tokenizer.fit\_on\_texts(train\_content)*, dilengkapi dengan ilustrasi atau gambar

Dimana variabel tokenizer akan melakukan vektorisasi kata menggunakan fungsi Tokenizer yang dimana jumlah kata yang ingin diubah kedalam bentuk token adalah 2000 kata. dan maksudnya kita akan melakukan fit tokenizer hanya untuk dat trainnya saja tidak dengan data test nya untuk kolom CONTENT. Dapat dilihat seperti gambar dibawah ini

```

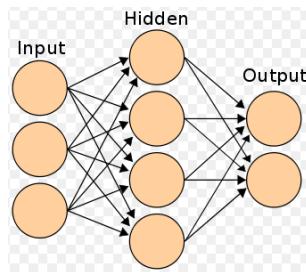
> labels
array([0, 2, 1, 2, 0])
`to_categorical` converts this into a matrix with as many
columns as there are classes. The number of rows
stays the same.
> to_categorical(labels)
array([[1., 0., 0.],
 [0., 1., 0.],
 [0., 0., 1.],
 [0., 0., 1.],
 [1., 0., 0.]], dtype=float32)

```

**Gambar 7.96** Ilustrasi Cara Membaca Hasil Tokenizer

6. Jelaskan apa maksud dari fungsi `d.train_inputs = tokenizer.texts_to_matrix(train_content, mode='tfidf')` dan `d.test_inputs = tokenizer.texts_to_matrix(test_content, mode='tfidf')`, dilengkapi dengan ilustrasi kode dan atau gambar

Maksudnya yaitu untuk variabel d\_train\_inputs akan melakukan tokenizer dari bentuk teks ke matrix dari data train\_content dengan mode matriksnya yaitu tfidf begitu juga dengan variabel d\_test\_inputs untuk data test. Dapat dilihat seperti gambar dibawah ini



**Gambar 7.97** Ilustrasi Hasil fungsi train input

7. Jelaskan apa maksud dari fungsi `d.train_inputs = d_train_inputs/npamax(np.abs(d_train_inputs))` dan `d.test_inputs = d_test_inputs/npamax(np.absolute(d_test_inputs))`, dilengkapi dengan ilustrasi atau gambar

Fungsi tersebut akan membagi matrix tfidf tadi dengan amax yaitu mengembalikan maksimum array atau maksimum sepanjang sumbu. Dapat dilihat seperti gambar dibawah ini

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|----|----|---|---|---|---|---|
| <table border="1"> <tbody> <tr><td>1</td><td>2</td><td>5</td><td>2</td><td>1</td><td>4</td></tr> <tr><td>3</td><td>2</td><td>5</td><td>1</td><td>3</td><td>5</td></tr> <tr><td>3</td><td>4</td><td>4</td><td>3</td><td>3</td><td>3</td></tr> <tr><td>5</td><td>3</td><td>5</td><td>1</td><td>3</td><td>3</td></tr> <tr><td>5</td><td>1</td><td>1</td><td>5</td><td>5</td><td>1</td></tr> <tr><td>5</td><td>5</td><td>4</td><td>3</td><td>3</td><td>1</td></tr> </tbody> </table> | 1  | 2 | 5 | 2 | 1 | 4 | 3 | 2 | 5 | 1 | 3 | 5 | 3 | 4 | 4 | 3 | 3 | 3 | 5 | 3 | 5 | 1 | 3 | 3 | 5 | 1 | 1 | 5 | 5 | 1 | 5 | 5 | 4 | 3 | 3 | 1 | Dengan g(x,y) | <table border="1"> <tbody> <tr><td>0</td><td>-1</td></tr> <tr><td>-1</td><td>0</td></tr> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td></tr> </tbody> </table> | 0 | -1 | -1 | 0 | 0 | 1 | 1 | 1 |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 2  | 5 | 2 | 1 | 4 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 2  | 5 | 1 | 3 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 4  | 4 | 3 | 3 | 3 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 3  | 5 | 1 | 3 | 3 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 1  | 1 | 5 | 5 | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 5  | 4 | 3 | 3 | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | -1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
| -1                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 0  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 1  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 1  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |               |                                                                                                                                                                        |   |    |    |   |   |   |   |   |

**Gambar 7.98** Ilustrasi train input dan test input

8. Jelaskan apa maksud fungsi dari `d.train_outputs = np_utils.to_categorical(d['CLASS'])` dan `d.test_outputs = np_utils.to_categorical(d['CLASS'].iloc[test_idx])` dalam kode program, dilengkapi dengan ilustrasi atau gambar

Dalam variabel d\_train\_output dan d\_test\_outputs akan dilakukan one hot encoding, dimana np\_utilsakan mengubah vektor dengan ben-

tuk integer ke matriks kelas biner untuk kolom CLASS dimana nantinya hanya akan ada dua pilihan yaitu 1 atau 0. 1 untuk spam 0 untuk non spam atau sebaliknya.Dapat dilihat seperti gambar dibawah ini

|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---------------------------------------------------------------------------------------------------------------------------------------------------|---|---|----|----|---|---|---|---|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|---|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| <table border="1"><tr><td>1</td><td>2</td><td>5</td><td>2</td><td>1</td><td>4</td></tr><tr><td>3</td><td>2</td><td>5</td><td>1</td><td>1</td><td>5</td></tr><tr><td>3</td><td>2</td><td>4</td><td>1</td><td>1</td><td>5</td></tr><tr><td>5</td><td>3</td><td>5</td><td>1</td><td>3</td><td>5</td></tr><tr><td>5</td><td>1</td><td>4</td><td>1</td><td>5</td><td>1</td></tr><tr><td>5</td><td>5</td><td>4</td><td>3</td><td>3</td><td>1</td></tr></table> | 1 | 2  | 5 | 2 | 1 | 4 | 3 | 2 | 5 | 1 | 1 | 5 | 3 | 2 | 4 | 1 | 1 | 5 | 5 | 3 | 5 | 1 | 3 | 5 | 5 | 1 | 4 | 1 | 5 | 1 | 5 | 5 | 4 | 3 | 3 | 1 | <table border="1"><tr><td>0</td><td>2</td><td>-1</td></tr><tr><td>-1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td></tr></table> | 0 | 2 | -1 | -1 | 0 | 2 | 0 | 1 | 1 | <table border="1"><tr><td>12</td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table> | 12 |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | 5  | 2 | 1 | 4 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | 5  | 1 | 1 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | 4  | 1 | 1 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 3 | 5  | 1 | 3 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 1 | 4  | 1 | 5 | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 5 | 4  | 3 | 3 | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | -1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| -1                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 0 | 2  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 1 | 1  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12                                                                                                                                                                                                                                                                                                                                                                                                                                                       |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Kompositus→                                                                                                                                                                                                                                                                                                                                                                                                                                              |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| <table border="1"><tr><td>1</td><td>2</td><td>5</td><td>2</td><td>1</td><td>4</td></tr><tr><td>3</td><td>2</td><td>5</td><td>1</td><td>1</td><td>5</td></tr><tr><td>3</td><td>2</td><td>4</td><td>1</td><td>1</td><td>5</td></tr><tr><td>5</td><td>3</td><td>5</td><td>1</td><td>3</td><td>5</td></tr><tr><td>5</td><td>1</td><td>4</td><td>1</td><td>5</td><td>1</td></tr><tr><td>5</td><td>5</td><td>4</td><td>3</td><td>3</td><td>1</td></tr></table> | 1 | 2  | 5 | 2 | 1 | 4 | 3 | 2 | 5 | 1 | 1 | 5 | 3 | 2 | 4 | 1 | 1 | 5 | 5 | 3 | 5 | 1 | 3 | 5 | 5 | 1 | 4 | 1 | 5 | 1 | 5 | 5 | 4 | 3 | 3 | 1 | <table border="1"><tr><td>0</td><td>2</td><td>-1</td></tr><tr><td>-1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td></tr></table> | 0 | 2 | -1 | -1 | 0 | 2 | 0 | 1 | 1 | <table border="1"><tr><td>12</td><td>6</td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>                                    | 12 | 6 |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | 5  | 2 | 1 | 4 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | 5  | 1 | 1 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | 4  | 1 | 1 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 3 | 5  | 1 | 3 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 1 | 4  | 1 | 5 | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 5 | 4  | 3 | 3 | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | -1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| -1                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 0 | 2  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 1 | 1  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 6 |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Kompositus→                                                                                                                                                                                                                                                                                                                                                                                                                                              |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| <table border="1"><tr><td>1</td><td>2</td><td>5</td><td>2</td><td>1</td><td>4</td></tr><tr><td>3</td><td>2</td><td>5</td><td>1</td><td>1</td><td>5</td></tr><tr><td>3</td><td>2</td><td>4</td><td>1</td><td>1</td><td>5</td></tr><tr><td>5</td><td>3</td><td>5</td><td>1</td><td>3</td><td>5</td></tr><tr><td>5</td><td>1</td><td>4</td><td>1</td><td>5</td><td>1</td></tr><tr><td>5</td><td>5</td><td>4</td><td>3</td><td>3</td><td>1</td></tr></table> | 1 | 2  | 5 | 2 | 1 | 4 | 3 | 2 | 5 | 1 | 1 | 5 | 3 | 2 | 4 | 1 | 1 | 5 | 5 | 3 | 5 | 1 | 3 | 5 | 5 | 1 | 4 | 1 | 5 | 1 | 5 | 5 | 4 | 3 | 3 | 1 | <table border="1"><tr><td>0</td><td>2</td><td>-1</td></tr><tr><td>-1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td></tr></table> | 0 | 2 | -1 | -1 | 0 | 2 | 0 | 1 | 1 | <table border="1"><tr><td>12</td><td>6</td><td>15</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>                                  | 12 | 6 | 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | 5  | 2 | 1 | 4 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | 5  | 1 | 1 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | 4  | 1 | 1 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 3 | 5  | 1 | 3 | 5 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 1 | 4  | 1 | 5 | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 5 | 4  | 3 | 3 | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 2 | -1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| -1                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 0 | 2  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 1 | 1  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 6 | 15 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Kompositus→                                                                                                                                                                                                                                                                                                                                                                                                                                              |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                   |   |   |    |    |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Gambar 7.99 Ilustrasi train output

9. Jelaskan apa maksud dari fungsi di listing dilengkapi dengan ilustrasi atau gambar

Melakukan pemodelan Sequential, dan Layer pertama dense dari 512 neuron untuk inputan dengan inputan tadi yang sudah dijadikan matriks sebanyak 2000. Activationnya menggunakan fungsi relu yaitu jika ada inputan dengan nilai maksimum maka inputan itu yang akan terpilih.Dapat dilihat seperti gambar dibawah ini

|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|---|----|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| $\begin{matrix} 1 & 2 & 5 & 2 & 1 & 4 \\ 3 & 2 & 5 & 1 & 1 & 5 \\ 3 & 2 & 4 & 1 & 1 & 5 \\ 5 & 3 & 5 & 1 & 3 & 5 \\ 5 & 1 & 4 & 1 & 5 & 1 \\ 5 & 5 & 4 & 3 & 3 & 1 \end{matrix}$ | $\begin{matrix} 0 & 2 & -1 \\ -1 & 0 & 2 \\ 0 & 1 & 1 \end{matrix}$ | <table border="1"><tr><td>12</td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>    | 12 |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12                                                                                                                                                                               |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Kompositus→                                                                                                                                                                      |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| $\begin{matrix} 1 & 2 & 5 & 2 & 1 & 4 \\ 3 & 2 & 5 & 1 & 1 & 5 \\ 3 & 2 & 4 & 1 & 1 & 5 \\ 5 & 3 & 5 & 1 & 3 & 5 \\ 5 & 1 & 4 & 1 & 5 & 1 \\ 5 & 5 & 4 & 3 & 3 & 1 \end{matrix}$ | $\begin{matrix} 0 & 2 & -1 \\ -1 & 0 & 2 \\ 0 & 1 & 1 \end{matrix}$ | <table border="1"><tr><td>12</td><td>6</td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>   | 12 | 6 |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12                                                                                                                                                                               | 6                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Kompositus→                                                                                                                                                                      |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| $\begin{matrix} 1 & 2 & 5 & 2 & 1 & 4 \\ 3 & 2 & 5 & 1 & 1 & 5 \\ 3 & 2 & 4 & 1 & 1 & 5 \\ 5 & 3 & 5 & 1 & 3 & 5 \\ 5 & 1 & 4 & 1 & 5 & 1 \\ 5 & 5 & 4 & 3 & 3 & 1 \end{matrix}$ | $\begin{matrix} 0 & 2 & -1 \\ -1 & 0 & 2 \\ 0 & 1 & 1 \end{matrix}$ | <table border="1"><tr><td>12</td><td>6</td><td>15</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table> | 12 | 6 | 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12                                                                                                                                                                               | 6                                                                   | 15                                                                                                                                                                                                                                                                                                                                                                                                                                          |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|                                                                                                                                                                                  |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Kompositus→                                                                                                                                                                      |                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                             |    |   |    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Gambar 7.100 Ilustrasi fungsi listing

10. Jelaskan apa maksud dari fungsi di listing ke 2 dilengkapi dengan ilustrasi atau gambar

Melakukan peng compile-an dari model Sequential tadi dengan Loss yandengang merupakan fungsi optimisasi skor menggunakan categorical, dan menggunakan algoritma adam sebagai optimizer. Adam yaitu algoritma pengoptimalan yang dapat digunakan sebagai ganti dari prosedur penurunan gradien stokastik klasik untuk memperbarui bobot jaringan yang berulang berdasarkan data training.Dengan metrik yaitu fungsi yang digunakan untuk menilai kinerja mode Anda disini menggunakan fungsi accuracy.Dapat dilihat seperti gambar dibawah ini

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | 2 | 5 | 2 | 1 | 4 |
| 3 | 2 | 5 | 1 | 1 | 5 |
| 3 | 2 | 4 | 1 | 1 | 3 |
| 5 | 3 | 1 | 1 | 1 | 5 |
| 5 | 1 | 1 | 5 | 1 | 1 |
| 5 | 1 | 4 | 3 | 3 | 1 |

|    |   |    |
|----|---|----|
| 0  | 2 | -1 |
| -1 | 0 | 2  |
| 0  | 1 | 1  |

konvolusi →

|    |   |    |
|----|---|----|
| 12 | 6 | 15 |
| 7  | 8 |    |
|    |   |    |

$(0^*4)+(1^*1)+(1^*5)+(1^*5)+(0^*1)+(2^*3)+(0^*1)+(1^*5)+(1^*5)=8$

$$(0^*4)+(2^*1)+(1^*5)+(1^*5)+(0^*1)+(2^*3)+(0^*1)+(1^*5)+(1^*5)=8$$

$$(0^*1)+(2^*5)+(1^*3)+(1^*1)+(0^*3)+(2^*5)+(0^*5)+(1^*5)+(1^*1)=22$$

**Gambar 7.101** Ilustrasi fungsi listing dari model sequential

### 11. Jelaskan apa itu Deep Learning

Deep Learning adalah subbidang machine learning yang berkaitan dengan algoritma yang terinspirasi oleh struktur dan fungsi otak yang disebut jaringan saraf tiruan atau Artificial Neural Networks. Jaringan saraf tiruan, algoritma yang terinspirasi oleh otak manusia, belajar dari sejumlah besar data. Demikian pula dengan bagaimana kita belajar dari pengalaman, algoritma pembelajaran yang mendalam akan melakukan tugas berulang kali, setiap kali sedikit mengubahnya untuk

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | 2 | 5 | 2 | 1 | 4 |
| 3 | 2 | 5 | 1 | 1 | 5 |
| 3 | 2 | 4 | 1 | 1 | 3 |
| 5 | 3 | 1 | 1 | 1 | 5 |
| 5 | 1 | 1 | 5 | 1 | 1 |
| 5 | 1 | 4 | 3 | 3 | 1 |

|    |   |    |
|----|---|----|
| 0  | 2 | -1 |
| -1 | 0 | 2  |
| 0  | 1 | 1  |

konvolusi →

|    |   |    |
|----|---|----|
| 12 | 6 | 15 |
| 7  | 8 | 22 |
| 2  |   |    |

$$(0^*5)+(2^*3)+(1^*5)+(1^*5)+(0^*1)+(2^*3)+(0^*5)+(1^*5)+(1^*4)=7$$

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | 2 | 5 | 2 | 1 | 4 |
| 3 | 2 | 5 | 1 | 1 | 5 |
| 3 | 2 | 4 | 1 | 1 | 3 |
| 5 | 3 | 1 | 1 | 1 | 5 |
| 5 | 1 | 1 | 5 | 1 | 1 |
| 5 | 1 | 4 | 3 | 3 | 1 |

|    |   |    |
|----|---|----|
| 0  | 2 | -1 |
| -1 | 0 | 2  |
| 0  | 1 | 1  |

konvolusi →

|    |    |    |
|----|----|----|
| 12 | 6  | 15 |
| 7  | 8  | 22 |
| 7  | 14 |    |

$$(0^*5)+(2^*3)+(1^*5)+(1^*5)+(0^*1)+(2^*3)+(0^*5)+(0^*4)+(1^*5)+(1^*3)=14$$

meningkatkan hasilnya.

learning

pengertian Deep

Jelaskan apa itu Deep Neural Network, dan apa bedanya dengan Deep Learning

Deep Neural Network adalah jaringan syaraf tiruan (JST) dengan beberapa lapisan antara lapisan input dan output. DNN menemukan manipulasi matematis yang benar untuk mengubah input menjadi output, apakah itu hubungan linear atau hubungan non-linear. Merupakan jaringan syaraf dengan tingkat kompleksitas tertentu, jaringan syaraf dengan lebih dari dua lapisan. Deep Neural Network menggunakan pemodelan matematika yang canggih untuk memproses data dengan cara yang kompleks.

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | 2 | 5 | 2 | 1 | 4 |
| 3 | 2 | 5 | 1 | 1 | 5 |
| 3 | 2 | 4 | 1 | 1 | 3 |
| 5 | 3 | 1 | 1 | 1 | 5 |
| 5 | 1 | 1 | 5 | 1 | 1 |
| 5 | 1 | 4 | 3 | 3 | 1 |

|    |   |    |
|----|---|----|
| 0  | 2 | -1 |
| -1 | 0 | 2  |
| 0  | 1 | 1  |

konvolusi →

|    |    |    |
|----|----|----|
| 12 | 6  | 15 |
| 7  | 8  | 22 |
| 7  | 14 |    |

$$(0^*1)+(2^*3)+(1^*5)+(1^*5)+(0^*1)+(2^*3)+(0^*5)+(0^*4)+(1^*5)+(1^*3)=1$$

**Gambar 7.102** perbedaan deep neural network dengan deep learning

Jelaskan dengan ilustrasi gambar buatan sendiri(langkah per langkah) bagaimana perhitungan algoritma konvolusi dengan ukuran stride (NPM mod3+1) x (NPM mod3+1) yang terdapat max poolingStridenya 3

Melakukan pemodelan Sequential, dan Layer pertama dense dari 512 neuron untuk inputan dengan inputan tadi yang sudah dijadikan matriks sebanyak 2000. Activationnya menggunakan fungsi relu yaitu jika ada inputan dengan nilai maksimum maka inputan itu yang akan terpilih. Dapat dilihat seperti gambar dibawah ini

|                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                |    |    |   |   |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|---|---|----|---|---|----|---|----|---|---|---|----|----|----|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hasil konvolusi akhir                                                                    | <table border="1"><tr><td>1</td><td>2</td><td>5</td><td>2</td><td>1</td><td>4</td></tr><tr><td>3</td><td>12</td><td>6</td><td>15</td><td>1</td><td>5</td></tr><tr><td>4</td><td>11</td><td>10</td><td>22</td><td>1</td><td>3</td></tr><tr><td>5</td><td>7</td><td>14</td><td>1</td><td>3</td><td>5</td></tr><tr><td>5</td><td>1</td><td>1</td><td>5</td><td>5</td><td>1</td></tr><tr><td>5</td><td>5</td><td>4</td><td>3</td><td>3</td><td>1</td></tr></table> | 1  | 2  | 5 | 2 | 1  | 4 | 3 | 12 | 6 | 15 | 1 | 5 | 4 | 11 | 10 | 22 | 1 | 3 | 5 | 7 | 14 | 1 | 3 | 5 | 5 | 1 | 1 | 5 | 5 | 1 | 5 | 5 | 4 | 3 | 3 | 1 |
| 1                                                                                        | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 5  | 2  | 1 | 4 |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 3                                                                                        | 12                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 6  | 15 | 1 | 5 |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 4                                                                                        | 11                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 10 | 22 | 1 | 3 |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5                                                                                        | 7                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 14 | 1  | 3 | 5 |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5                                                                                        | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 1  | 5  | 5 | 1 |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5                                                                                        | 5                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 4  | 3  | 3 | 1 |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Kemudian untuk max pooling 2x2 dengan stride 3 karena $(1164080 \bmod 3 + 1) = 3$ adalah |                                                                                                                                                                                                                                                                                                                                                                                                                                                                |    |    |   |   |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|                                                                                          | <table border="1"><tr><td>12</td><td>15</td><td>9</td></tr><tr><td>7</td><td>22</td><td>5</td></tr><tr><td>5</td><td>5</td><td>5</td></tr></table>                                                                                                                                                                                                                                                                                                             | 12 | 15 | 9 | 7 | 22 | 5 | 5 | 5  | 5 |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 12                                                                                       | 15                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 9  |    |   |   |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 7                                                                                        | 22                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 5  |    |   |   |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5                                                                                        | 5                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 5  |    |   |   |    |   |   |    |   |    |   |   |   |    |    |    |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

**Gambar 7.103** Ilustrasi fungsi listing

#### 7.4.2 Praktek

##### 1. No.1 Kode Program Blok In 1

- Pertama kita akan mengimpor librari csv. Dimana dari librai PIL atau Pillow atau Python Imaging Library akan diimporkan modul Image yang di inisiasikan sebagai pil.image. Modul Image menyediakan kelas dengan nama yang sama yang digunakan untuk mewakili gambar PIL. Modul ini juga menyediakan sejumlah fungsi pabrik, termasuk fungsi untuk memuat image dari file, dan untuk membuat image baru.mengimpor librari image dari keras .Yang menghasilkan kumpulan data gambar tensor dengan augmentasi data waktu nyata. Data akan diulang (dalam batch).Berikut Hasilnya :



**Gambar 7.104** kode blok satu

##### ▪ No.2 Kode Program Blok In 2

variabel imgs berisikan array kosong

Variabel classes berisikan array kosong

Membuka file csv dari Folder HSYv2 dengan nama file hasy-data-labels.csv sebagai csvfile

Variabel csvreader akan menggunakan fungsi reader pada library csv untuk membaca file csv tadi yang disimpan di csvfile.

Dimana variabel i dimulai dari nol.

Untuk setiap baris pada csvreader

Jika i lebih besar dari 0

Jadi itu akan mengambil contoh Gambar PIL dan mengubahnya menjadi array numpy dengan mengambil data dari HSYv2 dan dimulai dari baris ke nol.

Hasil dari variabel img akan dibagi dengan 255.0

append akan membuat list array baru untuk baris 0 baris 2 pada img.

Menyimpan setiap class nya pada baris 2

Penambahan i sebanyak 1.



**Gambar 7.105** hasil olah data blok kedua

- No.3 Kode Program Blok In 3

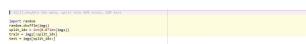
Impor librari Random dari Python

Melakukan pengacakan untuk imgs dengan Metode Shuffle untuk mengocok urutan di tempat. yaitu, mengubah posisi item dalam daftar.

Membagi data dari imgs dengan cara mengalikan 80% dengan jumlah data dari imgs.

Untuk data train mengambil hasil dari perhitungan sebelumnya.

Untuk data test mengambil sisa dari jumlah yang telah dijadikan data train



**Gambar 7.106** hasil olah blok in ketiga

- No.4 Kode Program Blok In 4

Impor librari Numpy yang di inisiasikan sebagai np

Variabel train\_input mengubah input menjadi sebuah array yang diambil dari baris 2, data train.

Variabel test\_input mengubah input menjadi sebuah array yang diambil dari baris 2, data test.

Variabel train\_output mengubah input menjadi sebuah array yang diambil dari baris 1, data train.

Variabel train\_output mengubah input menjadi sebuah array yang diambil dari baris 1, data test.

```

1 #!/usr/bin/python
2 import numpy as np
3
4 train_input = np.array([[lambda row: row[0], text]])
5 test_input = np.array([[lambda row: row[1], text]])
6
7 train_output = np.array([list(map(lambda row: row[1], text))])
8 test_output = np.array([list(map(lambda row: row[1], text))])

```

**Gambar 7.107** hasil olah data blok in keempat

- No.5 Kode Program Blok In 5
  - Impor Fungsi LabelEncoder
  - Impor Fungsi OneHotEncoder

```

In [1]: from sklearn.preprocessing import LabelEncoder
In [2]: from sklearn.preprocessing import OneHotEncoder

```

**Gambar 7.108** hasil olah data blok kelima

- No.6 Kode Program Blok In 6
  - Variabel label\_encoder akan memanggil fungsi LabelEncoder tadi.
  - variabel integer\_encoded akan menggunakan labelencoder untuk melakukan fit pada classes agar berubah datanya menjadi integer.

```

1 # In[6]: convert class names into one-hot encoding
2
3 # first, convert class names into integers
4 label_encoder = LabelEncoder()
5 integer_encoded = label_encoder.fit_transform(classes)

```

**Gambar 7.109** hasil olah data blok keenam

- No.7 Kode Program Blok In 7
  - Variabel onehot\_encoder akan memanggil fungsi OneHotEncoder dimana tidak berisikan matriks sparse.
  - Pada variabel integer\_encoded akan diubah bentuknya dimana setiap nilai integer akan direpresentasikan sebagai vektor binari dengan nilai 0 kecuali index dari integer tersebut ditandai dengan 1.
  - Melakukan fit untuk one hot encoder kedalam integer\_encoder.

```

1 # In[7]: convert integer_encoded into one-hot encoding
2
3 onehot_encoder = OneHotEncoder(sparse=False)
4 integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
5 onehot_encoder.fit(integer_encoded)

```

**Gambar 7.110** hasil olah data blok ke tujuh

- No.8 Kode Program Blok In 8
  - Variabel train\_output\_int akan mengubah data dari train\_output menjadi LabelEncoder

Variabel test\_output\_int akan mengubah data dari test\_output menjadi LabelEncoder

Dimana pada train\_output setelah diubah labelnya menjadi integer dilakukan one hot encoding diambil dari test\_output\_int dan menggunakan .reshape untuk memberikan bentuk baru ke array tanpa mengubah datanya dengan keterangan jika index dari integer tersebut ditandai dengan 1 dan sisanya yang bukan nol.

Variabel num\_classes akan menampilkan jumlah data dari classes yang telah dilakukan label encoder

Menampilkan tulisan "Number of classes : %d" dmana mengembalikan nilai integer dari num\_classes.

```
In[8]: Import LabelEncoder
train_output_int = label_encoder.transform(train_output)
test_output_int = label_encoder.transform(test_output)
train_output_int = train_output_int.reshape(len(train_output_int), 33)
test_output_int = test_output_int.reshape(len(test_output_int), 33)

num_classes = len(label_encoder.classes_)

print("Number of classes: ", num_classes)
```

**Gambar 7.111** hasil blok in 8

- No.9 Kode Program Blok In 9

Impor Sequential dari model pada librari Keras.

Impor Dense, Dropout, Flatten dari modul Layers pada librari Keras.

Impor Conv2D, MaxPooling2D dari modul Layers pada librari Keras.

```
In[9]: Import sequential
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
```

**Gambar 7.112** hasil blok in 9

- No.10 Kode Program Blok In 10

Melakukan pemodelan Sequential.

Menambahkan Konvolusi 2D dengan 32 filter konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu dengan data dari train\_input mulai dari baris nol.

Menambahkan Max Pooling dengan matriks 2x2.

Dilakukan lagi penambahan Konvolusi 2D dengan 32 filter konvolusi masing-masing berukuran 3x3 dengan algoritam activation relu.

Mendefinisikan inputan dengan 1024 neuron dan menggunakan algoritma tanh untuk activationnya.

Dropout terdiri dari pengaturan secara acak tingkat pecahan unit input ke 0 pada setiap pembaruan selama waktu pelatihan, yang membantu mencegah overfitting sebesar 50% .

Untuk output layer menggunakan data dari variabel num\_classes dengan fungsi activationnya softmax.

Mengonfigurasi proses pembelajaran, yang dilakukan melalui metode compile, sebelum melatih suatu model.

Menampilkan atau mencetak representasi ringkasan model yang telah dibuat.

```

1 #!/usr/bin/python
2
3 model = Sequential()
4 model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
5 input_shape=(28, 28, 1)))
6 model.add(MaxPooling2D(pool_size=(2, 2)))
7 model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
8 model.add(MaxPooling2D(pool_size=(2, 2)))
9 model.add(Flatten())
10 model.add(Dense(128, activation='tanh'))
11 model.add(Dropout(0.5))
12 model.add(Dense(num_classes, activation='softmax'))
13
14 model.compile(loss='categorical_crossentropy', optimizer='adam',
15 metrics=['accuracy'])
16
17 print(model.summary())

```

Gambar 7.113 hasil blok in 10

- Impor Modul Callbacks dari Librari Keras. Variabel callback mendefinisikan Callback ini untuk menulis log untuk TensorBoard, yang memungkinkan Anda untuk memvisualisasikan grafik dinamis dari pelatihan dan metrik pengujian Anda, serta histogram aktivasi untuk berbagai lapisan dalam model Anda.

```

1 #!/usr/bin/python
2
3 import tensorflow as tf
4 from keras.callbacks import TensorBoard
5
6 tensorboard = TensorBoard(log_dir='./logs/mnist-style')

```

Gambar 7.114 hasil blok in 11

- Melakukan fit model dengan 32 ukuran subset dari sampel pelatihan Anda Epoch sebanyak 10 kali. Vebrose=2 maksudnya menampilkan nomor dari epoch yang sedang berjalan atau yang sudah dijalankan. Validasi plit sebanyak 20% sebagai fraksi data pelatihan untuk digunakan sebagai data validasi. Menggunakan TensorBoard sebagai callback untuk diterapkan selama pelatihan dan validasi. Variabel score mengembalikan nilai evaluate untuk menampilkan data loss dan data accuracy dari test yang Menampilkan data loss dengan menghitung jumlah kemunculan nol. Menampilkan data accuracy dengan menghitung jumlah kemunculan 1.

```

1 #!/usr/bin/python
2
3 model.fit(train_input, train_output,
4 nb_epoch=10, batch_size=32,
5 verbose=2,
6 validation_split=0.2,
7 callbacks=[tensorboard])
8
9 score = model.evaluate(test_input, test_output, verbose=2)
10 print('test loss:', score[0])
11 print('test accuracy:', score[1])

```

Gambar 7.115 hasil blok in 12

- impor modul time dari python anaconda. Variabel result berisikan array kosong. Dengan Menggunakan convolution 2D yang dimana akan memiliki 1 atau 2 layer.

```

 if (n < 1) return 0;
 if (n == 1) return 1;
 if (n == 2) return 2;
 if (n == 3) return 4;
 if (n == 4) return 8;
 if (n == 5) return 16;
 if (n == 6) return 32;
 if (n == 7) return 64;
 if (n == 8) return 128;
 if (n == 9) return 256;
 if (n == 10) return 512;
 if (n == 11) return 1024;
 if (n == 12) return 2048;
 if (n == 13) return 4096;
 if (n == 14) return 8192;
 if (n == 15) return 16384;
 if (n == 16) return 32768;
 if (n == 17) return 65536;
 if (n == 18) return 131072;
 if (n == 19) return 262144;
 if (n == 20) return 524288;
 if (n == 21) return 1048576;
 if (n == 22) return 2097152;
 if (n == 23) return 4194304;
 if (n == 24) return 8388608;
 if (n == 25) return 16777216;
 if (n == 26) return 33554432;
 if (n == 27) return 67108864;
 if (n == 28) return 134217728;
 if (n == 29) return 268435456;
 if (n == 30) return 536870912;
 if (n == 31) return 1073741824;
 if (n == 32) return 2147483648;
 if (n == 33) return 4294967296;
 if (n == 34) return 8589934592;
 if (n == 35) return 17179869184;
 if (n == 36) return 34359738368;
 if (n == 37) return 68719476736;
 if (n == 38) return 137438953472;
 if (n == 39) return 274877906944;
 if (n == 40) return 549755813888;
 if (n == 41) return 1099511627776;
 if (n == 42) return 2199023255552;
 if (n == 43) return 4398046511104;
 if (n == 44) return 8796093022208;
 if (n == 45) return 17592186044416;
 if (n == 46) return 35184372088832;
 if (n == 47) return 70368744177664;
 if (n == 48) return 140737488355328;
 if (n == 49) return 281474976710656;
 if (n == 50) return 562949953421312;
 if (n == 51) return 1125899906842624;
 if (n == 52) return 2251799813685248;
 if (n == 53) return 4503599627370496;
 if (n == 54) return 9007199254740992;
 if (n == 55) return 18014398509481984;
 if (n == 56) return 36028797018963968;
 if (n == 57) return 72057594037927936;
 if (n == 58) return 144115188075855872;
 if (n == 59) return 288230376151711744;
 if (n == 60) return 576460752303423488;
 if (n == 61) return 1152921504606846976;
 if (n == 62) return 2305843009213693952;
 if (n == 63) return 4611686018427387904;
 if (n == 64) return 9223372036854775808;
 if (n == 65) return 18446744073709551616;
 if (n == 66) return 36893488147419103232;
 if (n == 67) return 73786976294838206464;
 if (n == 68) return 147573952589676412928;
 if (n == 69) return 295147905179352825856;
 if (n == 70) return 590295810358705651712;
 if (n == 71) return 1180591620717411303424;
 if (n == 72) return 2361183241434822606848;
 if (n == 73) return 4722366482869645213696;
 if (n == 74) return 9444732965739290427392;
 if (n == 75) return 1888946593147858085584;
 if (n == 76) return 3777893186295716161168;
 if (n == 77) return 7555786372591432322336;
 if (n == 78) return 1511157274582864644672;
 if (n == 79) return 3022314549165729289344;
 if (n == 80) return 6044629098331458578688;
 if (n == 81) return 12089258196662917157376;
 if (n == 82) return 24178516393325834314752;
 if (n == 83) return 48357032786651668629504;
 if (n == 84) return 96714065573303337259008;
 if (n == 85) return 193428131146606674580016;
 if (n == 86) return 386856262293213349160032;
 if (n == 87) return 773712524586426698320064;
 if (n == 88) return 1547425049172853976640128;
 if (n == 89) return 3094850098345707953280256;
 if (n == 90) return 6189700196691415906560512;
 if (n == 91) return 12379400393382831813121024;
 if (n == 92) return 24758800786765663626242048;
 if (n == 93) return 49517601573531327252484096;
 if (n == 94) return 99035203147062654504968192;
 if (n == 95) return 198070406294125309009936384;
 if (n == 96) return 396140812588250618019872768;
 if (n == 97) return 792281625176501236039745536;
 if (n == 98) return 1584563253353002712079491072;
 if (n == 99) return 316912650670600542415882144;
 if (n == 100) return 633825301341201084831764288;
 if (n == 101) return 1267650602682402169663525776;
 if (n == 102) return 2535301205364804339327051552;
 if (n == 103) return 5070602410729608678654103104;
 if (n == 104) return 10141204821459217577308206208;
 if (n == 105) return 20282409642918435154616412416;
 if (n == 106) return 40564819285836870309232824832;
 if (n == 107) return 81129638571673740618465649664;
 if (n == 108) return 16225927714334748123693129936;
 if (n == 109) return 32451855428669496247386259872;
 if (n == 110) return 64903710857338992494772519744;
 if (n == 111) return 129807421714677984989545039488;
 if (n == 112) return 259614843429355969979090078976;
 if (n == 113) return 519229686858711939958180157952;
 if (n == 114) return 1038459373717423879916360315904;
 if (n == 115) return 2076918747434847759832720631808;
 if (n == 116) return 4153837494869695519665441263616;
 if (n == 117) return 8307674989739391039330882527232;
 if (n == 118) return 1661534977947878207866176505464;
 if (n == 119) return 3323069955895756415732353010928;
 if (n == 120) return 6646139911791512831464706021856;
 if (n == 121) return 1329227982358302566292941204312;
 if (n == 122) return 2658455964716605132585882408624;
 if (n == 123) return 5316911929433210265171764817248;
 if (n == 124) return 10633823858866205130343529634496;
 if (n == 125) return 21267647717732410260687059268992;
 if (n == 126) return 42535295435464820521374118537984;
 if (n == 127) return 85070590870929641042748237075968;
 if (n == 128) return 170141181741859282085496474151936;
 if (n == 129) return 340282363483718564160992948303872;
 if (n == 130) return 680564726967437128321985896607744;
 if (n == 131) return 136112945393487425664397179321488;
 if (n == 132) return 272225890786974851328794358642976;
 if (n == 133) return 544451781573949702657588717325952;
 if (n == 134) return 108890356314789940531517743465904;
 if (n == 135) return 217780712629579881063035486931808;
 if (n == 136) return 435561425259159762126070973863616;
 if (n == 137) return 871122850518319524252141947727232;
 if (n == 138) return 174224570103663904850428389545464;
 if (n == 139) return 348449140207327809700856779090928;
 if (n == 140) return 696898280414655619401713558181856;
 if (n == 141) return 139379656082931123880342711636372;
 if (n == 142) return 278759312165862247760685423272744;
 if (n == 143) return 557518624331724495521370846545488;
 if (n == 144) return 111503724866348899104274169309096;
 if (n == 145) return 223007449732697798208548338618192;
 if (n == 146) return 446014899465395596417096677236384;
 if (n == 147) return 892029798930791192834193354472768;
 if (n == 148) return 178405959786158238566838670894552;
 if (n == 149) return 356811919572316477133677341789104;
 if (n == 150) return 713623839144632954267354683578208;
 if (n == 151) return 142724767828926590853470936755616;
 if (n == 152) return 285449535657853181706941873511232;
 if (n == 153) return 570898571315706363413883747022464;
 if (n == 154) return 114179714263141272682776749404492;
 if (n == 155) return 228359428526282545365553498808984;
 if (n == 156) return 456718857052565090731106997617968;
 if (n == 157) return 913437714105130181462213995235936;
 if (n == 158) return 182687542821026036292442798547872;
 if (n == 159) return 365375085642052072584885597095744;
 if (n == 160) return 730750171284104145169771194191488;
 if (n == 161) return 146150034256820829033954238838376;
 if (n == 162) return 292300068513641658067888477676752;
 if (n == 163) return 584600137027283316135776955353504;
 if (n == 164) return 1169200274054566632271553910707008;
 if (n == 165) return 2338400548109133264543107821414016;
 if (n == 166) return 4676801096218266529086215642828032;
 if (n == 167) return 9353602192436533058172431285656064;
 if (n == 168) return 18707204384873066116344862571312128;
 if (n == 169) return 37414408769746132232689725142624256;
 if (n == 170) return 74828817539492264465379450285248512;
 if (n == 171) return 149657635078984528930758900570497024;
 if (n == 172) return 299315270157969057861517801140994048;
 if (n == 173) return 598630540315938115723035602281988096;
 if (n == 174) return 1197261080631876231446072044563977192;
 if (n == 175) return 2394522161263752462892144089127954384;
 if (n == 176) return 4789044322527504925784288178255908768;
 if (n == 177) return 9578088645055009851568576356511817536;
 if (n == 178) return 19156177290110019703137532713023635072;
 if (n == 179) return 38312354580220039406275065426047270144;
 if (n == 180) return 76624709160440078812550130852094540288;
 if (n == 181) return 15324941832088015762510066170418900576;
 if (n == 182) return 30649883664176031525020012340837801152;
 if (n == 183) return 61299767328352063050400246681675602304;
 if (n == 184) return 122599534656704126100800493363351204608;
 if (n == 185) return 245199069313408252201600986726702411216;
 if (n == 186) return 490398138626816504403201973453404822432;
 if (n == 187) return 980796277253632008806403946906809644864;
 if (n == 188) return 196159255450726401761280789381361929728;
 if (n == 189) return 392318510901452803522561578762723859456;
 if (n == 190) return 784637021802905607045123157525447718912;
 if (n == 191) return 1569274043605811214090246315050895437824;
 if (n == 192) return 3138548087211622428180492630101790875648;
 if (n == 193) return 627709617442324485636098526020358175136;
 if (n == 194) return 1255419234884648971272197052040716350272;
 if (n == 195) return 2510838469769297942544394104081432700544;
 if (n == 196) return 5021676939538595885088788208162865401088;
 if (n == 197) return 10043353879077191770175764416325730802176;
 if (n == 198) return 20086707758154383540351528832651461604352;
 if (n == 199) return 4017341551630876708070305766530292320864;
 if (n == 200) return 8034683103261753416140611533060584641728;
 if (n == 201) return 1606936620652350683228222306612117328356;
 if (n == 202) return 3213873241304701366456444613224234656712;
 if (n == 203) return 6427746482609402732912889226448469313424;
 if (n == 204) return 12855492965218805465825778452896938626848;
 if (n == 205) return 25710985930437610931651556905793877253696;
 if (n == 206) return 51421971860875221863303113811587554507392;
 if (n == 207) return 10284394372175044372660626762317510914784;
 if (n == 208) return 2056878874435008874532125352463502182968;
 if (n == 209) return 4113757748870017749064250704927004365344;
 if (n == 210) return 8227515497740035498128501409854008730688;
 if (n == 211) return 1645503099548007099625702281970801741376;
 if (n == 212) return 3291006199096014199251404563941603482752;
 if (n == 213) return 658201239819202839850280912788320696504;
 if (n == 214) return 1316402479638405679700561825576641392088;
 if (n == 215) return 2632804959276811359401123651153282784176;
 if (n == 216) return 5265609918553622718802247302266565568352;
 if (n == 217) return 10531219837107245437604494604533131136704;
 if (n == 218) return 21062439674214490875208989208566262273408;
 if (n == 219) return 42124879348428981750417978417132524546816;
 if (n == 220) return 84249758696857963500835956834265049093632;
 if (n == 221) return 16849951739371592700167911366853009818664;
 if (n == 222) return 3369985347874318540033582273370601963728;
 if (n == 223) return 673997069574863708006716454674120392756;
 if (n == 224) return 1347994139149727416013432909348240785132;
 if (n == 225) return 2695988278299454832026865818696481570264;
 if (n == 226) return 5391976556598909664053731637392963140528;
 if (n == 227) return 10783953113197819328107463274785926281056;
 if (n == 228) return 21567906226395638656214926549571852562112;
 if (n == 229) return 43135812452791277312429853098543705124224;
 if (n == 230) return 86271624905582554624859706197087410248448;
 if (n == 231) return 172543249811165109249719412394174820496896;
 if (n == 232) return 345086499622330218499438824788349640993792;
 if (n == 233) return 690172999244660436998877649576699281987584;
 if (n == 234) return 1380345998489320873997755299153398563975168;
 if (n == 235) return 2760691996978641747995510598306797127950336;
 if (n == 236) return 5521383993957283495991021196613594255800672;
 if (n == 237) return 11042767987914566991920422393227884511601344;
 if (n == 238) return 22085535975829133983840844786455769023202688;
 if (n == 239) return 4417107195165826796
```

**Gambar 7.116** hasil blok in 13

- Dilakukan Max Pooling 2D dengan ukuran matriks  $2 \times 2$ . Untuk layer kedua, melakukan Convolusi lagi dengan kriteria yang sama tanpa menambahkan input, ini dilakukan untuk mendapatkan data yang terbaik

```

model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(train_input[0].shape[1],)))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dense(128, activation='tanh'))
model.add(Dense(128, activation='tanh'))
model.add(Dense(128, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())

```

**Gambar 7.117** hasil blok in 14

- I Melakukan fit dengan join data train dan test agar dapat dilakukan pelatihan untuk jaringan pada semua data yang dimiliki.

```
[1]: In[33]:#use train and test data as input to the network as all data we have available to us
model.fit(np.concatenate((train_input, test_input)),
 np.concatenate((train_output, test_output)),
 batch_size=32, epochs=10, verbose=2)
```

**Gambar 7.118** hasil blok in 15

- Menyimpan atau save model yang telah di latih dengan nama math-symbols.model

```
| In[16]: save the trained model
model.save("mathsymbols.model")
```

Gambar 7.119 hasil blok in 16

- Simpan label enkoder.

```
| In[17]: save_label_encoder (to reverse one-hot encoding)
np.save('classes.npy', label_encoder.classes_)
```

**Gambar 7.120** hasil blok in 17

- Impor models dari librari Keras. Variabel model2 akan memanggil model yang telah disave tadi. kemudian Menampilkan ringkasan dari hasil pemodelan

```
[1]: [In] load the pre-trained model, and predict the math model, for an arbitrary image.
a single image can be placed in a separate file
import keras.models
model2 = keras.models.load_model("mathsymbols.model")
print(model2.summary())
```

**Gambar 7.121** hasil blok in 18

- Memanggil fungsi LabelEncoder.

```
[1]: [In] (a) restore the class name to integer encoder
label_encoder2 = LabelEncoder()
label_encoder2.fit(np.loadtxt('classes.npy'))
def predict(img_path):
reading = keras.preprocessing.image.img_to_array(pil_image.open(img_path))
reading = np.expand_dims(reading, axis=0)
prediction = model2.predict(reading.reshape(1, 32, 32, 3))
inverted = label_encoder2.inverse_transform([np.argmax(prediction)]) # inverse fields in
print("Prediction: No. confidence %.2f" % (inverted[0], np.max(prediction)))
```

**Gambar 7.122** hasil blok in 19

- Melakukan prediksi dari pelatihan.

```
[1]: [In] (a) load the image (will just use a random training image for demonstration purposes)
predict("MNISTv2/test-data/v2-00010.png")
predict("MNISTv2/test-data/v2-00510.png")
predict("MNISTv2/test-data/v2-00700.png")
```

**Gambar 7.123** hasil blok in 20

