

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:

Syafrial Fachrie Pane

Khaera Tunnisia

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. Sariasisih 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

1 Chapter 1	1
2 Chapter 2	3
3 Chapter 3	31
4 Chapter 4	77
5 Chapter 5	85
6 Chapter 6	119

DAFTAR ISI

Foreword	xiii
Kata Pengantar	xv
Acknowledgments	xvii
Acronyms	xix
Glossary	xxi
List of Symbols	xxiii
Introduction	xxv
<i>Rolly Maulana Awangga, S.T., M.T.</i>	
1 Chapter 1	1
2 Chapter 2	3
2.1 Arrizak Furqona Gifary (1174070)	4
2.1.1 Teori	4
2.1.2 Praktek	10
2.1.3 Penanganan Error	17
2.1.4 Bukti Tidak Plagiat	17

2.1.5	Link Youtube:	17
2.2	Mochamad Arifqi Ramadhan (1174074)	18
2.2.1	Teori	18
2.2.2	Praktek	23
2.2.3	Penanganan Error	30
2.2.4	Bukti Tidak Plagiat	30
3	Chapter 3	31
3.1	Kaka Kamaludin (1174067)	31
3.1.1	Teori	31
3.1.2	Praktek	35
3.1.3	Penanganan Error	52
3.1.4	Link Youtube:	52
3.2	Mochamad Arifqi Ramadhan (1174074)	53
3.2.1	Teori	53
3.2.2	Praktek	57
3.2.3	Penanganan Error	75
3.2.4	Bukti Tidak Plagiat	75
4	Chapter 4	77
4.1	Mochamad Arifqi Ramadhan (1174074)	77
4.1.1	Teori	77
4.1.2	Praktek Program	79
4.1.3	Penanganan Error	83
4.1.4	Bukti Tidak Plagiat	84
5	Chapter 5	85
5.1	1174067 - Kaka Kamaludin	85
5.1.1	Teori	85
5.1.2	Praktek	88
5.1.3	Penanganan Error	97
5.1.4	Link Youtube	97
5.2	Mochamad Arifqi Ramadhan(1174074)	97
5.2.1	Teori	97
5.2.2	Praktek	100
5.2.3	Penanganan Error	109
5.2.4	Bukti Tidak Plagiat	110
5.3	Nurul Izza Hamka - 1174062	110

5.3.1	Teori	110
5.3.2	Praktek Program	112
5.3.3	Bukti Tidak Plagiat	116
6	Chapter 6	119
6.1	1174067 - Kaka Kamaludin	119
6.1.1	Teori	119
6.1.2	Praktek	122
6.1.3	Penanganan Error	130
6.1.4	Link Youtube	130
6.2	Mochamad Arifqi Ramadhan(1174074)	131
6.2.1	Teori	131
6.2.2	Praktek	134
6.2.3	Penanganan Error	141
6.2.4	Bukti Tidak Plagiat	142
6.3	Nurul Izza Hamka - 1174062	142
6.3.1	Teori	142
6.3.2	Praktek Program	145

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 Intership.

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

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 penghantar 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

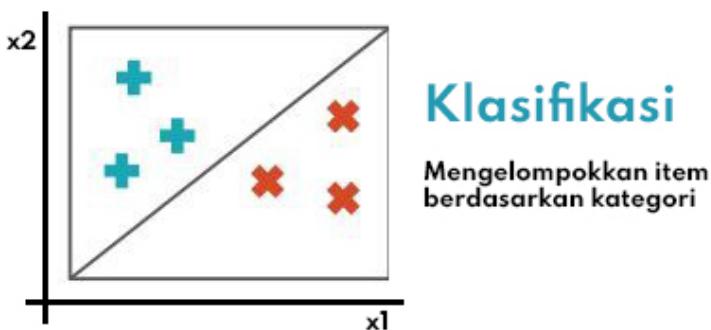
CHAPTER 1

BAB 2

CHAPTER 2

2.1 Arrizak Furqona Gifary (1174070)

2.1.1 Teori



Gambar 2.1 Binary Classification

2.1.1.1 Binary Classification Binary Classification (Klasifikasi Biner) adalah sebuah tugas yang mengklasifikasikan elemen-elemen dari himpunan yang diberikan ke dalam dua kelompok (memprediksi kelompok mana yang masing-masing dimiliki) berdasarkan aturan klasifikasi. Konteks yang membutuhkan keputusan apakah suatu item memiliki sifat kualitatif atau tidak, beberapa karakteristik tertentu, atau beberapa klasifikasi biner khas meliputi:

1. Tes medis

untuk menentukan apakah pasien memiliki penyakit tertentu atau tidak - properti klasifikasi adalah keberadaan penyakit.

2. Metode uji "lulus atau gagal"

Kontrol kualitas di pabrik, yaitu memutuskan apakah suatu spesifikasi telah atau belum terpenuhi - klasifikasi Go/no go.

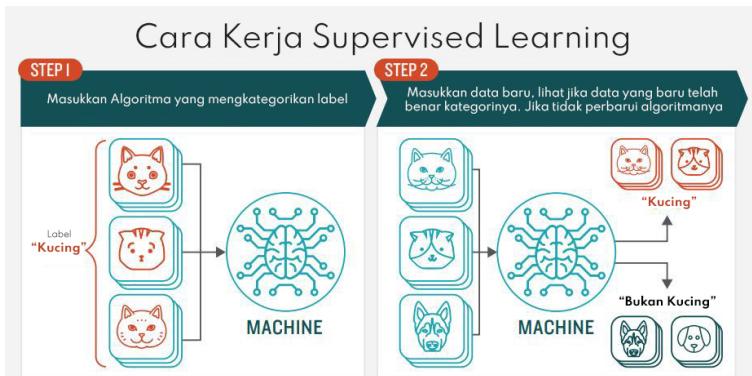
3. Pengambilan informasi

memutuskan apakah suatu halaman atau artikel harus ada dalam hasil pencarian atau tidak - properti klasifikasi adalah relevansi artikel.

2.1.1.2 Supervised Learning , Unsupervised Learning dan Clustering

1. Supervised Learning

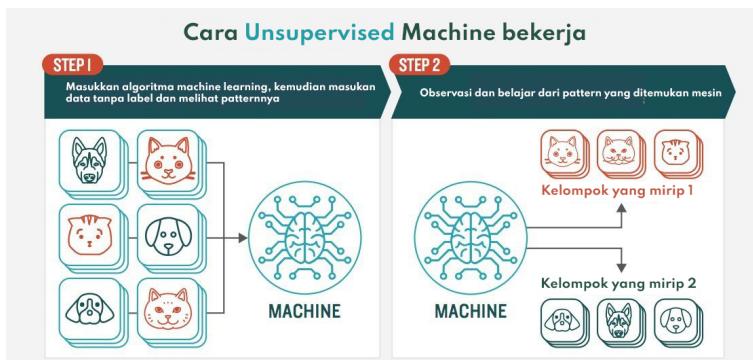
Supervised learning adalah suatu pembelajaran yang terawasi dimana jika output yang diharapkan telah diketahui sebelumnya. Biasanya pembelajaran ini dilakukan dengan menggunakan data yang telah ada. Dan Supervised Learning dalam bahasa indonesia adalah pembelajaran yang ada supervisornya. Maknudisdi disini ada supervisornya adalah label di tiap data nya. Label maksudnya adalah tag dari data yang ditambahkan dalam machine learning model. Contohnya gambar kucing di tag "kucing" di tiap masing masing image kucing dan gambar anjing di tag "anjing" di tiap masing gambar anjing.



Gambar 2.2 Supervised Learning

2. Unsupervised Learning

Unsupervised learning menggunakan kemiripan dari attribut yang dimiliki suatu item. Jika attribut dan sifat dari data yang diekstrak memiliki kemiripan, maka akan dikelompokkan (clustering). Sehingga hal ini akan menimbulkan kelompok (cluster). Jumlah cluster bisa unlimited. Dari kelompok-kelompok itu model dilabelkan, dan jika data baru mau di prediksi, maka akan dicocokkan dengan data kelompok yang mirip featurenya.



Gambar 2.3 Unsupervised Learning

3. Clustering

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



Gambar 2.4 Clustering

2.1.1.3 Evaluasi dan Akurasi Evaluasi adalah tentang bagaimana kita dapat mengevaluasi seberapa baik model bekerja dengan mengukur tingkat akurasinya. Dan akurasi akan didefinisikan sebagai persentase kasus yang diklasifikasikan dengan benar. Kita dapat menganalisis kesalahan yang dibuat oleh model, atau tingkat kebingungannya,

menggunakan matriks kebingungan(confusion matrix). Matriks kebingungan mengacu pada kebingungan dalam model,tetapi matriks kebingungan ini bisa menjadi sedikit sulit untuk dipahami ketika mereka menjadi sangat besar.

	Hasil Prediksi "Apel"	Hasil Prediksi "Jeruk"
True "Apel"	20	5
True "Jeruk"	3	22

Gambar 2.5 Evaluasi

2.1.1.4 Cara Membuat Confusion Matrix 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. Diantaranya 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

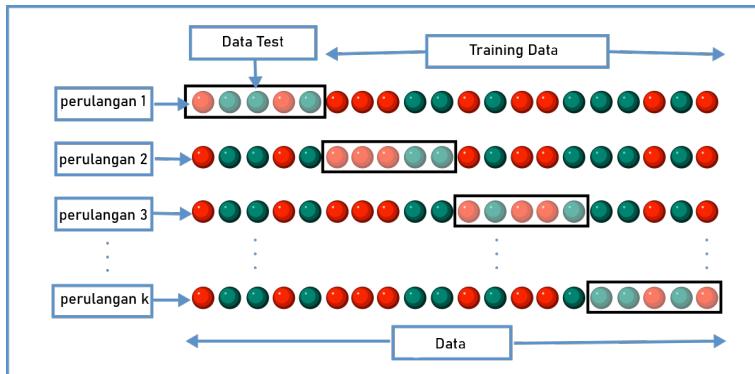
	Data Asli 1 (positif)	Data Asli 0 (negatif)
Prediksi 1 (positif)	True Positive (TP)	False Positive (FP)
Prediksi 0 (negatif)	False Negative (FN)	True Negative (TN)

Gambar 2.6 Contoh Confusion Matrix

2.1.1.5 Bagaimana K-fold cross validation bekerja

1. Total instance dibagi menjadi N bagian.
2. Fold yang pertama adalah bagian pertama menjadi data uji (testing data) dan sisanya menjadi training data.
3. Lalu hitung akurasi berdasarkan porsi data tersebut dengan menggunakan persamaan.
4. Fold yang kedua adalah bagian kedua, yang menjadi data uji(testing data)dan sisanya training data.
5. Kemudian hitung akurasi berdasarkan porsi data tersebut.
6. Dan seterusnya hingga habis mencapai fold ke-K.

7. Terakhir hitung rata-rata akurasi K buah.



Gambar 2.7 Contoh K-Fold Cross Validation

2.1.1.6 Apa itu Decision Tree Decision Tree adalah 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.



Gambar 2.8 Contoh Decision Tree membeli mobil

2.1.1.7 Information Gain dan Entropi

- Information Gain

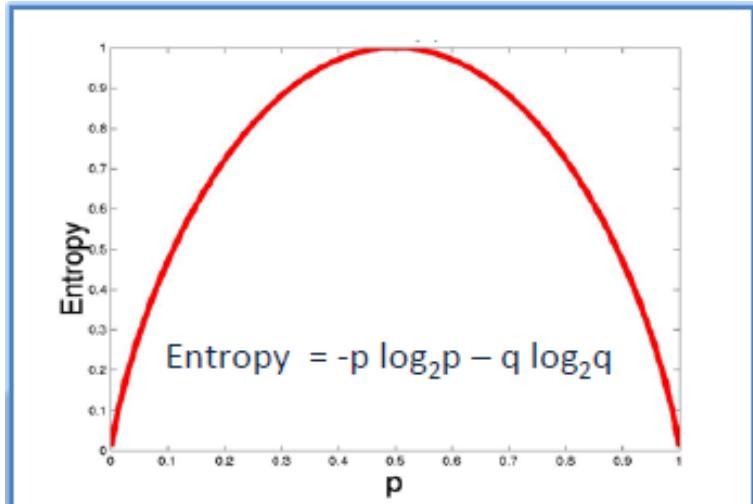
Information Gain merupakan total data yang didapat dari data - data acak yang data tersebut akan digunakan untuk analisis data lainnya. Information Gain ini digunakan pada decision tree sebagai label setiap aksi - aksi yang perlu dinilai validasinya.



Gambar 2.9 Information Gain

- Entropi

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.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Gambar 2.10 Entropi

2.1.2 Praktek

Tugas anda adalah, dataset ganti menggunakan student-mat.csv dan mengganti semua nama variabel dari kode di bawah ini dengan nama-nama makanan (NPM mod 3=0), kota (NPM mod 3=1), buah (NPM mod 3=2)

```
1 # In [0]
2 1174070 % 3 #Hasilnya 1 maka akan menggunakan nama Kota
```

2.1.2.1 Nomor 1

```
1 import pandas as pd #Import library pandas menggantinya nama yang
   akan dipanggil jadi pd
2 tokyo = pd.read_csv('dataset/student-mat.csv', sep=';') #Membuat
   variabel tokyo yang isinya memanggil fungsi membaca file csv
3 len(tokyo) #Menghitung jumlah data yang ada pada csv yang tadi sudah
   dibaca
```

The screenshot shows a Jupyter Notebook interface. On the left, a code cell contains Python code for importing a CSV file and creating a binary label column. A tooltip for the 'Cell' button is displayed, explaining its function: 'Pressing Cell in front of it, either on the Editor or on the toolbar, will run the selected cell. The result can also be shown automatically after writing via left parenthesis next to an object. You can activate this behavior in Preferences > Help.' Below the code cell, the output pane shows the command and its execution status.

```

1 # In [1]:
2 # coding: utf-8
3
4 # Created on Sun Mar  8 12:50:02 2018
5
6 # Author: Atik
7
8
9 # In [2]:
10 114866 8 $ download 3 min ago megaparser.html.htm
11
12 import pandas as pd
13 import numpy as np
14 import os
15 from sklearn import datasets
16 from sklearn.model_selection import train_test_split
17
18 # In [3]:
```

Gambar 2.11 Nomor 1

2.1.2.2 Nomor 2

```

1 # In[2]
2 # generate binary label (pass/fail) based on G1+G2+G3 (test grades,
3 # each 0–20 pts); threshold for passing is sum>=30
4 tokyo['pass'] = tokyo.apply(lambda row: 1 if (row['G1']+row['G2']+row
5 ['G3']) >= 35 else 0, axis=1) #Membuat label binary (pass/fail)
6 berdasarkan G1+G2+G3 (testgrade, semuanya 0–20 point); Batas
7 untuk pass adalah sum>=30
8 tokyo = tokyo.drop(['G1', 'G2', 'G3'], axis=1) #Meghilangkan data G1
9 G2 dan G3
10 tokyo.head() #Menampilkan data
```

The screenshot shows a Jupyter Notebook interface. On the left, a code cell contains Python code for generating a binary label column ('pass') based on the sum of three test grades ('G1', 'G2', 'G3'). A tooltip for the 'Cell' button is displayed, explaining its function: 'Pressing Cell in front of it, either on the Editor or on the toolbar, will run the selected cell. The result can also be shown automatically after writing via left parenthesis next to an object. You can activate this behavior in Preferences > Help.' Below the code cell, the output pane shows the command and its execution status.

```

# In[2]:
tokyo['pass'] = tokyo.apply(lambda row: 1 if (row['G1']+row['G2']+row['G3']) >= 35 else 0, axis=1)
tokyo.head()

# In[3]:
```

Gambar 2.12 Nomor 2

2.1.2.3 Nomor 3

```

1 # In[3]:
2 # use one-hot encoding on categorical columns
3 tokyo = pd.get_dummies(tokyo, columns=['sex', 'school', 'address',
4 'famsize', 'Pstatus', 'Mjob', 'Fjob',
5 'reason', 'guardian', 'schoolsup',
6 'famsup', 'paid', 'activities',
7 'nursery', 'higher', 'internet',
8 'romantic'])
9 tokyo.head()
```

```
# In [1]:  
# Import library  
import pandas as pd  
  
# Load dataset  
tokyo = pd.read_csv('dataset/tokyo.csv')  
tokyo.head()
```

In [1]: tokyo = pd.get_dummies(tokyo, columns=['sex', 'school', 'address', 'family', 'parent', 'grade', 'job', 'religion', 'nursery', 'higher', 'internet', 'romantic'])

Out[1]:

	sex	school	address	family	parent	grade	job	religion	nursery	higher	internet	romantic
0	Male	Fedu	... Internet_no	romantic_no	yes
1	Male	Fedu	... Internet_no	romantic_no	yes
2	Male	Fedu	... Internet_no	romantic_no	yes
3	Male	Fedu	... Internet_no	romantic_no	yes
4	Male	Fedu	... Internet_no	romantic_no	yes

[5 rows x 17 columns]

In [1]:

Gambar 2.13 Nomor 3

2.1.2.4 Nomor 4

```
1 # In [4]:  
2 # shuffle rows  
3 tokyo = tokyo.sample(frac=1) #Mengambil data sample dari tokyo  
4 # split training and testing data  
5 tokyo_train = tokyo[:500] #Membagi data untuk training  
tokyo_test = tokyo[500:] #Membagi data untuk test  
6  
7 tokyo_train_att = tokyo_train.drop(['pass'], axis=1) #Meghapus data  
yang telah pass dan memasukkannya  
tokyo_train_pass = tokyo_train['pass'] #Mengambil data yang pass saja  
10  
11 tokyo_test_att = tokyo_test.drop(['pass'], axis=1) #Meghapus data  
yang telah pass dan memasukkannya  
tokyo_test_pass = tokyo_test['pass'] #Mengambil data yang pass saja  
13  
14 tokyo_att = tokyo.drop(['pass'], axis=1) #Meghapus data yang telah  
pass dan memasukkannya  
tokyo_pass = tokyo['pass'] #Mengambil data yang pass saja  
16  
17 # number of passing students in whole dataset:  
18 import numpy as np #Mengimport library numpy sebagai np  
19 print("Passing: %d out of %d (%.2f%%)" % (np.sum(tokyo_pass), len(  
tokyo_pass), 100*float(np.sum(tokyo_pass)) / len(tokyo_pass))) #  
Menampilkan data
```

```
# In [2]:  
# Import library  
import pandas as pd  
  
# Load dataset  
tokyo = pd.read_csv('dataset/tokyo.csv')  
tokyo.head()
```

In [2]: tokyo = tokyo.sample(frac=1)

...
tokyo_train = tokyo[:500]
tokyo_test = tokyo[500:]

tokyo_train_att = tokyo_train.drop(['pass'], axis=1)
tokyo_train_pass = tokyo_train['pass']

tokyo_test_att = tokyo_test.drop(['pass'], axis=1)
tokyo_test_pass = tokyo_test['pass']

tokyo_att = tokyo.drop(['pass'], axis=1)
tokyo_pass = tokyo['pass']

...
number of passing students in whole dataset:
len(tokyo_pass), 100*float(np.sum(tokyo_pass)) / len(tokyo_pass))

In [2]:

ModuleNotFoundError: No module named 'graphviz'

In [2]:

In [2]: tokyo = tokyo.sample(frac=1)

...
tokyo_train = tokyo[:500]
tokyo_test = tokyo[500:]

tokyo_train_att = tokyo_train.drop(['pass'], axis=1)

tokyo_train_pass = tokyo_train['pass']

tokyo_test_att = tokyo_test.drop(['pass'], axis=1)

tokyo_test_pass = tokyo_test['pass']

tokyo_att = tokyo.drop(['pass'], axis=1)

tokyo_pass = tokyo['pass']

...
number of passing students in whole dataset:
len(tokyo_pass), 100*float(np.sum(tokyo_pass)) / len(tokyo_pass))

In [2]:

Passing: 166 out of 399 (42.00%)

In [2]:

Gambar 2.14 Nomor 4

2.1.2.5 Nomor 5

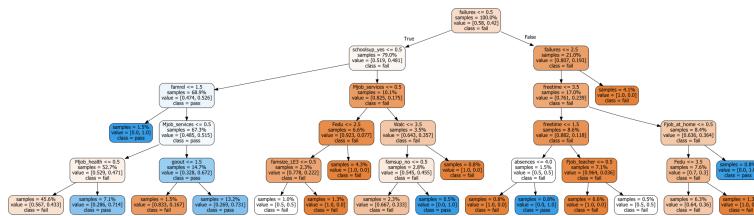
```
1 # In [5]:  
2 # fit a decision tree  
3 from sklearn import tree #import Decision tree dari library sklearn  
4 kyoto = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)  
#Membuat decision tree dengan maximal depthnya 5  
5 kyoto = kyoto.fit(tokyo_train_att, tokyo_train_pass) #Memasukkan data  
yang akan dijadikan decision treenya
```

```
In [24]: from sklearn import tree
... kyoto = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)
... kyoto = kyoto.fit(tokyo_train_att, tokyo_train_pass)
```

Gambar 2.15 Nomor 5

2.1.2.6 Nomor 6

```
1 # In [6]:  
2 # visualize tree  
3 import graphviz #Mengimport Library Graphviz untuk memvisualisasikan  
    decision tree  
4 dot_data = tree.export_graphviz(kyoto, out_file=None, label="all",  
    impurity=False, proportion=True,  
    feature_names=list(tokyo_train_att),  
    class_names=["fail", "pass"],  
    filled=True, rounded=True) #  
    Mendefinisikan dot_data yang isikan akan berisikan data yang akan  
    dijadikan gambar  
7 graph = graphviz.Source(dot_data) #Memasukkan data tadi menjadi  
    sebuah graph  
8 graph #Menampilkan graph menggunakan graphviz
```



Gambar 2.16 Nomor 6

2.1.2.7 Nomor 7

```
1 # In [7]:  
2 # save tree  
3 tree.export_graphviz(kyoto, out_file="student-performance.dot", label=  
4     ="all", impurity=False, proportion=True,  
5     feature_names=list(tokyo_train_att), class_names=  
6     =["fail", "pass"],  
7     filled=True, rounded=True) #Digunakan untuk  
mengexport graph tree tadi yang telah kita buat
```

```
In [32]: tree.export_graphviz(kyoto, out_file="student-performance.dot", label="all", impurity=False, proportion=True, feature_names=list(kyoto_train_att), class_names=['fail', 'pass'], filled=True, rounded=True) #digunakan untuk mengexporth graph tree tadi yang telah kita buat
```

Gambar 2.17 Nomor 7

2.1.2.8 Nomor 8

```

1 # In [8]:
2 kyoto.score(tokyo_test_att, tokyo_test_pass) #Menghitung prediksi
    nilai yang akan datang dimasa depan

```

In [60]: kyoto.score(tokyo_test_att, tokyo_test_pass)
Out[60]: 0.6778523489932886

Gambar 2.18 Nomor 8**2.1.2.9 Nomor 9**

```

1 # In [9]:
2 from sklearn.model_selection import cross_val_score #Mengimport
    fungsi cross_val_score dari library sklearn
3 nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5) #
    Mendefinisikan nagoya yang isinya pembagian data menjadi 5
4 # show average score and +/- two standard deviations away (covering
    95% of scores)
5 print("Accuracy: %0.2f (+/- %0.2f)" % (nagoya.mean(), nagoya.std() *
    2)) #Menampilkan data nilai dan +/- dari dua standar deviasi

```

In [69]: from sklearn.model_selection import cross_val_score
...: nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5)
...: # show average score and +/- two standard deviations away (covering 95% of scores)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (nagoya.mean(), nagoya.std() * 2))
Accuracy: 0.55 (+/- 0.10)

Gambar 2.19 Nomor 9**2.1.2.10 Nomor 10**

```

1 # In [10]:
2 for max_depth in range(1, 20): #Pengulangan menunjukkan seberapa
    dalam tree itu
3     kyoto = tree.DecisionTreeClassifier(criterion="entropy",
        max_depth=max_depth) #Membuat decision Tree
4     nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5) #
        Mendefinisikan nagoya yang isinya pembagian data menjadi 5
5     print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (max_depth,
        nagoya.mean(), nagoya.std() * 2)) #Menampilkan data nilai dan +/- dari dua standar deviasi

```

```
In [70]: for max_depth in range(1, 20):
...:     kyoto = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
...:     nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5)
...:     print("Max depth: %d, Accuracy: %.2f (+/- %.2f)" % (max_depth, nagoya.mean(), nagoya.std() * 2))
Max depth: 1, Accuracy: 0.56 (+/- 0.01)
Max depth: 2, Accuracy: 0.56 (+/- 0.08)
Max depth: 3, Accuracy: 0.56 (+/- 0.12)
Max depth: 4, Accuracy: 0.57 (+/- 0.08)
Max depth: 5, Accuracy: 0.55 (+/- 0.09)
Max depth: 6, Accuracy: 0.57 (+/- 0.18)
Max depth: 7, Accuracy: 0.57 (+/- 0.15)
Max depth: 8, Accuracy: 0.54 (+/- 0.18)
Max depth: 9, Accuracy: 0.54 (+/- 0.11)
Max depth: 10, Accuracy: 0.58 (+/- 0.11)
Max depth: 11, Accuracy: 0.56 (+/- 0.10)
Max depth: 12, Accuracy: 0.60 (+/- 0.08)
Max depth: 13, Accuracy: 0.59 (+/- 0.05)
Max depth: 14, Accuracy: 0.58 (+/- 0.05)
Max depth: 15, Accuracy: 0.57 (+/- 0.07)
Max depth: 16, Accuracy: 0.58 (+/- 0.06)
Max depth: 17, Accuracy: 0.60 (+/- 0.11)
Max depth: 18, Accuracy: 0.58 (+/- 0.08)
Max depth: 19, Accuracy: 0.58 (+/- 0.06)
```

Gambar 2.20 Nomor 10**2.1.2.11 Nomor 11**

```
1 # In[11]:
2 depth_acc = np.empty((19,3), float) #Membuat array baru
3 i = 0 #Membuat variable berisikan 0
4 for max_depth in range(1, 20): #Perulangan untuk memasukkan data
5     kyoto = tree.DecisionTreeClassifier(criterion="entropy",
6     max_depth=max_depth)#Membuat decision Tree
7     nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5) #
8     depth_acc[i,0] = max_depth #Memasukkan data max_depth ke array
9     depth_acc
10    depth_acc[i,1] = nagoya.mean() #Memasukkan data rata-rata dari
11    nagoya ke array depth_acc
12    depth_acc[i,2] = nagoya.std() * 2 #Memasukkan data akar 2 dari
13    nagoya ke array depth_acc
14    i += 1
15
16 depth_acc
```

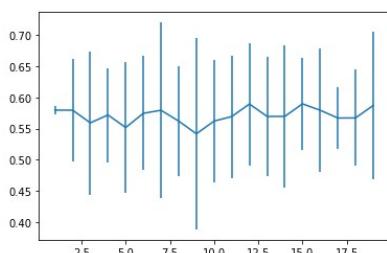
```
In [71]: depth_acc = np.empty((19,3), float)
...: i = 0
...: for max_depth in range(1, 20):
...:     kyoto = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
...:     nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5)
...:     depth_acc[i,0] = max_depth
...:     depth_acc[i,1] = nagoya.mean()
...:     depth_acc[i,2] = nagoya.std() * 2
...:     i += 1
...:
...:
...: depth_acc
Out[71]:
array([[1.00000000e+00, 5.79751704e-01, 6.30768599e-03],
       [2.00000000e+00, 5.79654333e-01, 8.25005697e-02],
       [3.00000000e+00, 5.54241318e-01, 1.21808510e-01],
       [4.00000000e+00, 5.71964460e-01, 8.72318860e-02],
       [5.00000000e+00, 5.46678027e-01, 9.66616528e-02],
       [6.00000000e+00, 5.69561819e-01, 1.08113451e-01],
       [7.00000000e+00, 5.67030996e-01, 1.40406275e-01],
       [8.00000000e+00, 5.47030996e-01, 1.22447311e-01],
       [9.00000000e+00, 5.51966082e-01, 6.6884125e-02],
       [1.00000000e+01, 5.87314184e-01, 7.12478298e-02],
       [1.10000000e+01, 5.77124310e-01, 7.61119153e-02],
       [1.20000000e+01, 5.89719247e-01, 4.34452239e-02],
       [1.30000000e+01, 5.97569783e-01, 3.87484760e-02],
       [1.40000000e+01, 5.77026939e-01, 7.68881184e-02],
       [1.50000000e+01, 5.97347452e-01, 5.69404888e-02],
       [1.60000000e+01, 5.74720058e-01, 1.29000478e-01],
       [1.70000000e+01, 5.82282538e-01, 5.71762018e-02],
       [1.80000000e+01, 5.74720870e-01, 3.56163418e-02],
       [1.90000000e+01, 5.77250893e-01, 8.75345835e-02]])
```

Gambar 2.21 Nomor 11

2.1.2.12 Nomor 12

```
1 # In[12]:
2 import matplotlib.pyplot as plt #Menimport fungsi pyplot dari library
   #matplotlib sebagai plt
3 fig, ax = plt.subplots() #Membuat plot baru
4 ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2]) #
   Mengisikan data plot
5 plt.show() #Menampilkan plot
```

```
In [73]: import matplotlib.pyplot as plt #Menimport fungsi pyplot dari library matplotlib sebagai plt
...: fig, ax = plt.subplots() #Membuat plot baru
...: ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2]) #Mengisikan data plot
...: plt.show() #Menampilkan plot
```



Gambar 2.22 Nomor 12

2.1.3 Penanganan Error

2.1.3.1 Error

1. ModuleNotFoundError

Traceback (most recent call last):

```
File "<ipython-input-25-af85b140ad99>", line 1, in <module>
    import graphviz
```

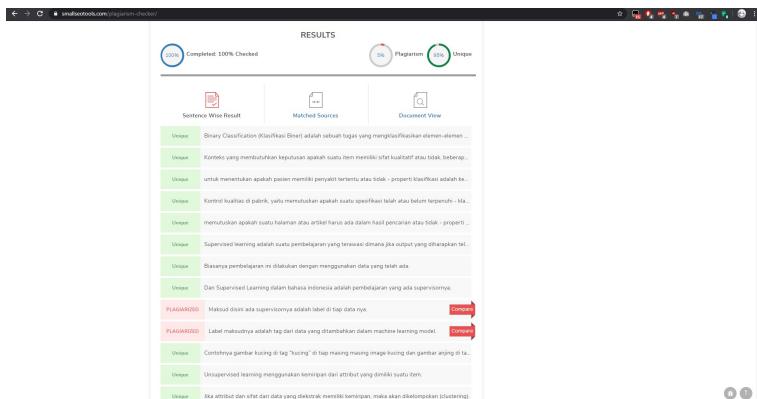
```
ModuleNotFoundError: No module named 'graphviz'
```

Gambar 2.23 ModuleNotFoundError

2.1.3.2 Solusi

1. Intall library Graphviz dengan cara download graphviz di google setelah instalasi buka anaconda prompt sebagai admin lalu mengetikkan `conda install graphviz`

2.1.4 Bukti Tidak Plagiat



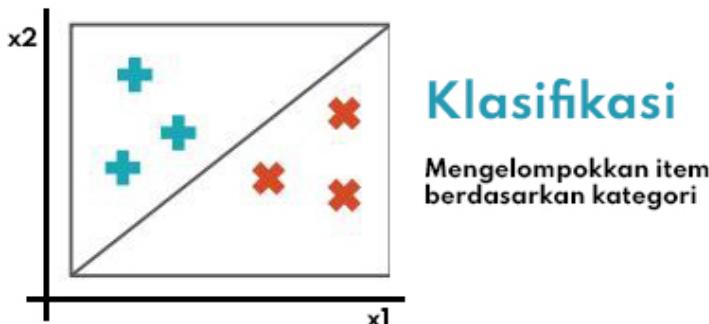
Gambar 2.24 Bukti Tidak Plagiat

2.1.5 Link Youtube:

https://youtu.be/_rB-Z2xMMdk

2.2 Mochamad Arifqi Ramadhan (1174074)

2.2.1 Teori



Gambar 2.25 Binary Classification

2.2.1.1 *Binary Classification* Binary Classification (Klasifikasi Biner) adalah sebuah tugas yang mengklasifikasikan elemen-elemen dari himpunan yang diberikan ke dalam dua kelompok (memprediksi kelompok mana yang masing-masing dimiliki) berdasarkan aturan klasifikasi. Konteks yang membutuhkan keputusan apakah suatu item memiliki sifat kualitatif atau tidak, beberapa karakteristik tertentu, atau beberapa klasifikasi biner khas meliputi:

1. Tes medis

untuk menentukan apakah pasien memiliki penyakit tertentu atau tidak - properti klasifikasi adalah keberadaan penyakit.

2. Metode uji "lulus atau gagal"

Kontrol kualitas di pabrik, yaitu memutuskan apakah suatu spesifikasi telah atau belum terpenuhi - klasifikasi Go/no go.

3. Pengambilan informasi

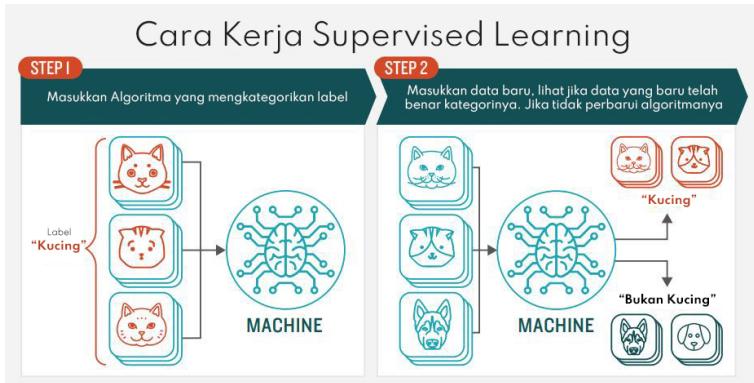
memutuskan apakah suatu halaman atau artikel harus ada dalam hasil pencarian atau tidak - properti klasifikasi adalah relevansi artikel.

2.2.1.2 *Supervised Learning , Unsupervised Learning dan Clustering*

1. Supervised Learning

Supervised learning adalah suatu pembelajaran yang terawasi dimana jika output yang diharapkan telah diketahui sebelumnya. Biasanya pembelajaran ini dilakukan dengan menggunakan data yang telah ada. Dan Supervised Learning

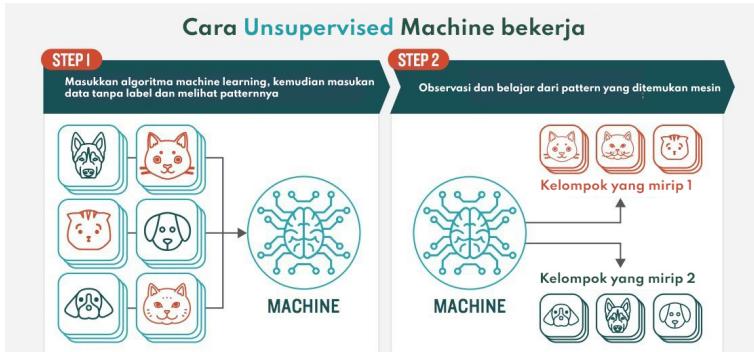
dalam bahasa indonesia adalah pembelajaran yang ada supervisornya. Mak-sud disini ada supervisornya adalah label di tiap data nya. Label maksudnya adalah tag dari data yang ditambahkan dalam machine learning model. Contohnya gambar kucing di tag “kucing” di tiap masing masing image kucing dan gambar anjing di tag “anjing” di tiap masing gambar anjing.



Gambar 2.26 Supervised Learning

2. Unsupervised Learning

Unsupervised learning menggunakan kemiripan dari attribut yang dimiliki suatu item. Jika attribut dan sifat dari data yang diekstrak memiliki kemiripan, maka akan dikelompokan (clustering). Sehingga hal ini akan menimbulkan kelompok (cluster). Jumlah cluster bisa unlimited. Dari kelompok-kelompok itu model dilabelkan, dan jika data baru mau di prediksi, maka akan dicocokkan dengan data kelompok yang mirip featurenya.



Gambar 2.27 Unsupervised Learning

3. Clustering

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



Gambar 2.28 Clustering

2.2.1.3 Evaluasi dan Akurasi Evaluasi adalah tentang bagaimana kita dapat mengevaluasi seberapa baik model bekerja dengan mengukur tingkat akurasinya. Dan akurasi akan didefinisikan sebagai persentase kasus yang diklasifikasikan dengan benar. Kita dapat menganalisis kesalahan yang dibuat oleh model, atau tingkat kebingungannya, menggunakan matriks kebingungan(confusion matrix). Matriks kebingungan mengacu pada kebingungan dalam model, tetapi matriks kebingungan ini bisa menjadi sedikit sulit untuk dipahami ketika mereka menjadi sangat besar.

	Hasil Prediksi "Apel"	Hasil Prediksi "Jeruk"
True "Apel"	20	5
True "Jeruk"	3	22

Gambar 2.29 Evaluasi

2.2.1.4 Cara Membuat Confusion Matrix 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. Diantaranya adalah:

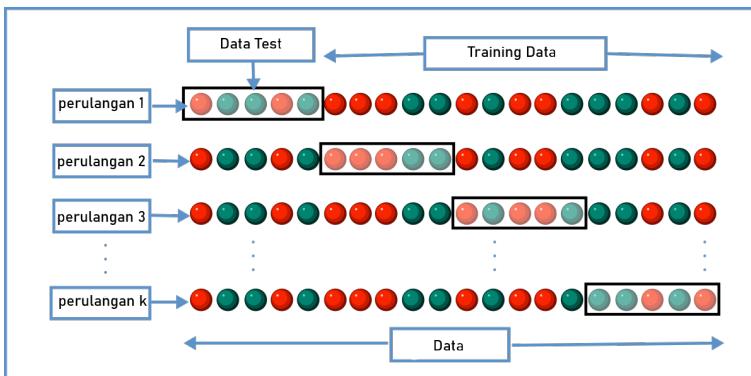
- True Positive: Data positif yang terdeteksi memiliki hasil benar
- False Positive: Data Positif yang terdeteksi memiliki hasil salah
- True Negative: Data negatif yang terdeteksi memiliki hasil benar
- False Negative: Data negatif yang terdeteksi memiliki hasil salah

	Data Asli 1 (positif)	Data Asli 0 (negatif)
Prediksi 1 (positif)	True Positive (TP)	False Positive (FP)
Prediksi 0 (negatif)	False Negative (FN)	True Negative (TN)

Gambar 2.30 Contoh Confusion Matrix

2.2.1.5 Bagaimana K-fold cross validation bekerja

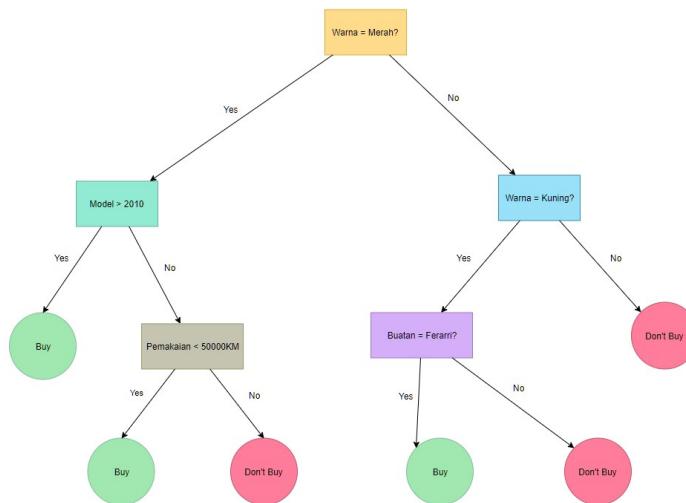
1. Total instance dibagi menjadi N bagian.
2. Fold yang pertama adalah bagian pertama menjadi data uji (testing data) dan sisanya menjadi training data.
3. Lalu hitung akurasi berdasarkan porsi data tersebut dengan menggunakan persamaan.
4. Fold yang kedua adalah bagian kedua, yang menjadi data uji(testing data) dan sisanya training data.
5. Kemudian hitung akurasi berdasarkan porsi data tersebut.
6. Dan seterusnya hingga habis mencapai fold ke-K.
7. Terakhir hitung rata-rata akurasi K buah.



Gambar 2.31 Contoh K-Fold Cross Validation

2.2.1.6 Apa itu Decision Tree

Decision Tree adalah 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.



Gambar 2.32 Contoh Decision Tree membeli mobil

2.2.1.7 *Information Gain dan Entropi*

- **Information Gain**

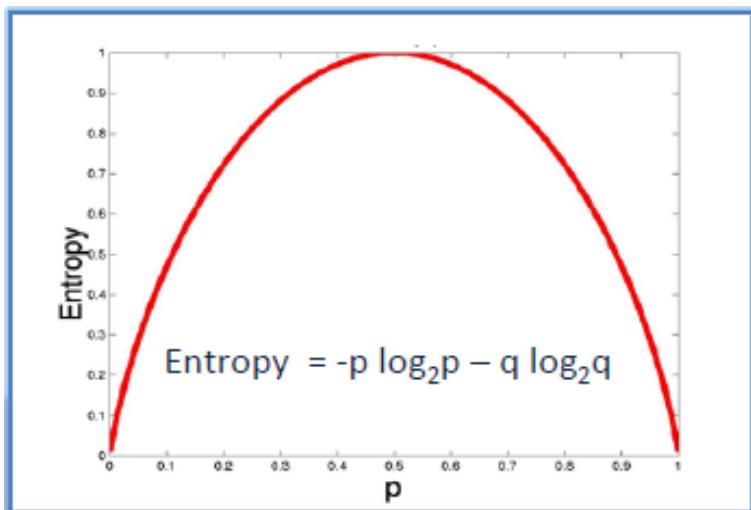
Information Gain merupakan total data yang didapat dari data - data acak yang data tersebut akan digunakan untuk analisis data lainnya. Information Gain ini digunakan pada decision tree sebagai label setiap aksi - aksi yang perlu dinilai validasinya.



Gambar 2.33 Information Gain

- **Entropi**

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.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Gambar 2.34 Entropi

2.2.2 Praktek

Tugas anda adalah, dataset ganti menggunakan student-mat.csv dan mengganti semua nama variabel dari kode di bawah ini dengan nama-nama makanan (NPM mod 3=0), kota (NPM mod 3=1), buah (NPM mod 3=2)

```
1 # In [0]
2 1174066 % 3 #Hasilnya 1 maka akan menggunakan nama Kota
```

2.2.2.1 Nomor 1

```
1 import pandas as pd #Import library pandas menggantinya nama yang
   akan dipanggil jadi pd
2 tokyo = pd.read_csv('dataset/student-mat.csv', sep=';') #Membuat
   variable tokyo yang isinya memanggil fungsi membaca file csv
3 len(tokyo) #Menghitung jumlah data yang ada pada csv yang tadi sudah
   dibaca
```

Gambar 2.35 Nomor 1

2.2.2.2 Nomor 2

```
1 # In[2]
2 # generate binary label (pass/fail) based on G1+G2+G3 (test grades ,
3 # each 0–20 pts); threshold for passing is sum>=30
4 tokyo[ 'pass' ] = tokyo .apply(lambda row: 1 if (row[ 'G1' ]+row[ 'G2' ]+row[ 'G3' ]) >= 35 else 0, axis=1) #Membuat label binary (pass/fail)
5 # berdasarkan G1+G2+G3 (testgrade , semuanya 0–20 point); Batas
6 # untuk pass adalah sum>=30
7 tokyo = tokyo .drop([ 'G1' , 'G2' , 'G3' ], axis=1) #Meghilangkan data G1
8 G2 dan G3
9 tokyo .head() #Menampilkan data
```

```
[4]: token = token.apply(lambda row: 1 if (row['U1'][row['U2'][row['U3'][row['U4']]]) == 35 else 0, axis=1)
token.head()

[5]: token.drop(['U1', 'U2', 'U3', 'U4'], axis=1)

[6]: token.head()
```

Variable explorer: PR editor: Help

Python console: 1M

```
In [4]: import pandas as pd
import numpy as np
from sklearn import datasets
dataset = datasets.load_iris()
X = dataset.data
y = dataset.target
X = pd.DataFrame(X, columns = dataset.feature_names)
y = pd.Series(y, name = 'target')

Out[4]: 395

In [5]: token['U1'] = d.apply(lambda row: 1 if (row['U2'][row['U3'][row['U4']]]) == 35 else 0, axis=1)
token['U2'] = d.apply(lambda row: 1 if (row['U1'][row['U3'][row['U4']]]) == 35 else 0, axis=1)
token['U3'] = d.apply(lambda row: 1 if (row['U1'][row['U2'][row['U4']]]) == 35 else 0, axis=1)
token['U4'] = d.apply(lambda row: 1 if (row['U1'][row['U2'][row['U3']]]) == 35 else 0, axis=1)

Tokenized data frame (cell limit): 1000000

File "C:\Users\reut\PycharmProjects\Iris\iris.ipynb", line 1, in <module>
    token['U1'] = d.apply(lambda row: 1 if (row['U2'][row['U3'][row['U4']]]) == 35 else 0, axis=1)

NameError: name 'd' is not defined

In [6]: token = token.drop(['U1', 'U2', 'U3', 'U4'], axis=1)

Out[6]: 395
```

Gambar 2.36 Nomor 2

2.2.2.3 Nomor 3

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

```
# In [1]:
# Import library
import pandas as pd

# Read CSV file
tokyo = pd.read_csv('tokyo.csv', columns=['sex', 'school', 'address', 'families', 'parents', 'age', 'job', 'relatives', 'nursery', 'higher', 'internet', 'romantic'])

# Print head
tokyo.head()
```

In [1]: tokyo = pd.read_csv('tokyo.csv', columns=['sex', 'school', 'address', 'families', 'parents', 'age', 'job', 'relatives', 'nursery', 'higher', 'internet', 'romantic'])
tokyo.head()
Out[1]:

	sex	school	address	families	parents	age	job	relatives	nursery	higher	internet	romantic
0	M	Publ	Publ	1	1	1	1	1	1	1	1	1
1	L	Publ	Publ	1	1	1	1	1	1	1	1	1
2	M	Publ	Publ	1	1	1	1	1	1	1	1	1
3	M	Publ	Publ	1	1	1	1	1	1	1	1	1
4	M	Publ	Publ	1	1	1	1	1	1	1	1	1

[5 rows x 13 columns]

In [1]:

Gambar 2.37 Nomor 3

2.2.2.4 Nomor 4

```
1 # In [4]:
2 # shuffle rows
3 tokyo = tokyo.sample(frac=1) #Mengambil data sample dari tokyo
4 # split training and testing data
5 tokyo_train = tokyo[:500] #Membagi data untuk training
6 tokyo_test = tokyo[500:] #Membagi data untuk test
7
8 tokyo_train_att = tokyo_train.drop(['pass'], axis=1) #Meghapus data yang telah pass dan memasukkannya
9 tokyo_train_pass = tokyo_train['pass'] #Mengambil data yang pass saja
10
11 tokyo_test_att = tokyo_test.drop(['pass'], axis=1) #Meghapus data yang telah pass dan memasukkannya
12 tokyo_test_pass = tokyo_test['pass'] #Mengambil data yang pass saja
13
14 tokyo_att = tokyo.drop(['pass'], axis=1) #Meghapus data yang telah pass dan memasukkannya
15 tokyo_pass = tokyo['pass'] #Mengambil data yang pass saja
16
17 # number of passing students in whole dataset:
18 import numpy as np #Mengimport library numpy sebagai np
19 print("Passing: %d out of %d (%.2f%%)" % (np.sum(tokyo_pass), len(tokyo_pass), 100*float(np.sum(tokyo_pass)) / len(tokyo_pass))) # Menampilkan data
```

```
# In [2]:
# Import library
import pandas as pd

# Read CSV file
tokyo = pd.read_csv('tokyo.csv', columns=['sex', 'school', 'address', 'families', 'parents', 'age', 'job', 'relatives', 'nursery', 'higher', 'internet', 'romantic'])

# Print head
tokyo.head()

# Split training and testing data
tokyo_train = tokyo[:500]
tokyo_test = tokyo[500:]

# Drop 'pass' column from training and testing datasets
tokyo_train_att = tokyo_train.drop(['pass'], axis=1)
tokyo_train_pass = tokyo_train['pass']

tokyo_test_att = tokyo_test.drop(['pass'], axis=1)
tokyo_test_pass = tokyo_test['pass']

# Drop 'pass' column from whole dataset
tokyo_att = tokyo.drop(['pass'], axis=1)
tokyo_pass = tokyo['pass']

# Number of passing students in whole dataset
len(tokyo_pass)

# Print Passing: 166 out of 300 (55.33%)
print("Passing: %d out of %d (%.2f%%)" % (np.sum(tokyo_pass), len(tokyo_pass), 100*float(np.sum(tokyo_pass)) / len(tokyo_pass)))
```

In [2]: No module named 'graphviz'
tokyo = pd.read_csv('tokyo.csv', columns=['sex', 'school', 'address', 'families', 'parents', 'age', 'job', 'relatives', 'nursery', 'higher', 'internet', 'romantic'])
tokyo.head()
tokyo_train = tokyo[:500]
tokyo_test = tokyo[500:]
tokyo_train_att = tokyo_train.drop(['pass'], axis=1)
tokyo_train_pass = tokyo_train['pass']
tokyo_test_att = tokyo_test.drop(['pass'], axis=1)
tokyo_test_pass = tokyo_test['pass']
tokyo_att = tokyo.drop(['pass'], axis=1)
tokyo_pass = tokyo['pass']
len(tokyo_pass)
Passing: 166 out of 300 (55.33%)
In [2]:

Gambar 2.38 Nomor 4

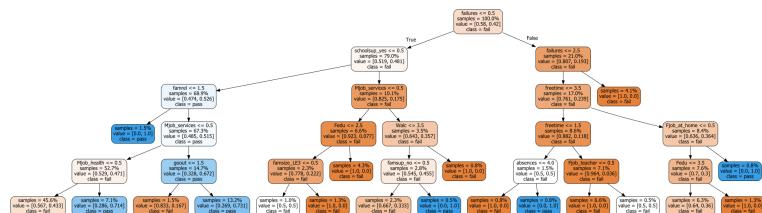
2.2.2.5 Nomor 5

```
1 # In [5]:
2 # fit a decision tree
3 from sklearn import tree #import Decision tree dari library sklearn
4 kyoto = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)
5 #Membuat decision tree dengan maximal depthnya 5
6 kyoto = kyoto.fit(tokyo_train_att, tokyo_train_pass) #Memasukkan data yang akan dijadikan decision treenya
```

```
In [24]: from sklearn import tree
...: kyoto = tree.DecisionTreeClassifier(criterion="entropy", max_depth=5)
...: kyoto = kyoto.fit(tokyo_train_att, tokyo_train_pass)
```

Gambar 2.39 Nomor 5**2.2.2.6 Nomor 6**

```
1 # In [6]:
2 # visualize tree
3 import graphviz #Mengimport Library Graphviz untuk memvisualisasikan
4 decision tree
5 dot_data = tree.export_graphviz(kyoto, out_file=None, label="all",
6     impurity=False, proportion=True,
7     feature_names=list(tokyo_train_att),
8     class_names=["fail", "pass"],
9     filled=True, rounded=True) #
10 Mendefinisikan dot_data yang isikan akan berisikan data yang akan
11 dijadikan gambar
12 graph = graphviz.Source(dot_data) #Memasukkan data tadi menjadi
13 sebuah graph
14 graph #Menampilkan graph menggunakan graphviz
```

**Gambar 2.40** Nomor 6**2.2.2.7 Nomor 7**

```
1 kyoto.score(tokyo_test_att, tokyo_test_pass) #Menghitung prediksi
2 nilai yang akan datang dimasa depan
```

```
In [32]: tree.export_graphviz(kyoto, out_file="student-performance.dot", label="all", impurity=False, proportion=True,
...: feature_names=list(tokyo_train_att), class_names=["fail", "pass"],
...: filled=True, rounded=True) #Digunakan untuk mengekspor graph tree tadi yang telah kita buat
```

Gambar 2.41 Nomor 7**2.2.2.8 Nomor 8**

```
1 # In [8]:
2 kyoto.score(tokyo_test_att, tokyo_test_pass) #Menghitung prediksi
3 nilai yang akan datang dimasa depan
```

```
In [60]: kyoto.score(tokyo_test_att, tokyo_test_pass)
Out[60]: 0.6778523489932886
```

Gambar 2.42 Nomor 8**2.2.2.9 Nomor 9**

```
1 # In [9]:
2 from sklearn.model_selection import cross_val_score #Mengimport
   fungsi cross_val_score dari library sklearn
3 nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5) #
   Mendefinisikan nagoya yang isinya pembagian data menjadi 5
4 # show average score and +/- two standard deviations away (covering
   95% of scores)
5 print("Accuracy: %0.2f (+/- %0.2f)" % (nagoya.mean(), nagoya.std() * 2)) #Menampilkan data nilai dan +/- dari dua standar deviasi
```

```
In [69]: from sklearn.model_selection import cross_val_score
...: nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5)
...: # show average score and +/- two standard deviations away (covering 95% of scores)
...: print("Accuracy: %0.2f (+/- %0.2f)" % (nagoya.mean(), nagoya.std() * 2))
Accuracy: 0.55 (+/- 0.10)
```

Gambar 2.43 Nomor 9**2.2.2.10 Nomor 10**

```
1 # In [10]:
2 for max_depth in range(1, 20): #Pengulangan menunjukkan seberapa
   dalam tree itu
3   kyoto = tree.DecisionTreeClassifier(criterion="entropy",
   max_depth=max_depth) #Membuat decision Tree
4   nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5) #
   Mendefinisikan nagoya yang isinya pembagian data menjadi 5
5   print("Max depth: %d, Accuracy: %0.2f (+/- %0.2f)" % (max_depth,
   nagoya.mean(), nagoya.std() * 2)) #Menampilkan data nilai dan +/- dari dua standar deviasi
```

```
In [70]: for max_depth in range(1, 20):
...:     kyoto = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
...:     nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5)
...:     print("Max depth: %d, Accuracy: %.2f (+/- %.2f)" % (max_depth, nagoya.mean(), nagoya.std() * 2))
Max depth: 1, Accuracy: 0.56 (+/- 0.01)
Max depth: 2, Accuracy: 0.56 (+/- 0.08)
Max depth: 3, Accuracy: 0.56 (+/- 0.12)
Max depth: 4, Accuracy: 0.57 (+/- 0.08)
Max depth: 5, Accuracy: 0.55 (+/- 0.09)
Max depth: 6, Accuracy: 0.57 (+/- 0.18)
Max depth: 7, Accuracy: 0.57 (+/- 0.15)
Max depth: 8, Accuracy: 0.54 (+/- 0.18)
Max depth: 9, Accuracy: 0.54 (+/- 0.11)
Max depth: 10, Accuracy: 0.58 (+/- 0.11)
Max depth: 11, Accuracy: 0.56 (+/- 0.10)
Max depth: 12, Accuracy: 0.60 (+/- 0.08)
Max depth: 13, Accuracy: 0.59 (+/- 0.05)
Max depth: 14, Accuracy: 0.58 (+/- 0.05)
Max depth: 15, Accuracy: 0.57 (+/- 0.07)
Max depth: 16, Accuracy: 0.58 (+/- 0.06)
Max depth: 17, Accuracy: 0.60 (+/- 0.11)
Max depth: 18, Accuracy: 0.58 (+/- 0.08)
Max depth: 19, Accuracy: 0.58 (+/- 0.06)
```

Gambar 2.44 Nomor 10

2.2.2.11 Nomor 11

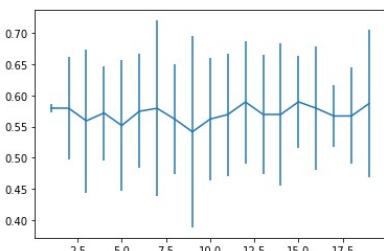
```
1 # In[11]:
2 depth_acc = np.empty((19,3), float) #Membuat array baru
3 i = 0 #Membuat variable berisikan 0
4 for max_depth in range(1, 20): #Perulangan untuk memasukkan data
5     kyoto = tree.DecisionTreeClassifier(criterion="entropy",
6         max_depth=max_depth)#Membuat decision Tree
7     nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5) #
8     depth_acc[i,0] = max_depth #Memasukkan data max_depth ke array
9     depth_acc
10    depth_acc[i,1] = nagoya.mean() #Memasukkan data rata-rata dari
11    nagoya ke array depth_acc
12    depth_acc[i,2] = nagoya.std() * 2 #Memasukkan data akar 2 dari
13    nagoya ke array depth_acc
14    i += 1
15
16 depth_acc
```

```
In [71]: depth_acc = np.empty((19,3), float)
...: i = 0
...: for max_depth in range(1, 20):
...:     kyoto = tree.DecisionTreeClassifier(criterion="entropy", max_depth=max_depth)
...:     nagoya = cross_val_score(kyoto, tokyo_att, tokyo_pass, cv=5)
...:     depth_acc[i,0] = max_depth
...:     depth_acc[i,1] = nagoya.mean()
...:     depth_acc[i,2] = nagoya.std() * 2
...:     i += 1
...:
...:
...: depth_acc
Out[71]:
array([[1.00000000e+00, 5.79751704e-01, 6.30768599e-03],
       [2.00000000e+00, 5.79654333e-01, 8.25005697e-02],
       [3.00000000e+00, 5.54241318e-01, 1.21808510e-01],
       [4.00000000e+00, 5.71964460e-01, 8.72318860e-02],
       [5.00000000e+00, 5.46678027e-01, 9.66616528e-02],
       [6.00000000e+00, 5.69561819e-01, 1.08113451e-01],
       [7.00000000e+00, 5.67030996e-01, 1.40406275e-01],
       [8.00000000e+00, 5.47030996e-01, 1.22447311e-01],
       [9.00000000e+00, 5.51966082e-01, 6.68884125e-02],
       [1.00000000e+01, 5.87314184e-01, 7.12478298e-02],
       [1.10000000e+01, 5.77124310e-01, 7.61119153e-02],
       [1.20000000e+01, 5.89719247e-01, 4.34452239e-02],
       [1.30000000e+01, 5.97569783e-01, 3.87484760e-02],
       [1.40000000e+01, 5.77026939e-01, 7.68881184e-02],
       [1.50000000e+01, 5.97347452e-01, 5.69404888e-02],
       [1.60000000e+01, 5.74720058e-01, 1.29000478e-01],
       [1.70000000e+01, 5.82282538e-01, 5.71762018e-02],
       [1.80000000e+01, 5.74720870e-01, 3.56163418e-02],
       [1.90000000e+01, 5.77250893e-01, 8.75345835e-02]])
```

Gambar 2.45 Nomor 11**2.2.2.12 Nomor 12**

```
1 # In [12]:
2 import matplotlib.pyplot as plt #Menimport fungsi pyplot dari library
   #matplotlib sebagai plt
3 fig, ax = plt.subplots() #Membuat plot baru
4 ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2]) #
   Mengisikan data plot
5 plt.show() #Menampilkan plot
```

```
In [73]: import matplotlib.pyplot as plt #Menimport fungsi pyplot dari library matplotlib sebagai plt
...: fig, ax = plt.subplots() #Membuat plot baru
...: ax.errorbar(depth_acc[:,0], depth_acc[:,1], yerr=depth_acc[:,2]) #Mengisikan data plot
...: plt.show() #Menampilkan plot
```

**Gambar 2.46** Nomor 12

2.2.3 Penanganan Error

2.2.3.1 Error

1. ModuleNotFoundError

Traceback (most recent call last):

```
File "<ipython-input-25-af85b140ad99>", line 1, in <module>
    import graphviz
```

```
ModuleNotFoundError: No module named 'graphviz'
```

Gambar 2.47 ModuleNotFoundError

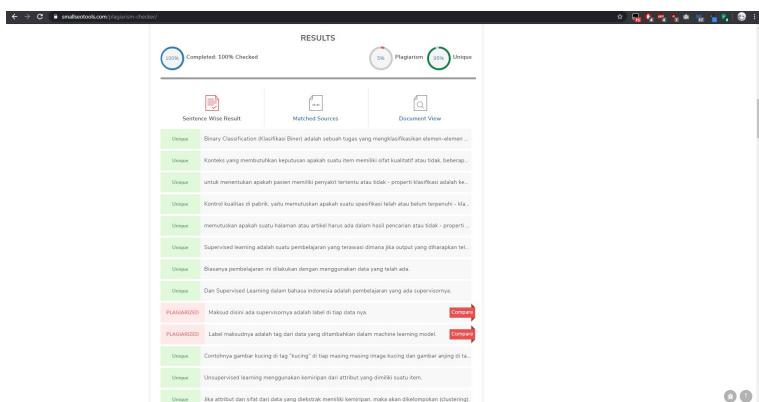
2.2.3.2 Solusi

1. Intall library Graphviz dengan cara download graphviz di google

setelah instalasi buka anaconda prompt sebagai admin lalu mengetikkan

```
conda install graphviz
```

2.2.4 Bukti Tidak Plagiat



Gambar 2.48 Bukti Tidak Plagiat

BAB 3

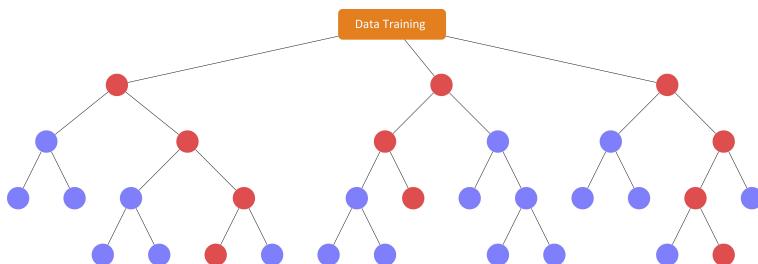
CHAPTER 3

3.1 Kaka Kamaludin (1174067)

3.1.1 Teori

3.1.1.1 Apa Itu Random Forest

Random Forest adalah sebuah 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.



Gambar 3.1 Random Forest

3.1.1.2 cara membaca dataset kasus 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 file bernama .txt dan didalamnya terdapat class dari field.
- Misalnya saja pada data jenis burung memiliki file index dan angka, dimana index berisi angka yang memiliki makna berupa jenis burung.
- bahkan nama burung sedangkan field 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 field yang di isikan berupa angka.
- Artinya angka 0 berarti tidak dan angka 1 berarti Ya.

3.1.1.3 Cross Validation

Cross Validation adalah sebuah teknik validasi model yang berfungsi untuk melakukan penilaian bagaimana hasil analisis statistik akan digeneralisasi ke data yang lebih independen. Cross validation digunakan dengan tujuan prediksi, dan bila kita ingin memperkirakan seberapa akurat model prediksi yang dilakukan dalam sebuah praktek. Tujuan dari cross validation yaitu untuk mendefinisikan dataset guna mengejut dalam fase pelatihan untuk membatasi masalah seperti overfitting dan underfitting serta mendapatkan wawasan tentang bagaimana model akan digeneralisasikan ke set data independen.

3.1.1.4 Arti score 44 % pada random forest, 27% pada decission tree dan 29 % dari SVM.

Dimana Score 44 % didapatkan 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 dicision 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.

3.1.1.5 Cara membaca confusion matriks dan contohnya memakai gambar atau ilustrasi sendiri.

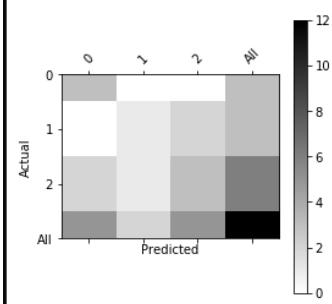
Perhitungan Confusion Matriks dapat dilakukan dengan cara dibawah ini.

- 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 Tue Mar 14 01:32:47 2020
4
5 @author: root
6 """
7 import numpy as np
8 import matplotlib.pyplot as plt
9 import pandas as pd
10 y_actu = pd.Series([2, 0, 2, 2, 0, 1, 1, 2, 2, 0, 1, 2], name='Actual')
11 y_pred = pd.Series([0, 0, 2, 1, 0, 2, 1, 0, 2, 0, 2, 2], name='Predicted')
12 df_confusion = pd.crosstab(y_actu, y_pred)
13 df_confusion = pd.crosstab(y_actu, y_pred, rownames=['Actual'],
14                           colnames=['Predicted'], margins=True)
14 def plot_confusion_matrix(df_confusion, title='Confusion matrix',
15                           cmap=plt.cm.gray_r):
15     plt.matshow(df_confusion, cmap=cmap) # imshow
16     #plt.title(title)
17     plt.colorbar()
18     tick_marks = np.arange(len(df_confusion.columns))
19     plt.xticks(tick_marks, df_confusion.columns, rotation=45)
20     plt.yticks(tick_marks, df_confusion.index)
21     #plt.tight_layout()
22     plt.ylabel(df_confusion.index.name)
23     plt.xlabel(df_confusion.columns.name)
24 plot_confusion_matrix(df_confusion)
25 plt.show()
```

```
In [1]: runfile('D:/OneDrive - Hybi.god/KULIAH/Semester 6/AI/KB3C/src/1174067/3/commatriks.py', wdir='D:/OneDrive - Hybi.god/KULIAH/Semester 6/AI/KB3C/src/1174067/3')
```



Gambar 3.2 Confusion Matriks

3.1.1.6 Apa itu voting pada random forest

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

Untuk menggunakan Voting pada Random Forest dapat dilihat code ini:

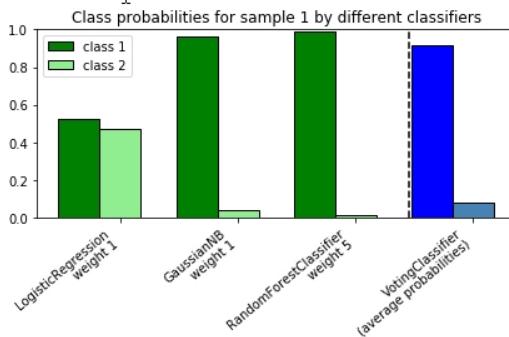
```
1 # --- coding: utf-8 ---
2 """
3 Created on Tue Mar 14 01:32:47 2020
4
5 @author: root
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10 from sklearn.linear_model import LogisticRegression
11 from sklearn.naive_bayes import GaussianNB
12 from sklearn.ensemble import RandomForestClassifier
13 from sklearn.ensemble import VotingClassifier
14 clf1 = LogisticRegression(solver='lbfgs', max_iter=1000, random_state
   =123)
15 clf2 = RandomForestClassifier(n_estimators=100, random_state=123)
16 clf3 = GaussianNB()
17 X = np.array([[-1.0, -1.0], [-1.2, -1.4], [-3.4, -2.2], [1.1, 1.2]])
18 y = np.array([1, 1, 2, 2])
19 eclf = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('gnb'
   , clf3)],
   voting='soft',
   weights=[1, 1, 5])
20 # predict class probabilities for all classifiers
21 probas = [c.fit(X, y).predict_proba(X) for c in (clf1, clf2, clf3,
   eclf)]
22 # get class probabilities for the first sample in the dataset
23 class1_1 = [pr[0, 0] for pr in probas]
24 class2_1 = [pr[0, 1] for pr in probas]
25 # plotting
26 N = 4 # number of groups
27 ind = np.arange(N) # group positions
```

```

30 width = 0.35 # bar width
31 fig, ax = plt.subplots()
32 # bars for classifier 1-3
33 p1 = ax.bar(ind, np.hstack(([class1_1[:-1], [0]])), width,
34             color='green', edgecolor='k')
35 p2 = ax.bar(ind + width, np.hstack(([class2_1[:-1], [0]])), width,
36             color='lightgreen', edgecolor='k')
37 # bars for VotingClassifier
38 p3 = ax.bar(ind, [0, 0, 0, class1_1[-1]], width,
39             color='blue', edgecolor='k')
40 p4 = ax.bar(ind + width, [0, 0, 0, class2_1[-1]], width,
41             color='steelblue', edgecolor='k')
42 # plot annotations
43 plt.axvline(2.8, color='k', linestyle='dashed')
44 ax.set_xticks(ind + width)
45 ax.set_xticklabels(['LogisticRegression\nweight 1',
46                     'GaussianNB\nweight 1',
47                     'RandomForestClassifier\nweight 5',
48                     'VotingClassifier\n(average probabilities)'],
49                     rotation=40,
50                     ha='right')
51 plt.ylim([0, 1])
52 plt.title('Class probabilities for sample 1 by different classifiers')
53 plt.legend([p1[0], p2[0]], ['class 1', 'class 2'], loc='upper left')
54 plt.tight_layout()
55 plt.show()

```

In [2]: runfile('D:/OneDrive - Hybi.god/KULIAH/Semester 6/AI/KB3C/src/1174067/3/untitled1.py', wdir='D:/OneDrive - Hybi.god/KULIAH/Semester 6/AI/KB3C/src/1174067/3')



Gambar 3.3 Voting Random Matriks

3.1.2 Praktek

3.1.2.1 Nomor 1

In [0]

```

1 import pandas as KK # Melakukan import library pandas menjadi nama
2 sendiri yaitu KK
3
4 buah = {"Nama buah" : ['apel','ceri','boom','melon']} # Membuat
5 varibel yang bernama aplikasi , dan mengisi dataframanya dengan
6 nama nama aplikasi koding
7 x = KK.DataFrame(buah) # Membuat variabel x yang akan membuat
8 DataFrame dari library pandas yang akan memanggil variabel
9 aplikasi .
10 print (' KK makan buah: ' + x) #print hasil dari x

```

```

In [7]: runfile('D:/OneDrive - Hybi.god/KULIAH/Semester 6/AI/KB3C/src/
1174067/3/1174067.py', wdir='D:/OneDrive - Hybi.god/KULIAH/Semester 6/AI/KB3C/src/
1174067/3')
          Nama Aplikasi
0      KK pake aplikasi: VSCode
1      KK pake aplikasi: Atom
2      KK pake aplikasi: Sublime
3      KK pake aplikasi: Notepad++           I

```

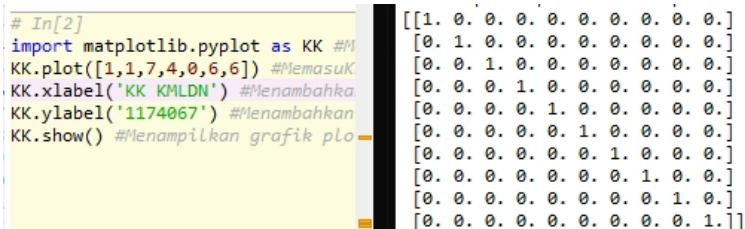
Gambar 3.4 Membuat Aplikasi pakai pandas

3.1.2.2 Nomor 2

```

1 import numpy as KK #Melakukan import library numpy menjadi nama
2 sendiri yaitu KK
3
4 matrix_x = KK.eye(10) #Membuat sebuah matrix pake numpy dengan
5 menggunakan fungsi eye
6 matrix_x #Mendeklarasikan matrix_x yang tadi dibuat
7
8 print (matrix_x) #print matrix_x yang tadi dibuat yang berbentuk 10
9 x10

```



```

# In[2]
import matplotlib.pyplot as KK #M
KK.plot([1,1,7,4,0,6,6]) #Memasuk
KK.xlabel('KK KMLDN') #Menambahka
KK.ylabel('1174067') #Menambahkan
KK.show() #Menampilkan grafik plo
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]

```

Gambar 3.5 Membuat Aplikasi pakai numpy

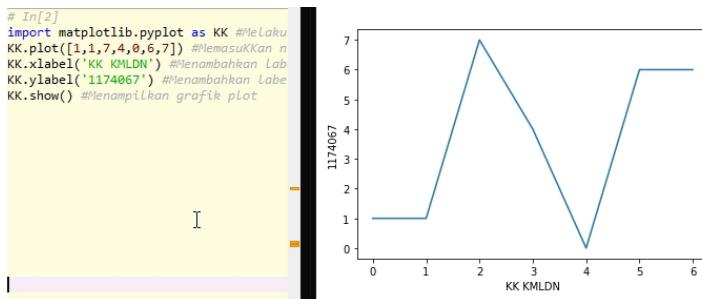
3.1.2.3 Nomor 3

```

1 # In [2]
2 import matplotlib.pyplot as KK #Melakukan import library numpy
3 menjadi nama sendiri yaitu KK
4 KK.plot([1,1,7,4,0,6,7]) #Memasukkan nilai pada plot

```

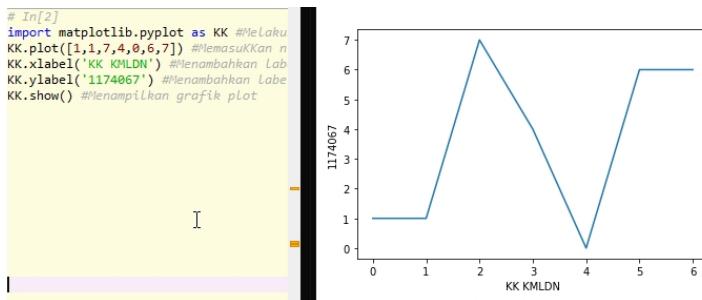
```
4 KK.xlabel('KK KMLDN') #Menambahkan label pada x
5 KK.ylabel('1174067') #Menambahkan label pada y
```



Gambar 3.6 Membuat Aplikasi pakai matplotlib

3.1.2.4 Nomor 4

- Source Code pertama pada random forest berfungsi untuk membaca dataset yang memiliki format text file dengan mendefinisikan variabel yang bernama imgatt. Variabel tersebut berisi value untuk membaca data.



Gambar 3.7 Hasil 4 Bagian 1

- Pada source code berikutnya akan mengembalikan baris teratas dari DataFrame variabel imgatt.

```
In [15]: import pandas as pd #Melakukan import library numpy menjadi pd
...
...: imgatt = pd.read_csv("D:/OneDrive - Hybi.god/KULIAH/Semester 6/AI/KB3C/src/1174067/3/
CUB_200_2011/attributes/image_attribute_labels.txt",
...: sep='|', header=None, error_bad_lines=False, warn_bad_lines=False,
...: usecols=[0,1,2], names=['imgid', 'atid', 'present']) #membuat variabel
imgatt untuk membaca file csv dari dataset, dengan ketentuan yang ada.
```

Gambar 3.8 Hasil 4 Bagian 2

- Pada output berikutnya akan menampilkan jumlah kolom dan baris dari DataFrame variable imgatt.

```
In [13]: imgatt.head() #Menampilkan 5 baris pertama pada DataFrame imgatt
Out[13]:
      imgid  attid  present
0        1       1        0
1        1       2        0
2        1       3        0
3        1       4        0
4        1       5        1
```

Gambar 3.9 Hasil 4 Bagian 3

- Variabel imgatt2 telah menggunakan fungsi yang bernama pivot agar mengubah kolom jadi baris dan sebaliknya dari DataFrame imgatt sebelumnya.

```
1 sep='\\s+' , header=None , error_bad_lines=
   False , warn_bad_lines=False ,
```

```
In [15]: imgatt2 = imgatt.pivot(index='imgid', columns='attid', values='present') #Membuat sebuah variabel baru dari fungsi
imgatt, dengan mengganti index menjadi kolom dan kolom menjadi index
```

Gambar 3.10 Hasil 4 Bagian 4

- Variabel imgatt2 head berfungsi untuk mengembalikan value teratas pada DataFrame imgatt2.

```
1 # In [4]
```

```
In [16]: imgatt2.head() #Menampilkan data yang sudah dibaca tadi tapi cuma data paling atas
Out[16]:
      attid  1    2    3    4    5    6    7    ...  306   307   308   309   310   311   312
      imgid
1        0    0    0    0    1    0    0    ...    0    0    1    0    0    0    0
2        0    0    0    0    0    0    0    ...    0    0    0    0    0    0    0
3        0    0    0    0    1    0    0    ...    0    0    1    0    0    1    0
4        0    0    0    0    1    0    0    ...    1    0    0    1    0    0    0
5        0    0    0    0    1    0    0    ...    0    0    0    0    0    0    0
```

[5 rows x 312 columns]

Gambar 3.11 Hasil 4 Bagian 5

- Menghasilkan jumlah kolom dan baris pada DataFrame imgatt2.

```
In [17]: imgatt2.shape #Menampilkan jumlah seluruh data, kolom-nya
Out[17]: (11788, 312)
```

Gambar 3.12 Hasil 4 Bagian 6

- Menunjukkan dalam melakukan pivot yang mana imgid menjadi sebuah index yang unik.

```
1 imgatt2.head() #Menampilkan data yang sudah dibaca tadi tapi
     cuman data paling atas
2
3 # In [8]
4 imgatt2.shape #Menampilkan jumlah seluruh data , kolom-nya
```

```
In [21]: imglabels = pd.read_csv("CUB_200_2011/image_class_labels.txt",
...:                         sep=' ', header=None, names=['imgid', 'label']) #Membaca data dimasukkan ke variable
...:
...: imgLabels = imglabels.set_index('imgid') #Variable imgLabels dan set index (imgid)
```

Gambar 3.13 Hasil 4 Bagian 7

- Akan melakukan load jawabannya yang berisi apakah burung tersebut termasuk spesies yang mana. Kolom tersebut yaitu imgid dan label.

```
1 sep=' ', header=None, names=['imgid', 'label']) #baca data csv dengan ketentuan yang ada
```

```
In [22]: imglabels.head()
Out[22]:
label
imgid
1      1
2      1
3      1
4      1
5      1
```

Gambar 3.14 Hasil 4 Bagian 8

- Menunjukkan bahwa jumlah baris sebanyak 11788 dan kolom 1 yang dimana kolom tersebut adalah jenis spesies pada burung.

```
In [23]: imglabels.shape
Out[23]: (11788, 1)
```

Gambar 3.15 Hasil 4 Bagian 9

- Melakukan join antara imgatt2 dengan imglabels dikarenakan memiliki isi yang sama sehingga akan mendapatkan sebuah data ciri-ciri dan data jawaban sehingga bisa dikategorikan sebagai supervised learning.

```
1         sep=' ', header=None, names=['imgid', 'label'])
#Membaca data dimasukkan ke variable imglabels
```

```
In [24]: df = imgatt2.join(imglabels) #Variabel df dimasukkan fungsi join dari data imgatt2 ke variabel imglabels
...: df = df.sample(frac=1) #Variabel df sebagai sample dengan ketentuan frac=1
```

Gambar 3.16 Hasil 4 Bagian 10

- Melakukan drop pada label yang ada didepan dan akan menggunakan label yang baru di joinkan.

```
1 # In[10]
2 imglabels.head() #Menampilkan data yang sudah dibaca tadi tapi
    cuman data paling atas
```

In [25]: df_att = df.iloc[:, :312] #Membuat kolom dengan ketentuan 312
...: df_label = df.iloc[:, 312:] #Membuat kolom dengan ketentuan 312

Gambar 3.17 Hasil 4 Bagian 11

- Mengecek isi 5 data teratas pada df.att.

```
1 imglabels.shape #Menampilkan jumlah seluruh data , kolom-nya
```

In [26]: df_att.head() #Menampilkan data yang sudah dibaca tadi tapi cuman data paling atas
Out[26]:

imgid	1	2	3	4	5	6	7	...	306	307	308	309	310	311	312
10397	0	0	0	0	0	0	1	...	0	0	0	0	0	0	1
1108	0	0	0	0	0	0	0	1	...	0	0	0	1	0	0
2382	0	0	0	0	0	0	0	1	...	0	1	0	1	0	0
9374	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0
9371	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0

[5 rows x 312 columns]

Gambar 3.18 Hasil 4 Bagian 12

- Mengecek isi data teratas dari df.label.

```
1 df = imgatt2.join(imglabels) #Varibel df dimasukkan fungsi join
    dari data imgatt2 ke variabel imglabels
```

```
In [27]: df_label.head()
Out[27]:
label
imgid
10397    177
1108      20
2382      42
9374      160
9371      160
```

Gambar 3.19 Hasil 4 Bagian 13

- Membagi 8000 row pertama menjadi data training dan sisanya adalah data testing.

```
1 # In[13]
2 df_att = df.iloc[:, :312] #Membuat kolom dengan ketentuan 312
3 df_label = df.iloc[:, 312:] #Membuat kolom dengan ketentuan 312
4
5 # In[14]
6 df_att.head() #Menampilkan data yang sudah dibaca tadi tapi cuman
                 data paling atas
```

```
In [29]: df_train_att = df_att[:8000] #Data akan dibagi dari 8000 row pertama menjadi data training dan sisanya adalah data
testing
...: df_train_label = df_label[:8000] #Data akan dibagi dari 8000 row pertama menjadi data training dan sisanya adalah
data testing
...: df_test_att = df_att[8000:] #Berbalik dari sebelumnya data akan dibagi mulai dari 8000 row pertama menjadi data
training dan sisanya adalah data testing
...: df_test_label = df_label[8000:] #Berbalik dari sebelumnya data akan dibagi mulai dari 8000 row pertama menjadi data
training dan sisanya adalah data testing
...:
...: df_train_label = df_train_label['label'] #Menambahkan Label
...: df_test_label = df_test_label['label'] #Menambahkan Label
```

Gambar 3.20 Hasil 4 Bagian 14

- Pemanggilan class RandomForestClassifier. Dimana artinya menunjukkan banyak kolom pada setiap tree adalah 50.

```
1
2 # In[16]
```

```
In [30]: from sklearn.ensemble import RandomForestClassifier #import randomforestclassifier
...: clf = RandomForestClassifier(max_features=50, random_state=0, n_estimators=100) #clf sebagai variabel untuk
...: klasifikasi random forest
```

Gambar 3.21 Hasil 4 Bagian 15

- Menunjukkan hasil prediksi dari Random Forest.

```
| df_test_att = df_att[8000:] #Berbalik dari sebelumnya data akan
|     dibagi mulai dari 8000 row pertama menjadi data training dan
|     sisanya adalah data testing
```

```
In [31]: clf.fit(df_train_att, df_train_label) #Variablnle clf untuk fit yaitu menjadi data training
Out[31]:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=None, max_features=50, 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, n_estimators=100,
n_jobs=None, oob_score=False, random_state=0, verbose=0,
warm_start=False)
```

Gambar 3.22 Hasil 4 Bagian 16

- Menampilkan besaran akurasi dari prediksi pada Random Forest yang merupakan score perolehan klarifikasi.

```
| df_train_label = df_train_label['label'] #Menambahkan label
```

```
In [32]: print(clf.predict(df_train_att.head()))
data paling atas
[177  20  42 160 160]
```

Gambar 3.23 Hasil 4 Bagian 17

- Menampilkan besaran akurasi dari prediksi pada Random Forest yang merupakan score perolehan klarifikasi.

```
| # In [17]
```

```
In [33]: clf.score(df_test_att, df_test_label)
Out[33]: 0.44931362196409713
```

Gambar 3.24 Hasil 4 Bagian 18

3.1.2.5 Nomor 5

```

1 clf.fit(df_train_att, df_train_label) #Variabne clf untuk fit yaitu
2     menjadi data training
3 # In[19]

```

```

In [34]: from sklearn.metrics import confusion_matrix #Mengimport Confusion Matrix
...: pred_labels = clf.predict(df_test_att) #membuat variable pred_labels yang data testing dari sebelumnya
...: cm = confusion_matrix(df_test_label, pred_labels) #Variable cm sebagai variabel data label
...

```

Gambar 3.25 Hasil 5 Bagian 1

```

1 # In[20]

```

```

In [35]: cm #Menampilkan data label berbentuk array
Out[35]:
array([[ 6,  0,  1, ...,  0,  0,  0],
       [ 0, 11,  0, ...,  0,  0,  0],
       [ 5,  1,  9, ...,  0,  0,  0],
       ...,
       [ 1,  0,  0, ...,  1,  0,  0],
       [ 0,  1,  0, ...,  0,  8,  0],
       [ 0,  0,  0, ...,  0,  0, 19]], dtype=int64)

```

Gambar 3.26 Hasil 5 Bagian 2

```

1
2
3 # In[21]: Confusion Matrix
4 from sklearn.metrics import confusion_matrix #Mengimport Confusion
5     Matrix
6 pred_labels = clf.predict(df_test_att) #Membuat variable pred_labels
7     dari data testing
8 cm = confusion_matrix(df_test_label, pred_labels) #cm sebagai
9     variabel data label
10
11 # In[22]
12 cm #Memunculkan data label berbentuk array
13
14 # In[23]
15 import matplotlib.pyplot as plt #Mengimport library matplotlib
16     sebagai plt
17 import itertools #Mengimport library itertools
18 def plot_confusion_matrix(cm, classes,
19     normalize=False,
20     title='Confusion matrix',
21     cmap=plt.cm.Blues): #Membuat fungsi dengan
22     ketentuan data yang ada pada cm
23
24     if normalize:
25         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

```

```

21     print("Normalized confusion matrix") #Jika normalisasi
22     sebagai ketentuan yang ada maka print normalized confusion matrix
23 else:
24     print('Confusion matrix, without normalization') #Jika tidak
25     memenuhi kondisi if maka pring else
26
27     print(cm) #Print data cm
28
29     plt.imshow(cm, interpolation='nearest', cmap=cmap) #plt sebagai
30     fungsi untuk membuat plot
31     plt.title(title) #Membuat title pada plot

```

```

In [36]: import matplotlib.pyplot as plt #mengimport library matplotlib sebagai plt
...: import itertools #mengimport Library itertools
...: def plot_confusion_matrix(cm, classes,
...:                         normalize=False,
...:                         title='Confusion matrix',
...:                         cmap=plt.cm.Blues): #membuat fungsi dengan ketentuan data yang ada pada cm
...:
...:     if normalize:
...:         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
...:         print("Normalized confusion matrix") #jika normalisasi sebagai ketentuan yang ada maka print normalized
...:         confusion matrix
...:     else:
...:         print('Confusion matrix, without normalization') #jika tidak memenuhi kondisi if maka pring else
...:
...:     print(cm) #print data cm
...:
...:     plt.imshow(cm, interpolation='nearest', cmap=cmap) #plt sebagai fungsi untuk membuat plot
...:     plt.title(title) #membuat title pada plot
...:     plt.colorbar()
...:     tick_marks = np.arange(len(classes)) #membuat marks pada plot
...:     plt.xticks(tick_marks, classes, rotation=90) #membuat ticks pada x
...:     plt.yticks(tick_marks, classes) #membuat ticks pada y
...:
...:     fmt = '.2f' if normalize else 'd' #fmt sebagai normalisasi
...:     thresh = cm.max() / 2. #variabel thresh menambil data max pada cm kemudian dibagi 2
...:
...:     plt.tight_layout() #mengatur layout pada plot
...:     plt.ylabel('True label') #memberi nama label pada sumbu y
...:     plt.xlabel('Predicted label') #memberi nama label pada sumbu x

```

Gambar 3.27 Hasil 5 Bagian 3

```

1     plt.yticks(tick_marks, classes) #Membuat ticks pada y
2
3     fmt = '.2f' if normalize else 'd' #fmt sebagai normalisasi
4     thresh = cm.max() / 2. #Variable thresh menambil data max pada
cm kemudian dibagi 2

```

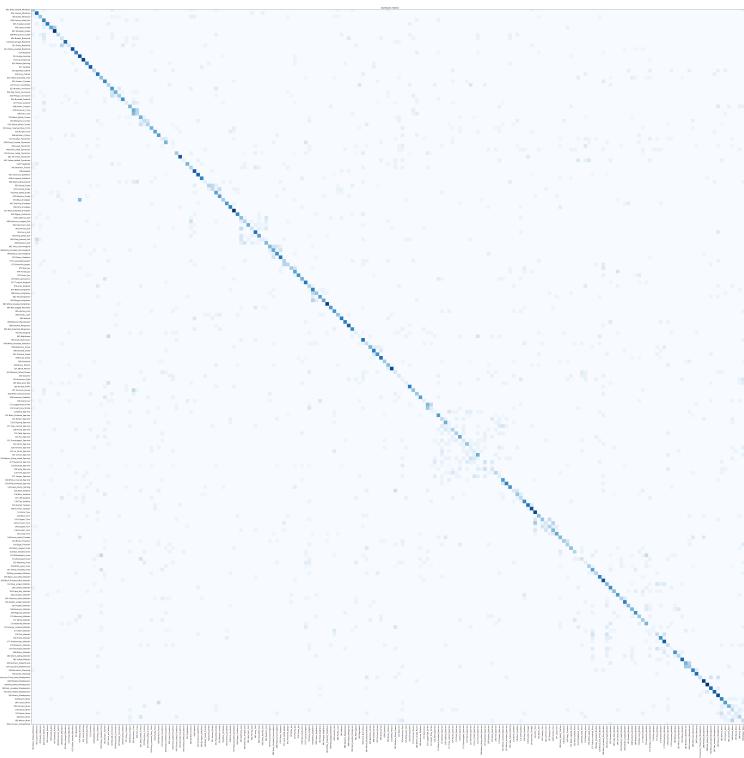
```
In [37]: birds = pd.read_csv("CUB_200_2011/cla
...:                                     sep='\\s+', header=
birdname
...: birds = birds['birdname'] #nama birds
...: birds #menampilkan data birds
Out[37]:
0      001.Black_footed_Albatross
1      002.Laysan_Albatross
2      003.Sooty_Albatross
3      004.Groove_billed_Ani
4      005.Crested_Auklet
...
195      196.House_Wren
196      197.Marsh_Wren
197      198.Rock_Wren
198      199.Winter_Wren
199      200.Common_Yellowthroat
Name: birdname, Length: 200, dtype: object
```

Gambar 3.28 Hasil 5 Bagian 4

```
1 plt.xlabel('Predicted label') #Menambahkan nama label pada sumbu
x
2
3
4 # In[24]
5 birds = pd.read_csv("N:/Tugas/Kuliah/Semester 6/Kecerdasan Buatan/
KB3C Ngerjain/src/1174067/3/CUB_200_2011/classes.txt",
6                                     sep='\\s+', header=None, usecols=[1], names=[ ',
birdname']) #membaca csv dengan ketentuan nama birdname
```

```
In [39]: import numpy as np #import library numpy sebagai np
...: np.set_printoptions(precision=2) #np sebagai variabel yang membuat set precision=2
...: plt.figure(figsize=(60,60), dpi=300) #plot sebagai figure dengan ketentuan size 60,60 dan dpi 300
...: plot.confusion_matrix(cm, classes=birds, normalize=True) #data cm dan clas birds dibuat sebagai plot
...: plt.savefig('hasil.png')
Normalized confusion matrix
[[0.37 0. 0. ... 0. 0.05 0. ]
 [0.08 0.77 0. ... 0. 0. 0. ]
 [0.04 0.04 0.39 ... 0. 0. 0. ]
 ...
 [0. 0. 0. ... 0.31 0. 0. ]
 [0. 0. 0. ... 0. 0.24 0. ]
 [0. 0. 0. ... 0. 0.04 0.73]]
```

Gambar 3.29 Hasil 5 Bagian 5



Gambar 3.30 Plot Hasil 5 Bagian 5

3.1.2.6 Nomor 6

```
1 # In[25]
2 import numpy as np #Mengimport library numpy sebagai np
3 np.set_printoptions(precision=2) #np sebagai variabel yang membuat
   set precision=2
4 plt.figure(figsize=(60,60), dpi=300) #Plot sebagai figure dengan
   ketentuan size 60,60 dan dpi 300
```

```
In [24]: from sklearn import tree #Mengimport library
....: clftree = tree.DecisionTreeClassifier() #clf
....: clftree.fit(df_train_att, df_train_label) #M
....: clftree.score(df_test_att, df_test_label) #M
Out[24]: 0.264519535374868
```

Gambar 3.31 Hasil 6 Bagian 1

```
1 #plt.savefig('hasil.png') #
```

```
In [31]: from sklearn.model_selection import
...: scores = cross_val_score(clf, df_dari_data_training
...: print("Accuracy: %0.2f (+/- %0.2f"
C:\ProgramData\Anaconda3\lib\site-packages\from 'auto' to 'scale' in version 0.22 to ;
avoid this warning.
"avoid this warning.", FutureWarning)
Accuracy: 0.27 (+/- 0.01)
```

Gambar 3.32 Hasil 6 Bagian 2**3.1.2.7 Nomor 7**

```
1 clftree = tree.DecisionTreeClassifier() #clftree sebagai variabel
  untuk decision tree
2 clftree.fit(df_train_att, df_train_label) #Mengatur data training
3 clftree.score(df_test_att, df_test_label) #Mengatur data testing
```

```
In [32]: scorestree = cross_val_scores dan metode tree
...: print("Accuracy: %0.2f ada
Accuracy: 0.44 (+/- 0.02)
```

Gambar 3.33 Hasil 7 Bagian 1

```
1 from sklearn import svm #Mengimport library svm
2 clfsvm = svm.SVC() #clfsvm sebagai variabel untuk mengatur fungsi SVC
```

In [28]: scorestree = cross_val_scores dan metode tree
: print("Accuracy: %0.2f ada
 Accuracy: 0.27 (+/- 0.02)

Gambar 3.34 Hasil 7 Bagian 2

1
 2 # In[28]: Pengecekan Cross Validation

```
In [27]: scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5) #Membuat variable data training
...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2)) #Menampilkan data testing dan output akurasi
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 warning.
"avoid this warning", FutureWarning)
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 warning.
"avoid this warning", FutureWarning)
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 warning.
"avoid this warning", FutureWarning)
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 warning.
"avoid this warning", FutureWarning)
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 warning.
"avoid this warning", FutureWarning)
Accuracy: 0.27 (+/- 0.03)
```

Gambar 3.35 Hasil 7 Bagian 3

3.1.2.8 Nomor 8

1 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() *
 2)) #Print data scores dengan ketentuan akurasi
 2
 3 # In[29]
 4 scorestree = cross_val_score(clftree, df_train_att, df_train_label,
 cv=5) #Membuat variable prediksi menggunakan scores dan metode
 tree
 5 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.
 std() * 2)) #Menampilkan dengan ketentuan yang ada
 6
 7 # In[30]
 8 scoressvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv
 =5) #Membuat variable data training
 9 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.
 std() * 2)) #Menampilkan data testing dan output akurasi
 10
 11 # In[31]: Pengamatan komponen informasi

```

12 max_features_opts = range(5, 50, 5) #Variable max_features_opts
    sebagai variabel untuk membuat range 5,50,5
13 n_estimators_opts = range(10, 200, 20) #Variabel n_estimators_opts
    sebagai variabel untuk membuat range 10,200,20
14 rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),
    ,4), float) #Variabel rf_params sebagai variabel untuk
    menjumlahkan yang sudah di tentukan sebelumnya
15 i = 0
16 for max_features in max_features_opts: #Perulangan

```

```

....          ....
....      rf_params[i,3] = scores.std() * 2 #index 3
....      i += 1 #Dengan ketentuan i += 1
....      print("Max features: %d, num estimators: %d, acc
n_estimators, scores.mean(), scores.std() * 2))
....      #Print hasil pengulangan yang sudah ditentukan
Max features: 5, num estimators: 10, accuracy: 0.26 (+/- 0.03)
Max features: 5, num estimators: 30, accuracy: 0.36 (+/- 0.03)
Max features: 5, num estimators: 50, accuracy: 0.39 (+/- 0.03)
Max features: 5, num estimators: 70, accuracy: 0.41 (+/- 0.02)
Max features: 5, num estimators: 90, accuracy: 0.42 (+/- 0.02)
Max features: 5, num estimators: 110, accuracy: 0.43 (+/- 0.03)
Max features: 5, num estimators: 130, accuracy: 0.43 (+/- 0.02)
Max features: 5, num estimators: 150, accuracy: 0.44 (+/- 0.02)
Max features: 5, num estimators: 170, accuracy: 0.44 (+/- 0.03)
Max features: 5, num estimators: 190, accuracy: 0.45 (+/- 0.01)
Max features: 10, num estimators: 10, accuracy: 0.29 (+/- 0.02)
Max features: 10, num estimators: 30, accuracy: 0.38 (+/- 0.02)
Max features: 10, num estimators: 50, accuracy: 0.41 (+/- 0.03)
Max features: 10, num estimators: 70, accuracy: 0.43 (+/- 0.02)
Max features: 10, num estimators: 90, accuracy: 0.43 (+/- 0.02)
Max features: 10, num estimators: 110, accuracy: 0.44 (+/- 0.02)
Max features: 10, num estimators: 130, accuracy: 0.45 (+/- 0.03)
Max features: 10, num estimators: 150, accuracy: 0.45 (+/- 0.03)
Max features: 10, num estimators: 170, accuracy: 0.45 (+/- 0.02)
Max features: 10, num estimators: 190, accuracy: 0.45 (+/- 0.03)
Max features: 15, num estimators: 10, accuracy: 0.31 (+/- 0.03)
Max features: 15, num estimators: 30, accuracy: 0.40 (+/- 0.03)
Max features: 15, num estimators: 50, accuracy: 0.42 (+/- 0.03)
Max features: 15, num estimators: 70, accuracy: 0.43 (+/- 0.02)
Max features: 15, num estimators: 90, accuracy: 0.43 (+/- 0.03)
Max features: 15, num estimators: 110, accuracy: 0.44 (+/- 0.02)
Max features: 15, num estimators: 130, accuracy: 0.44 (+/- 0.02)
Max features: 15, num estimators: 150, accuracy: 0.45 (+/- 0.02)
Max features: 15, num estimators: 170, accuracy: 0.45 (+/- 0.02)
Max features: 15, num estimators: 190, accuracy: 0.45 (+/- 0.02)

```

Gambar 3.36 Hasil 8 Bagian 1

```

Max features: 40, num estimators: 190, accuracy: 0.45 (+/- 0.02)
Max features: 45, num estimators: 10, accuracy: 0.34 (+/- 0.02)
Max features: 45, num estimators: 30, accuracy: 0.41 (+/- 0.02)
Max features: 45, num estimators: 50, accuracy: 0.43 (+/- 0.02)
Max features: 45, num estimators: 70, accuracy: 0.44 (+/- 0.03)
Max features: 45, num estimators: 90, accuracy: 0.44 (+/- 0.03)
Max features: 45, num estimators: 110, accuracy: 0.45 (+/- 0.03)
Max features: 45, num estimators: 130, accuracy: 0.45 (+/- 0.02)
Max features: 45, num estimators: 150, accuracy: 0.45 (+/- 0.03)
Max features: 45, num estimators: 170, accuracy: 0.45 (+/- 0.03)
Max features: 45, num estimators: 190, accuracy: 0.45 (+/- 0.02)

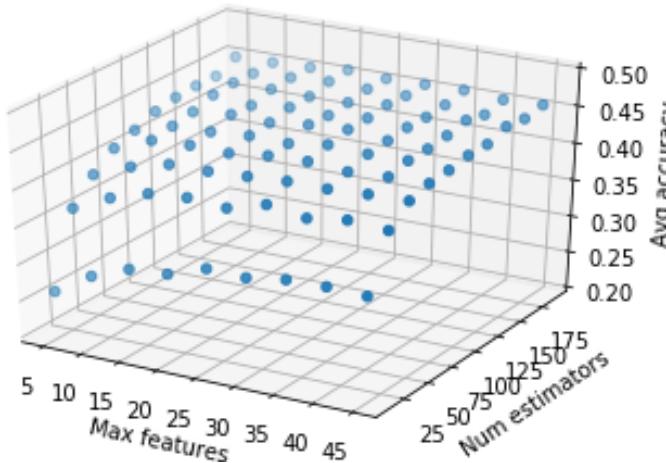
```

Gambar 3.37 Hasil 8 Bagian 1 Akhir kode

```

1      clf = RandomForestClassifier(max_features=max_features ,
2          n_estimators=n_estimators) #Menampilkan variabel csf
3          scores = cross_val_score(clf, df_train_att, df_train_label,
4          cv=5) #Variable scores sebagai variabel training
5          rf_params[i,0] = max_features #index 0
6          rf_params[i,1] = n_estimators #index 1
7          rf_params[i,2] = scores.mean() #index 2
8          rf_params[i,3] = scores.std() * 2 #index 3
9          i += 1 #Dengan ketentuan i += 1
10         print("Max features: %d, num estimators: %d, accuracy: %0.2f
11         (+/- %0.2f)" % (max_features, n_estimators, scores.mean(),
12         scores.std() * 2))
13         #Print hasil pengulangan yang sudah ditentukan
14
15 # In[32]
16 import matplotlib.pyplot as plt #Mengimport library matplotlib
17     sebagai plt
18 from mpl_toolkits.mplot3d import Axes3D #Mengimport axes3D untuk
19     menampilkan plot 3 dimensi
20 from matplotlib import cm #Memanggil data cm yang sudah tersedia
21 fig = plt.figure() #Menghasilkan plot sebagai figure

```



Gambar 3.38 Hasil 8 Bagian 2

3.1.3 Penanganan Error

1. ScreenShoot Error

```
C:\Windows\system32\cmd.exe /c ll C:\Users\Acer\Downloads\dataset\diabetes.csv & python C:\Users\Acer\Downloads\dataset\diabetes.py & ll C:\Users\Acer\Downloads\dataset\diabetes.csv
```

Gambar 3.39 FileNotFoundError

2. Tuliskan Kode Error dan Jenis Error

- FileNotFoundError

3. Cara Penangan Error

- FileNotFoundError

Error terdapat pada kesalahan saat baca file csv, yang tidak terbaca. Dikarenakan letak file yang dibaca tidak pada direktori yang sama. Seharusnya letakkan file di direktori yang sama.

3.1.4 Link Youtube:

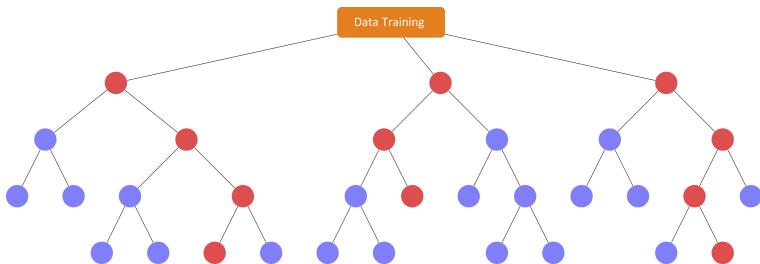
https://youtu.be/69BWGlcf_CQ

3.2 Mochamad Arifqi Ramadhan (1174074)

3.2.1 Teori

3.2.1.1 Apa Itu Random Forest

Random Forest adalah sebuah 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.



Gambar 3.40 Random Forest

3.2.1.2 cara membaca dataset kasus 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 file bernama .txt dan didalamnya terdapat class dari field.
- Misalnya saja pada data jenis burung memiliki file index dan angka, dimana index berisi angka yang memiliki makna berupa jenis burung.
- bahkan nama burung sedangkan field 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 field yang di isikan berupa angka.
- Artinya angka 0 berarti tidak dan angka 1 berarti Ya.

3.2.1.3 Cross Validation

Cross Validation adalah sebuah teknik validasi model yang berfungsi untuk melakukan penilaian bagaimana hasil analisis statistik akan digeneralisasi ke data yang lebih 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 mendefinisikan dataset guna mengejauhi dalam fase pelatihan untuk membatasi masalah seperti overfitting dan underfitting serta mendapatkan wawasan tentang bagaimana model akan digeneralisasikan ke set data independen.

3.2.1.4 Arti score 44 % pada random forest, 27% pada decision tree dan 29 % dari SVM.

Dimana Score 44 % didapatkan 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.

3.2.1.5 Cara membaca confusion matriks dan contohnya memakai gambar atau ilustrasi sendiri.

Perhitungan Confusion Matriks dapat dilakukan dengan cara dibawah ini.

- 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 Tue Mar 17 21:24:46 2020
4
5 @author: Rin
6 """
7 import numpy as np
8 import matplotlib.pyplot as plt
9 import pandas as pd
10 y_actu = pd.Series([2, 0, 2, 2, 0, 1, 1, 2, 2, 0, 1, 2], name='
    Actual')
11 y_pred = pd.Series([0, 0, 2, 1, 0, 2, 1, 0, 2, 0, 2, 2], name='
    Predicted')
12 df_confusion = pd.crosstab(y_actu, y_pred)
```

```

13 df_confusion = pd.crosstab(y_actu, y_pred, rownames=['Actual'],
14                             colnames=['Predicted'], margins=True)
15 def plot_confusion_matrix(df_confusion, title='Confusion matrix',
16                           cmap=plt.cm.gray_r):
17     plt.matshow(df_confusion, cmap=cmap) # imshow
18     #plt.title(title)
19     plt.colorbar()
20     tick_marks = np.arange(len(df_confusion.columns))
21     plt.xticks(tick_marks, df_confusion.columns, rotation=45)
22     plt.yticks(tick_marks, df_confusion.index)
23     #plt.tight_layout()
24     plt.ylabel(df_confusion.index.name)
25     plt.xlabel(df_confusion.columns.name)
26 plot_confusion_matrix(df_confusion)
27 plt.show()

```



Gambar 3.41 Confusion Matriks

3.2.1.6 Apa itu voting pada random forest

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

Untuk menggunakan Voting pada Random Forest dapat dilihat code ini:

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Mar 17 21:32:47 2020
4
5 @author: Rin
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10 from sklearn.linear_model import LogisticRegression
11 from sklearn.naive_bayes import GaussianNB
12 from sklearn.ensemble import RandomForestClassifier
13 from sklearn.ensemble import VotingClassifier
14 clf1 = LogisticRegression(solver='lbfgs', max_iter=1000, random_state=123)
15 clf2 = RandomForestClassifier(n_estimators=100, random_state=123)
16 clf3 = GaussianNB()
17 X = np.array([[-1.0, -1.0], [-1.2, -1.4], [-3.4, -2.2], [1.1, 1.2]])

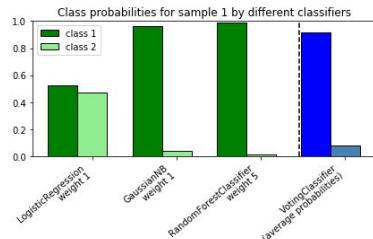
```

```

18 y = np.array([1, 1, 2, 2])
19 eclf = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('gnb',
   , clf3)],
20                         voting='soft',
21                         weights=[1, 1, 5])
22 # predict class probabilities for all classifiers
23 probas = [c.fit(X, y).predict_proba(X) for c in (clf1, clf2, clf3,
   eclf)]
24 # get class probabilities for the first sample in the dataset
25 class1_1 = [pr[0, 0] for pr in probas]
26 class2_1 = [pr[0, 1] for pr in probas]
27 # plotting
28 N = 4 # number of groups
29 ind = np.arange(N) # group positions
30 width = 0.35 # bar width
31 fig, ax = plt.subplots()
32 # bars for classifier 1-3
33 p1 = ax.bar(ind, np.hstack(([class1_1[:-1], [0]])), width,
   color='green', edgecolor='k')
34 p2 = ax.bar(ind + width, np.hstack(([class2_1[:-1], [0]])), width,
   color='lightgreen', edgecolor='k')
35 # bars for VotingClassifier
36 p3 = ax.bar(ind, [0, 0, 0, class1_1[-1]], width,
   color='blue', edgecolor='k')
37 p4 = ax.bar(ind + width, [0, 0, 0, class2_1[-1]], width,
   color='steelblue', edgecolor='k')
38 # plot annotations
39 plt.axvline(2.8, color='k', linestyle='dashed')
40 ax.set_xticks(ind + width)
41 ax.set_xticklabels(['LogisticRegression\nweight 1',
   'GaussianNB\nweight 1',
   'RandomForestClassifier\nweight 5',
   'VotingClassifier\n(average probabilities)'],
   rotation=40,
   ha='right')
42 plt.ylim([0, 1])
43 plt.title('Class probabilities for sample 1 by different classifiers')
44 plt.legend([p1[0], p2[0]], ['class 1', 'class 2'], loc='upper left')
45 plt.tight_layout()
46 plt.show()

```

In [2]: runfile('N:/Tugas/Kuliah/Semester 6/Kecerdasan Buatan/KB3C Ngerjain/src/1174066/3/untitled1.py', wdir='N:/Tugas/Kuliah/Semester 6/Kecerdasan Buatan/KB3C Ngerjain/src/1174066/3')



Gambar 3.42 Voting Random Matriks

3.2.2 Praktek

3.2.2.1 Nomor 1

```

1 # In [0]
2 import pandas as dirga # Melakukan import library pandas menjadi nama
   sendiri yaitu dirga
3
4 aplikasi = {"Nama Aplikasi" : ['VSCode','Atom','Sublime','Notepad++']
   ]} # Membuat varibel yang bernama aplikasi , dan mengisi
   dataframanya dengan nama nama aplikasi koding
5 x = dirga.DataFrame(aplikasi) # Membuat variabel x yang akan membuat
   DataFrame dari library pandas yang akan memanggil variabel
   aplikasi.
6 print (' Dirga pake aplikasi: ' + x) #print hasil dari x

```

In [6]: import pandas as dirga # Melakukan import library pandas menjadi nama sendiri yaitu dirga

```

....:
....: aplikasi = {"Nama Aplikasi" : [
....: mengisi dataframanya dengan nama nama aplikasi
....: ]} # Membuat varibel yang bernama aplikasi , dan mengisi
....: dataframanya dengan nama nama aplikasi koding
....: x = dirga.DataFrame(aplikasi) # Membuat variabel x yang akan membuat
....: DataFrame dari library pandas yang akan memanggil variabel
....: aplikasi.
....: print (' Dirga pake aplikasi: '
....:       Nama Aplikasi
0      Dirga pake aplikasi: VSCode
1      Dirga pake aplikasi: Atom
2      Dirga pake aplikasi: Sublime
3      Dirga pake aplikasi: Notepad++

```

Gambar 3.43 Membuat Aplikasi pakai pandas

3.2.2.2 Nomor 2

```

1 import numpy as dirga #Melakukan import library numpy menjadi nama
   sendiri yaitu dirga
2
3 matrix_x = dirga.eye(10) #Membuat sebuah matrix pake numpy dengan
   menggunakan fungsi eye
4 matrix_x #Mendeklarasikan matrix_x yang tadi dibuat
5
6 print (matrix_x) #print matrix_x yang tadi dibuat yang berbentuk 10
   x10

```

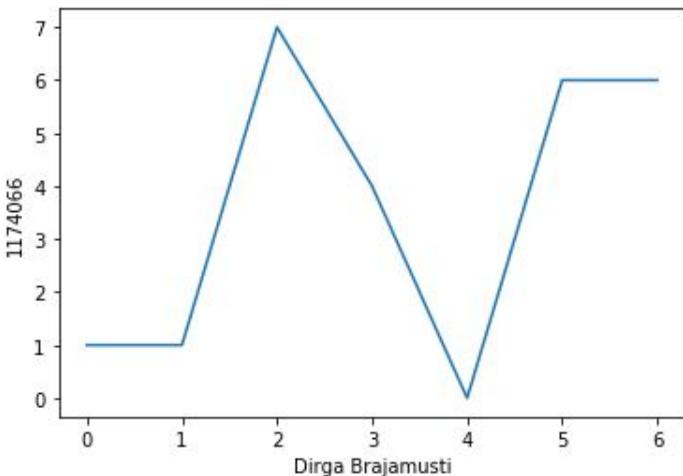
```
In [7]: import numpy as dirga #Melakukan
....:
....: matrix_x = dirga.eye(10) #Membuat
....: matrix_x #Menampilkan matriks
....:
....: print (matrix_x) #print matriks
[[1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 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 3.44 Membuat Aplikasi pakai numpy

3.2.2.3 Nomor 3

```
1 import matplotlib.pyplot as dirga #Melakukan import library numpy
   menjadi nama sendiri yaitu dirga
2 dirga.plot([1,1,7,4,0,6,6]) #Memasukkan nilai pada plot
3 dirga.xlabel('Dirga Brajamusti') #Menambahkan label pada x
4 dirga.ylabel('1174066') #Menambahkan label pada y
5 dirga.show() #Menampilkan grafik plot
```

```
In [8]: import matplotlib.pyplot as dirga #Melakukan import
...: dirga.plot([1,1,7,4,0,6,6]) #Memasukkan nilai pada list
...: dirga.xlabel('Dirga Brajamusti') #Menambahkan Label pada x
...: dirga.ylabel('1174066') #Menambahkan label pada y
...: dirga.show() #Menampilkan grafik plot
```



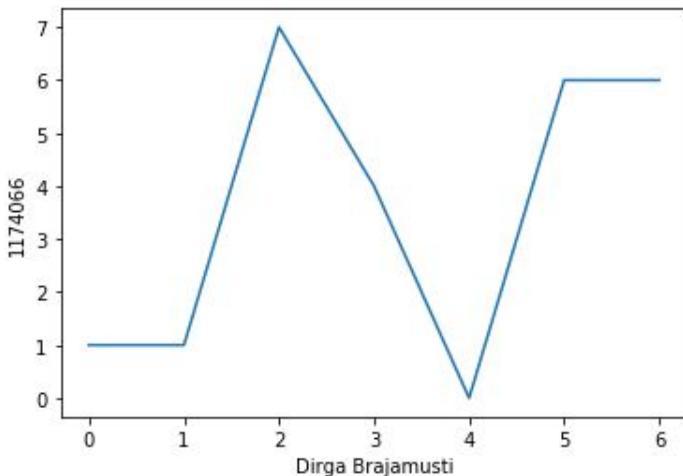
Gambar 3.45 Membuat Aplikasi pakai matplotlib

3.2.2.4 Nomor 4

- Source Code pertama pada random forest berfungsi untuk membaca dataset yang memiliki format text file dengan mendefinisikan variabel yang bernama imgatt. Variabel tersebut berisi value untuk membaca data.

```
1 import pandas as pd #Melakukan import library numpy menjadi pd
2
3 imgatt = pd.read_csv("N:/Tugas/Kuliah/Semester 6/Kecerdasan
   Buatan/KB3C Ngerjain/src/1174066/3/CUB_200_2011/attributes/
   image_attribute_labels.txt",
4                         sep='\s+', header=None, error_bad_lines=
   False, warn_bad_lines=False,
5                         usecols=[0,1,2], names=['imgid', 'attid', 'present']) #Membuat variable imgatt untuk membaca file csv
   dari dataset, dengan ketentuan yang ada.
```

```
In [8]: import matplotlib.pyplot as dirga #Melakukan import
....: dirga.plot([1,1,7,4,0,6,6]) #Memasukkan nilai pada list
....: dirga.xlabel('Dirga Brajamusti') #Menambahkan Label pada x
....: dirga.ylabel('1174066') #Menambahkan label pada y
....: dirga.show() #Menampilkan grafik plot
```



Gambar 3.46 Hasil 4 Bagian 1

- Pada source code berikutnya akan mengembalikan baris teratas dari DataFrame variabel imgatt.

```
1 imgatt.head() #Menampilkan data yang sudah dibaca tadi tapi cuman
   data paling atas
```

```
In [12]: import pandas as pd #Melakukan import library numpy menjadi pd
....:
....: imgatt = pd.read_csv("CIB_200_2011/attributes/image_attribute_labels.txt",
....: sep='\t', header=None, error_bad_lines=False, warn_bad_lines=False,
....: usecols=[0,1,2], names=['imgid', 'attid', 'present'],
file csv dari dataset, dengan ketentuan yang ada.
```

Gambar 3.47 Hasil 4 Bagian 2

- Pada output berikutnya akan menampilkan jumlah kolom dan baris dari DataFrame variable imgatt.

```
1 imgatt.head() #Menampilkan data yang sudah dibaca tadi tapi cuman
   data paling atas
```

```
In [13]: imgatt.head() #Menampilkan data yang sudah dibaca tadi tapi cuman data paling atas
Out[13]:
      imgid  attid  present
0        1       1        0
1        1       2        0
2        1       3        0
3        1       4        0
4        1       5        1
```

Gambar 3.48 Hasil 4 Bagian 3

- Variabel imgatt2 telah menggunakan fungsi yang bernama pivot agar mengubah kolom jadi baris dan sebaliknya dari DataFrame imgatt sebelumnya.

```
1 imgatt2 = imgatt.pivot(index='imgid', columns='attid', values='present') #Membuat sebuah variabel baru dari fungsi imgatt, dengan mengganti index menjadi kolom dan kolom menjadi index
```

```
In [15]: imgatt2 = imgatt.pivot(index='imgid', columns='attid', values='present') #Membuat sebuah variabel baru dari fungsi imgatt, dengan mengganti index menjadi kolom dan kolom menjadi index
```

Gambar 3.49 Hasil 4 Bagian 4

- Variabel imgatt2 head berfungsi untuk mengembalikan value teratas pada DataFrame imgatt2.

```
1 imgatt2.head() #Menampilkan data yang sudah dibaca tadi tapi cuman data paling atas
```

```
In [16]: imgatt2.head() #Menampilkan data yang sudah dibaca tadi tapi cuman data paling atas
Out[16]:
      attid  1   2   3   4   5   6   7   ...  306  307  308  309  310  311  312
      imgid
1        0   0   0   0   1   0   0   ...   0   0   1   0   0   0   0
2        0   0   0   0   0   0   0   ...   0   0   0   0   0   0   0
3        0   0   0   0   0   1   0   0   ...   0   0   1   0   0   0   1
4        0   0   0   0   1   0   0   0   ...   1   0   0   1   0   0   0
5        0   0   0   0   0   1   0   0   ...   0   0   0   0   0   0   0
```

[5 rows x 312 columns]

Gambar 3.50 Hasil 4 Bagian 5

- Menghasilkan jumlah kolom dan baris pada DataFrame imgatt2.

```
| imgatt2.shape #Menampilkan jumlah seluruh data , kolom-nya
```

In [17]: `imgatt2.shape` #Menampilkan jumlah seluruh data, kolom-nya
Out[17]: (11788, 312)

Gambar 3.51 Hasil 4 Bagian 6

- Menunjukkan dalam melakukan pivot yang mana imgid menjadi sebuah index yang unik.

```
| imglabels = pd.read_csv("N:/Tugas/Kuliah/Semester 6/Kecerdasan  
Buatan/KB3C_Ngerjain/src/1174066/3/CUB_200_2011/  
image_class_labels.txt",  
2 sep=' ', header=None, names=['imgid', '  
label']) #Membaca data dimasukkan ke variable imglabels  
3  
4 imglabels = imglabels.set_index('imgid') #Variable imglabels dan  
set index (imgid)
```

In [21]: `imglabels = pd.read_csv("CUB_200_2011/image_class_labels.txt",
...
imglabels
...
...
...: imglabels = imglabels.set_index('imgid') #Variable imglabels dan set index (imgid)`

Gambar 3.52 Hasil 4 Bagian 7

- Akan melakukan load jawabannya yang berisi apakah burung tersebut termasuk spesies yang mana. Kolom tersebut yaitu imgid dan label.

```
| imglabels.head() #Menampilkan data yang sudah dibaca tadi tapi  
cuman data paling atas
```

```
In [22]: imglabels.head()
Out[22]:
label
imgid
1      1
2      1
3      1
4      1
5      1
```

Gambar 3.53 Hasil 4 Bagian 8

- Menunjukkan bahwa jumlah baris sebanyak 11788 dan kolom 1 yang dimana kolom tersebut adalah jenis spesies pada burung.

```
1 imglabels.shape #Menampilkan jumlah seluruh data , kolom-nya
```

```
In [23]: imglabels.shape
Out[23]: (11788, 1)
```

Gambar 3.54 Hasil 4 Bagian 9

- Melakukan join antara imgatt2 dengan imglabels dikarenakan memiliki isi yang sama sehingga akan mendapatkan sebuah data ciri-ciri dan data jawaban sehingga bisa dikategorikan sebagai supervised learning.

```
1 df = imgatt2.join(imglabels) #Varibel df dimasukkan fungsi join
   dari data imgatt2 ke variabel imglabels
2 df = df.sample(frac=1) #Variabel df sebagai sample dengan
   ketentuan frac=1
```

```
In [24]: df = imgatt2.join(imglabels) #Variabel df dimasukkan fungsi join dari data imgatt2 ke variabel imglabels
...: df = df.sample(frac=1) #Variabel df sebagai sample dengan ketentuan frac=1
```

Gambar 3.55 Hasil 4 Bagian 10

- Melakukan drop pada label yang ada didepan dan akan menggunakan label yang baru di joinkan.

```
1 df_att = df.iloc[:, :312] #Membuat kolom dengan ketentuan 312
2 df_label = df.iloc[:, 312:] #Membuat kolom dengan ketentuan 312
```

```
In [25]: df_att = df.iloc[:, :312] #Membuat kolom dengan ketentuan 312
...: df_label = df.iloc[:, 312:] #Membuat kolom dengan ketentuan 312
```

Gambar 3.56 Hasil 4 Bagian 11

- Mengecek isi 5 data teratas pada df.att.

```
1 df_att.head() #Menampilkan data yang sudah dibaca tadi tapi cuman
    data paling atas
```

```
In [26]: df_att.head() #Menampilkan data yang sudah dibaca tadi tapi cuman data paling atas
Out[26]:
   1   2   3   4   5   6   7   ...  306  307  308  309  310  311  312
imgid
10397  0   0   0   0   0   0   1   ...   0   0   0   0   0   0   0   1
1108   0   0   0   0   0   0   1   ...   0   0   0   0   1   0   0   0
2382   0   0   0   0   0   0   1   ...   0   1   0   1   0   0   0   0
9374   0   0   0   0   0   0   1   ...   0   0   0   0   0   0   0   0
9371   0   0   0   0   0   0   0   ...   0   0   0   0   1   0   0   0
```

[5 rows x 312 columns]

Gambar 3.57 Hasil 4 Bagian 12

- Mengecek isi data teratas dari df.label.

```
1 df_label.head() #Menampilkan data yang sudah dibaca tadi tapi
    cuman data paling atas
```

```
In [27]: df_label.head()
Out[27]:
      label
imgid
10397    177
1108      20
2382      42
9374     160
9371     160
```

Gambar 3.58 Hasil 4 Bagian 13

- Membagi 8000 row pertama menjadi data training dan sisanya adalah data testing.

```
1 df_train_att = df_att[:8000] #Data akan dibagi dari 8000 row
   pertama menjadi data training dan sisanya adalah data testing
2 df_train_label = df_label[:8000] #Data akan dibagi dari 8000 row
   pertama menjadi data training dan sisanya adalah data testing
3 df_test_att = df_att[8000:] #Berbalik dari sebelumnya data akan
   dibagi mulai dari 8000 row pertama menjadi data training dan
   sisanya adalah data testing
4 df_test_label = df_label[8000:] #Berbalik dari sebelumnya data
   akan dibagi mulai dari 8000 row pertama menjadi data training
   dan sisanya adalah data testing
5
6 df_train_label = df_train_label['label'] #Menambahkan label
7 df_test_label = df_test_label['label'] #Menambahkan label
```

```
In [29]: df_train_att = df_att[:8000] #Data akan dibagi dari 8000 row pertama menjadi data training dan sisanya adalah data
testing
...: df_train_label = df_label[:8000] #Data akan dibagi dari 8000 row pertama menjadi data training dan sisanya adalah
data testing
...: df_test_att = df_att[8000:] #Berbalik dari sebelumnya data akan dibagi mulai dari 8000 row pertama menjadi data
training dan sisanya adalah data testing
...: df_test_label = df_label[8000:] #Berbalik dari sebelumnya data akan dibagi mulai dari 8000 row pertama menjadi data
training dan sisanya adalah data testing
...:
...: df_train_label = df_train_label['label'] #Menambahkan label
...: df_test_label = df_test_label['label'] #Menambahkan label
```

Gambar 3.59 Hasil 4 Bagian 14

- Pemanggilan class RandomForestClassifier. Dimana artinya menunjukkan banyak kolom pada setiap tree adalah 50.

```
1 from sklearn.ensemble import RandomForestClassifier #Import fungsi randomforestclassifier
2 clf = RandomForestClassifier(max_features=50, random_state=0,
   n_estimators=100) #clf sebagai variabel untuk klafifikasi
   random forest
```

In [30]: from sklearn.ensemble import RandomForestClassifier #import randomforestclassifier
...: clf = RandomForestClassifier(max_features=50, random_state=0, n_estimators=100) #clf sebagai variabel untuk klafifikasi random forest

Gambar 3.60 Hasil 4 Bagian 15

- Menunjukkan hasil prediksi dari Random Forest.

```
1 clf.fit(df_train_att, df_train_label) #Variabnle clf untuk fit
   yaitu menjadi data training
```

In [31]: clf.fit(df_train_att, df_train_label) #Variabnle clf untuk fit yaitu menjadi data training
Out[31]:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
 max_depth=None, max_features=50, max_leaf_nodes=None,
 min_impurity_decrease=0.0, min_impurity_split=None,
 min_samples_leaf=1, min_samples_splits=2,
 min_weight_fraction_leaf=0.0, n_estimators=100,
 n_jobs=None, oob_score=False, random_state=0, verbose=0,
 warm_start=False)

Gambar 3.61 Hasil 4 Bagian 16

- Menampilkan besaran akurasi dari prediksi pada Random Forest yang merupakan score perolehan klarifikasi.

```
1 print(clf.predict(df_train_att.head())) #Print clf yang di sudah
   prediksi dari training tetapi hanya menampilkan data paling
   atas
```

In [32]: print(clf.predict(df_train_att.head()))
data paling atas
[177 20 42 160 160]

Gambar 3.62 Hasil 4 Bagian 17

- Menampilkan besaran akurasi dari prediksi pada Random Forest yang merupakan score perolehan klarifikasi.

```
1 clf.score(df_test_att, df_test_label) #Memunculkan clf sebagai
   testing yang sudah di training tadi
```

```
In [33]: clf.score(df_test_att, df_test_label)
Out[33]: 0.44931362196409713
```

Gambar 3.63 Hasil 4 Bagian 18

3.2.2.5 Nomor 5

```
1 from sklearn.metrics import confusion_matrix #Mengimport Confusion
   Matrix
2 pred_labels = clf.predict(df_test_att) #Membuat variable pred_labels
   dari data testing
3 cm = confusion_matrix(df_test_label, pred_labels) #cm sebagai
   variabel data label
```

```
In [34]: from sklearn.metrics import confusion_matrix #Mengimport Confusion Matrix
...: pred_labels = clf.predict(df_test_att) #Membuat variable pred_label yang data testing dari sebelumnya
...: cm = confusion_matrix(df_test_label, pred_labels) #Variable cm sebagai variabel data label
```

Gambar 3.64 Hasil 5 Bagian 1

```
1 cm #Memunculkan data label berbentuk array
```

```
In [35]: cm #Menampilkan data label berbentuk array
Out[35]:
array([[ 6,  0,  1, ...,  0,  0,  0],
       [ 0, 11,  0, ...,  0,  0,  0],
       [ 5,  1,  9, ...,  0,  0,  0],
       ...,
       [ 1,  0,  0, ...,  1,  0,  0],
       [ 0,  1,  0, ...,  0,  8,  0],
       [ 0,  0,  0, ...,  0,  0, 19]], dtype=int64)
```

Gambar 3.65 Hasil 5 Bagian 2

```
1 import matplotlib.pyplot as plt #Mengimport library matplotlib
   sebagai plt
2 import itertools #Mengimport library itertools
3 def plot_confusion_matrix(cm, classes,
                           normalize=False,
                           title='Confusion matrix',
                           cmap=plt.cm.Blues): #Membuat fungsi dengan
   ketentuan data yang ada pada cm
4
5
6
7
8     if normalize:
9         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
10        print("Normalized confusion matrix") #Jika normalisasi
   sebagai ketentuan yang ada maka print normalized confusion matrix
```

```

11 else:
12     print('Confusion matrix, without normalization') #Jika tidak
memenuhi kondisi if maka print else
13
14 print(cm) #Print data cm
15
16 plt.imshow(cm, interpolation='nearest', cmap=cmap) #plt sebagai
fungsi untuk membuat plot
17 plt.title(title) #Membuat title pada plot
18 #plt.colorbar()
19 tick_marks = np.arange(len(classes)) #Membuat marks pada plot
20 plt.xticks(tick_marks, classes, rotation=90) #Membuat ticks pada
x
21 plt.yticks(tick_marks, classes) #Membuat ticks pada y
22
23 fmt = '.2f' if normalize else 'd' #fmt sebagai normalisasi
24 thresh = cm.max() / 2. #Variable thresh menambil data max pada
cm kemudian dibagi 2
25
26 plt.tight_layout() #Mengatur layout pada plot
27 plt.ylabel('True label') #Menambahkan nama label pada sumbu y
28 plt.xlabel('Predicted label') #Menambahkan nama label pada sumbu
x

```

```

In [36]: import matplotlib.pyplot as plt #Mengimport library matplotlib sebagai plt
...: import itertools #Mengimport library itertools
...: def plot_confusion_matrix(cm, classes,
...:                         normalize=False,
...:                         title='Confusion matrix',
...:                         cmap=plt.cm.Blues): #Membuat fungsi dengan ketentuan data yang ada pada cm
...:
...:     if normalize:
...:         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
...:         print("Normalized confusion matrix") #jika normalisasi sebagai ketentuan yang ada maka print normalized
confusion matrix
...:     else:
...:         print('Confusion matrix, without normalization') #jika tidak memenuhi kondisi if maka print else
...:
...:     print(cm) #print data cm
...:
...:     plt.imshow(cm, interpolation='nearest', cmap=cmap) #plt sebagai fungsi untuk membuat plot
...:     plt.title(title) #membuat title pada plot
...:     #plt.colorbar()
...:     tick_marks = np.arange(len(classes)) #membuat marks pada plot
...:     plt.xticks(tick_marks, classes, rotation=90) #membuat ticks pada x
...:     plt.yticks(tick_marks, classes) #membuat ticks pada y
...:
...:     fmt = '.2f' if normalize else 'd' #fmt sebagai normalisasi
...:     thresh = cm.max() / 2. #variabel thresh menambil data max pada cm kemudian dibagi 2
...:
...:     plt.tight_layout() #mengatur layout pada plot
...:     plt.ylabel('True label') #memberi nama label pada sumbu y
...:     plt.xlabel('Predicted label') #memberi nama label pada sumbu x

```

Gambar 3.66 Hasil 5 Bagian 3

```

1 birds = pd.read_csv("N:/Tugas/Kuliah/Semester 6/Kecerdasan Buatan/
KB3C Ngerjain/src/1174066/3/CUB_200_2011/classes.txt",
2                     sep='\s+', header=None, usecols=[1], names=['
birdname']) #membaca csv dengan ketentuan nama birdname
3 birds = birds['birdname'] #nama birds dengan ketentuan birdname
4 birds #Menampilkan data birds

```

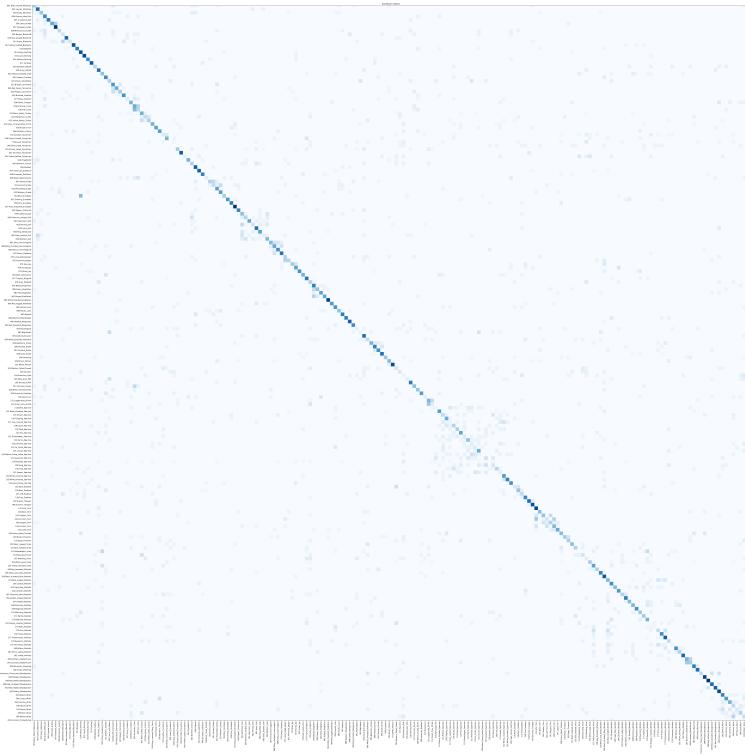
```
In [37]: birds = pd.read_csv("CUB_200_2011/cla
...:                                     sep='\t', header=0)
birdname
...: birds = birds['birdname'] #nama birds
...: birds #menampilkan data birds
Out[37]:
0      001.Black_footed_Albatross
1      002.Laysan_Albatross
2      003.Sooty_Albatross
3      004.Groove_billed_Ani
4      005.Crested_Auklet
...
195      196.House_Wren
196      197.Marsh_Wren
197      198.Rock_Wren
198      199.Winter_Wren
199      200.Common_Yellowthroat
Name: birdname, Length: 200, dtype: object
```

Gambar 3.67 Hasil 5 Bagian 4

```
1 import numpy as np #Mengimport library numpy sebagai np
2 np.set_printoptions(precision=2) #np sebagai variabel yang membuat
   set precision=2
3 plt.figure(figsize=(60,60), dpi=300) #Plot sebagai figure dengan
   ketentuan size 60,60 dan dpi 300
4 plot_confusion_matrix(cm, classes=birds, normalize=True) #Data cm dan
   clas birds dibuat sebagai plot
5 #plt.show()
6 #plt.savefig('hasil.png') #
```

```
In [39]: import numpy as np #import library numpy sebagai np
...: np.set_printoptions(precision=2) #np sebagai variabel yang membuat set precision=2
...: plt.figure(figsize=(60,60), dpi=300) #plot sebagai figure dengan ketentuan size 60,60 dan dpi 300
...: plot_confusion_matrix(cm, classes=birds, normalize=True) #data cm dan clas birds dibuat sebagai plot
...: plt.savefig('hasil.png')
Normalized confusion matrix
[[0.37 0. 0. ... 0. 0.05 0. ]
 [0.08 0.77 0. ... 0. 0. 0. ]
 [0.04 0.04 0.39 ... 0. 0. 0. ]
 ...
 [0. 0. 0. ... 0.31 0. 0. ]
 [0. 0. 0. ... 0. 0.24 0. ]
 [0. 0. 0. ... 0. 0.04 0.73]]
```

Gambar 3.68 Hasil 5 Bagian 5



Gambar 3.69 Plot Hasil 5 Bagian 5

3.2.2.6 Nomor 6

```

1 from sklearn import tree #Mengimport library tree
2 clftree = tree.DecisionTreeClassifier() #clftree sebagai variabel
   untuk decision tree
3 clftree.fit(df_train_att, df_train_label) #Mengatur data training
4 clftree.score(df_test_att, df_test_label) #Mengatur data testing

```

```

In [24]: from sklearn import tree #Mengimport library
...: clf = tree.DecisionTreeClassifier() #clf
...: clf.fit(df_train_att, df_train_label) #M
...: clf.score(df_test_att, df_test_label) #M
Out[24]: 0.264519535374868

```

Gambar 3.70 Hasil 6 Bagian 1

```

1 from sklearn import svm #Mengimport library svm
2 clfsvm = svm.SVC() #clfsvm sebagai variabel untuk mengatur fungsi SVC

```

```
3 clfsvm.fit(df_train_att, df_train_label) #Mengatur data training
4 clfsvm.score(df_test_att, df_test_label) #Mengatur data testing
```

```
In [31]: from sklearn.model_selection import
...: scores = cross_val_score(clf, df_
dari data training
...: print("Accuracy: %0.2f (+/- %0.2f"
C:\ProgramData\Anaconda3\lib\site-packages\scikit-learn\utils\validation.py:140: UserWarning: From version 0.22 onwards, if no scoring is passed to cross_val_score() or cross_val_predict(), it will default from 'auto' to 'scale' in version 0.22 to avoid this warning.
"avoid this warning.", FutureWarning)
Accuracy: 0.27 (+/- 0.01)
```

Gambar 3.71 Hasil 6 Bagian 2**3.2.2.7 Nomor 7**

```
1 from sklearn.model_selection import cross_val_score #Mengimport
    cross_val_score
2 scores = cross_val_score(clf, df_train_att, df_train_label, cv=5) #Membuat variable scores sebagai variabel prediksi dari data
    training
3 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() *
    2)) #Print data scores dengan ketentuan akurasi
```

```
In [32]: scorestree = cross_val_score(scores dan metode tree
...: print("Accuracy: %0.2f ada")
Accuracy: 0.44 (+/- 0.02)
```

Gambar 3.72 Hasil 7 Bagian 1

```
1 scorestree = cross_val_score(clftree, df_train_att, df_train_label,
    cv=5) #Membuat variable prediksi menggunakan scores dan metode
    tree
2 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.
    std() * 2)) #Menampilkan dengan ketentuan yang ada
```

```
In [28]: scorestree = cross_val_scores dan metode tree
....: print("Accuracy: %0.2f ada
Accuracy: 0.27 (+/- 0.02)
```

Gambar 3.73 Hasil 7 Bagian 2

```
1 scoresvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv
=5) #Membuat variable data training
2 print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2)) #Menampilkan data testing dan output akurasi
```

```
In [27]: scoresvm = cross_val_score(clfsvm, df_train_att, df_train_label, cv=5) #Membuat variable data training
....: print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(), scoresvm.std() * 2)) #Menampilkan data testing dan output akurasi
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 warning.
    "avoid this warning.", FutureWarning)
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 warning.
    "avoid this warning.", FutureWarning)
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 warning.
    "avoid this warning.", FutureWarning)
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 warning.
    "avoid this warning.", FutureWarning)
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 warning.
    "avoid this warning.", FutureWarning)
Accuracy: 0.27 (+/- 0.03)
```

Gambar 3.74 Hasil 7 Bagian 3

3.2.2.8 Nomor 8

```
1 max_features_opts = range(5, 50, 5) #Variable max_features_opts
    sebagai variabel untuk membuat range 5,50,5
2 n_estimators_opts = range(10, 200, 20) #Variablen_estimators_opts
    sebagai variabel untuk membuat range 10,200,20
3 rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts))
    ,4), float) #Variablerf_params sebagai variabel untuk
    menjumlahkan yang sudah di tentukan sebelumnya
4 i = 0
5 for max_features in max_features_opts: #Perulangan
6     for n_estimators in n_estimators_opts: #Perulangan
7         clf = RandomForestClassifier(max_features=max_features,
n_estimators=n_estimators) #Menampilkan variabel csf
8         scores = cross_val_score(clf, df_train_att, df_train_label,
cv=5) #Variable scores sebagai variabel training
9         rf_params[i,0] = max_features #index 0
```

```

10     rf_params[i,1] = n_estimators #index 1
11     rf_params[i,2] = scores.mean() #index 2
12     rf_params[i,3] = scores.std() * 2 #index 3
13     i += 1 #Dengan ketentuan i += 1
14     print("Max features: %d, num estimators: %d, accuracy: %0.2f
15 (+/- %0.2f)" % (max_features, n_estimators, scores.mean(),
    scores.std() * 2))
        #Print hasil pengulangan yang sudah ditentukan

```

```

....          ....
....      rf_params[i,3] = scores.std() * 2 #index 3
....      i += 1 #Dengan ketentuan i += 1
....      print("Max features: %d, num estimators: %d, acc
n_estimators, scores.mean(), scores.std() * 2))
....      #Print hasil pengulangan yang sudah ditentukan
Max features: 5, num estimators: 10, accuracy: 0.26 (+/- 0.03)
Max features: 5, num estimators: 30, accuracy: 0.36 (+/- 0.03)
Max features: 5, num estimators: 50, accuracy: 0.39 (+/- 0.03)
Max features: 5, num estimators: 70, accuracy: 0.41 (+/- 0.02)
Max features: 5, num estimators: 90, accuracy: 0.42 (+/- 0.02)
Max features: 5, num estimators: 110, accuracy: 0.43 (+/- 0.03)
Max features: 5, num estimators: 130, accuracy: 0.43 (+/- 0.02)
Max features: 5, num estimators: 150, accuracy: 0.44 (+/- 0.02)
Max features: 5, num estimators: 170, accuracy: 0.44 (+/- 0.03)
Max features: 5, num estimators: 190, accuracy: 0.45 (+/- 0.01)
Max features: 10, num estimators: 10, accuracy: 0.29 (+/- 0.02)
Max features: 10, num estimators: 30, accuracy: 0.38 (+/- 0.02)
Max features: 10, num estimators: 50, accuracy: 0.41 (+/- 0.03)
Max features: 10, num estimators: 70, accuracy: 0.43 (+/- 0.02)
Max features: 10, num estimators: 90, accuracy: 0.43 (+/- 0.02)
Max features: 10, num estimators: 110, accuracy: 0.44 (+/- 0.02)
Max features: 10, num estimators: 130, accuracy: 0.45 (+/- 0.03)
Max features: 10, num estimators: 150, accuracy: 0.45 (+/- 0.03)
Max features: 10, num estimators: 170, accuracy: 0.45 (+/- 0.02)
Max features: 10, num estimators: 190, accuracy: 0.45 (+/- 0.03)
Max features: 15, num estimators: 10, accuracy: 0.31 (+/- 0.03)
Max features: 15, num estimators: 30, accuracy: 0.40 (+/- 0.03)
Max features: 15, num estimators: 50, accuracy: 0.42 (+/- 0.03)
Max features: 15, num estimators: 70, accuracy: 0.43 (+/- 0.02)
Max features: 15, num estimators: 90, accuracy: 0.43 (+/- 0.03)
Max features: 15, num estimators: 110, accuracy: 0.44 (+/- 0.02)
Max features: 15, num estimators: 130, accuracy: 0.44 (+/- 0.02)
Max features: 15, num estimators: 150, accuracy: 0.45 (+/- 0.02)
Max features: 15, num estimators: 170, accuracy: 0.45 (+/- 0.02)
Max features: 15, num estimators: 190, accuracy: 0.45 (+/- 0.02)

```

Gambar 3.75 Hasil 8 Bagian 1

```

Max features: 40, num estimators: 190, accuracy: 0.45 (+/- 0.02)
Max features: 45, num estimators: 10, accuracy: 0.34 (+/- 0.02)
Max features: 45, num estimators: 30, accuracy: 0.41 (+/- 0.02)
Max features: 45, num estimators: 50, accuracy: 0.43 (+/- 0.02)
Max features: 45, num estimators: 70, accuracy: 0.44 (+/- 0.03)
Max features: 45, num estimators: 90, accuracy: 0.44 (+/- 0.03)
Max features: 45, num estimators: 110, accuracy: 0.45 (+/- 0.03)
Max features: 45, num estimators: 130, accuracy: 0.45 (+/- 0.02)
Max features: 45, num estimators: 150, accuracy: 0.45 (+/- 0.03)
Max features: 45, num estimators: 170, accuracy: 0.45 (+/- 0.03)
Max features: 45, num estimators: 190, accuracy: 0.45 (+/- 0.02)

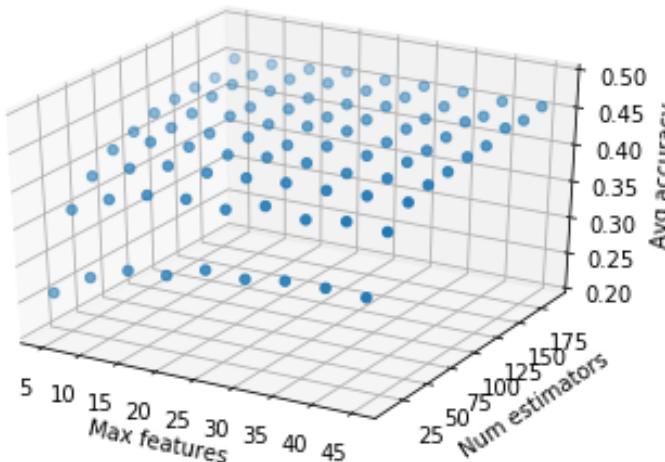
```

Gambar 3.76 Hasil 8 Bagian 1 Akhir kode

```

1 import matplotlib.pyplot as plt #Mengimport library matplotlib
2 sebagai plt
3 from mpl_toolkits.mplot3d import Axes3D #Mengimport axes3D untuk
4 menampilkan plot 3 dimensi
5 from matplotlib import cm #Memanggil data cm yang sudah tersedia
6 fig = plt.figure() #Menghasilkan plot sebagai figure
7 fig.clf() #Figure di ambil dari clf
8 ax = fig.gca(projection='3d') #ax sebagai projection 3d
9 x = rf_params[:,0] #x sebagai index 0
10 y = rf_params[:,1] #y sebagai index 1
11 z = rf_params[:,2] #z sebagai index 2
12 ax.scatter(x, y, z) #Membuat plot scatter x y z
13 ax.set_zlim(0.2, 0.5) #Set zlim dengan ketentuan yang ada
14 ax.set_xlabel('Max features') #Memberikan nama label x
15 ax.set_ylabel('Num estimators') #Memberikan nama label y
16 ax.set_zlabel('Avg accuracy') #Memberikan nama label z
17 plt.show() #Print hasil plot yang sudah dibuat.

```

**Gambar 3.77** Hasil 8 Bagian 2

3.2.3 Penanganan Error

1. ScreenShoot Error



```
[Errno 2] file 'B:\Tugas\Tugas\Kelas\semester 6\Kode\Python\Aplikasi\CSV\ApplikasiCSV\CSV\CSV_001.csv' [Errno 2] file 'B:\Tugas\Tugas\Kelas\semester 6\Kode\Python\Aplikasi\CSV\CSV_001.csv' [Errno 2] file 'B:\Tugas\Tugas\Kelas\semester 6\Kode\Python\Aplikasi\CSV\CSV_001.csv'
```

Gambar 3.78 FileNotFoundError

2. Tuliskan Kode Error dan Jenis Error

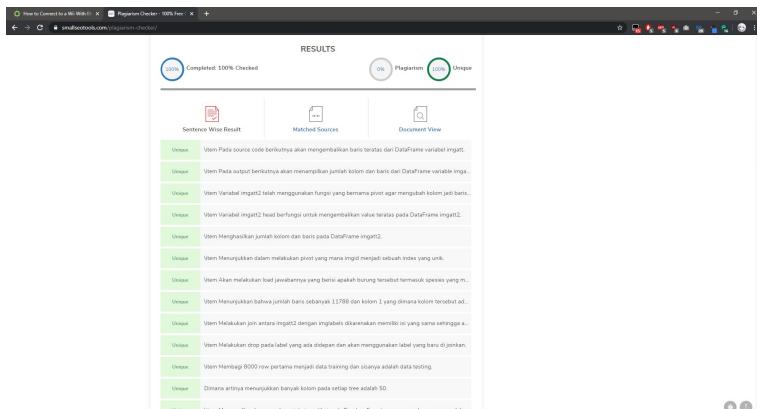
- FileNotFoundError

3. Cara Penanganan Error

- FileNotFoundError

Error terdapat pada kesalahan saat baca file csv, yang tidak terbaca. Dikarenakan letak file yang dibaca tidak pada direktori yang sama. Seharusnya letakkan file di direktori yang sama.

3.2.4 Bukti Tidak Plagiat



Gambar 3.79 Bukti Tidak Plagiat

BAB 4

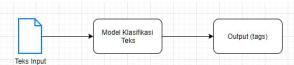
CHAPTER 4

4.1 Mochamad Arifqi Ramadhan (1174074)

4.1.1 Teori

4.1.1.1 Klasifikasi Teks

Klasifikasi teks merupakan sebuah model yang biasa digunakan untuk mengkategorikan sebuah teks ke dalam kelompok-kelompok yang lebih terorganisir. Jadi untuk setiap kalimat yang di masukan ke dalam mesin, mesin tersebut akan menjadikan setiap kata dari kalimat tersebut menjadi sebuah kolom. Untuk ilustrasinya bisa dilihat pada gambar berikut :



Gambar 4.1 Klasifikasi Teks.

4.1.1.2 Jelaskan mengapa hal ini bisa terjadi, klasifikasi bunga tidak bisa digunakan untuk machine learning

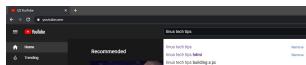
Karena machine learning tidak dapat menampilkan inputan sesuai dengan apa yang kita inputkan. Karena inputan tersebut serupa namun mesin memberikan output yang berbeda, biasanya output atau error ini disebut dengan istilah noise. Untuk contoh sederhananya misalkan kita inputkan salah satu label yang terdapat pada bunga, output yang dihasilkan oleh mesin tersebut ialah label yang lain. Itu dikarenakan bunga banyak jenis yang serupa namun tidak sama. Untuk ilustrasinya bisa dilihat pada gambar berikut:



Gambar 4.2 Klasifikasi Bunga.

4.1.1.3 Jelaskan bagaimana yang dimaksud dengan teknik pembelajaran mesin pada teks yang digunakan

Teknik yang digunakan pada youtube salah satunya ialah keywords. Dengan keywords tersebut mesin dapat memberikan video sesuai dengan keyword yang kita inputkan pada kolom pencarian. Teknik pembelajarannya tergantung user memberikan input teks seperti apa, karena pada youtube itu sendiri akan menyesuaikan dengan apa yang biasa kita inputkan dan akan memfilter video secara otomatis sesuai dengan keyword yang biasa kita inputkan. Contoh ilustrasi sederhananya seperti berikut:



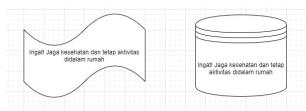
Gambar 4.3 Klasifikasi Teks pada Youtube.

4.1.1.4 Vektorisasi Data

Vektorisasi data ialah suatu pemecahan atau pembagian data berupa teks, sebagai contoh terdapat 5 paragraf, data teks tersebut di pecah menjadi kalimat-kalimat yang lebih sederhana, lalu di pecah lagi menjadi kata untuk setiap kalimatnya.

4.1.1.5 Bag of Words

Representasi penyederhanaan sebuah kalimat atau perhitungan setiap kata pada suatu kalimat dengan presentase berapa kali muncul kata tersebut untuk setiap kalimatnya. Contoh ilustrasi sederhananya seperti berikut:



Gambar 4.4 Bag of Words.

4.1.1.6 TF-IDF

TF-IDF merupakan metode untuk menghitung bobot setiap kata pada suatu kalimat

yang paling sering digunakan. TF-IDF ini akan menghitung nilai Term Frequency dan Inverse Document Frequency pada setiap kata dalam setiap kalimat yang muncul dengan diimbangi dengan jumlah dokumen dalam korpus yang mengandung kata. Contoh ilustrasi sederhananya seperti gambar berikut:

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

	Doc 1	Doc 2	...	Doc n
Term(s) 1	12	2	...	1
Term(s) 2	0	1	...	0
...
Term(s) n	0	6	...	3

Gambar 4.5 TF-IDF.

4.1.2 Praktek Program

4.1.2.1 Nomor 1

```
1 import pandas as pd #digunakan untuk mengimport library pandas dengan
    alias pd
2 pd = pd.read_csv("N:/Tugas/Kuliah/Semester 6/Kecerdasan Buatan/KB3C
    Ngerjain/src/1174066/4/csv.csv") #membaca file csv
```

Gambar 4.6 Nomor 1

4.1.2.2 Nomor 2

```
1 d_train=pd[:450] #membagi data training menjadi 450
2 d_test=pd[450:] #membagi data menjadi 50 atau sisa dari data yang
    tersedia
```

Gambar 4.7 Nomor 2

4.1.2.3 Nomor 3

```
1 import pandas as pd #digunakan untuk mengimport library pandas dengan
    alias pd
```

```

2 d = pd.read_csv("N:/Tugas/Kuliah/Semester 6/Kecerdasan Buatan/KB3C
3   Ngerjain/src/1174066/4/Youtube04-Eminem.csv") #Membaca file csv
4
5 from sklearn.feature_extraction.text import CountVectorizer #import
6   fungsi countvectorize dari sklearn
7 vectorizer = CountVectorizer() #membuat instansi CountVectorizer
8
9 dvec = vectorizer.fit_transform(d['CONTENT']) #Memasukkan data ke
10   dvec
11 dvec #Melihat data yang dimasukkan ke dvec
12
13 daptarkata = vectorizer.get_feature_names() #Mendapatkan data dan
14   memasukkannya ke daptarkata
15
16 dshuf = d.sample(frac=1) #Memasukkan sample kedalam variable dshuf
17
18 d_train = dshuf[:300] #Membuat data training
19 d_test = dshuf[300:] #Membuat data test
20
21 d_train_att = vectorizer.fit_transform(d_train['CONTENT']) ##
22   Memasukkan data training dari vectorizer
23 d_train_att #Melihat data training
24
25 d_test_att = vectorizer.transform(d_test['CONTENT']) #Memasukkan data
26   test dari vectorizer
27 d_test_att #Melihat data training
28
29 d_train_label = d_train['CLASS'] #Memberi label
30 d_test_label = d_test['CLASS'] #Memberi Label

```

```

In [1]: d = pd.read_csv("N:/Tugas/Kuliah/Semester 6/Kecerdasan Buatan/KB3C
... Ngerjain/src/1174066/4/Youtube04-Eminem.csv")
... from sklearn.feature_extraction.text import CountVectorizer
... vectorizer = CountVectorizer()
... dvec = vectorizer.fit_transform(d['CONTENT'])
... dvec
... daptarkata = vectorizer.get_feature_names()
... daptarkata
... dshuf = d.sample(frac=1)
... d_train = dshuf[:300]
... d_test = dshuf[300:]
... d_train_att = vectorizer.fit_transform(d_train['CONTENT'])
... d_train_att
... daptarkata
... daptarkata
... d_train_label = d_train['CLASS']
... d_test_label = d_test['CLASS']

Out[1]:
<ipython-input-1-1234567890>
In [2]: d = pd.read_csv("N:/Tugas/Kuliah/Semester 6/Kecerdasan Buatan/KB3C
... Ngerjain/src/1174066/4/Youtube04-Eminem.csv")
... from sklearn.feature_extraction.text import CountVectorizer
... vectorizer = CountVectorizer()
... dvec = vectorizer.fit_transform(d['CONTENT'])
... dvec
... daptarkata = vectorizer.get_feature_names()
... daptarkata
... dshuf = d.sample(frac=1)
... d_train = dshuf[:300]
... d_test = dshuf[300:]
... d_train_att = vectorizer.fit_transform(d_train['CONTENT'])
... d_train_att
... daptarkata
... daptarkata
... d_train_label = d_train['CLASS']
... d_test_label = d_test['CLASS']

Out[2]:
<ipython-input-2-1234567890>

```

Gambar 4.8 Nomor 3

4.1.2.4 Nomor 4

```

1 from sklearn import svm #Mengimport svm dari sklearn
2 clfsvm = svm.SVC() #Membuat svc kedalam variable svm
3 clfsvm.fit(d_train_att, d_train_label) #Memprediksi data dari data
4   training
5 clfsvm.score(d_test_att, d_test_label) #Memunculkan clf sebagai
6   testing yang sudah di training tadi

```

Gambar 4.9 Nomor 4**4.1.2.5 Nomor 5**

```

1 from sklearn import tree #Mengimport tree dari sklearn
2 clftree = tree.DecisionTreeClassifier() #Membuat decision tree
3 clftree.fit(d_train_att , d_train_label) #Memprediksi data dari data
   training
4 clftree.score(d_test_att , d_test_label) #Memunculkan clf sebagai
   testing yang sudah di training tadi

```

```

In [45]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(d_train_att, d_train_label)
...: clftree.score(d_test_att, d_test_label)
Out[45]: 0.9594594594594594

```

Gambar 4.10 Nomor 5**4.1.2.6 Nomor 6**

```

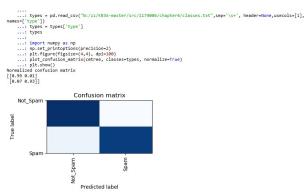
1 from sklearn.metrics import confusion_matrix
2 pred_labelstree = clftree.predict(d_test_att)
3 cmtree = confusion_matrix(d_test_label , pred_labelstree)
4 cmtree
5
6 import matplotlib.pyplot as plt
7 import itertools
8 def plot_confusion_matrix(cm, classes,
                           normalize=False,
                           title='Confusion matrix',
                           cmap=plt.cm.Blues):
9     """
10     This function prints and plots the confusion matrix.
11     Normalization can be applied by setting 'normalize=True'.
12     """
13     if normalize:
14         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
15         print("Normalized confusion matrix")
16     else:
17         print('Confusion matrix, without normalization')
18
19     print(cm)
20
21     plt.imshow(cm, interpolation='nearest', cmap=cmap)
22     plt.title(title)
23     #plt.colorbar()
24     tick_marks = np.arange(len(classes))
25     plt.xticks(tick_marks, classes, rotation=90)
26     plt.yticks(tick_marks, classes)
27
28     fmt = '.2f' if normalize else 'd'
29
30

```

```

32 thresh = cm.max() / 2.
33 #for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
34 #    plt.text(j, i, format(cm[i, j], fmt),
35 #              horizontalalignment="center",
36 #              color="white" if cm[i, j] > thresh else "black")
37
38 plt.tight_layout()
39 plt.ylabel('True label')
40 plt.xlabel('Predicted label')
41
42 types = pd.read_csv("N:/zz/KB3A-master/src/1174006/chapter4/classes.txt",sep='\s+', header=None, usecols=[1], names=['type'])
43 types = types['type']
44 types
45
46 import numpy as np
47 np.set_printoptions(precision=2)
48 plt.figure(figsize=(4,4), dpi=100)
49 plot_confusion_matrix(cmtree, classes=types, normalize=True)
50 plt.show()

```



Gambar 4.11 Nomor 6

4.1.2.7 Nomor 7

```

1 from sklearn.model_selection import cross_val_score
2
3 scorestree = cross_val_score(clftree, d_train_att, d_train_label, cv=5)
4 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(), scorestree.std() * 2))
5
6 scoressvm = cross_val_score(clfsvm, d_train_att, d_train_label, cv=5)
7 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(), scoressvm.std() * 2))
8
9 scores = cross_val_score(clf, d_train_att, d_train_label, cv=5)
10 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

```

Gambar 4.12 Nomor 7

4.1.2.8 Nomor 8

```

1 max_features_opts = range(5, 50, 5)
2 n_estimators_opts = range(10, 200, 20)
3 rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),
4 ,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,
9             n_estimators=n_estimators)
10        scores = cross_val_score(clf, d_train_att, d_train_label, cv
11        =5)
12        rf_params[i,0] = max_features
13        rf_params[i,1] = n_estimators
14        rf_params[i,2] = scores.mean()
15        rf_params[i,3] = scores.std() * 2
16        i += 1
17        print("Max features: %d, num estimators: %d, accuracy: %0.2f
18 (+/- %0.2f)" % (max_features, n_estimators, scores.mean(), scores
19 .std() * 2))

```

Gambar 4.13 Nomor 8

4.1.3 Penanganan Error

1. FileNotFoundError

```
File "pandas\_libs/parsers.pyx", line 689, in
pandas._libs.parsers.TextReader._setup_parser_source

FileNotFoundError: [Errno 2] File b'Ni/Tugas/a/Semester 6/Kecerdasan Buatan/K83C
Ngerjain/src/1174866/4.csv' does not exist: b'Ni/Tugas/a/Semester 6/Kecerdasan
Buatan/K83C Ngerjain/src/1174866/4.csv'
```

Gambar 4.14 FileNotFoundError

2. Cara Penangan Error

- **FileNotFoundException**

Dengan menyesuaikan letak file csv pada codingannya, sehingga dapat dibaca atau dipanggil file csvnya.

4.1.4 Bukti Tidak Plagiat



Gambar 4.15 Bukti Tidak Melakukan Plagiat Chapter 4

BAB 5

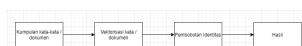
CHAPTER 5

5.1 1174067 - Kaka Kamaludin

5.1.1 Teori

5.1.1.1 *Jelaskan kenapa kata-kata harus di lakukan vektorisasi*

Algoritma machine learning perlu mengubah teks menjadi angka terlebih dahulu agar dapat dibaca oleh komputer, karena komputer hanya mengerti bahasa biner. Vektorisasi membantu mengubah teks kedalam bentuk vektor yang dapat dimengerti oleh komputer.

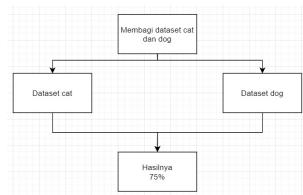


Gambar 5.1 Teori 1

5.1.1.2 *Jelaskan mengapa dimensi dari vektor dataset google bisa sampai 300.*

Karena setiap objek memiliki identitas khusus, misalnya sederhana. Dalam dataset Google ini memiliki 3 objek, kucing, anjing, dan ember yang disetujui. Kemudian

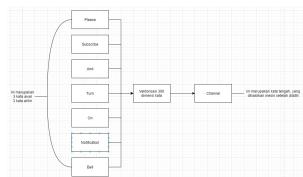
dari masing-masing objek dibandingkan dataset antara kucing dan anjing kemudian kucing dan ember. Hasil yang diperoleh untuk kucing dan anjing sekitar 75% sedangkan untuk kucing dan ember yang memiliki persentase 15%. Untuk lebih lengkapnya bisa dilihat pada gambar berikut:



Gambar 5.2 Teori 2

5.1.1.3 Jelaskan konsep vektorisasi untuk kata

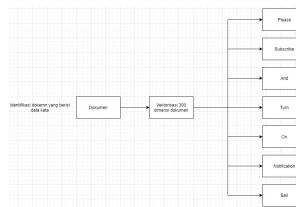
Konsep untuk vektorisasi kata sama dengan masukan atau input pada kata-kata di mesin pencari. Maka anggaplah itu akan dikeluarkan sebagai saran tentang kata tersebut. Jadi kata data diperoleh dari hasil yang diolah dalam kalimat sebelumnya yang telah diproses. Contoh sederhana dalam kalimat berikut (Please Subscribe and Turn on Notification Bell), dalam kalimat itu terkait dengan kalimat saluran, kata akan dibuat data pelatihan untuk mesin. Jadi kita kompilasi kata channel, maka mesin akan menampilkan hubungannya dengan kata tersebut. Contoh ilustrasi sederhananya seperti berikut:



Gambar 5.3 Teori 3

5.1.1.4 Jelaskan konsep vektorisasi untuk dokumen

Sama halnya dengan vektorisasi kata, yang membedakan hanya pada proses awalnya. Untuk vektorisasi dokumen ini, mesin akan membaca semua kalimat yang terdapat pada dokumen tersebut, lalu kalimat yang terdapat pada dokumen akan dipecah menjadi kata-kata. Perhatikan gambar berikut:

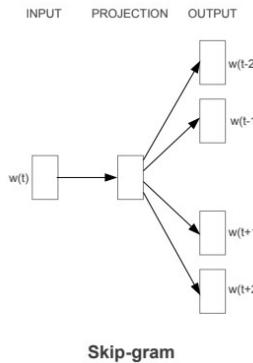
**Gambar 5.4** Teori 4**5.1.1.5 Jelaskan apa mean dan standar deviasi**

Mean adalah nilai rata-rata. Untuk mendapatkan makna ini, kita hanya perlu menambahkan data yang tersedia dan kemudian membaginya dengan jumlah data. Sedangkan standar deviasi adalah nilai statistik yang digunakan untuk menentukan bagaimana data didistribusikan dalam sampel, dan menentukan titik data individu dengan nilai rata-rata nilai sampel. Rumusnya ialah sebagai berikut:

Mean $X = \frac{X_1 + X_2 + \dots + X_n}{n}$ atau $X = \frac{\sum_{i=1}^n X_i}{n}$	$\bar{X} = \text{rata - rata}$ $\sum_{i=1}^n X_i = \text{jumlah seluruh nilai data}$ $n = \text{jumlah seluruh frekuensi}$
Rumus Varian $s^2 = \frac{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}{n(n-1)}$	
Rumus Standar Deviasi $s = \sqrt{\frac{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}{n(n-1)}}$	

Gambar 5.5 Teori 5**5.1.1.6 Jelaskan apa itu skip-gram**

Skip-gram sama halnya dengan vektorisasi kata, namun untuk skip-gram ia dibalik prosesnya. Yang sebelumnya dari kalimat lalu diolah untuk menemukan salah satu kata, kali ini dari keyword tersebut akan diolah menjadi suatu kalimat yang memiliki keterkaitannya dengan keyword tersebut. Contoh sederhananya seperti gambar berikut:

**Gambar 5.6** Teori 6

5.1.2 Praktek

5.1.2.1 Nomor 1

```

1 #%%
2 model_kk['faith']
3 #%%

```

Kode digunakan untuk import library gensim. Gensim itu sendiri berguna untuk melakukan pemodelan dengan dataset atau topik yang telah ditentukan. Untuk keluaran 100 logging itu library opsional karena logging hanya untuk menampilkan berupa log untuk setiap code yang dijalankan. Dan keluaran 101 itu hasil load data dari file vektor google itu, disini saya menggunakan limit karena kondisi laptop yang tidak mempunyai untuk melakukan running data sebesar 3 juta file, jadi saya batasi hanya melakukan running 500 ribu data saja, hasilnya ialah sebagai berikut:

```

[In 1]: import gensim, logging
        ...: logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
        ...: from gensim.models import Word2Vec
        ...: word2vec = Word2Vec.load_word2vec_format('GoogleNews-vectors-negative300.txt',
        ...:                                         limit=500000, unicode_errors='utf8')
[2020-05-25 21:31:09,387] [INFO] : Loading projection weights from /k/GoogleNews-vectors-negative300.txt
[2020-05-25 21:31:09,457] [INFO] : Loaded 500000 word vectors from /k/GoogleNews-vectors-negative300.txt

```

Gambar 5.7 Hasil dari Nomor 1-1

```

1 model_kk['sick']
2 #%%
3 model_kk['clear']
4 #%%
5 model_kk['shine']
6 #%%
7 model_kk['bag']
8 #%%
9 model_kk['car']
10 #%%
11 model_kk['wash']
12 #%%
13 model_kk['motor']
14 #%%
15 model_kk['cycle']
16 #%%
17 model_kk.similarity('wash', 'clear')
18 #%%
19 model_kk.similarity('bag', 'love')
20 #%%
21 model_kk.similarity('motor', 'car')

```

Menampilkan data hasil vektorisasi data:

```
In [4]: model_dirgs['love']

Out[4]:
array([-0.13534579,  0.05879191, -0.16939986, -0.16939986,
       0.068680453,  0.29369757, -0.26367188, -0.148625, -0.20171886,
      -0.02624512, -0.08283125, -0.02770996, -0.03510451, -0.23515156,
      -0.18055864,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.03857422,  0.07959884,  0.22949219, -0.14354569,  0.16798875,
     -0.03153625,  0.09517578,  0.18693353,  0.11161641, -0.16985954,
     -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.07715444,  0.26371875,  0.08642578, -0.92514646,  0.33398438,
     -0.18055864, -0.28998694,  0.07809778, -0.02609098, -0.16646151,
     -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.18985128,  0.14941406, -0.18693359,  0.01556396,  0.08094375,
     -0.18985128,  0.14941406, -0.18693359,  0.01556396,  0.08094375,
     -0.24316498,  0.08447286, -0.07808078,  0.18864648,  0.03515625,
     -0.09667959, -0.21972656, -0.08528054, -0.0318242,  0.18457031,
     -0.09667959, -0.21972656, -0.08528054, -0.0318242,  0.18457031,
     -0.09220518, -0.1894512,  0.01516072,  0.1855468,  0.343737 ,
     -0.31956588,  0.22559594,  0.08740254, -0.2280625, -0.29492185,
     -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.08958525,  0.23632812, -0.17871094, -0.12451172, -0.17281556,
     -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.19726562,  0.19324219,  0.09512484,  0.08515253,  0.12597656,
     -0.08673264, -0.0442832,  0.0363965,  0.01623553, -0.164625,
     -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.06494141,  0.08953217, -0.164862, -0.01367189,  0.18945312,
     -0.0566468,  0.08956853,  0.01422119,  0.15917969,  0.07421875,
     -0.146825 , -0.13183594, -0.12792669,  0.12898947,  0.05863789,
     -0.08662231,  0.01928711,  0.09423828, -0.13183594, -0.27929688,
     -0.08662231,  0.01928711,  0.09423828, -0.13183594,  0.11425781, -0.27734375,
     -0.31839598,  0.10858954,  0.10858954,  0.31839598,
```

Gambar 5.8 Data love

```
In [5]: model_dirgs['faith']

Out[5]:
array([ 0.23367188, -0.04158391,  0.1951312,  0.13476562, -0.14648438,
       0.11962891,  0.04345783,  0.18931562,  0.12207931,  0.13476562,
      -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.19381215,  0.18449219,  0.36132812, -0.1953125, -0.18148662,
     -0.15332831, -0.18639644,  0.18239986, -0.01367189,  0.23144531,
     -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.17828132,  0.08673264,  0.01135254,  0.0211116,  0.18554688,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562,
     -0.02818624,  0.01257234, -0.09512484, -0.18666486,  0.146825 ,
     -0.02818624,  0.01257234, -0.09512484, -0.18666486,  0.146825 ,
     -0.27350982, -0.20095469,  0.10521562, -0.29214584, -0.1875 ,
     -0.10992188,  0.13574219,  0.13769531,  0.16354954, -0.03818356,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562,
     -0.02331543, -0.04248497, -0.08283125,  0.16915625,  0.04158931,
     -0.16681562, -0.13671875,  0.09619141,  0.32617188,  0.08251593,
     -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.17828132,  0.22469698,  0.03466797,  0.19824219, -0.08837091,
     -0.18593757,  0.07324219,  0.1178175, -0.53967575,  0.16796875,
     -0.18593757,  0.07324219,  0.1178175, -0.53967575,  0.16796875,
     -0.15234375,  0.02966389,  0.3389125,  0.28320112,  0.09375 ,
     -0.15234375,  0.02966389,  0.3389125,  0.28320112,  0.09375 ,
     -0.08673264,  0.02966389,  0.3389125,  0.28320112,  0.09375 ,
     -0.25926785,  0.2558938,  0.08436298,  0.146825,  0.05818547,
     -0.21582031,  0.0291728,  0.02929688,  0.28619531,  0.34669838,
     -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.05688477,  0.18490847, -0.04785156, -0.15136719, -0.07714844,
     -0.45894438, -0.29492188, -0.322125, -0.28996094,  0.38671875,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562,
     -0.05837695, -0.15234375,  0.07519531, -0.1374219, -0.10642669,
     -0.08521484,  0.07861528,  0.04793528,  0.3785125, -0.02324375 ,
     -0.25852552, -0.12866486,  0.08436298,  0.146825,  0.05818547,
```

Gambar 5.9 Data faith

```
In [6]: model_dirgs['fall']

Out[6]:
array([-0.04727461,  0.14942188, -0.00277344,  0.168094531, -0.1328125 ,
       0.16693359,  0.04212289,  0.01964207,  0.14646438,  0.15039862,
      -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.13835156,  0.01929229, -0.08389781, -0.01852089, -0.1953125 ,
     -0.1815625,  0.13671875,  0.09228516, -0.12180975,  0.12695512,
     -0.18985128,  0.14516516, -0.15437232, -0.16061322, -0.14332227,
     -0.26953125, -0.0888877, -0.07763672, -0.15527344, -0.0393555 ,
     -0.08673264,  0.25895325,  0.25895325,  0.25895325, -0.02324375 ,
     -0.1374219, -0.25895325,  0.37798052, -0.07354457, -0.0838984 ,
     -0.1935938,  0.0546875, -0.04956985,  0.16915625, -0.0393555 ,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562,
     -0.03335954,  0.08234061,  0.01422119, -0.01397705, -0.03935585 ,
     -0.025527 ,  0.06229882,  0.0780807, -0.47617188,  0.05452969 ,
     -0.2189375 ,  0.0812795 ,  0.08798962,  0.19623986,  0.1137955 ,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562,
     -0.06225586, -0.37609312, -0.08737385, -0.08354844,  0.08893775 ,
     -0.00448686, -0.14100156, -0.04541865, -0.4781125,  0.0608559 ,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562,
     -0.03466797,  0.04830872,  0.1483753 ,  0.01789884, -0.08789862 ,
     -0.1494466 , -0.02331543, -0.0395878 , -0.10460391, -0.1416156 ,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562 ,
     -0.1289625 ,  0.18957903,  0.17578125,  0.08660453 ,  0.3496938 ,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562 ,
     -0.0254375 , -0.8694941 ,  0.09057969, -0.04539864,  0.04958955 ,
     -0.08673264,  0.08421278, -0.164625 , -0.03274484,  0.0780125 ,
     -0.20019531 ,  0.08886719, -0.18847556, -0.23823125,  0.07578789 ,
     -0.14453125 , -0.18695894, -0.0853737 ,  0.18289587, -0.03935725 ,
     -0.08673264,  0.12011719,  0.17488609, -0.22167969,  0.13476562 ,
     -0.01977539 ,  0.87470785,  0.1767578 ,  0.21289862,  0.04589844 ,
```

Gambar 5.10 Data fall

```
In [7]: model_dirge['sick']

Out[7]:
array([-0.2617388e-01, 1.4941486e-01, -4.0527343e-02, 1.6486250e-01,
       -1.5976525e-01, 3.2226525e-01, -1.77823125e-01, -1.4766951e-01,
       1.0167421e-01, 5.4687560e-01, 1.66992188e-01, 1.6894531e-01,
       -0.1123952e-01, 1.16375812e-01, 1.66992188e-01, 1.6894531e-01,
       1.6691525e-01, -1.7987859e-01, 5.92841016e-03, 2.45171765e-01,
       -0.240538e-02, 2.56347455e-02, 3.41796275e-01, 4.98867575e-02,
       1.7197852e-01, 2.15824122e-01, 1.29868172e-01, 1.2597652e-01,
       1.6164625e-01, -2.6562598e-01, -1.45978121e-01, 1.0953595e-01,
       1.9453853e-01, 1.66992188e-01, 1.6894531e-01, 1.6894531e-01,
       1.1859234e-01, 1.23946875e-01, -6.03927344e-02, 4.6875000e-01,
       9.1889535e-02, -3.1259800e-01, 1.84578131e-01, -1.5137153e-01,
       1.9348852e-01, 1.66992188e-01, 1.6894531e-01, 1.6894531e-01,
       -0.1171875e-01, 1.86281172e-02, 1.29868172e-01, 1.2597652e-01,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       7.1298025e-02, 4.37911719e-02, 2.05078125e-01, 5.7129802e-02,
       8.44726562e-02, 2.15824122e-01, -1.20955125e-01, 8.7989625e-02,
       1.8389375e-01, 1.66992188e-01, 1.6894531e-01, 1.6894531e-01,
       -1.5742175e-01, 3.02124023e-03, 2.08878121e-01, 5.85371004e-02,
       2.01893522e-02, 1.23946875e-02, 2.08878121e-01, 5.3321735e-02,
       -0.1416945e-02, 1.6379590e-02, 1.66992188e-01, 1.6894531e-01,
       3.5469866e-02, 1.5625980e-01, 2.03857422e-02, 2.26562500e-01,
       -0.1808963e-02, 1.66992188e-01, 1.6894531e-01, 1.6894531e-01,
       2.01171875e-02, -2.01171875e-01, -1.59756855e-02, 2.61718750e-01,
       1.1889444e-02, -4.2197800e-01, 2.22167959e-02, 1.46484375e-01,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       -0.4238212e-02, -3.1259800e-02, 6.47495655e-02, 2.47892754e-02,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       2.2226525e-02, -3.1259800e-02, 6.47495655e-02, 2.47892754e-02,
       2.2226525e-02, -3.1259800e-02, 6.47495655e-02, 2.47892754e-02,
       2.5589375e-02, -1.5429875e-01, 8.30078125e-03, -9.8145312e-02,
       1.80589375e-02, 1.66992188e-01, 2.42019922e-01, -1.87500000e-01,
       1.9683750e-02, 1.77749719e-02, 1.08578121e-02, 1.08578121e-02]
```

Gambar 5.11 Data sick

```
In [8]: model_dirge['clear']

Out[8]:
array([-4.4144025e-04, -1.82998781e-01, -1.49414862e-01, -4.24894658e-02,
       -1.0683759e-01, -1.46468175e-01, 1.76797121e-01, 1.46468175e-01,
       1.24511739e-01, 2.23632812e-01, -2.13867188e-01, 3.10855944e-02,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       -0.75787890e-02, -6.04248487e-03, -7.37984688e-02, -1.7285152e-01,
       1.8089444e-02, 1.66992188e-01, 1.6894531e-01, 1.6894531e-01,
       7.9161526e-02, 1.8791856e-01, 1.18515152e-01, -8.3468938e-02,
       1.2226525e-02, -7.1298025e-02, 2.83293125e-01, -2.75598825e-01,
       2.5589375e-02, -1.1298025e-02, 2.83293125e-01, -2.75598825e-01,
       -4.4660975e-02, 3.7193156e-02, -9.17968750e-02, 1.30859375e-01,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       2.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       2.7497812e-02, -1.2598800e-01, -5.73778469e-02, -2.74658203e-02,
       -1.37729402e-02, 2.16897188e-01, -1.17292989e-01, 4.08867575e-02,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       -1.57479783e-02, -1.37695112e-01, 3.88183949e-02, 1.5722652e-01,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       1.55277430e-01, 1.65998902e-01, -1.66992188e-01, 3.14941686e-02,
       2.85156250e-01, 2.69531250e-01, 3.32831598e-02, 2.41219193e-01,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       2.51646444e-02, -2.05156250e-01, -1.24023439e-01, -8.8947655e-02,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       -2.61238459e-02, -3.0151372e-02, 8.66699213e-03, -1.30859375e-01,
       3.05183994e-02, 6.6959750e-03, 5.11685159e-03, -6.05468750e-02,
       1.2226525e-02, 1.23946875e-02, 1.29868172e-01, 1.2597652e-01,
       1.14746094e-01, -2.47795078e-02, 9.71797988e-02, -1.66992188e-01,
       5.08715625e-02, -2.76154862e-03, -1.05489859e-01, -1.05489854e-01,
       -1.07421750e-02, 1.23946875e-02, -1.27871875e-01, -1.27871875e-01,
       9.96893750e-01, 1.35742188e-01, -6.83593750e-01, 2.26562500e-01,
```

Gambar 5.12 Data clear

```
In [9]: model_dirge['shine']

Out[9]:
array([1.1482344e-01, 0.25979652e-01, -0.15017969e-01, -0.27734375e-01, 0.39273458e-01,
       0.89968938e-01, 0.39257912e-01, -0.22949219e-01, -0.18559375e-01, 0.3671875e-01,
       -0.18982734e-01, 0.13671375e-01, 0.2598625e-01, 0.87212898e-01, 0.82539862e-01,
       -0.11239469e-01, 0.18898394e-01, 0.80515250e-01, 0.80515250e-01, 0.16484535e-01,
       -0.88587897e-01, 0.86127935e-01, -0.18549818e-01, 0.87617188e-01, 0.95182359e-01,
       -0.34755035e-01, 0.25633477e-01, -0.18541266e-01, -0.84521666e-01, 0.79571252e-01,
       -0.24121994e-01, -0.18494512e-01, 0.15234735e-01, -0.85493164e-01, 0.81634326e-01,
       0.24023438e-01, -0.36132312e-01, -0.12792969e-01, 0.10995703e-01, 0.89921209e-01,
       -0.82465482e-01, 0.32226962e-01, 0.11736953e-01, 0.18164460e-01, 0.84931641e-01,
       -0.14257118e-01, -0.20898938e-01, 0.0927344e-01, -0.8559375e-01, 0.85193550e-01,
       -0.14367188e-01, -0.20898938e-01, -0.0927344e-01, -0.8559375e-01, 0.85193550e-01,
       -0.84768742e-01, 0.88837390e-01, -0.84722461e-01, 0.85988283e-01, 0.87212898e-01,
       -0.81519775e-01, -0.11621304e-01, 0.07123806e-01, 0.81468389e-01, 0.10844531e-01,
       -0.11239469e-01, 0.18898394e-01, 0.80515250e-01, 0.80515250e-01, 0.16484535e-01,
       0.3359375e-01, -0.1875e-01, 0.14559781e-01, 0.80464385e-01, 0.87617188e-01,
       -0.40921394e-01, 0.25633477e-01, -0.18541266e-01, -0.84521666e-01, 0.79571252e-01,
       -0.25398625e-01, 0.85280195e-01, 0.27399682e-01, -0.88598438e-01, 0.81054669e-01,
       -0.22949219e-01, 0.14941486e-01, -0.1953125e-01, 0.88464389e-01, -0.80753764e-01,
       -0.84768742e-01, 0.18898394e-01, 0.80515250e-01, 0.80515250e-01, 0.16484535e-01,
       -0.80748024e-01, 0.10985594e-01, -0.11744609e-01, 0.131644602e-01, 0.13578980e-01,
       -0.11239469e-01, -0.88008109e-01, 0.08951406e-01, 0.03989594e-01, 0.03982975e-01,
       -0.08623975e-01, 0.15234735e-01, 0.19642869e-01, 0.84989869e-01, 0.25e-01,
       -0.08671563e-01, 0.84598944e-01, -0.05182359e-01, -0.84652754e-01, 0.85491211e-01,
       0.08821456e-01, 0.26367188e-01, -0.28718938e-01, 0.89863281e-01, -0.89862803e-01,
       0.0291415e-01, 0.26367188e-01, -0.28718938e-01, 0.89863281e-01, -0.89862803e-01,
       -0.083125e-01, -0.13769533e-01, -0.05737305e-01, 0.38867188e-01, -0.421875e-01]
```

Gambar 5.13 Data shine

```
In [11]: model_dirgs['car']
Out[11]:
array([-0.12085938,  0.00842285,  0.03344727, -0.05883789,  0.04803906,
       0.14257812,  0.04931641, -0.16094531,  0.20898483,  0.11962891,
       0.18664486,  -0.25,  0.18400391, -0.10742188, -0.01879883,
       0.01391267,  0.02316575,  0.12500525,  0.11760751,  0.01911195,
       0.16113281, -0.01511528, -0.03808575,  0.08447266,  0.16218938,
       0.01488544,  0.01573334,  0.2598025,  0.33981775,  0.01689556,
       -0.25893938,  0.01733334,  0.2598025,  0.33981775,  0.01689556,
       -0.09931289,  0.16593986,  0.08684766, -0.18943112,  0.02832031,
       -0.01488544,  0.01573334,  0.2598025,  0.33981775,  0.01689556,
       0.12355161,  0.00829499,  0.1484937,  0.33200125,  0.05249023,
       0.20805531,  0.37695312,  0.12255959,  0.11425781,  0.17675791,
       0.11885394,  0.09931455, -0.3125,  0.05315625,  0.02832031,
       0.11885394,  0.09931455, -0.3125,  0.05315625,  0.02832031,
       0.21242188,  0.0234375, -0.20117188, -0.13476552,  0.26371188,
       0.06760943,  0.20597112,  0.01780884,  0.12968281,  0.04711914,
       0.14984588,  0.01824219,  0.048986, -0.16992188,  0.10858594,
       0.04722461,  0.19824219,  0.048986, -0.16992188,  0.10858594,
       -0.87988078,  0.05688476, -0.09539341, -0.07225562,  0.04852812,
       0.07324219,  0.11893156,  0.04858398, -0.17675781, -0.33798662,
       0.22558594,  0.16380894,  0.05102539, -0.08251953,  0.07950964,
```

Gambar 5.14 Data bag

```
In [12]: model_dirgs['wash']
Out[12]:
array([ 1.4109105e-01, -3.4607985e-02,  1.34790625e-01,
       3.3821598e-01,  3.2421759e-01, -8.4472532e-02, -1.2998232e-01,
       1.0791056e-01,  2.53986250e-01,  1.1352391e-01, -1.6695218e-01,
       1.2187890e-01,  2.0598750e-01,  1.7611870e-01, -2.0410150e-01,
       -2.4187900e-02,  3.4868750e-01,  2.1194862e-01, -8.8867187e-02,
       2.6778125e-01,  2.1289625e-01,  1.74568547e-01,  2.82941859e-03,
       1.9901565e-01,  2.3197868e-01,  2.1289625e-01, -1.1474669e-01,
       -2.6792090e-03, -0.1389538e-02,  2.3628159e-01,  2.2363212e-01,
       -0.48541992,  0.20889458,  0.28757812,  0.27929688,  0.17889844,
       -0.1757125,  -0.02779998,  0.28410156,  0.82929578,  0.03125,
       -0.23242188,  0.0234375, -0.20117188, -0.13476552,  0.26371188,
       0.06760943,  0.20597112,  0.01780884,  0.12968281,  0.04711914,
       0.14984588,  0.01824219,  0.048986, -0.16992188,  0.10858594,
       0.04722461,  0.19824219,  0.048986, -0.16992188,  0.10858594,
       -0.87988078,  0.05688476, -0.09539341, -0.07225562,  0.04852812,
       0.07324219,  0.11893156,  0.04858398, -0.17675781, -0.33798662,
       0.22558594,  0.16380894,  0.05102539, -0.08251953,  0.07950964,
```

Gambar 5.15 Data car

```
In [13]: model_dirgs['water']
Out[13]:
array( 5.7378469e-02, -0.593904e-01, -6.6162704e-02, -1.1321406e-01,
       2.31728159e-01, -0.49294757e-01,  2.68052144e-02, -4.2730000e-02,
       2.34735900e-02,  2.57812500e-01,  1.74894659e-01,  2.44308035e-02,
       -2.51593125e-01, -5.76713175e-02,  8.15429688e-01,  1.86767578e-02,
       2.57812500e-02, -0.42486468e-02,  1.32812500e-01,  2.11954802e-01,
       1.56250000e-01, -4.24864688e-02, -1.32812500e-01,  2.11954802e-01,
       1.22051562e-01,  0.69979575e-01,  1.55273430e-01,  4.16748219e-01,
       2.12205156e-02,  0.28128735e-01,  0.87374219e-01,
       -2.06530800e-01, -5.39559701e-01,  1.37695130e-01, -5.85690984e-02,
       -2.26562000e-01, -5.44433950e-01,  1.37695130e-01, -5.85690984e-02,
       -1.56445312e-02,  1.77734375e-01,  3.22978125e-01,  3.71937398e-02,
       -1.10839444e-01,  6.83939758e-01,  2.52685547e-01, -1.27929688e-01,
       1.77901535e-02, -0.1389538e-02,  2.21079868e-01,  3.18559375e-01,
       1.08398458e-01,  2.45171886e-01,  2.08951750e-01,
       1.22051562e-01,  0.39773150e-01,  2.45171886e-01,  2.08951750e-01,
       0.04984669e-01, -3.47956250e-01, -5.20819311e-02,  2.24699375e-01,
       1.22051562e-01,  0.39773150e-01,  2.45171886e-01,  2.08951750e-01,
       2.17205156e-02,  1.12394612e-02, -1.14257812e-01,  7.22656250e-02,
       1.22051562e-02,  0.28128735e-01,  0.87374219e-01,
       -2.06530800e-01, -5.39559701e-01,  1.37695130e-01, -5.85690984e-02,
       -2.26562000e-01, -5.44433950e-01,  1.37695130e-01, -5.85690984e-02,
       5.66486598e-02,  2.13867188e-01, -2.86865234e-02, -1.70898438e-01,
       2.92668759e-02, -1.05939862e-01,  2.71684735e-01,  1.58203105e-01,
       9.17988759e-02,  2.41699249e-02, -1.77734375e-01, 2.71684735e-01,
```

Gambar 5.16 Data wash

```
In [14]: model_dirga['cycle']

Out[14]:
array([ 0.34541016,  0.21679688, -0.02789961,  0.12353516, -0.20793125,
       -0.1328125 ,  0.26367188, -0.12989625, -0.125 ,  0.15328311,
       -0.18261719, -0.15828012, -0.06176758,  0.21972656, -0.15828312,
       -0.18261719, -0.15828012, -0.06176758,  0.21972656, -0.15828311,
       -0.13779986, -0.11669922, -0.339375 ,  0.878125 ,  0.88447266,
       0.87228562, -0.06445312,  0.05517978,  0.14941498,  0.15071875,
       0.14941498,  0.15071875,  0.05517978,  0.88447266, -0.06445312,
       -0.85541992, -0.29492185, -0.40839862,  0.06347056, -0.88447266,
       0.14257811,  0.26880781, -0.06979182,  0.07275391,  0.15939862,
       0.18864466, -0.28515325, -0.04452754,  0.81806441,  0.00331116,
       0.00331116, -0.06787189,  0.05957031,  0.05517978,  0.22656279,
       -0.06787189, -0.07808097,  0.144475 ,  -0.20214848, -0.03935555,
       0.03935555, -0.05173828,  0.05957031,  0.05517978,  0.22656279,
       -0.17889844,  0.1484375 ,  0.10546875,  0.86445312,  0.01031404,
       -0.05263719, -0.05000525, -0.05026172,  0.05026172,  0.14257812,
       0.14257812, -0.05000525, -0.05263719,  0.01031404,  0.01031404,
       -0.05163281,  0.16894531, -0.02856445,  0.18791816, -0.24241875,
       0.2578125 ,  0.12989625, -0.24988538, -0.82929888, -0.19628986,
       -0.2578125 , -0.3673875 ,  0.01481154,  0.20703125,  0.99667969,
       0.01481154,  0.20703125, -0.3673875 ,  0.2578125 ,  0.99667969,
       0.06689452, -0.1796875 ,  0.02783289,  0.1598125 ,  0.02026357,
       0.0247907 , -0.13479562,  0.15527344,  0.11152812, -0.01055986,
       0.15527344,  0.11152812, -0.13479562,  0.0247907 ,  0.01055986,
       0.18986328, -0.20895469,  0.07275391, -0.35546875, -0.02746532,
       -0.20895469,  0.18986328,  0.07275391, -0.35546875, -0.02746532,
       -0.08970459, -0.22265925, -0.10498847,  0.24121094, -0.06938275,
       -0.234375 ,  0.12011719, -0.12898625,  0.05444336, -0.14746894,
       -0.234375 ,  0.12011719, -0.12898625,  0.05444336, -0.14746894,
       -0.089608938, -0.17889844,  0.09179688,  0.36328125,  0.88449385,
```

Gambar 5.17 Data motor

```
In [15]: model_dirga.similarity('wash', 'clear')

Out[15]: 0.09019176
```

Gambar 5.18 Data cycle

```
1 model_kk.similarity('sick', 'faith')
2 #%%%
3 model_kk.similarity('cycle', 'shine')
4
5 #%% Soal 2
6 import re
7
8 test_string = "Kaka Kamaludin, adalah nama aku"
9 print ("Faktanya: " + test_string)
```

Merupakan persentase dari perbandingan kata:

```
In [16]: model_dirga.similarity('bag', 'love')

Out[16]: 0.07536096
```

Gambar 5.19 Data wash dan clear

```
In [17]: model_dirga.similarity('motor', 'car')

Out[17]: 0.4810173
```

Gambar 5.20 Data bag dan love

```
In [18]: model_dirga.similarity('sick', 'faith')
Out[18]: 0.123673285
```

Gambar 5.21 Data sick dan faith

```
In [16]: model_dirga.similarity('bag', 'love')
Out[16]: 0.07556096
```

Gambar 5.22 Data cycle dan shine

5.1.2.2 Nomor 2

```
1 #%%
2 import random
3 sent_matrix = [ ['Ini', 'Data'], ['Untuk', 'Merandom'], ['Isi', 'Yang'],
4                 ], ['Ada', 'Disini']]
5 result = ""
6 for elem in sent_matrix:
7     result += random.choice(elem) + " "
7 print (result)
```

Kode ini berguna untuk membuat string memakai import re, dengan memakai test_string sebagai datanya

```
In [23]: import re
...: test_string = "Dirga Brajamusti, adalah nama aku"
...: print ("Faktanya: " + test_string)
...: res = re.findall('(\w+)', test_string)
...: print ("The list of words is : " + str(res))
Faktanya: Dirga Brajamusti, adalah nama aku
The list of words is : ['Dirga', 'Brajamusti', 'adalah', 'nama', 'aku']
```

Gambar 5.23 Nomor 2

```
1 #%% Soal 3
2 from gensim.models.doc2vec import Doc2Vec, TaggedDocument
3 ## Exampmple document (list of sentences)
4 doc = ["I love machine learning",
5         "I love coding in python",
6         "This is a good pc",
```

Kode ini berguna untuk membuat string dengan import random, memakai variable sent_matrix untuk membuat string, dan result sebagai print dari random data yang diacak, hasilnya akan berubah-ubah:

```
In [23]: import re
...: test_string = "Dirga Brajamusti, adalah nama aku"
...: print ("Faktanya: " + test_string)
...: res = re.findall('(\w+)', test_string)
...: print ("The list of words is : " + str(res))
Faktanya: Dirga Brajamusti, adalah nama aku
The list of words is : ['Dirga', 'Brajamusti', 'adalah', 'nama', 'aku']
```

Gambar 5.24 Nomor 2-1

5.1.2.3 Nomor 3

```
1     "This is a good phone"]
2 tokenized_doc = ['love']
3 tokenized_doc
4
5 print(doc)
6
7 #%%%
8 tagged_data = [TaggedDocument(d, [i]) for i, d in enumerate(
9     tokenized_doc)]
10 tagged_data
11 ## Train doc2vec model
12 model = Doc2Vec(tagged_data, vector_size=20, window=2, min_count=1,
13     workers=4, epochs = 100)
```

Fungsi dari library gensim untuk pemodelan topik unsupervised learning. Fungsi dari doc2vec itu sendiri adalah untuk membandingkan bobot data yang terdapat pada dokumen lainnya, apakah kata-katanya ada yang sama atau tidak. Lalu untuk tagged document itu memasukan kata-kata pada setiap dokumennya untuk di vektorisasi, dan model untuk membuat model dan save file model.

```
In [8]: from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from collections import deque
import random

# Create training data
docs = ["I love machine learning",
        "I love coding in python",
        "This is a good pic",
        "This is a good mac",
        "I love playing soccer"]
tagged_docs = []
for i, doc in enumerate(docs):
    tagged_docs.append(TaggedDocument(doc, [i]))
    tokens_and_doc = [doc]
    tokens_and_doc.append(tagged_docs[i])
    tagged_and_doc = [tagged_docs[i]]
    tagged_and_doc.append(tokens_and_doc)
    tagged_and_doc.append([i])
    tagged_and_doc.append(tokens_and_doc)
    tagged_and_doc.append(tokens_and_doc)

# Create a Doc2Vec model
model = Doc2Vec(tagged_docs, vector_size=10, window=2, min_count=1, workers=4)

# Test the model
model.wv['I']
model.wv['love']
model.wv['machine']
model.wv['learning']
model.wv['coding']
model.wv['python']
model.wv['good']
model.wv['pic']
model.wv['mac']
model.wv['soccer']

# Test the model's ability to find similar words
model.wv.most_similar('I')
model.wv.most_similar('love')
model.wv.most_similar('machine')
model.wv.most_similar('learning')
model.wv.most_similar('coding')
model.wv.most_similar('python')
model.wv.most_similar('good')
model.wv.most_similar('pic')
model.wv.most_similar('mac')
model.wv.most_similar('soccer')

# Test the model's ability to find similar documents
model.docvecs.most_similar('I')
model.docvecs.most_similar('love')
model.docvecs.most_similar('machine')
model.docvecs.most_similar('learning')
model.docvecs.most_similar('coding')
model.docvecs.most_similar('python')
model.docvecs.most_similar('good')
model.docvecs.most_similar('pic')
model.docvecs.most_similar('mac')
model.docvecs.most_similar('soccer')
```

Gambar 5.25 Nomor 3

```
1 ## Load saved doc2vec model
2 model= Doc2Vec.load("test_doc2vec.model")
3 ## Print model vocabulary
4 model.wv.vocab
5
6
7 #%% Soal 4
8 import re
9 import os
10 unsup_sentences = []
```

```
In [3]: from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from collections import deque
import random

# Create training documents
docs = ["I love machine learning",
        "I love this language in python",
        "This is a good movie",
        "I like this movie too",
        "This is a good movie"]
tagged_docs = []
for i, doc in enumerate(docs):
    tagged_docs.append(TaggedDocument(doc, [i]))
print(tagged_docs)

# Create a Doc2Vec model
model = Doc2Vec(tagged_docs, vector_size=10, window=2, min_count=1, workers=4)
model.train(tagged_docs, total_examples=len(tagged_docs), epochs=100)

# Test the model
print(model.infer_vector(["I", "love", "this", "language", "in", "python"]))
print(model.infer_vector(["I", "like", "this", "movie", "too"]))
print(model.infer_vector(["This", "is", "a", "good", "movie"]))

# Test the model with unknown words
print(model.infer_vector(["This", "is", "a", "good", "movie", "This", "is", "a", "good", "movie"]))
print(model.infer_vector(["DAM", "is", "a", "good", "movie"]))
```

Gambar 5.26 Nomor 3-1

5.1.2.4 Nomor 4

```
1     for fname in sorted(os.listdir("D:/Documents/KULIAH/Semester 6/AI  
2         /aclImdb/" + dirname)):  
3             if fname[-4:] == '.txt':  
4                 with open("D:/Documents/KULIAH/Semester 6/AI/aclImdb/" +  
5                     dirname + "/" + fname, encoding='UTF-8') as f:
```

```
4         sent = f.read()
5         words = (sent)
6         unsup_sentences.append(TaggedDocument(words, [dirname
7             + "/" + fname]))
8
9 #%% soal 5
10 #Pengacakan data
11 mute = (unsup_sentences)
```

Disini memakai dataset dari aclImdb. Untuk menambahkan data training kita melakukan import library os, library os fungsinya untuk melakukan interaksi antara python dengan windows atau os kita, setelah itu buat variable unsup sentences. Lalu pilih direktori tempat data akan disimpan. Lalu untuk mensortir data yang terdapat pada folder aclImdb dan membaca file tersebut dengan ekstensi .txt. Hasil dari code pertama tersebut adalah adanya data hasil running dari folder aclImdb.

Gambar 5.27 Nomor 4

5.1.2.5 Nomor 5

```
1 #soal 6  
2 #Save data  
3 model.save('kk.d2v')
```

Pada bagian pengacakan data berguna untuk mengacak data agar saat data di running bisa berjalan lebih baik dan hasil presentase akan lebih baik. Sedangkan untuk pembersihan data untuk memberikan ruang bagi ram laptop kita setelah melakukan running data lebih dari 3 juta , agar lebih ringan saat proses selanjutnya.

```
In [32]: mute = (unsup_sentences)
.....
....: #Pembersihan data
....: model.delete_temporary_training_data(keep_inference=True)
```

5.1.2.6 Nomor 6

```
1 #%% soal 7  
2 model.infer_vector(res)  
3  
4 #%% soal 8
```

```
5 from sklearn.metrics.pairwise import cosine_similarity
```

Save berfungsi untuk menyimpan hasil dari proses pelatihan data sebelumnya ke dalam sebuah file, model tersebut dilakukan penyimpanan untuk memberikan keringanan pada ram agar saat akan melakukan pelatihan lagi, model tersebut bisa load tanpa harus melakukan pelatihan dari awal dan akan menghemat waktu. Sedangkan untuk delete temporary training data ini berfungsi untuk menghapus data latihan yang sebelumnya sudah dilakukan dan disimpan, bertujuan untuk memberikan keringanan pada ram. Karena setelah melakukan proses pelatihan ram biasanya jadi penuh sampai komputer menjadi lag. Itulah fungsi dari delete temporary training data

```
In [36]: model.save("dirga.dcv")
.....
Out[36]: model.delete_temporary_training_data(keep_inference=True)
2020-04-05 21:30:32,528 : INFO : saving DocVec object under dirga.dcv, separately None
2020-04-05 21:30:32,482 : INFO : saved dirga.dcv
```

Gambar 5.29 Nomor 6

5.1.2.7 Nomor 7

```
1 [model.infer_vector(["pusing", "data"])]
```

Infer_vector berfungsi untuk membandingkan kata yang tercantum dengan vektor yang mana pada dokumen yang sudah diload pada step sebelumnya. Selain itu infer_vector juga untuk menghitung atau mengkalkulasikan vektor dari kata yang dicantumkan dari model yang telah dibuat. Alangkah baiknya kata yang dicantumkan itu lebih panjang lagi agar hasilnya bisa lebih baik lagi.

```
In [79]: model.infer_vector(res)
Out[79]:
array([ 0.02322483, -0.01122747, -0.00295282, -0.00356366,
       0.02363259,
      -0.00529885,  0.00381824, -0.00417452, -0.00091396,  0.0075980,
      0.01598696, -0.00897159, -0.01558069,  0.01741282,  0.01791294,
      0.01989056,  0.01433126,  0.00834827, -0.01111711,  0.01451616],
      dtype='float32')
```

Gambar 5.30 Nomor 7

5.1.2.8 Nomor 8

```
1 from sklearn import datasets, linear_model
2 from sklearn.model_selection import cross_val_score
3 diabetes = datasets.load_diabetes()
4 X = diabetes.data[:150]
```

Cosine_similarity berfungsi untuk membandingkan vektorisasi data diantara kedua kata yang di inputkan, jika hasil presentase dari kedua kata tersebut lebih dari 50% itu memiliki kemungkinan kata tersebut terdapat dalam 1 file. Namun jika kurang dari 50% itu kemungkinan kata tersebut tidak terdapat dalam 1 file. Hasil yang didapatkan pada code tersebut hanya 0.49% itu dikarenakan kata pertama dan kedua tidak memiliki kesamaan vektorisasi dan tidak terdapat pada salah satu dokumen.

```
In [83]: from sklearn.metrics.pairwise import cosine_similarity
...: ...
...: model.lsa_vector(["banyak", "banyak"])
...: model.lsa_vector(["positive", "data"])
Out[83]: array([-0.49988132], dtype=float32)
```

Gambar 5.31 Nomor 8

5.1.2.9 Nomor 9

```
1 print(cross_val_score(lasso , X, y, cv=3))
```

Melakukan perhitungan presentase dengan menggunakan cross_validation dengan metode kneighborsClassifier. Menggunakan dataset iris

```
In [78]: from sklearn import datasets, linear_model
...: from sklearn.model_selection import cross_val_score
...: diabetes = datasets.load_diabetes()
...: X = diabetes.data[1:150]
...: y = diabetes.target[1:150]
...: lasso = linear_model.Lasso()
...: print(cross_val_score(lasso, X, y, cv=3))
[0.331509724 0.08022311 0.03937764]
```

Gambar 5.32 Nomor 9

5.1.3 Penanganan Error

- Error

- FileNotFoundException

```
FileNotFoundException: [Errno 2] No such file or directory: 'N:/XBa/GoogleNews-vectors-negative300.bin'
```

Gambar 5.33 Error 1

- Solusi

- FileNotFoundException

Perhatikan letak file ada dimana, pastikan path telah benar

5.1.4 Link Youtube

<https://www.youtube.com/playlist?list=PL4dhp4u89PHbhX9jrGyM3N12gmwhY3uIe>

5.2 Mochamad Arifqi Ramadhan(1174074)

5.2.1 Teori

5.2.1.1 Jelaskan kenapa kata-kata harus di lakukan vektorisasi

Kata kata harus dilakukan vektorisasi dikarenakan atau bertujuan utk mengukur nilai kemunculan suatau kata yang serupa dari sebuah kalimat sehingga kata-kata tersebut

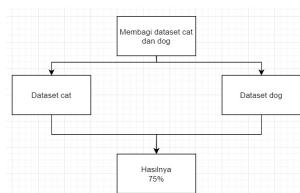
dapat di prediksi kemunculanya. atau juga di buatnya vektorisasi data digunakan untuk memprediksi bobot dari suatu kata misalkan ayam dan kucing sama-sama hewan maka akan dibuat prediksi apakah kata tersebut akan muncul pada kalimat yang kira-kira memiliki bobot yang sama.



Gambar 5.34 Teori 1

5.2.1.2 Jelaskan mengapa dimensi dari vektor dataset google bisa sampai 300.

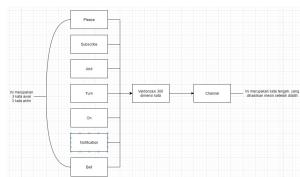
Karena setiap objek memiliki identitas khusus, misalnya sederhana. Dalam dataset Google ini memiliki 3 objek, kucing, anjing, dan ember yang disetujui. Kemudian dari masing-masing objek dibandingkan dataset antara kucing dan anjing kemudian kucing dan ember. Hasil yang diperoleh untuk kucing dan anjing sekitar 75% sedangkan untuk kucing dan ember yang memiliki persentase 15%. Untuk lebih lengkapnya bisa dilihat pada gambar berikut:



Gambar 5.35 Teori 2

5.2.1.3 Jelaskan konsep vektorisasi untuk kata

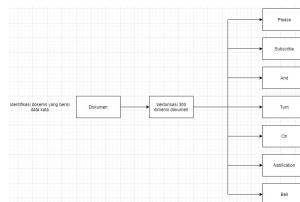
Konsep untuk vektorisasi kata sama dengan masukan atau input pada kata-kata di mesin pencari. Maka anggaplah itu akan dikeluarkan sebagai saran tentang kata tersebut. Jadi kata data diperoleh dari hasil yang diolah dalam kalimat sebelumnya yang telah diproses. Contoh sederhana dalam kalimat berikut (Please Subscribe and Turn on Notification Bell), dalam kalimat itu terkait dengan kalimat saluran, kata akan dibuat data pelatihan untuk mesin. Jadi kita komplasiasi kata channel, maka mesin akan menampilkan hubungannya dengan kata tersebut. Contoh ilustrasi sederhananya seperti berikut:



Gambar 5.36 Teori 3

5.2.1.4 Jelaskan konsep vektorisasi untuk dokumen

Sama halnya dengan vektorisasi kata, yang membedakan hanya pada proses awalnya. Untuk vektorisasi dokumen ini, mesin akan membaca semua kalimat yang terdapat pada dokumen tersebut, lalu kalimat yang terdapat pada dokumen akan dipecah menjadi kata-kata. Perhatikan gambar berikut:



Gambar 5.37 Teori 4

5.2.1.5 Jelaskan apa mean dan standar deviasi

Mean adalah nilai rata-rata. Untuk mendapatkan makna ini, kita hanya perlu menambahkan data yang tersedia dan kemudian membaginya dengan jumlah data. Sedangkan standar deviasi adalah nilai statistik yang digunakan untuk menentukan bagaimana data didistribusikan dalam sampel, dan menentukan titik data individu dengan nilai rata-rata nilai sampel. Rumusnya ialah sebagai berikut:

Mean $\bar{X} = \frac{x_1 + x_2 + \dots + x_n}{n}$ atau $\bar{X} = \frac{\sum_{i=1}^n X_i}{n}$	$\bar{X} = \text{rata - rata}$ $\sum_{i=1}^n X_i = \text{jumlah seluruh nilai data}$ $n = \text{jumlah seluruh frekuensi}$
--	--

Rumus Varian

$$s^2 = \frac{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}{n(n-1)}$$

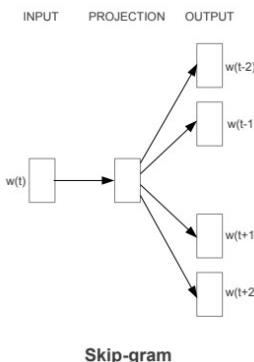
Rumus Standar Deviasi

$$s = \sqrt{\frac{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}{n(n-1)}}$$

Gambar 5.38 Teori 5

5.2.1.6 Jelaskan apa itu skip-gram

Skip-gram sama halnya dengan vektorisasi kata, namun untuk skip-gram ia dibalik prosesnya. Yang sebelumnya dari kalimat lalu diolah untuk menemukan salah satu kata, kali ini dari keyword tersebut akan diolah menjadi suatu kalimat yang memiliki keterkaitannya dengan keyword tersebut. Contoh sederhananya seperti gambar berikut:

**Gambar 5.39** Teori 6

5.2.2 Praktek

5.2.2.1 Nomor 1

```

1 import gensim, logging
2 logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
3 , level=logging.INFO)
3 model_dirga = gensim.models.KeyedVectors.load_word2vec_format('N:/KB/
GoogleNews-vectors-negative300.bin', binary=True, limit=500000)
  
```

Kode digunakan untuk import library gensim. Gensim itu sendiri berguna untuk melakukan pemodelan dengan dataset atau topik yang telah ditentukan. Untuk keluaran 100 logging itu library opsional karena logging hanya untuk menampilkan berupa log untuk setiap code yang dijalankan. Dan keluaran 101 itu hasil load data dari file vektor google itu, disini saya menggunakan limit karena kondisi laptop yang tidak mempunyai untuk melakukan running data sebesar 3 juta file, jadi saya batasi hanya melakukan running 500 ribu data saja, hasilnya ialah sebagai berikut:

```

[1]: import gensim, logging
...: logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
...: , level=logging.INFO)
...: model_dirga = gensim.models.KeyedVectors.load_word2vec_format('N:/KB/
GoogleNews-vectors-negative300.bin', binary=True, limit=500000)
2018-04-05 19:31:19,851 : INFO : Loading projection weights. From N:/KB/
GoogleNews-vectors-negative300.bin
2018-04-05 19:31:19,851 : INFO : Loaded (500000, 300) words from N:/KB/
GoogleNews-vectors-negative300.bin
  
```

Gambar 5.40 Hasil dari Nomor 1-1

```

1 model_dirga['love']
2 #%%
3 model_dirga['faith']
4 #%%
5 model_dirga['fall']
6 #%%
7 model_dirga['sick']
8 #%%
  
```

```

9 model_dirga[ 'clear' ]
10 #%%
11 model_dirga[ 'shine' ]
12 #%%
13 model_dirga[ 'bag' ]
14 #%%
15 model_dirga[ 'car' ]
16 #%%
17 model_dirga[ 'wash' ]
18 #%%
19 model_dirga[ 'motor' ]
20 #%%
21 model_dirga[ 'cycle' ]

```

Menampilkan data hasil vektorisasi data:

```

In [4]: model_dirga['love']
Out[4]:
array([ 0.1830774, -0.15234579,  0.05079091, -0.1603996, -0.1603996,
       0.06604053,  0.29206275, -0.26367188, -0.140625 , -0.2017108,
      -0.02624512, -0.02083125, -0.02709056, -0.039404531, -0.29351516,
       0.08574422,  0.08574422,  0.08574422,  0.08574422,  0.08574422,
      -0.03574422,  0.07958864,  0.22949219, -0.14359469,  0.1798875,
      -0.0351525 ,  0.09517576,  0.18053359,  0.1116164 , -0.1609594,
      -0.1609594,  0.1609594,  0.1609594,  0.1609594,  0.1609594,
      0.07714544,  0.26171875,  0.08642578, -0.02514648,  0.33394835,
      0.18652344, -0.20996805,  0.07608078,  0.0260005 , -0.10644511,
      -0.10644511,  0.10644511,  0.10644511,  0.10644511,  0.10644511,
      -0.18985328,  0.14941486, -0.10839359,  0.01556396,  0.08984375,
      0.08984375, -0.08370317, -0.08370317, -0.08370317, -0.08370317, -0.08370317,
      0.24130609,  0.24130609,  0.24130609,  0.24130609,  0.24130609,
      -0.09667989, -0.21972656, -0.08520854, -0.03108242,  0.18457031,
      0.18457031,  0.18457031,  0.18457031,  0.18457031,  0.18457031,
      -0.09220516, -0.18045112,  0.01516072,  0.18554608,  0.34375 ,
      -0.31854688,  0.22559504,  0.08740234, -0.2205625 , -0.29492188,
      0.08740234,  0.08740234,  0.08740234,  0.08740234,  0.08740234,
      -0.08958252,  0.23032812, -0.17871094, -0.12405172, -0.17285156,
      -0.17285156,  0.17285156,  0.17285156,  0.17285156,  0.17285156,
      -0.17203078,  0.19242419, -0.09521464,  0.09515253,  0.12597656 ,
      0.08673624, -0.0482832 ,  0.03630965,  0.01623553, -0.1646625 ,
      0.01646625,  0.01646625,  0.01646625,  0.01646625,  0.01646625,
      0.06494141,  0.05932617, -0.164862 ,  0.01371889,  0.18945312,
      0.05566496, -0.0596853 ,  0.01422119,  0.1591796 ,  0.07421875,
      0.07421875,  0.07421875,  0.07421875,  0.07421875,  0.07421875,
      -0.148625 , -0.13185954, -0.12792969,  0.12809547,  0.05683709,
      -0.05683709,  0.05683709,  0.05683709,  0.05683709,  0.05683709,
      -0.09275 ,  0.0895665 ,  0.04803066,  0.03772958,  0.13070715,
      0.03772958,  0.03772958,  0.03772958,  0.03772958,  0.03772958,
      0.08662231, -0.01928711,  0.09423828, -0.13113594, -0.27929688,
      0.27734375,  0.31035938,  0.10858954,  0.11425781, -0.27734375,

```

Gambar 5.41 Data love

```

In [5]: model_dirga['faith']
Out[5]:
array([-0.23637188, -0.04150391,  0.1953125 ,  0.13476562, -0.14648438,
       0.11962891,  0.08435783,  0.12200562,  0.12200561,  0.13476562,
      -0.1568125 ,  0.1568125 ,  0.1568125 ,  0.1568125 ,  0.1568125 ,
      0.32981125,  0.18449219,  0.36132812, -0.1953125 , -0.18164662,
      -0.02331543, -0.08380444,  0.08380444, -0.08380444,  0.08380444,
      -0.05688502, -0.05688502,  0.05688502,  0.05688502,  0.05688502,
      -0.08008502, -0.08008502,  0.08008502,  0.08008502,  0.08008502,
      -0.1328125 ,  0.21484375,  0.01135254,  0.02111816,  0.18554668,
      0.18554668,  0.18554668,  0.18554668,  0.18554668,
      0.3125,  0.06464025, -0.17675781, -0.01780884, -0.1649625 ,
      -0.02818024,  0.01257224, -0.09521464, -0.1805406 , -0.146625 ,
      0.02818024,  0.01257224, -0.09521464, -0.1805406 , -0.146625 ,
      -0.27539862, -0.20695449,  0.10515262, -0.29214844, -0.1875 ,
      0.10515262,  0.10515262, -0.29214844, -0.1875 ,
      0.10997138,  0.15757219,  0.17709531,  0.16380594, -0.03881836,
      -0.03881836,  0.03881836,  0.03881836,  0.03881836,
      -0.02331543, -0.04248047, -0.08380312,  0.08380312,  0.04158051,
      -0.05688502, -0.05688502,  0.05688502,  0.05688502,  0.05688502,
      -0.08008502, -0.08008502,  0.08008502,  0.08008502,  0.08008502,
      -0.13492125,  0.04159219,  0.05346054, -0.12081788,  0.09138055,
      -0.17382812, -0.02469038,  0.05466797,  0.19624219, -0.08837091,
      0.08837091,  0.08837091,  0.08837091,  0.08837091,
      -0.13574219, -0.30678125, -0.00469771,  0.0666559 , -0.29296875,
      0.15234375,  0.02965389,  0.3328125 ,  0.28238012,  0.0975 ,
      0.0975 ,  0.0975 ,  0.0975 ,  0.0975 ,
      -0.20926675,  0.2558938 ,  0.00438206,  0.146625 ,  0.05185457,
      0.21582031,  0.0293748 ,  0.02929688,  0.20819531,  0.3466938 ,
      -0.05688502,  0.18449219,  0.36132812, -0.1953125 , -0.18164662,
      -0.05688502, -0.18449219,  0.08380444, -0.08380444,  0.08380444,
      -0.08008502, -0.08008502,  0.08008502,  0.08008502,  0.08008502,
      -0.0899778 ,  0.07373847, -0.01229703,  0.22469038,  0.14598701,
      0.03637695, -0.15234375,  0.07159331, -0.1374219 , -0.10042669,
      0.09521484,  0.07801328,  0.04793232,  0.378 , -0.0234375 ,
      0.2558938 ,  0.00438206,  0.146625 ,  0.05185457,

```

Gambar 5.42 Data faith

```
In [6]: model_dิง['fall']
Out[6]:
array([-0.04772461,  0.19742188, -0.0077354,  0.16804531, -0.1226152,
       -0.18693359,  0.84721289,  0.01964297,  0.14648438,  0.15830862,
      -0.08691486,  0.84492188,  0.154574,  0.86691406, -0.19824219,
      -0.18693359,  0.84721289,  0.01964297,  0.14648438,  0.15830862,
     -0.1015625,  0.13671875,  0.99228516, -0.12109375,  0.12895312,
    0.03417969,  0.21869737,  0.61973539,  0.125,  0.05154189,
     0.04499219,  0.29828212, -0.18554688,  0.08486894, -0.02087482,
     0.18295978,  0.15136494,  0.0044557,  0.04463904,  0.02753984,
     0.18295978,  0.15136494,  0.0044557,  0.04463904,  0.02753984,
     -0.19359378,  0.0546877,  0.0495895,  0.0737375, -0.0329894,
     0.18295978,  0.15136494,  0.0044557,  0.04463904,  0.02753984,
     0.0255227,  0.06279828,  0.07080878, -0.07617183,  0.06542969,
    0.01672363,  0.04711514,  0.19628986, -0.08984375,  0.078125 ,
     0.00542466,  0.08131,  0.12988211,  0.03279586,  0.14594781,
     0.00542466,  0.08131,  0.12988211,  0.03279586,  0.14594781,
     0.00448889, -0.14108159,  0.0451816, -0.078125,  0.06988589,
    0.26757912,  0.82091953, -0.12895312, -0.04482812,  0.18945312,
     0.18261719,  0.08876321,  0.04598844, -0.0289625, -0.03540809,
     0.18261719,  0.08876321,  0.04598844, -0.0289625, -0.03540809,
     0.0429875,  0.89932381, -0.08956641, -0.06268832,  0.12255895,
     0.0429875,  0.89932381, -0.08956641, -0.06268832,  0.12255895,
     0.08897112,  0.08482178, -0.1646625, -0.03274484,  0.070125 ,
     0.07959864,  0.1269625,  0.0187983, -0.17773434,  0.061203945,
     0.07959864,  0.1269625,  0.0187983, -0.17773434,  0.061203945,
     -0.14453125,  0.1085855, -0.085375,  0.18280975, -0.03938725,
    0.04853984,  0.88135514, -0.15828312, -0.09130895, -0.12255895,
     -0.01277759,  0.074707082,  0.117975792,  0.12289862,  0.04589844,
```

Gambar 5.43 Data fall

```
In [7]: model_dิง['sick']
Out[7]:
array([-0.82171886e-01,  1.49414862e-01, -4.05273439e-02,  1.64862500e-01,
       -5.5976525e-01,  3.2236525e-01, -1.73828125e-01, -1.47469518e-01,
       -0.12238662e-01,  0.12238662e-01,  0.12238662e-01,
      2.24304199e-03,  9.6679875e-02, -1.00001563e-01, -1.12384666e-01,
     1.66815625e-01, -1.79687300e-01,  5.92648165e-03, -2.45171886e-01,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     1.7871930e-01, -9.9182189e-01,  8.8671875e-02, -1.95312500e-01,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     9.42302312e-02, -3.12598000e-02,  1.98974699e-02, -6.39648415e-02,
    1.16852344e-01,  1.2394675e-01, -0.03627344e-01,  4.68754000e-01,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     5.70013281e-02, -1.04988469e-01,  1.69854512e-01, -0.00781250e-02,
     -0.01171875e-01,  1.06208152e-01, -1.06208152e-01,  0.00000000e+00,
     -0.01171875e-01,  1.06208152e-01, -1.06208152e-01,  0.00000000e+00,
     7.12990625e-02,  4.37911719e-02,  2.05978125e-01,  5.71218906e-02,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     2.4884675e-01, -6.5429875e-02, -2.02836712e-02,  1.52534798e-01,
     -0.57421675e-01,  0.82124037e-01, -2.00800000e-01, -5.95717004e-02,
     2.22256525e-01, -1.53128125e-01,  1.53128125e-01,  2.77343750e-01,
     -0.49414602e-02,  3.67769531e-02,  1.91692500e-01,  2.77343750e-01,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     4.66380594e-02, -5.1778125e-02,  1.63689538e-01,  4.17488469e-02,
     0.01171875e-02, -2.0171875e-01, -1.50750500e-02,  2.61718750e-01,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     4.19921875e-01, -6.88479562e-02,  9.42302312e-02, -1.96239002e-01,
     -0.49414602e-02,  1.53128125e-01,  1.53128125e-01,  0.00000000e+00,
     -0.61132812e-01, -1.53128125e-01,  1.53128125e-01, -1.8148655e-01,
     -0.22256525e-02, -1.43554688e-01,  8.30878125e-03, -9.81453112e-02,
     3.55993750e-01, -1.54312500e-01,  3.45915012e-02, -1.00000000e+00,
     1.50000000e-02, -2.07734739e-01,  2.46907325e-02,  0.82734375e-01,
```

Gambar 5.44 Data sick

```
In [8]: model_dิง['clear']
Out[8]:
array([-0.41480625e-04, -1.49414862e-01, -1.49414862e-01, -4.24846458e-02,
       -1.78719300e-01, -0.04688469e-01,  1.76579312e-01, -1.46488375e-01,
      2.26562300e-01,  9.76562300e-01, -2.07578125e-01, -1.99828125e-01,
     1.24511739e-01, -2.23632812e-01, -2.13971886e-01,  3.10855354e-02,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     3.22256525e-01,  3.14453125e-01, -1.18164866e-01,  8.007150e-02,
     -0.76780750e-02, -6.91339000e-01, -1.76719300e-01, -1.46488375e-01,
     0.6679875e-02, -6.91339000e-01, -1.76719300e-01, -1.46488375e-01,
     7.91915625e-02,  1.0791456e-01, -1.10515162e-01, -0.34669308e-02,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     2.55899375e-01, -7.12990625e-02,  2.83293125e-01, -2.7559805e-01,
     -0.49414602e-02,  8.0088231e-01, -2.00800000e-01, -5.95717004e-02,
     3.47987125e-02, -5.29880000e-01, -5.73779469e-02, -2.74658203e-02,
     -0.49414602e-02,  8.0088231e-01, -2.00800000e-01, -5.95717004e-02,
     -1.92302312e-01,  1.32012500e-01, -5.08088231e-01, -1.66001565e-01,
     -1.57479038e-02, -1.37905312e-01, -3.8813594e-02,  1.57225525e-01,
     -0.49414602e-02,  8.0088231e-01, -2.00800000e-01, -5.95717004e-02,
     1.5527538e-01,  1.60998062e-01, -1.60998062e-01, -1.49414602e-02,
     -1.39160556e-02,  5.12995312e-02,  9.76562300e-02,  3.14941460e-02,
     2.51464644e-02, -2.05156250e-01, -1.24623438e-01, -6.05476552e-02,
     -0.49414602e-02,  3.0811672e-01, -2.66699219e-01, -1.30859375e-01,
     -0.61232812e-01, -3.01511672e-02,  8.66699219e-03, -1.30859375e-01,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     0.17446964e-01, -1.47795678e-02,  9.71679688e-02, -1.66699219e-01,
     0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
     0.87421675e-01, -1.7088438e-01,  7.61718750e-02, -1.54682300e-01,
     9.96993750e-02,  1.35742188e-01, -6.83593750e-02,  2.25562300e-01,
```

Gambar 5.45 Data clear

```
In [9]: model_dirg['shine']
Out[9]:
array([-0.13482344,  0.29576562, -0.15017969, -0.27734375,  0.30273438,
       0.89968038,  0.39527512, -0.22549219, -0.18559375,  0.3671875,
      -0.18582734,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.18582734,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.85883789,  0.8612739, -0.01549818,  0.87617188,  0.85182539,
      0.26031531,  0.38805938,  0.00162586, -0.85692927,  0.14648458,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.24121894, -0.18945132, -0.15234375, -0.85493164,  0.81434326,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.24023486,  0.36132062, -0.12507159,  0.87617188,  0.85182539,
      -0.0246556,  0.32226562,  0.11579655,  0.18164602,  0.84931644,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.13567188, -0.26898458, -0.07801328, -0.8059375,  0.85959598,
      -0.14257812, -0.149625,  0.0927344, -0.14455125,  0.359375 ,
      -0.14257812, -0.149625,  0.0927344, -0.14455125,  0.359375 ,
      -0.04768742,  0.08537391, -0.04724661,  0.05998283,  0.87228986,
      0.01513775, -0.11621954,  0.07128986,  0.81434326, -0.18644531,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.3359975, -0.187,  0.14589781,  0.88463867,  0.87617188,
      -0.09521484,  0.08816266,  0.20117188,  0.11238469,  0.33984775,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.22949219,  0.14944486,  0.1953125,  0.88496904,  0.80753784,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.08740234,  0.10850383, -0.11474605, -0.86516486,  0.32651888,
      -0.11238469,  0.08808109,  0.08591406,  0.83988594,  0.8029235,
      -0.08823975, -0.15234375,  0.19042369,  0.04988469,  0.25 ,
      -0.00671661,  0.04589844,  0.05182359,  0.04952754,  0.05491111,
      -0.02801416,  0.26367188, -0.28718938,  0.89863281, -0.89862831,
      -0.03125, -0.13769531, -0.05737305,  0.38867188, -0.421875
```

Gambar 5.46 Data shine

```
In [11]: model_dirg['car']
Out[11]:
array([-0.13482348,  0.00942358,  0.03344272, -0.95631769,  0.04839396,
       0.12577578,  0.04951641, -0.16945351, -0.20898455,  0.11930261,
      -0.1866446, -0.25 , -0.18080591, -0.18742188, -0.01879883,
      0.95208019,  0.08816266,  0.20117188,  0.11238469,  0.33984775,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.04467773, -0.15527544,  0.2598025,  0.33984775,  0.00756536,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.09912189,  0.15693086,  0.08684766, -0.19495112,  0.02832031,
      -0.05346426, -0.03963965,  0.11639664,  0.24121894, -0.234375 ,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.26010531,  0.37695312,  0.12255859,  0.11429781,  0.17675781,
      0.18000766,  0.0093038,  0.26757812,  0.20117188,  0.07710938,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.26171875, -0.08542578, -0.02583801, -0.8534961,  0.08773754,
      -0.11767578, -0.04429675,  0.17285156,  0.04549351, -0.23867575,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.32617188, -0.04492188, -0.11429781,  0.22851562,  0.01647949,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.1815625,  0.08812771,  0.18793151, -0.24698975, -0.189375 ,
      -0.09375, -0.0162335,  0.20214544,  0.23144531,  0.05444336,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.17578125, -0.02779996,  0.20110356,  0.02392978,  0.03125 ,
      -0.25398125, -0.125 , -0.05939164, -0.17582112,  0.28515625,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.00769843,  0.20597812, -0.01789804, -0.12988281,  0.04711954,
      0.95208019,  0.08816266,  0.20117188,  0.11238469,  0.33984775,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.04722052,  0.15682112, -0.15859396, -0.06654688,  0.19435594,
      -0.14212894,  0.13671875,  0.2595825,  0.87128986,  0.85239862,
      -0.07324219,  0.16895156,  0.04585986,  0.16380594, -0.08251593,  0.07958964,
```

Gambar 5.47 Data bag

```
In [12]: model_dirg['wash']
Out[12]:
array([-0.4604422e-03, -0.41081562e-01, -5.46879800e-02,  1.34765625e-01,
       -3.38281590e-01,  3.24218750e-01, -8.44728562e-02, -1.29832612e-01,
       -1.79541801e-02,  2.80987812e-01, -4.27246894e-01, -0.95468750e-01,
      -7.41875008e-02,  3.04867500e-01,  2.11014682e-01, -8.88671750e-02,
      -2.29861211e-02,  0.62189750e-01,  1.93593750e-01, -2.17285156e-01,
      -2.47844688e-02, -0.78748118e-01,  0.11238215e-01,  1.65906180e-01,
      -0.89781500e-02, -3.80959375e-02, -1.00978500e-01, -1.39648510e-01,
      1.74804688e-01,  0.78748130e-01,  1.11238215e-01,  1.65906180e-01,
      -0.89781500e-02, -3.80959375e-02, -1.00978500e-01, -1.39648510e-01,
      -3.47750000e-02, -1.02895781e-01,  3.80859750e-01, -5.05171904e-02,
      2.84423282e-02, -2.29492188e-01, -5.21854888e-01,  4.5160156e-02,
      -5.6445312e-02,  1.77734750e-01,  1.27802112e-01,  3.71937508e-02,
      -1.02895781e-01,  3.80859750e-01,  1.27802112e-01,  3.71937508e-02,
      4.21575000e-01,  5.32226562e-02, -3.92578150e-01,  1.74804688e-01,
      1.0701553e-02, -2.05978125e-02,  2.21798860e-01,  3.18593750e-01,
      -0.89781500e-02, -3.80959375e-02, -1.00978500e-01, -1.39648510e-01,
      1.58867130e-02,  8.39078125e-02,  1.68943120e-01,  2.79541816e-02,
      1.40921750e-02,  2.13867188e-01,  3.18593750e-01,  2.79541816e-02,
      1.69677734e-02, 1.69912179e-01, -1.46484756e-01, 2.65625000e-01,
      2.17205356e-02, 1.12984605e-02, -1.14257812e-01, 7.22652590e-02,
      4.04220500e-02, 1.02895781e-01, 3.80859750e-01, 1.65906180e-01,
      -0.98144531e-01, 5.44433994e-02, 3.79468750e-01, 5.62500000e-01,
      -2.53280312e-01, -5.32226562e-02, 3.92578150e-01, 1.74804688e-01,
      5.66468250e-02, 2.13867188e-01, -2.86865324e-02, -1.70898438e-01,
      3.70898438e-02, -2.13867188e-01, 3.18593750e-01, 2.79541816e-02,
      9.17668750e-02, -2.41692108e-02, -1.77734750e-01, 2.71484374e-01
```

Gambar 5.48 Data car

```
In [13]: model_dirga['motor']

Out[13]:
array([-1.73738649e-02, 1.5090802e-02, -4.63425781e-02, -1.32812590e-01,
       -2.59975525e-01, -1.77794375e-01, 3.68652344e-02, -4.37950000e-01,
       -2.34759800e-02, 2.57812598e-01, 1.74894689e-01, 2.44348625e-02,
       -1.83380115e-01, 1.58283155e-01, 5.89577952e-02, 1.12846486e-01,
       -1.20000000e-01, 1.69921757e-01, -1.55274358e-01, 4.58954375e-01,
       -1.23846875e-01, 1.69921757e-01, -1.55274358e-01, 4.58954375e-01,
       -3.02734755e-01, 1.53328312e-01, 1.69921757e-01, -1.03874219e-01,
       -3.08595775e-01, 1.53328312e-01, 1.69921757e-01, -1.03874219e-01,
       -1.54268750e-01, -8.88617197e-02, 2.36328125e-01, 5.61533438e-03,
       -4.00875000e-01, -3.08595775e-01, 1.53328312e-01, 1.69921757e-01, -1.03874219e-01,
       -3.08595775e-01, -6.50624414e-01, 6.34765256e-02, 4.02812011e-02,
       -1.00000000e-01, 1.53328312e-01, 1.69921757e-01, -1.03874219e-01,
       -5.08980250e-01, -1.37695312e-01, 9.96803750e-01, 1.28962508e-01,
       -3.44726556e-01, -1.80886172e-02, 2.14843798e-01, -9.354358e-02,
       -9.00000000e-01, 1.53328312e-01, 1.69921757e-01, -1.03874219e-01,
       -2.21679888e-01, 2.33398438e-01, 5.05371604e-02, -3.37986255e-01,
       -1.12813200e-01, 1.53328312e-01, 1.69921757e-01, -1.03874219e-01,
       -8.00781250e-02, -1.14257812e-01, -9.71679658e-02, -2.61373750e-01,
       -3.84765525e-01, -1.87988000e-01, 1.10381515e-01, 1.06859598e-01,
       -1.10862500e-01, -1.87988000e-01, 1.10381515e-01, 1.06859598e-01,
       -2.40234755e-01, 8.3466938e-02, 4.19921875e-02, -1.91858931e-02,
       -9.00000000e-01, 1.53328312e-01, 1.69921757e-01, -1.03874219e-01,
       -2.02125200e-02, -3.22831250e-01, 3.27148415e-02, 5.71239902e-02,
       -1.77734755e-01, -9.57831250e-02, 2.45171258e-01, 6.68476552e-02,
       -1.00000000e-01, 1.53328312e-01, 1.69921757e-01, -1.03874219e-01,
       -4.00598025e-01, 6.69348025e-02, 3.08751250e-01, -1.99218750e-01,
       -1.23846875e-01, 1.53328312e-01, 1.69921757e-01, -1.03874219e-01,
       -5.70513251e-02, 7.80897122e-02, -2.27398022e-01, -3.02754375e-01,
```

Gambar 5.49 Data wash

```
In [14]: model_dirga['cycle']

Out[14]:
array([ 0.04541016,  0.21679688, -0.02789961,  0.12353515,
       -0.12281321,  0.26367188, -0.12989625, -0.125
       -0.15328201,  0.10000000,  0.10000000,  0.10000000,
       -0.02563477, -0.07585395, -0.0625
       -0.08464258, -0.10546488,
       -0.13739896, -0.11669922, -0.33575
       -0.078125,  0.078125,  0.08447266,
       -0.10000000,  0.10000000,  0.10000000,
       -0.18302734,  0.02172852, -0.10893359,
       -0.82698234, -0.18044531,
       -0.09963281, -0.02038374, -0.08780662,
       -0.07226562, -0.09423828,
       -0.17809844,  0.1484375
       -0.1846875,  0.06445112, -0.02038354,
       -0.09963281,  0.02172852, -0.10893359,
       -0.82698234, -0.18044531,
       -0.37109357, -0.15722656,  0.09326172,
       -0.34969038, -0.00081553,
       -0.04335489,  0.03173782, -0.07421875,
       -0.02417888, -0.149625,
       -0.14355489,  0.03173782, -0.07421875,
       -0.02417888, -0.149625,
       -0.08433548, -0.12698625, -0.34698538,
       -0.82929662, -0.19628986,
       -0.18931562,  0.03173782, -0.07421875,
       -0.02417888, -0.149625,
       -0.18931562, -0.13054688,  0.02344238,
       -0.18400031,  0.17773438,
       -0.06694552, -0.1796875
       -0.02783205,  0.15625
       -0.02026957,  0.02026957,  0.02026957,
       -0.07959864,  0.01989746,
       -0.25589538,  0.13778986,  0.02539862,
       -0.09960938,  0.18490947, -0.0625
       -0.28800781, -0.18490947, -0.01177979,
       -0.17828212,
       -0.08979659, -0.22265625, -0.10498847,
       -0.24512109,  0.0638275,
       -0.2345794,  0.12031159,  0.12031159,
       -0.09779889,  0.17889344,  0.09779889,
       -0.09960938,
```

Gambar 5.50 Data motor

```
In [15]: model_dirga.similarity('wash', 'clear')

Out[15]: 0.09019176
```

Gambar 5.51 Data cycle

```
1 model_dirga.similarity('wash', 'clear')
2 #%%%
3 model_dirga.similarity('bag', 'love')
4 #%%%
5 model_dirga.similarity('motor', 'car')
6 #%%%
7 model_dirga.similarity('sick', 'faith')
8 #%%%
9 model_dirga.similarity('cycle', 'shine')
```

Merupakan persentase dari perbandingan kata:

```
In [16]: model_dirga.similarity('bag', 'love')
Out[16]: 0.07536096
```

Gambar 5.52 Data wash dan clear

```
In [17]: model_dirga.similarity('motor', 'car')
Out[17]: 0.4810173
```

Gambar 5.53 Data bag dan love

```
In [18]: model_dirga.similarity('sick', 'faith')
Out[18]: 0.123073205
```

Gambar 5.54 Data sick dan faith

```
In [16]: model_dirga.similarity('bag', 'love')
Out[16]: 0.07536096
```

Gambar 5.55 Data cycle dan shine

5.2.2.2 Nomor 2

```
1 import re
2
3 test_string = "Dirga Brajamusti, adalah nama aku"
4 print ("Faktanya: " + test_string)
5 res = re.findall(r'\w+', test_string)
6 print ("The list of words is : " + str(res))
7 #%%
```

Kode ini berguna untuk membuat string memakai import re, dengan memakai test_string sebagai datanya

```
In [23]: import re
...
...: test_string = "Dirga Brajamusti, adalah nama aku"
...: print ("Faktanya: " + test_string)
...: res = re.findall(r'\w+', test_string)
...: print ("The list of words is : " + str(res))
Faktanya: Dirga Brajamusti, adalah nama aku
The list of words is : ['Dirga', 'Brajamusti', 'adalah', 'nama', 'aku']
```

Gambar 5.56 Nomor 2

```
1 import random
2 sent_matrix = [ ['Ini', 'Data'], ['Untuk', 'Merandom'], ['Isi', 'Yang'],
   ], ['Ada', 'Disini']]
3 result = ""
```

```

4 for elem in sent_matrix:
5     result += random.choice(elem) + " "
6 print (result)

```

Kode ini berguna untuk membuat string dengan import random, memakai variable sent_matrix untuk membuat string, dan result sebagai print dari random data yang diacak, hasilnya akan berubah-ubah:

```

In [23]: import re
...: test_string = "Dirga Brajemusti, adalah nama aku"
...: print ("Faktanya: " + test_string)
...: res = re.findall(r'\w+', test_string)
...: print ("The list of words is : " + str(res))
Faktanya: Dirga Brajemusti, adalah nama aku
The list of words is : ['Dirga', 'Brajemusti', 'adalah', 'nama', 'aku']

```

Gambar 5.57 Nomor 2-1

5.2.2.3 Nomor 3

```

1 from gensim.models.doc2vec import Doc2Vec, TaggedDocument
2 ## Exapmple document (list of sentences)
3 doc = ["I love machine learning",
4         "I love coding in python",
5         "This is a good pc",
6         "This is a good mac",
7         "This is a good phone"]
8 tokenized_doc = ['love']
9 tokenized_doc
10
11 print(doc)

```

Fungsi dari library gensim untuk pemodelan topik unsupervised learning. Fungsi dari doc2vec itu sendiri adalah untuk membandingkan bobot data yang terdapat pada dokumen lainnya, apakah kata-katanya ada yang sama atau tidak. Lalu untuk tagged document itu memasukan kata-kata pada setiap dokumennya untuk di vektorisasi, dan model untuk membuat model dan save file model.

```

In [24]: from gensim.models.doc2vec import Doc2Vec, TaggedDocument
...: doc = ["I love machine learning",
...:       "I love coding in python",
...:       "This is a good pc",
...:       "This is a good mac",
...:       "This is a good phone"]
...: tokenized_doc = ['love']
...: print(doc)
...: print(tokenized_doc)
['I love machine learning', 'I love coding in python', 'This is a good pc', 'This is a good mac', 'This is a good phone']
['I love machine learning', 'I love coding in python', 'This is a good pc', 'This is a good mac', 'This is a good phone']

```

Gambar 5.58 Nomor 3

```

1 tagged_data = [TaggedDocument(d, [i]) for i, d in enumerate(
2     tokenized_doc)]
3 tagged_data
## Train doc2vec model
4 model = Doc2Vec(tagged_data, vector_size=20, window=2, min_count=1,
5                 workers=4, epochs = 100)
# Save trained doc2vec model
6 model.save("test_doc2vec.model")
## Load saved doc2vec model
7 model= Doc2Vec.load("test_doc2vec.model")
## Print model vocabulary
10 model.wv.vocab

```

```

    # from google.cloud import storage
    # storage.Client().list_blobs('my-test-bucket')

    doc = [{"text": "I love machine learning", "label": "positive"}, {"text": "This is a good pc", "label": "positive"}, {"text": "This is a good mac", "label": "positive"}, {"text": "This is a good phone", "label": "positive"}]
    test_doc = ["I love"]

    print(test_doc)
    print(doc)

```

Gambar 5.59 Nomor 3-1

5.2.2.4 Nomor 4

```
1 import re
2 import os
3 unsup_sentences = []
4
5 # source: http://ai.stanford.edu/~amaas/data/sentiment/, data from
6 # IMDB
7 for dirname in ["train/pos", "train/neg", "train/unsup", "test/pos",
8 "test/neg"]:
9     for fname in sorted(os.listdir("N:/KB/aclImdb/" + dirname)):
10         if fname[-4:] == '.txt':
11             with open("N:/KB/aclImdb/" + dirname + "/" + fname,
12 encoding='UTF-8') as f:
13                 sent = f.read()
14                 words = (sent)
15                 unsup_sentences.append(TaggedDocument(words, [dirname
16 + "/" + fname]))
```

Disini memakai dataset dari aclImdb. Untuk menambahkan data training kita melakukan import library os, library os fungsinya untuk melakukan interaksi antara python dengan windows atau os kita, setelah itu buat variable unsup sentences. Lalu pilih direktori tempat data akan disimpan. Lalu untuk mensortir data yang terdapat pada folder aclImdb dan membaca file tersebut dengan ekstensi .txt. Hasil dari code pertama tersebut adalah adanya data hasil running dari folder aclImdb.

Gambar 5.60 Nomor 4

5.2.2.5 Nomor 5

```
1 #Pengacakan data
2 mute = (unsup_sentences)
3
4 #Pembersihan data
5 model.delete_temporary_training_data(keep_inference=True)
```

Pada bagian pengacakan data berguna untuk mengacak data agar saat data di running bisa berjalan lebih baik dan hasil presentase akan lebih baik. Sedangkan untuk

pembersihan data untuk memberikan ruang bagi ram laptop kita setelah melakukan running data lebih dari 3 juta , agar lebih ringan saat proses selanjutnya.

```
In [32]: mute = (unsup_sentences)
.....
.... membershion_data
.... model.delete_temporary_training_data(keep_inference=True)
```

Gambar 5.61 Nomor 5

5.2.2.6 Nomor 6

```
1 #Save data
2 model.save('dirga.d2v')
3
4 #Delete temporary data
5 model.delete_temporary_training_data(keep_inference=True)
```

Save berfungsi untuk menyimpan hasil dari proses pelatihan data sebelumnya ke dalam sebuah file, model tersebut dilakukan penyimpanan untuk memberikan keringanan pada ram agar saat akan melakukan pelatihan lagi, model tersebut bisa load tanpa harus melakukan pelatihan dari awal dan akan menghemat waktu. Sedangkan untuk delete temporary training data ini berfungsi untuk menghapus data latihan yang sebelumnya sudah dilakukan dan disimpan, bertujuan untuk memberikan keringanan pada ram. Karena setelah melakukan proses pelatihan ram biasanya jadi penuh sampai komputer menjadi lag. Itulah fungsi dari delete temporary training data

```
In [33]: model.save('dirga.d2v')
.....
.....
.... model.delete_temporary_training_data(keep_inference=True)
2020-04-05 21:30:22,528 : INFO : saving Doc2Vec object under dirga.d2v, separately None
2020-04-05 21:30:22,682 : INFO : saved dirga.d2v
```

Gambar 5.62 Nomor 6

5.2.2.7 Nomor 7

```
1 model.infer_vector(res)
```

Infer_vector berfungsi untuk membandingkan kata yang tercantum dengan vektor yang mana pada dokumen yang sudah diload pada step sebelumnya. Selain itu infer_vector juga untuk menghitung atau mengkalkulasikan vektor dari kata yang dicantumkan dari model yang telah dibuat. Alangkah baiknya kata yang dicantumkan itu lebih panjang lagi agar hasilnya bisa lebih baik lagi.

```
In [79]: model.infer_vector(res)
Out[79]:
array([-0.87325483, -0.45122747, -0.00295182, -0.001936466, 0.00201252,
       0.00529885, 0.00531824, -0.00417452, -0.00003136, 0.00755909,
       0.01598696, -0.00097159, 0.0155869, 0.01741282, 0.01701254,
       0.018909536, 0.01433238, 0.00834827, -0.01111711, 0.01431616],
      dtype='float32')
```

Gambar 5.63 Nomor 7

5.2.2.8 Nomor 8

```

1 from sklearn.metrics.pairwise import cosine_similarity
2 cosine_similarity(
3     [model.infer_vector(["datanya", "banyak"])],
4     [model.infer_vector(["pusing", "data"])])

```

Cosine_similarity berfungsi untuk membandingkan vektorisasi data diantara kedua kata yang di inputkan, jika hasil presentase dari kedua kata tersebut lebih dari 50% itu memiliki kemungkinan kata tersebut terdapat dalam 1 file. Namun jika kurang dari 50% itu kemungkinan kata tersebut tidak terdapat dalam 1 file. Hasil yang didapatkan pada code tersebut hanya 0.49% itu dikarenakan kata pertama dan kedua tidak memiliki kesamaan vektorisasi dan tidak terdapat pada salah satu dokumen.

```

In [83]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity(
...:     [model.infer_vector(["datanya", "banyak"])],
...:     [model.infer_vector(["pusing", "data"])])
Out[83]: array([-0.490003132], dtype=float32)

```

Gambar 5.64 Nomor 8

5.2.2.9 Nomor 9

```

1 from sklearn import datasets, linear_model
2 from sklearn.model_selection import cross_val_score
3 diabetes = datasets.load_diabetes()
4 X = diabetes.data[:150]
5 y = diabetes.target[:150]
6 lasso = linear_model.Lasso()
7 print(cross_val_score(lasso, X, y, cv=3))

```

Melakukan perhitungan presentase dengan menggunakan cross_validation dengan metode kneighborsClassifier. Menggunakan dataset iris

```

In [78]: from sklearn import datasets, linear_model
...: from sklearn.model_selection import cross_val_score
...: diabetes = datasets.load_diabetes()
...: X = diabetes.data[:150]
...: y = diabetes.target[:150]
...: lasso = linear_model.Lasso()
...: print(cross_val_score(lasso, X, y, cv=3))
[0.33150734 0.0802231 0.05531764]

```

Gambar 5.65 Nomor 9

5.2.3 Penanganan Error

- Error

1. FileNotFoundError

```

FileNotFoundError: [Errno 2] No such file or directory: 'N:/XBa/GoogleNews-vectors-negative300.bin'

```

Gambar 5.66 Error 1

- Solusi

1. FileNotFoundError

Perhatikan letak file ada dimana, pastikan path telah benar



Gambar 5.68 Vektorisasi Kata



Gambar 5.69 Dimensi Vektor

5.2.4 Bukti Tidak Plagiat



Gambar 5.67 Bukti Tidak Plagiat

5.3 Nurul Izza Hamka - 1174062

5.3.1 Teori

1. Jelaskan kenapa kata-kata harus dilakukan vektorisasi, dilengkapi dengan ilustrasi atau gambar.

Vektorisasi pada kata-kata harus dilakukan karena dengan dilakukannya vektorisasi kita dapat mengetahui berapa banyak jumlah kata yang muncul di setiap kalimat. Vektorisasi ini juga bertujuan untuk menentukan kata keywords atau kata kunci yang nantinya memudahkan pembaca dalam mencari kata atau kalimat yang dicari.

2. Jelaskan mengapa dimensi dari vektor dataset google bisa sampai 300, dilengkapi dengan ilustrasi atau gambar.

Karena setiap objek memiliki identitas khusus, misalnya dalam dataset Google ini memiliki 3 objek, kucing, anjing, dan ayam yang disetujui. Kemudian dari masing-masing objek dibandingkan dataset antara kucing dan anjing kemudian kucing dan ayam. Hasil yang diperoleh untuk kucing dan anjing sekitar 75 persen sedangkan untuk kucing dan ayam yang memiliki persentase 15 persen.

3. Jelaskan konsep vektorisasi untuk kata, dilengkapi dengan ilustrasi atau gambar

$$\bar{x} = \frac{x_1 + x_2 + x_3 + \dots + x_n}{n}$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

Gambar 5.70 Rumus Mean

Konsep untuk vektorisasi kata sama dengan masukan atau input pada kata-kata di mesin pencari. Maka anggaplah itu akan dikeluarkan sebagai saran tentang kata tersebut. Jadi kata data diperoleh dari hasil yang diolah dalam kalimat sebelumnya yang telah diproses.

Contoh sederhana dalam kalimat berikut (Silakan berlangganan saluran saya, terima kasih teman), dalam kalimat itu terkait dengan kalimat saluran, kata akan dibuat data pelatihan untuk mesin. Jadi kita kompilasi kata channel, maka mesin akan menampilkan hubungannya dengan kata tersebut.

4. Jelaskan konsep vektorisasi untuk dokumen, dilengkapi dengan ilustrasi atau gambar.

Dokumen bukanlah data terstruktur karena jauh dari bentuk tabel (baris dan kolom). Perlu metodologi pembentukan suatu data terstruktur untuk mewakili dokumen. Langkah awal adalah harus menentukan features yang mewakili seluruh kumpulan dokumen.

Sama halnya dengan vektorisasi kata, yang membedakan hanya pada proses awalnya. Untuk vektorisasi dokumen ini, mesin akan membaca semua kalimat yang terdapat pada dokumen tersebut, lalu kalimat yang terdapat pada dokumen akan dipecah menjadi kata-kata.

5. Jelaskan apa mean dan standar deviasi, dilengkapi dengan ilustrasi atau gambar.

Mean adalah perhitungan nilai rata-rata dari data yang ada. Nilai mean ini dapat kita ambil dari hasil membagi jumlah data dengan banyaknya data yang ada. Sedangkan standar deviasi adalah salah satu pengukuran statistika yang digunakan untuk menjelaskan bagaimana sebaran data dalam sampel, dan seberapa dekat titik data individu ke rata-rata nilai sampel.

6. Jelaskan apa itu skip-gram, dilengkapi dengan ilustrasi atau gambar.

Skip-gram adalah salah satu teknik pembelajaran unsupervised-learning yang digunakan untuk menemukan kata yang paling terkait untuk kata yang diberikan. Skipgram digunakan untuk memprediksi kata konteks untuk kata target yang diberikan. Ini kebalikan dari algoritma CBOW. Di sini, kata target adalah input sedangkan kata konteks adalah output.

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 satuan
- (b) Rumus untuk frekuensi lebih dari satu

Gambar 5.71 Standar Deviasi**Gambar 5.72** Skip-Gram**5.3.2 Praktek Program**

- Cobalah dataset google, dan jelaskan vektor dari kata Love, faith, fall, sick, clear, shine, bag, car, wash, motor, cycle, dan cobalah untuk melakukan perbandingan similitas dari masing-masing kata tersebut. Jelaskan arti outputan sinilariti dan setiap baris kode yang dibuat.

```

1 #%% Soal no 1
2
3 import gensim, logging
4 logging.basicConfig(format='%(asctime)s : %(levelname)s : %(
      message)s', level=logging.INFO)
5 izza_model = gensim.models.KeyedVectors.load_word2vec_format('
      GoogleNews-vectors-negative300.bin', binary=True, limit
      =500000)
  
```

Kode diatas untuk melakukan import library gensim. Gensim ini untuk melakukan pemodelan dengan dataset atau topik yang sudah ditentukan.

```

1
2 #%%
3 izza_model['love']
4 #%%
5 izza_model['faith']
6 #%%
7 izza_model['fall']
8 #%%
9 izza_model['sick']
10 #%%
11 izza_model['clear']
12 #%%
13 izza_model['shine']
  
```

```

14 #%%
15 izza_model[ 'bag' ]
16 #%%
17 izza_model[ 'car' ]
18 #%%
19 izza_model[ 'wash' ]
20 #%%
21 izza_model[ 'motor' ]
22 #%%
23 izza_model[ 'cycle' ]
24 #%%
25 izza_model.similarity('wash', 'clear')
26 #%%
27 izza_model.similarity('bag', 'love')
28 #%%
29 izza_model.similarity('motor', 'car')
30 #%%
31 izza_model.similarity('sick', 'faith')
32 #%%
33 izza_model.similarity('cycle', 'shine')

```

2. Jelaskan dengan kata dan ilustrasi fungsi dari extract words dan PermuteSentences (harus beda dengan teman sekelas)

```

1 #%% Soal no 2
2 import re
3
4
5 test_string = "Buah Apel, Makanan Sehat dan Bervitamin"
6 print ("Sangat bagus di komsumsi : " + test_string)
7 res = re.findall(r'\w+', test_string)
8
9 print ("The list of words is : " + str(res))

```

Kode diatas pertama melakukan import re dengan test_string yang berfungsi sebagai string.

Kemudian untuk kode diatas adalah berguna sebagai string dan memakai import random dengan memakai set_matrix untuk membuat string. Sedangkan result untuk print random yang akan diacak.

```

1 #%%
2
3
4 import random
5
6 sent_matrix = [ [ 'Buah', 'Apel' ],
7                 [ 'Mengandung', 'Vitamin' ],
8                 [ 'A', 'Dan' ],
9                 [ 'Mengandung', 'Kalium' ] ]

```

```

10 ]
11
12 result = ""
13 for elem in sent_matrix:
14     result += random.choice(elem) + " "

```

3. Jelaskan fungsi dari Librarian gensim TaggedDocument dan DocVec disertai praktek pemakaianya.

```

1
2     genres = ['blues', 'classical', 'country', 'disco', 'hiphop',
3                 'jazz', 'metal', 'pop', 'reggae', 'rock']
4     for genre in genres:
5         sound_files = glob.glob('dataset/genres//'+genre+'/*.au')
6         print('Processing %d songs in %s genre ...' % (len(
7             sound_files), genre))
8         for f in sound_files:
9             features = extract_features_song(f)
10            all_features.append(features)
11            all_labels.append(genre)
12
13 # convert labels to one-hot encoding cth blues : 1000000000
14 #           classic 0100000000
15 label_uniq_ids, label_row_ids = np.unique(all_labels,
16 return_inverse=True)#ke integer
17 label_row_ids = label_row_ids.astype(np.int32, copy=False)
18 onehot_labels = to_categorical(label_row_ids, len(
19 label_uniq_ids))#ke one hot
20
21 return np.stack(all_features), onehot_labels
22
23
24
25 # In[5]: Praktek Nomor 5
26 features, labels = generate_features_and_labels()
27 # In[5]: Soal Nomor 5
28 print(np.shape(features))
29 print(np.shape(labels))
30
31
32 # In[6]: Praktek Nomor 6
33 training_split = 0.8
34 # In[6]: Soal Nomor 6
35 alldata = np.column_stack((features, labels))

```

Fungsi dari doc2Vec adalah untuk membandingkan bobot data yang ada didalam dokumen lainnya.

4. Jelaskan dengan kata dan praktek cara menambahkan data training dari file yang dimasukkan kepada variabel dalam rangka melatih model doc2vec. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1
2
3 #%% Soal no 4
4 import re
5 import os
6 unsup_sentences = []
7
8 # source: http://ai.stanford.edu/~amaas/data/sentiment/, data
   from IMDB
9 for dirname in ["train/pos", "train/neg", "train/unsup", "test/"
   "pos", "test/neg"]:
10    for fname in sorted(os.listdir("aclImdb/" + dirname)):
11       if fname[-4:] == '.txt':
12          with open("aclImdb/" + dirname + "/" + fname,
13                     encoding='UTF-8') as f:
14             sent = f.read()
15             words = (sent)

```

Pertama kita lakukan import library re dan os. Libarary os untuk melakukan interaksi antara python dan laptop user. Selanjutnya kita gunakan dataset dengan nama aclimdb.

- Jelaskan dengan kata dan praktek kenapa harus dilakukan pengocokan dan pembersihan data.

```

1
2
3 #%% soal no 5
4 #Pengacakan data
5 mute = (unsup_sentences)
6
7 #Pembersihan data

```

- Jelaskan dengan kata dan praktek kenapa model harus di save dan kenapa temporary training harus dihapus. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1
2
3 #%% soal no 6
4 #Save data
5 model.save('izzah.d2v')
6
7 #Delete temporary data

```

Kode diatas adala save data ini befungsi untuk melakukan penyimpanan file asil dari proses latihan yang telah dilakukan.

Sedangkan delete temporary training data ini untuk menghapus data latihan yang sebelumnya.

7. Jalankan dengan kata dan praktek maksud dari infer code.

```

1
2 #%% soal no 7

```

Kode diatas adalah untuk membandingkan kata yang tercantum dengan vektor pada dokumen yang telah diload pada langkah sebelumnya.

8. Jelaskan dengan praktek dan kata maksud dari cosine similarity.

```

1 validation_split=0.2)
2
3
4 # In[10]: Praktek Nomor 10
5 loss , acc = model.evaluate(test_input , test_labels , batch_size
     =32)
6 # In[10]: Soal Nomor 10

```

Cosine similarity berfungsi untuk membandingkan vektorisasi data dianatara kedua kata yang diinputkan.

9. Jelaskan dengan praktek score dari cross validation masing-masing metode.

```

1
2 #%% soal no 9
3
4 from sklearn import datasets , linear_model
5 from sklearn.model_selection import cross_val_score
6 diabetes = datasets.load_diabetes()
7 X = diabetes.data[:150]
8 y = diabetes.target[:150]
9 lasso = linear_model.Lasso()

```

Kode diatas adalah melakukan perhitungan persentase dengan menggunakan cross validation dengan metode KneigborsClassifier

5.3.3 Bukti Tidak Plagiat



Gambar 5.73 Tidak Plagiat

BAB 6

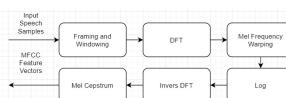
CHAPTER 6

6.1 1174067 - Kaka Kamaludin

6.1.1 Teori

6.1.1.1 *Kenapa file suara harus di lakukan MFCC*

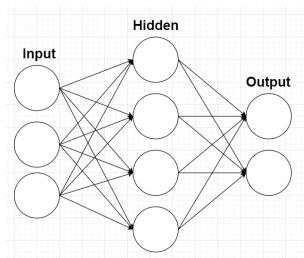
Karena MFCC adalah koefisien yang mewakili audio. Ekstraksi fitur dalam proses ini ditandai dengan konversi data suara menjadi gambar spektrum gelombang. File audio dilakukan oleh MFCC sehingga objek suara dapat dikonversi menjadi matriks. Suara akan menjadi vektor yang akan diproses sebagai output. Selain mempermudah mesin dalam bahasa ini karena mesin tidak dapat membaca teks, maka MFCC perlu mengubah suara menjadi vektor.



Gambar 6.1 Teori 1

6.1.1.2 Konsep dasar neural network

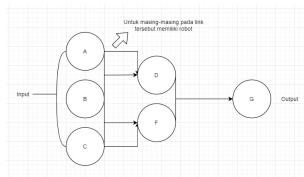
Konsep sederhana dari neural network atau jaringan saraf sederhana dengan proses pembelajaran pada anak-anak dengan memetakan pola-pola baru yang diperoleh dari input untuk membuat pola-pola baru pada output. Contoh sederhana ini men-ganalogikan kinerja otak manusia. Jaringan saraf itu sendiri terdiri dari unit pemrosesan yang disebut neuron yang berisi fungsi adder dan aktivasi. Fungsi akti-vasi itu sendiri untuk publikasi diberikan oleh neuron. Jaringan saraf yang men-dukung pemikiran sistem atau aplikasi yang melibatkan otak manusia, baik untuk mendukung berbagai elemen sinyal yang diterima, memeriksa kesalahan, dan juga memproses secara paralel. Karakteristik neural network atau jaringan saraf dilihat dari pola hubungan antar neuron, metode penentuan bobot setiap koneksi, dan fungsi aktivasi mereka.



Gambar 6.2 Teori 2

6.1.1.3 Konsep pembobotan dalam neural network

Pembobotan di dalam neural network juga akan menentukan penanda konektivitas. Dalam proses neural network mulai dari input yang diterima oleh neuron bersama dengan nilai bobot masing-masing input. Setelah memasuki neuron, nilai input akan ditambahkan oleh fungsi penerima. Hasil penambahan ini akan diproses oleh masing-masing fungsi neuron, hasil penambahan ini akan dibandingkan dengan nilai ambang tertentu. Jika nilai jumlah ini melebihi nilai ambang batas, aktivasi neuron akan dibatalkan, tetapi sebaliknya jika jumlah hasil di bawah nilai ambang batas, neuron akan diaktifkan. Setelah neuron aktif maka akan mengirimkan nilai output melalui bobot outputnya ke semua neuron yang terkait.



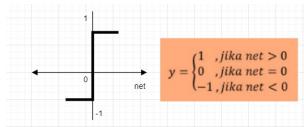
Gambar 6.3 Teori 3

6.1.1.4 Konsep fungsi aktifasi dalam neural network

Fungsi aktivasi dalam neural network ialah merupakan suatu operasi matematik yang

dikenakan pada sinyal output. Fungsi ini digunakan untuk mengaktifkan atau menon-aktifkan neuron. Fungsi aktivasi ini terbagi setidaknya menjadi 6, adapun sebagai berikut:

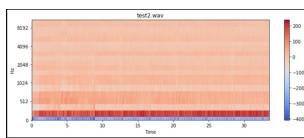
1. Fungsi Undak Biner Hard Limit, fungsi ini biasanya digunakan oleh jaringan lapiran tunggal untuk mengkonversi nilai input dari suatu variabel yang bernilai kontinu ke suatu nilai output biner 0 atau 1.
2. Fungsi Undak Biner Threshold, fungsi ini menggunakan nilai ambang sebagai batasnya.
3. Fungsi Bipolar Symetric Hard Limit, fungsi ini memiliki output bernilai 1, 0 atau -1.
4. Fungsi Bipolar dengan Threshold, fungsi ini mempunyai output yang bernilai 1, 0 atau -1 untuk batas nilai ambang tertentu.
5. Fungsi Linear atau Identitas.



Gambar 6.4 Teori 4

6.1.1.5 Cara membaca hasil plot dari MFCC

Perhatikan gambar dibawah: Gambar menjelaskan bahwa pada waktu ke-5 kekuatan atau layak yang dikeluarkan pada nada ini paling keras pada 20 Hz, selain itu pada 40 -120 Hz daya atau nilai dikeluarkan pada musik yang telah diplot. Demikian juga, warna tersebut adalah kekuatan atau nilai tertinggi dibandingkan dengan warna canggih. Yang merah terdengar di bawah pendengaran manusia, sehingga tidak bisa didengar secara langsung.



Gambar 6.5 Teori 5

6.1.1.6 one-hot encoding

one-hot encoding ini adalah untuk mengubah hasil data vektorisasi menjadi angka biner 0 dan 1 dan membuat informasi tentang atribut yang diberi label.

```

label_and_ids = label_row_ids + np.arange(label_ids.size, dtype='int32')
label_row_ids = label_row_ids.type(np.int32).copy('same')
onehot_labels = to_categorical(label_row_ids, len(label.unique_ids))#no one hot
onehot_labels = np.stack(all_features, onehot_labels)

```

Gambar 6.6 Teori 6-1

No	Color
0	Red
1	Green
2	Blue
3	Red
4	Blue

→

No	Red	Green	Blue
0	1	0	0
1	0	1	0
2	0	0	1
3	1	0	0
4	0	0	1

Gambar 6.7 Teori 6-2

6.1.1.7 Apa fungsi dari np.unique dan to_categorical dalam kode program

Fungsi np.unique adalah untuk menemukan berbagai elemen atau array unik, dan dapat mengembalikan elemen unik dari array yang diurutkan. Sebagai ilustrasi, cukup bisa dilihat pada gambar di bawah ini. Gambar tersebut menjelaskan bahwa unik itu sendiri akan mengambil data yang berbeda dari variabel a dalam fungsi array.

```

>>> import numpy as np
>>> x = np.array([[1,2],[1,1],[2,3],[4,5]])
>>> np.unique(x)
array([1, 2, 3, 4, 5])

```

Gambar 6.8 Teori 7-1

Fungsi dari to_categorical ialah untuk mengubah suatu vektor yang berupa integer menjadi matrix dengan kelas biner. Untuk ilustrasinya bisa dilihat pada gambar ??

```

>>> import pandas as pd
>>> var = pd.Categorical(["s","a","y","s","d","t","s","a"])
>>> print(var)
[s, a, y, s, d, t, s, a]
Categories (7, object): [s, d, t, a, s, y, s]

```

Gambar 6.9 Teori 7-2

6.1.1.8 Apa fungsi dari Sequential dalam kode program

Fungsi dari Sequential sebagai salah satu jenis model yang digunakan dalam perhitungan. Sequential ini membangun tumpukan linear yang berurutan.

```

In [38]: model = Sequential([
...     Dense(100, input_dim=np.shape(train_input)[1]),
...     Activation("relu"),
...     Dense(10),
...     Activation("softmax"),
... ])

```

Gambar 6.10 Teori 8

6.1.2 Praktek

6.1.2.1 Nomor 1

```

1 from keras.models import Sequential
2 from keras.utils.np_utils import to_categorical

```

```

3
4 def display_mfcc(song):
5     y, _ = librosa.load(song)
6     mfcc = librosa.feature.mfcc(y)
7
8     plt.figure(figsize=(10, 4))
9     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
10    plt.colorbar()
11    plt.title(song)
12    plt.tight_layout()
13    plt.show()
14 # In [2]: Soal 2
15 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/yuu_asterisk.wav')
16 # In [2]: Soal 2
17 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/Gosudarstvenny Gimn
    Soyuza Sovetskikh Sotsialisticheskikh Respublik.wav')
18 # In [2]: Soal 2
19 display_mfcc("D:/Documents/KULIAH/Semester 6/AI/genres/disco/disco
    .00006.wav")
20 # In [2]: Soal 2

```

Kode di atas menjelaskan isi data GTZAN. Ini adalah kumpulan data yang berisi 10 genre lagu dengan masing-masing genre memiliki 100 lagu yang akan kami lakukan proses MFCC dan juga lagu yang saya pilih, jika GTZAN memiliki beberapa genre jika freesound hanya untuk 1 lagu dan disini kita membuat fungsi untuk membaca file audio dan outputnya sebagai plot

```

In [1]: In [1]:
...: import librosa
...: import librosa.feature
...: import librosa.display
...: import numpy as np
...: import matplotlib.pyplot as plt
...: from tensorflow.keras.layers import Dense, Activation
...: from tensorflow.keras.models import Sequential
...: from tensorflow.keras.utils import to_categorical
...:
...: ...: def display_mfcc(song):
...: ...:     y, sr = 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()
...:
Using TensorFlow backend.

```

Gambar 6.11 Nomor 1

6.1.2.2 Nomor 2

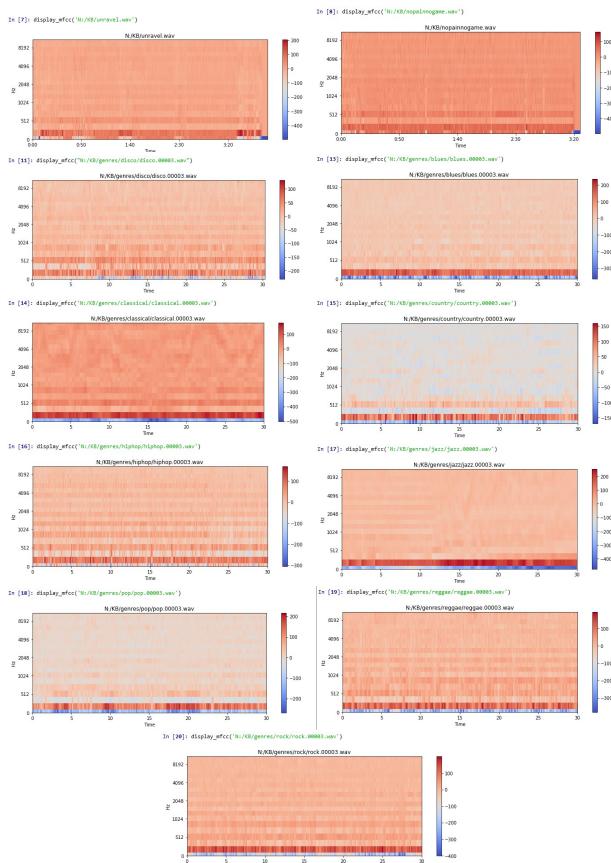
```

1 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/genres/blues/blues
    .00006.wav')
2 # In [2]: Soal 2
3 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/genres/classical/
    classical.00006.wav')
4 # In [2]: Soal 2
5 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/genres/country /
    country.00006.wav')
6 # In [2]: Soal 2
7 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/genres/hiphop/hiphop
    .00006.wav')
8 # In [2]: Soal 2
9 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/genres/jazz/jazz
    .00006.wav')

```

```
10 # In [2]: Soal 2
11 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/genres/pop/pop.00006.
   wav')
12 # In [2]: Soal 2
13 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/genres/reggae/reggae
   .00006.wav')
14 # In [2]: Soal 2
15 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/genres/rock/rock
   .00006.wav')
16 # In [2]: Soal 2
17 display_mfcc('D:/Documents/KULIAH/Semester 6/AI/genres/metal/metal
   .00006.wav')
18 # In [3]: Soal 3
19
20 def extract_features_song(f):
21     y, _ = librosa.load(f)
```

Kode di atas akan menampilkan hasil dari proses mfcc yang sudah dibuat fungsi pada soal 1, yaitu display_mfcc() dan akan menampilkan plot dari pembacaan file audio. Berikut adalah hasil setelah saya lakukan running dan pembacaan file audio:



Gambar 6.12 Nomor 2

6.1.2.3 Nomor 3

```

1 # normalize values between -1,1 (divide by max)
2 mfcc /= np.amax(np.absolute(mfcc))
3
4 return np.ndarray.flatten(mfcc)[:25000]
5
6 # In [4]: Soal 4
7
8 def generate_features_and_labels():
9     all_features = []

```

Baris pertama itu untuk membuat fungsi `extract_features_song(f)`. Pada baris kedua itu akan me-load data inputan dengan menggunakan librosa. Lalu selanjutnya untuk membuat sebuah fitur untuk mfcc dari `y` atau parameter inputan. Lalu akan me-return menjadi array dan akan mengambil 25000 data saja dari hasil vektorisasi dalam 1 lagu.

```
In [21]: def extract_features_song(f):
    ...
    y, sr = librosa.load(f)
    ...
    mfcc = librosa.feature.mfcc(y)
    ...
    # normalize feature between -1, 1 (divide by max)
    ...
    mfcc /= np.max(np.abs(mfcc))
    ...
    return np.ndarray.flatten(mfcc)[:25000]
```

Gambar 6.13 Nomor 3

6.1.2.4 Nomor 4

```
1   for genre in genres:
2       sound_files = glob.glob('D:/Documents/KULIAH/Semester 6/AI/
3   genres/*'+genre+'/*.wav')
4       print('Processing %d songs in %s genre ...' % (len(sound_files),
5   ), genre)
6       for f in sound_files:
7           features = extract_features_song(f)
8           all_features.append(features)
9           all_labels.append(genre)
10
11      # convert labels to one-hot encoding cth blues : 1000000000
12      # classic 0100000000
13      label_uniq_ids, label_row_ids = np.unique(all_labels,
14      return_inverse=True)#ke integer
15      label_row_ids = label_row_ids.astype(np.int32, copy=False)
16      onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids))
17      #ke one hot
18      return np.stack(all_features), onehot_labels
```

Kode di atas dapat digunakan untuk melakukan fungsi yang sebelumnya telah kita lakukan. Kemudian di bagian genre yang disesuaikan dengan dataset nama folder. Untuk baris berikutnya akan mengulang genre folder dengan ekstensi .au. Maka itu akan memanggil fungsi ekstrak lagu. Setiap file dalam folder itu akan diekstraksi menjadi vektor dan akan ditambahkan ke fitur. Dan fungsi yang ditambahkan adalah untuk menumpuk file yang telah di-vektor-kan. Hasil kode tidak menampilkan output.

```
In [26]: def generate_features_and_labels():
    ...
    ...
    genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae', 'rock']
    ...
    for genre in genres:
        genre_dir = os.path.join('genres', genre)
        for file in os.listdir(genre_dir):
            if file.endswith('.au'):
                f = os.path.join(genre_dir, file)
                features = generate_features(f)
                all_labels.append(genre)
    ...
    #stacking
    labels, labels = np.unique(all_labels, return_inverse=True)
    labels, labels = np.ndarray.flatten(labels), np.ndarray.flatten(labels)
    return np.stack(features), labels
```

Gambar 6.14 Nomor 4

6.1.2.5 Nomor 5

```
# In [6]: Soal 6
```

```
2 alldata = np.column_stack((features, labels))
3 # In [6]: Soal 6
```

Kode diatas berfungsi untuk melakukan load variabel features dan labels. Mengapa memakan waktu yang lama ? Karena mesin akan melakukan vektorisasi terhadap semua file yang berada pada setiap foldernya, di sini terdapat 10 folder dengan masing-masing folder terdiri atas 100 buah lagu, setiap lagu tersebut akan dilakukan vektorisasi atau ekstraksi data menggunakan mfcc.

```
In [27]: features, labels = generate_features_and_labels()
...: print(np.shape(features))
...: print(np.shape(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...
(1000, 25000)
(1000, 10)
```

features	float32	(1000, 25000)	[1.0, 0.0, ..., 0.0, 0.0, ..., 0.73384417, ..., -0.01833494, 0.0107234]
labels	float32	(1000, 10)	[1, 0, 0, ..., 0, 0, 0, ..., 0, 0, 0]

Gambar 6.15 Nomor 5

6.1.2.6 Nomor 6

```
1 np.random.shuffle(alldata)
2 splitidx = int(len(alldata) * training_split)
3 train, test = alldata[:splitidx ,:], alldata[splitidx :, :]
4 # In [6]: Soal 6
5 print(np.shape(train))
6 print(np.shape(test))
7 # In [6]: Soal 6
8 train_input = train[:, :-10]
9 train_labels = train[:, -10:]
10 # In [6]: Soal 6
11 test_input = test[:, :-10]
12 test_labels = test[:, -10:]
13 # In [6]: Soal 6
14 print(np.shape(train_input))
15 print(np.shape(train_labels))
16
17 # In [7]: Soal 7
18 model = Sequential([
19     Dense(100, input_dim=np.shape(train_input)[1]),
20     Activation('relu'),
```

Kode diatas berfungsi untuk melakukan training split 80%. Karena supaya mesin dapat terus belajar tentang data baru, jadi ketika prediksi dibuat tentang data yang terlatih itu bisa mendapatkan persentase yang cukup bagus.

The screenshot shows a Jupyter Notebook interface with several code cells and output sections.

- Cell 28:** Prints `training_split = 0.8`.
- Cell 29:** Prints the shapes of `alldata`, `features`, and `labels`. `alldata` is a float32 array of shape (1000, 2000). `features` is a float32 array of shape (1000, 2000). `labels` is a float32 array of shape (1000, 1).
- Cell 30:** Prints `train` and `test` arrays. Both are float32 arrays of shape (800, 2000) and (200, 2000) respectively.
- Cell 31:** Prints the shapes of `train` and `test`. Both have shapes (800, 25010).
- Cell 32:** Prints `train_input` and `train_labels`. Both have shapes (800, 25000) and (800, 10) respectively.
- Cell 33:** Prints `test_input` and `test_labels`. Both have shapes (200, 25000) and (200, 10) respectively.

Gambar 6.16 Nomor 6

6.1.2.7 Nomor 7

```

1     ])
2 # In [8]: Soal 8
3 model.compile(optimizer='adam',
4                 loss='categorical_crossentropy',
5                 metrics=['accuracy'])
6 print(model.summary())

```

Fungsi Sequential() ialah sebuah model untuk menentukan izin pada setiap neuron, di sini adalah 100 dense yang merupakan 100 neuron pertama dari data pelatihan. Fungsi dari relay itu sendiri adalah untuk mengaktifkan neuron atau input yang memiliki nilai maksimum. Sedangkan untuk dense 10 itu adalah output dari hasil neuron yang telah berhasil diaktifkan, untuk dense 10 diaktifkan menggunakan softmax.

```

In [35]: model = Sequential([
...     Dense(100, input_dim=np.shape(train_input)[1]),
...     Activation('relu'),
...     Dense(10),
...     Activation('softmax'),
... ])

```

Gambar 6.17 Nomor 7

6.1.2.8 Nomor 8

```

1 model.fit(train_input, train_labels, epochs=10, batch_size=32,

```

```

2 validation_split=0.2)
3 # In [10]: Soal 10
4 loss, acc = model.evaluate(test_input, test_labels, batch_size=32)

```

Model Compile di perjelas dengan gambar dibawah, Hasil output pada kode tersebut seperti gambar menjelaskan bahwa dense pertama itu memiliki 100 neurons dengan parameter sekitar 2 juta lebih dengan aktivasi 100, jadi untuk setiap neurons memiliki masing-masing 1 aktivasi. Sama halnya seperti dense 2 memiliki jumlah neurons sebanyak 10 dengan parameter 1010 dan jumlah aktivasinya 10 untuk setiap neurons tersebut dan total parameternya sekitar 2.5 juta data yang akan dilatih pada mesin tersebut.

```

In [36]: model.compile(optimizer='sgd',
...:         loss='categorical_crossentropy',
...:         metrics=['accuracy'])
...: print(model.summary())
Model: "sequential_1"
-----  

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 6.18 Nomor 8

6.1.2.9 Nomor 9

```

1 print("Loss: %.4f, accuracy: %.4f" % (loss, acc))

```

Kode tersebut berfungsi untuk melatih mesin dengan data training input dan training label. Epochs ini merupakan iterasi atau pengulangan berapa kali data tersebut akan dilakukan. Batch_size ini adalah jumlah file yang akan dilakukan pelatihan pada setiap 1 kali pengulangan. Sedangkan validation_split itu untuk menentukan persentase dari cross validation atau k-fold sebanyak 20% dari masing-masing data pengulangan.

```

In [37]: model.fit_generator(generator, steps_per_epoch=100, validation_steps=100,
...:                         epochs=3, validation_data=(val_input, val_labels),
...:                         verbose=1)
Epoch 1/20
100/100 [=====] - 2s/step - loss: 2.1899 - accuracy: 0.324 - val_loss: 1.7679 - val_accuracy:
Epoch 2/20
100/100 [=====] - 2s/step - loss: 1.0999 - accuracy: 0.6324 - val_loss: 1.4029 - val_accuracy:
Epoch 3/20
100/100 [=====] - 2s/step - loss: 1.5211 - accuracy: 0.4884 - val_loss: 1.4239 - val_accuracy:
Epoch 4/20
100/100 [=====] - 2s/step - loss: 1.1399 - accuracy: 0.6481 - val_loss: 1.3999 - val_accuracy:
Epoch 5/20
100/100 [=====] - 2s/step - loss: 1.0700 - accuracy: 0.6754 - val_loss: 1.5000 - val_accuracy:
Epoch 6/20
100/100 [=====] - 2s/step - loss: 0.8866 - accuracy: 0.7713 - val_loss: 1.0000 - val_accuracy:
Epoch 7/20
100/100 [=====] - 2s/step - loss: 0.9103 - accuracy: 0.7409 - val_loss: 1.4500 - val_accuracy:
Epoch 8/20
100/100 [=====] - 2s/step - loss: 0.9103 - accuracy: 0.7409 - val_loss: 1.2600 - val_accuracy:
Epoch 9/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.2400 - val_accuracy:
Epoch 10/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.0100 - val_accuracy:
Epoch 11/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.2700 - val_accuracy:
Epoch 12/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.0100 - val_accuracy:
Epoch 13/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.2700 - val_accuracy:
Epoch 14/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.0100 - val_accuracy:
Epoch 15/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.2700 - val_accuracy:
Epoch 16/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.0100 - val_accuracy:
Epoch 17/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.2700 - val_accuracy:
Epoch 18/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.0100 - val_accuracy:
Epoch 19/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.2700 - val_accuracy:
Epoch 20/20
100/100 [=====] - 2s/step - loss: 0.6228 - accuracy: 0.8578 - val_loss: 1.0100 - val_accuracy:
Out[37]: <keras.callbacks.callbacks.History at 0x10000000>

```

Gambar 6.19 Nomor 9

6.1.2.10 Nomor 10

```

1 model.predict(test_input[:1])

```

Fungsi evaluate atau evaluasi ini adalah untuk menguji data pengujian setiap file. Di sini ada prediksi yang hilang, artinya mesin memprediksi data, sedangkan untuk keseluruhan perjanjian sekitar 55%.

```
In [38]: loss, acc = model.evaluate(test_input, test_labels, batch_size=32
... :     print("Done!")
... :     print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
200/200 [=====] - 0s 45us/step
Done!
Loss: 1.3600 accuracy: 0.5150
```

Gambar 6.20 Nomor 10

6.1.2.11 Nomor 11

Fungsi Predict ialah untuk menghasilkan suatu nilai yang sudah di prediksi dari data training sebelumnya. Gambar dibawah ini menjelaskan file yang di jalankan tersebut termasuk ke dalam genre apa, hasilnya bisa dilihat pada gambar tersebut presentase yang paling besar yakni genre rock. Maka lagu tersebut termasuk ke dalam genre rock dengan perbandingan presentase hasil prediksi.

```
In [39]: model.predict(test_input[:1])
Out[39]:
array([[3.0364603e-01, 1.6553223e-04, 4.0861975e-02, 3.5557214e-02,
       1.2559776e-01, 1.8497287e-01, 1.2352003e-04, 1.2221429e-03,
       1.3322420e-01, 3.6558147e-01], [deutsche-fleet3]])
```

Gambar 6.21 Nomor 11

6.1.3 Penanganan Error

6.1.3.1 Error

- NoBackendError

Gambar 6.22 NoBackendError

6.1.3.2 Solusi Error

- NoBackendError

Cek kembali file format audionya dipastikan menggunakan wav

6.1.4 Link Youtube

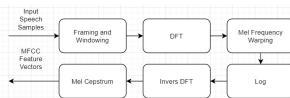
<https://www.youtube.com/playlist?list=PL4dhp4u89PHbhX9jrGyM3N12gmwhY3uE>

6.2 Mochamad Arifqi Ramadhan(1174074)

6.2.1 Teori

6.2.1.1 Kenapa file suara harus di lakukan MFCC

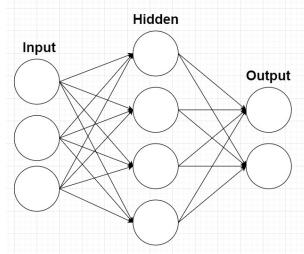
Karena MFCC adalah koefisien yang mewakili audio. Ekstraksi fitur dalam proses ini ditandai dengan konversi data suara menjadi gambar spektrum gelombang. File audio dilakukan oleh MFCC sehingga objek suara dapat dikonversi menjadi matriks. Suara akan menjadi vektor yang akan diproses sebagai output. Selain mempermudah mesin dalam bahasa ini karena mesin tidak dapat membaca teks, maka MFCC perlu mengubah suara menjadi vektor.



Gambar 6.23 Teori 1

6.2.1.2 Konsep dasar neural network

Konsep sederhana dari neural network atau jaringan saraf sederhana dengan proses pembelajaran pada anak-anak dengan memetakan pola-pola baru yang diperoleh dari input untuk membuat pola-pola baru pada output. Contoh sederhana ini men-ganalogikan kinerja otak manusia. Jaringan saraf itu sendiri terdiri dari unit pemrosesan yang disebut neuron yang berisi fungsi adder dan aktivasi. Fungsi aktivasi itu sendiri untuk publikasi diberikan oleh neuron. Jaringan saraf yang mendukung pemikiran sistem atau aplikasi yang melibatkan otak manusia, baik untuk mendukung berbagai elemen sinyal yang diterima, memeriksa kesalahan, dan juga memproses secara paralel. Karakteristik neural network atau jaringan saraf dilihat dari pola hubungan antar neuron, metode penentuan bobot setiap koneksi, dan fungsi aktivasi mereka.

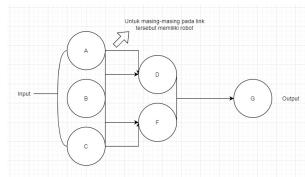


Gambar 6.24 Teori 2

6.2.1.3 Konsep pembobotan dalam neural network

Pembobotan di dalam neural network juga akan menentukan penanda konektivitas. Dalam proses neural network mulai dari input yang diterima oleh neuron bersama

dengan nilai bobot masing-masing input. Setelah memasuki neuron, nilai input akan ditambahkan oleh fungsi penerima. Hasil penambahan ini akan diproses oleh masing-masing fungsi neuron, hasil penambahan ini akan dibandingkan dengan nilai ambang tertentu. Jika nilai jumlah ini melebihi nilai ambang batas, aktivasi neuron akan dibatalkan, tetapi sebaliknya jika jumlah hasil di bawah nilai ambang batas, neuron akan diaktifkan. Setelah neuron aktif maka akan mengirimkan nilai output melalui bobot outputnya ke semua neuron yang terkait.

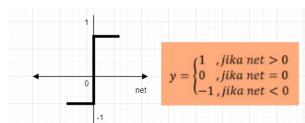


Gambar 6.25 Teori 3

6.2.1.4 Konsep fungsi aktifasi dalam neural network

Fungsi aktivasi dalam neural network ialah merupakan suatu operasi matematik yang dikenakan pada sinyal output. Fungsi ini digunakan untuk mengaktifkan atau menon-aktifkan neuron. Fungsi aktivasi ini terbagi setidaknya menjadi 6, adapun sebagai berikut:

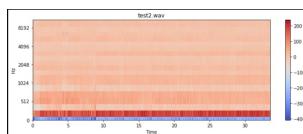
1. Fungsi Undak Biner Hard Limit, fungsi ini biasanya digunakan oleh jaringan lapiran tunggal untuk mengkonversi nilai input dari suatu variabel yang bernilai kontinu ke suatu nilai output biner 0 atau 1.
2. Fungsi Undak Biner Threshold, fungsi ini menggunakan nilai ambang sebagai batasnya.
3. Fungsi Bipolar Symetric Hard Limit, fungsi ini memiliki output bernilai 1, 0 atau -1.
4. Fungsi Bipolar dengan Threshold, fungsi ini mempunyai output yang bernilai 1, 0 atau -1 untuk batas nilai ambang tertentu.
5. Fungsi Linear atau Identitas.



Gambar 6.26 Teori 4

6.2.1.5 Cara membaca hasil plot dari MFCC

Perhatikan gambar dibawah: Gambar menjelaskan bahwa pada waktu ke-5 kekuatan atau layak yang dikeluarkan pada nada ini paling keras pada 20 Hz, selain itu pada 40 -120 Hz daya atau nilai dikeluarkan pada musik yang telah diplot. Demikian juga, warna tersebut adalah kekuatan atau nilai tertinggi dibandingkan dengan warna canggih. Yang merah terdengar di bawah pendengaran manusia, sehingga tidak bisa didengar secara langsung.



Gambar 6.27 Teori 5

6.2.1.6 one-hot encoding

one-hot encoding ini adalah untuk mengubah hasil data vektorisasi menjadi angka biner 0 dan 1 dan membuat informasi tentang atribut yang diberi label.

```
1 # convert labels to one-hot encoding cth blues : 1000000000.classic 9100000000
2 label_uniq_ids, label_row_ids = np.unique(all_labels, return_inverse=True) #like integer
3 label_row_ids = label_row_ids.astype(np.int32, copy=False)
4 oneshot_labels = -1 * categorical(label_row_ids, len(label_uniq_ids)) #like one hot
5 oneshot_labels = np.where(oneshot_labels >= 0, 1, oneshot_labels)
```

No	Color		No	Red	Green	Blue
0	Red	→	0	1	0	0
1	Green		1	0	1	0
2	Blue		2	0	0	1
3	Red		3	1	0	0
4	Blue		4	0	0	1

Gambar 6.29 Teori 6-2

6.2.1.7 Apa fungsi dari np.unique dan to categorical dalam kode program

Fungsi `np.unique` adalah untuk menemukan berbagai elemen atau array unik, dan dapat mengembalikan elemen unik dari array yang diurutkan. Sebagai ilustrasi, cukup bisa dilihat pada gambar di bawah ini. Gambar tersebut menjelaskan bahwa unik itu sendiri akan mengambil data yang berbeda dari variabel `a` dalam fungsi array.

```
>>> import numpy as np      ← 49 \be
>>> x = np.array([[1,2],[1,1],[2,3],[4,5]])    ← 50
>>> np.unique(x)           ← 51
array([1, 2, 3, 4, 5])    ← 52 \ce
```

Gambar 6.30 Teori 7-1

Fungsi dari `to_categorical` ialah untuk mengubah suatu vektor yang berupa integer menjadi matrix dengan kelas biner. Untuk ilustrasinya bisa dilihat pada gambar ??

```
>>> import pandas as pd
>>> var = pd.Categorical(['a', 'a', 'y', 'a', 'd', 't', 'r', 'g', 'a'])
>>> var
Categorical([a, a, y, a, d, t, r, g, a], categories=[a, d, g, t, r, s, y], ordered=False)
```

Gambar 6.31 Teori 7-2

6.2.1.8 Apa fungsi dari Sequential dalam kode program

Fungsi dari Sequential sebagai salah satu jenis model yang digunakan dalam perhitungan. Sequential ini membangun tumpukan linear yang berurutan.

```
In [35]: model = Sequential([
...:     Dense(100, input_dim=np.shape(train_input)[1]),
...:     Activation('relu'),
...:     Dense(10),
...:     Activation('softmax'),
...: ])
```

Gambar 6.32 Teori 8

6.2.2 Praktek

6.2.2.1 Nomor 1

```
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 def display_mfcc(song):
12     y, _ = librosa.load(song)
13     mfcc = librosa.feature.mfcc(y)
14
15     plt.figure(figsize=(10, 4))
16     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
17     plt.colorbar()
18     plt.title(song)
19     plt.tight_layout()
20     plt.show()
```

Kode di atas menjelaskan isi data GTZAN. Ini adalah kumpulan data yang berisi 10 genre lagu dengan masing-masing genre memiliki 100 lagu yang akan kami lakukan proses MFCC dan juga lagu yang saya pilih, jika GTZAN memiliki beberapa genre jika freesound hanya untuk 1 lagu dan disini kita membuat fungsi untuk membaca file audio dan outputnya sebagai plot

```

In [1]: import librosa
...: import librosa.feature
...: import librosa.display
...: import numpy as np
...: from matplotlib import pyplot as plt
...: from keras.layers import Dense, Activation
...: from keras.models import Sequential
...: from keras.utils import to_categorical
...:
...: def display_mfcc(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()
...:
Using TensorFlow backend.

```

Gambar 6.33 Nomor 1

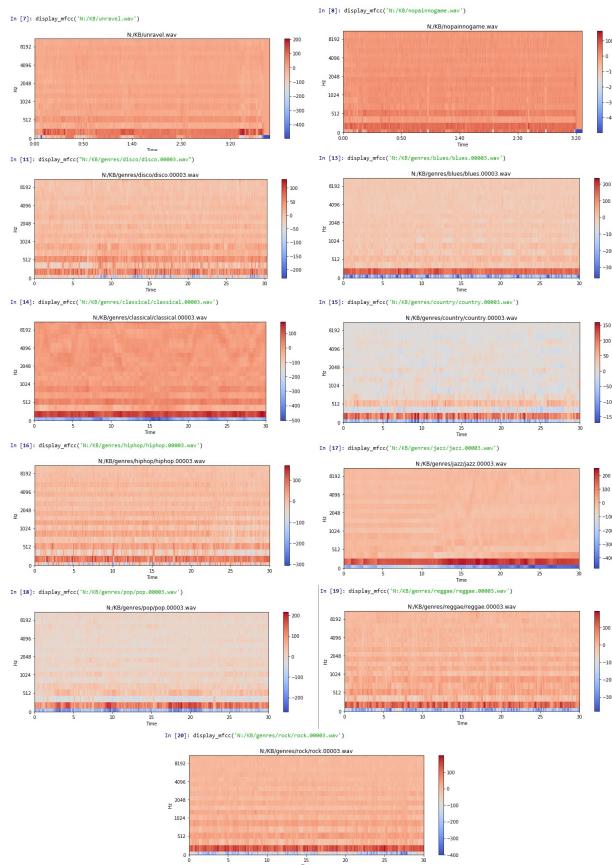
6.2.2.2 Nomor 2

```

1 # In[2]: Soal 2
2 display_mfcc('N:/KB/unravel.wav')
3 # In[2]: Soal 2
4 display_mfcc('N:/KB/nopainnogame.wav')
5 # In[2]: Soal 2
6 display_mfcc("N:/KB/genres/disco/disco.00003.wav")
7 # In[2]: Soal 2
8 display_mfcc('N:/KB/genres/blues/blues.00003.wav')
9 # In[2]: Soal 2
10 display_mfcc('N:/KB/genres/classical/classical.00003.wav')
11 # In[2]: Soal 2
12 display_mfcc('N:/KB/genres/country/country.00003.wav')
13 # In[2]: Soal 2
14 display_mfcc('N:/KB/genres/hiphop/hiphop.00003.wav')
15 # In[2]: Soal 2
16 display_mfcc('N:/KB/genres/jazz/jazz.00003.wav')
17 # In[2]: Soal 2
18 display_mfcc('N:/KB/genres/pop/pop.00003.wav')
19 # In[2]: Soal 2
20 display_mfcc('N:/KB/genres/reggae/reggae.00003.wav')
21 # In[2]: Soal 2
22 display_mfcc('N:/KB/genres/rock/rock.00003.wav')

```

Kode di atas akan menampilkan hasil dari proses mfcc yang sudah dibuat fungsi pada soal 1, yaitu display_mfcc() dan akan menampilkan plot dari pembacaan file audio. Berikut adalah hasil setelah saya lakukan running dan pembacaan file audio:



Gambar 6.34 Nomor 2

6.2.2.3 Nomor 3

```

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.amax(np.absolute(mfcc))
8
9     return np.ndarray.flatten(mfcc)[:25000]

```

Baris pertama itu untuk membuat fungsi `extract_features_song(f)`. Pada baris kedua itu akan me-load data inputan dengan menggunakan `librosa`. Lalu selanjutnya untuk membuat sebuah fitur untuk mfcc dari `y` atau parameter inputan. Lalu akan me-return menjadi array dan akan mengambil 25000 data saja dari hasil vektorisasi dalam 1 lagu.

```
In [21]: def extract_features_song(f):
    ...
    ...
    # get Mel-frequency cepstral coefficients
    ...
    # mfc = librosa.feature.mfcc(y)
    ...
    # normalize between -1,1 (divide by max)
    ...
    mfcc /= np.max(np.abs(mfcc))
    ...
    return np.ndarray.flatten(mfcc)[:25000]
```

Gambar 6.35 Nomor 3**6.2.2.4 Nomor 4**

```
1 def generate_features_and_labels():
2     all_features = []
3     all_labels = []
4
5     genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz',
6               'metal', 'pop', 'reggae', 'rock']
7     for genre in genres:
8         sound_files = glob.glob('N:/KB/genres/' + genre + '/*.wav')
9         print('Processing %d songs in %s genre ...' % (len(sound_files),
10 ), 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 : 1000000000
17     # 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(label_uniq_ids))
22     #ke one hot
23
24     return np.stack(all_features), onehot_labels
```

Kode di atas dapat digunakan untuk melakukan fungsi yang sebelumnya telah kita lakukan. Kemudian di bagian genre yang disesuaikan dengan dataset nama folder. Untuk baris berikutnya akan mengulang genre folder dengan ekstensi .au. Maka itu akan memanggil fungsi ekstrak lagu. Setiap file dalam folder itu akan diekstraksi menjadi vektor dan akan ditambahkan ke fitur. Dan fungsi yang ditambahkan adalah untuk menumpuk file yang telah di-vektor-kan. Hasil kode tidak menampilkan output.

```
In [26]: def generate_features_and_labels():
    ...
    ...
    genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae', 'rock']
    ...
    for genre in genres:
        genre_dir = os.path.join('N:/KB/genres', genre)
        for file in os.listdir(genre_dir):
            if file.endswith('.au'):
                f = os.path.join(genre_dir, file)
                features = extract_features_song(f)
                all_features.append(features)
    ...
    # convert labels to one-hot encoding
    label_uniq_ids, label_row_ids = np.unique(all_labels, return_inverse=True) #ke integer
    onehot_labels = to_categorical(label_row_ids, len(label_uniq_ids))
    #ke one hot
    return np.stack(all_features), onehot_labels
```

Gambar 6.36 Nomor 4**6.2.2.5 Nomor 5**

```
1 features, labels = generate_features_and_labels()
```

```
2 print(np.shape(features))
3 print(np.shape(labels))
```

Kode diatas berfungsi untuk melakukan load variabel features dan labels. Mengapa memakan waktu yang lama ? Karena mesin akan melakukan vektorisasi terhadap semua file yang berada pada setiap foldernya, di sini terdapat 10 folder dengan masing-masing folder terdiri atas 100 buah lagu, setiap lagu tersebut akan dilakukan vektorisasi atau ekstraksi data menggunakan mfcc.

```
In [27]: features, labels = generate_features_and_labels()
...: print(np.shape(features))
...: print(np.shape(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...
(1000, 25000)
(1000, 10)
```

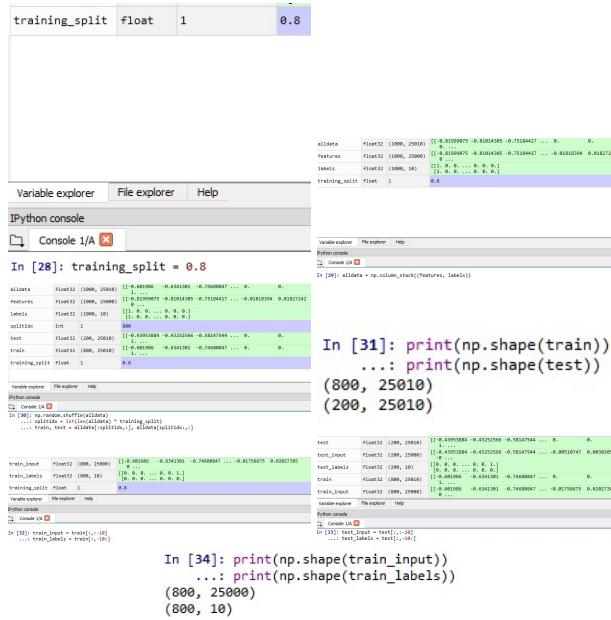
features	float32	(1000, 25000)	[1.0, -0.1234567, -0.3456789, -0.7894567, ...]	-0.01234567	-0.01234567
labels	float32	(1000, 10)	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	1, 0, 0, 0, 0, 0, 0, 0, 0, 0	

Gambar 6.37 Nomor 5

6.2.2.6 Nomor 6

```
1 # In [6]: Soal 6
2 training_split = 0.8
3 # In [6]: Soal 6
4 alldata = np.column_stack((features, labels))
5 # In [6]: Soal 6
6 np.random.shuffle(alldata)
7 splitidx = int(len(alldata) * training_split)
8 train, test = alldata[:splitidx, :], alldata[splitidx:, :]
9 # In [6]: Soal 6
10 print(np.shape(train))
11 print(np.shape(test))
12 # In [6]: Soal 6
13 train_input = train[:, :-10]
14 train_labels = train[:, -10:]
15 # In [6]: Soal 6
16 test_input = test[:, :-10]
17 test_labels = test[:, -10:]
18 # In [6]: Soal 6
19 print(np.shape(train_input))
20 print(np.shape(train_labels))
```

Kode diatas berfungsi untuk melakukan training split 80%. Karena supaya mesin dapat terus belajar tentang data baru, jadi ketika prediksi dibuat tentang data yang terlatih itu bisa mendapatkan persentase yang cukup bagus.



Gambar 6.38 Nomor 6

6.2.2.7 Nomor 7

```
1 model = Sequential([
2     Dense(100, input_dim=np.shape(train_input)[1]),
3     Activation('relu'),
4     Dense(10),
5     Activation('softmax'),
6 ])
```

Fungsi Sequential() ialah sebuah model untuk menentukan izin pada setiap neuron, di sini adalah 100 dense yang merupakan 100 neuron pertama dari data pelatihan. Fungsi dari relay itu sendiri adalah untuk mengaktifkan neuron atau input yang memiliki nilai maksimum. Sedangkan untuk dense 10 itu adalah output dari hasil neuron yang telah berhasil diaktifkan, untuk dense 10 diaktifkan menggunakan softmax.

```
In [35]: model = Sequential([
...     Dense(100, input_dim=np.shape(train_input)[1]),
...     Activation('relu'),
...     Dense(10),
...     Activation('softmax'),
... ])
```

Gambar 6.39 Nomor 7

6.2.2.8 Nomor 8

```
1 model.compile(optimizer='adam',
```

```

1 loss='categorical_crossentropy',
2 metrics=['accuracy'])
3 print(model.summary())
4

```

Model Compile di perjelas dengan gambar dibawah, Hasil output pada kode tersebut seperti gambar menjelaskan bahwa dense pertama itu memiliki 100 neurons dengan parameter sekitar 2 juta lebih dengan aktivasi 100, jadi untuk setiap neurons memiliki masing-masing 1 aktivasi. Sama halnya seperti dense 2 memiliki jumlah neurons sebanyak 10 dengan parameter 1010 dan jumlah aktivasinya 10 untuk setiap neurons tersebut dan total parameternya sekitar 2.5 juta data yang akan dilatih pada mesin tersebut.

```

In [36]: model.compile(optimizer='adam',
...:                 loss='categorical_crossentropy',
...:                 metrics=['accuracy'])
...: print(model.summary())
Model: "sequential_1"
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 6.40 Nomor 8

6.2.2.9 Nomor 9

```

1 model.fit(train_input, train_labels, epochs=10, batch_size=32,
2 validation_split=0.2)

```

Kode tersebut berfungsi untuk melatih mesin dengan data training input dan training label. Epochs ini merupakan iterasi atau pengulangan berapa kali data tersebut akan dilakukan. Batch_size ini adalah jumlah file yang akan dilakukan pelatihan pada setiap 1 kali pengulangan. Sedangkan validation_split itu untuk menentukan presentase dari cross validation atau k-fold sebanyak 20% dari masing-masing data pengulangan.

```

In [37]: model.fit(train_input, train_labels, epochs=10, batch_size=32,
...: validation_split=0.2)
Epoch 0/10 | 0s/0s - loss: 2.0000 - accuracy: 0.0000 | val_loss: 1.7070 - val_accuracy:
0.0000
Epoch 1/10 | 0s/0s - loss: 2.0000 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000
Epoch 2/10 | 0s/0s - loss: 1.7024 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000
Epoch 3/10 | 0s/0s - loss: 1.7024 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000
Epoch 4/10 | 0s/0s - loss: 1.7024 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000
Epoch 5/10 | 0s/0s - loss: 1.7024 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000
Epoch 6/10 | 0s/0s - loss: 1.7024 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000
Epoch 7/10 | 0s/0s - loss: 1.7024 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000
Epoch 8/10 | 0s/0s - loss: 1.7024 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000
Epoch 9/10 | 0s/0s - loss: 1.7024 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000
Epoch 10/10 | 0s/0s - loss: 1.7024 - accuracy: 0.0000 | val_loss: 1.7024 - val_accuracy:
0.0000

```

Gambar 6.41 Nomor 9

6.2.2.10 Nomor 10

```

1 loss, acc = model.evaluate(test_input, test_labels, batch_size=32)
2 print("Done!")
3 print("Loss: %.4f, accuracy: %.4f" % (loss, acc))

```

Fungsi evaluate atau evaluasi ini ialah untuk menguji data pengujian setiap file. Di sini ada prediksi yang hilang, artinya mesin memprediksi data, sedangkan untuk keseluruhan perjanjian sekitar 55%.

```
In [38]: loss, acc = model.evaluate(test_input, test_labels, batch_size=32)
...:   ...:   print("loss: %f, accuracy: %f" % (loss, acc))
...:   200/200 [=====] - 0s 453ms/step
Done! loss: 1.3699, accuracy: 0.5158
```

Gambar 6.42 Nomor 10

6.2.2.11 Nomor 11

```
| model.predict(test_input[:1])
```

Fungsi Predict ialah untuk menghasilkan suatu nilai yang sudah di prediksi dari data training sebelumnya. Gambar dibawah ini menjelaskan file yang di jalankan tersebut termasuk ke dalam genre apa, hasilnya bisa dilihat pada gambar tersebut presentase yang paling besar yakni genre rock. Maka lagu tersebut termasuk ke dalam genre rock dengan perbandingan presentase hasil prediksi.

```
In [39]: model.predict(test_input[:1])
Out[39]:
array([[3.0364608e-01, 6.8861075e-02, 3.5557214e-02,
       1.2599778e-01, 1.8497297e-01, 1.3252003e-04, 1.2221429e-05,
       1.3120548e-05, 1.0693814e-03]], dtype=float32)
```

Gambar 6.43 Nomor 11

6.2.3 Penanganan Error

6.2.3.1 Error

- NoBackendError

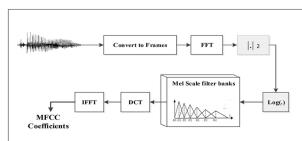
```
In [2]: display.winfo('nvidia-smi.log')
C:\Program Files\NVIDIA GPU P\bin\nvidia-smi.log:1: UserWarning: PybindFile failed. Trying address
location.
PybindFile('PybindFile', 'Trying without shared libfile')
traceback (most recent call last):
File "...", line 1, in __init__
    display_error('PybindFile', self._err)
File "...", line 1, in display_error
    File "...", line 1, in __init__
```

Gambar 6.44 NoBackendError

6.2.3.2 Solusi Error

- NoBackendError

Cek kembali file format audionya dipastikan menggunakan wav



Gambar 6.46 Mel Frequency Cepstrum Coefficient

6.2.4 Bukti Tidak Plagiat



Gambar 6.45 Bukti tidak plagiat

6.3 Nurul Izza Hamka - 1174062

6.3.1 Teori

1. Jelaskan kenapa file suara harus dilakukan MFCC, dilengkapi dengan ilustrasi atau gambar

Mel Frequency Cepstrum Coefficient (MFCC) adalah salah satu metode yang digunakan pada bidang speech recognition. Metode MFCC ini digunakan untuk melakukan feature extraction, yang mana sebuah proses yang mengkonversikan sinyal suara menjadi beberapa parameter. Metode MFCC memiliki keunggulan yaitu:

- Mampu menangkap karakteristik suara dan informasi-informasi yang sangat penting yang ada dalam sinyal suara.
- Mengurangi data seminimal mungkin, tapi tidak menghilangkan data dan informasi yang penting.
- Dapat mereplikasi organ pendengaran pada manusia dalam persepsi sinyal suara.

2. Jelaskan konsep Neural Network dilengkapi dengan ilustrasi atau gambar.

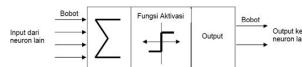
Ada beberapa konsep dari Neural Network yaitu:

- Proses kerja jaringan saraf pada otak manusia.

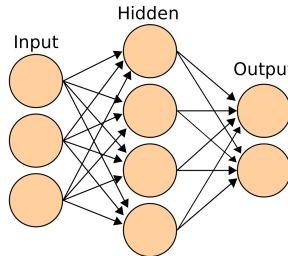
Ide dasar dari neural network ini dimulai dari otak manusia, yang mana otak memuat sekitar 1011 neuron. Satu neuron ini terdapat sat akson, dan min 1 dendrit.

- Struktur Neural network

Ide dasar dari Artifial Neural Network adalah mengadopsi mekanisme berpikir sebuah sistem atau aplikasi yang menyerupai otak manusia, baik untuk pemros-



Gambar 6.47 Konsep Dasar Neural Network



Gambar 6.48 Konsep Pembobotan Neural Network

esan berbagai sinyal.

Dari gambar di atas berikut penjelasannya:

- a. Input, sebagai dendrite
- b. Output, sebagai akson
- c. Fungsi aktivasi, sebagai sinapsis

3. Jelaskan konsep pembobotan dalam neural network dilengkapi dengan ilustrasi atau gambar.

Fleksibilitas yang dimiliki neural network adalah pemilihan input model, fungsi aktifasi yang digunakan dan juga pemilihan metode optimasi pada jaringan untuk mendapatkan bobot yang optimal. Bobot-bobot dalam Neural network biasanya diinisialisasi secara random dengan nilai random yang kecil. Jika dalam inisialisasi nilai bobot random yang diperoleh jauh dari solusi yang baik, atau dekat dengan nilai yang optimum local yang masih kurang baik, maka proses training akan cukup lama.

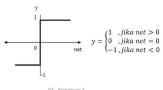
4. Jelaskan konsep fungsi aktifasi dalam neural network, dilengkapi dengan ilustrasi atau gambar

Fungsi aktifasi dalam neural network adalah fungsi matematis yang digunakan untuk mendapatkan sebuah output neuron dari nilai yang diinputkan. Ada beberapa fungsi aktifasi yang sering digunakan yaitu: hard limiter, signum activation dan sigmoid activation. Pada perceptron, fungsi aktifasi ini hanya terdapat pada neuron yang ada di output layer.

5. Jelaskan cara membaca asil plot dari MFCC dilengkapi dengan ilustrasi atau gambar sendiri.

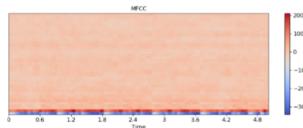
Fungsi Aktivasi

- Fungsi aktivasi adalah fungsi yang digunakan untuk mengaktifkan nilai keluaran (output) dari sebuah *neuron*.
- Pada perceptron, fungsi aktivasi hanya terdapat pada *neuron* yang berada di *output layer*.
- Fungsi aktivasi yang dipakai pada perceptron adalah *hard limit bipolar*.



37

Gambar 6.49 Fungsi Aktifasi Neural Network



Gambar 6.50 Plot MFCC

Label Encoding		One Hot Encoding
Food Name	Categorical #	
Apple	1	Apple Chicken Broccoli Calories
Chicken	2	0 1 0 231
Broccoli	3	0 0 1 50

Gambar 6.51 One-Hot Encoding

Dari gambar diatas diketahui:

- Terdapat dua dimensi yaitu x (waktu) dan y (power dan disable),
- Jika berwarna artinya power dari suara tersebut berarti rendah, dan jika merah artinya suara tinggi,
- Sedangkan untuk warna merah yang agak pudar artinya tidak ada suara sama sekali.

6. Jelaskan apa itu one-hot encoding dilengkapi ilustrasi kode dan atau gambar

One-hot encoding adalah proses dimana variabel kategorikal dikonversi menjadi bentuk yang dapat disediakan untuk algoritma ML untuk melakukan pekerjaan yang lebih baik dalam prediksi.

7. Jelaskan apa fungsi dari np.unique dalam kode program dilengkapi dengan ilustrasi atau gambar.

Fungsi dari np.unique sendiri adalah menemukan elemen unik dari array, dan juga mengembalikan sebuah elemen unik array yang telah diurutkan. Kemudian terdapat juga tiga output selain dari elemen unik yaitu:

- Indeks array input, yaitu untuk memberikan sebuah nilai yang unik,
- Indeks array unik, yaitu untuk melakukan rekonstruksi array input tadi,
- Dan yang terakhir adalah beberapa kali setiap nilai unik muncul dalam array

input tersebut.

8. Jelaskan apa fungsi dari sequential dalam kode program dilengkapi dengan ilustrasi atau gambar. Sequential sendiri berfungsi sebagai tumpukan lincar lapisan.

6.3.2 Praktek Program

1. Jelaskan isi dari data GTZAN Genre Collection dan data dari freesound. Buat kode program untuk meload data tersebut untuk digunakan pada MFCC.

```

1 # In [1]: Praktek Nomor 1
2
3 import librosa
4 import librosa.feature
5 import librosa.display
6 import glob
7 import numpy as np
8 import matplotlib.pyplot as plt
9 from keras.layers import Dense, Activation
10 from keras.models import Sequential
11 from keras.utils import np_utils
12
13 def display_mfcc(song):
14     y, sr = librosa.load(song)
15     mfcc = librosa.feature.mfcc(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()
```

2. Jelaskan perbaris kode program dengan kata-kata dan dilengkapi dengan ilustrasi gambar dari display mfcc.

```

1 # In [2]: Praktek Nomor 2
2 display_mfcc('test1.wav')
3 # In [2]: Soal Nomor 2
4 display_mfcc('test2.wav')
5 # In [2]: Soal Nomor 2
6 display_mfcc('dataset/genres/disco/disco.00069.au')
7 # In [2]: Soal Nomor 2
8 display_mfcc('dataset/genres/blues/blues.00069.au')
9 # In [2]: Soal Nomor 2
10 display_mfcc('dataset/genres/classical/classical.00069.au')
11 # In [2]: Soal Nomor 2
12 display_mfcc('dataset/genres/country/country.00069.au')
13 # In [2]: Soal Nomor 2
```

```
14 display_mfcc('dataset/genres/hiphop/hiphop.00069.au')
15 # In [2]: Soal Nomor 2
16 display_mfcc('dataset/genres/jazz/jazz.00069.au')
17 # In [2]: Soal Nomor 2
18 display_mfcc('dataset/genres/pop/pop.00069.au')
19 # In [2]: Soal Nomor 2
20 display_mfcc('dataset/genres/reggae/reggae.00069.au')
21 # In [2]: Soal Nomor 2
22 display_mfcc('dataset/genres/rock/rock.00069.au')
```

3. Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari extract feature song. Jelaskan mengapa yang diambil 25.000 baris pertama?

```
1 # In [3]: Praktek Nomor 3
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.amax(np.absolute(mfcc))
10
11    return np.ndarray.flatten(mfcc)[:25000]
```

4. Jelaskan perbaris kode program dengan kata-kata dan dilengkapi ilustrasi gambar fungsi dari generate features and labels().

```

17     label_row_ids = label_row_ids.astype(np.int32, copy=False)
18     onehot_labels = to_categorical(label_row_ids, len(
19         label_uniq_ids))#ke one hot
        return np.stack(all_features), onehot_labels

```

5. Jelaskan dengan kata dan praktek kanapa penggunaan fungsi generate feature and labels() sangat lama ketika meload dataset genre.

```

1 # In[5]: Praktek Nomor 5
2 features, labels = generate_features_and_labels()
3 # In[5]: Soal Nomor 5
4 print(np.shape(features))
5 print(np.shape(labels))

```

6. Jelaskan mengapa harus dilakukan pemisahan data training dan dataset sebesar 80 persen? Praktekkan dengan kode dan tunjukkan keluarannya.

```

1 # In[6]: Praktek Nomor 6
2 training_split = 0.8
3 # In[6]: Soal Nomor 6
4 alldata = np.column_stack((features, labels))
5 # In[6]: Soal Nomor 6
6 np.random.shuffle(alldata)
7 splitidx = int(len(alldata) * training_split)
8 train, test = alldata[:splitidx, :], alldata[splitidx:, :]
9 # In[6]: Soal Nomor 6
10 print(np.shape(train))
11 print(np.shape(test))
12 # In[6]: Soal Nomor 6
13 train_input = train[:, :-10]
14 train_labels = train[:, -10:]
15 # In[6]: Soal Nomor 6
16 test_input = test[:, :-10]
17 test_labels = test[:, -10:]
18 # In[6]: Soal Nomor 6
19 print(np.shape(train_input))
20 print(np.shape(train_labels))

```

7. Praktekkan dan jelaskan masing-masing parameter dari fungsi Sequential(). Tunjukkan keluarannya dari komputer sendiri.

```

1 # In[7]: Praktek Nomor 7
2 model = Sequential([
3     Dense(100, input_dim=np.shape(train_input)[1]),
4     Activation('relu'),
5     Dense(10),

```

```

6 Activation('softmax'),
7 ])

```

8. Praktekkan dan jelaskan masing-masing parameter dari fungsi fit(). Tunjukkan keluarannya dengan fungsi summary dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1 # In [8]: Praktek Nomor 8
2 model.compile(optimizer='adam',
3                 loss='categorical_crossentropy',
4                 metrics=['accuracy'])
5 print(model.summary())

```

9. Praktekkan dan jelaskan masing-masing parameter dari fungsi fit(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1 # In [9]: Praktek Nomor 9
2 model.fit(train_input, train_labels, epochs=10, batch_size=32,
3 validation_split=0.2)

```

10. Praktekkan dan jelaskan masing-masing parameter dari fungsi evaluate(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1 # In [10]: Praktek Nomor 10
2 loss, acc = model.evaluate(test_input, test_labels, batch_size
3                            =32)
4 # In [10]: Soal Nomor 10
5 print("Done!")
5 print("Loss: %.4f, accuracy: %.4f" % (loss, acc))

```

11. Praktekkan dan jelaskan masing-masing parameter dari fungsi predict(). Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1 # In [11]: Praktek Nomor 11
2 model.predict(test_input[:1])

```