

## **BAB 1**

---

## **CHAPTER 1**

---



## **BAB 2**

---

## **CHAPTER 2**

---



## **BAB 3**

---

## **CHAPTER 3**

---



## BAB 4

---

# CHAPTER 4

---

### 4.1 1174006 - Kadek Diva Krishna Murti

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

```
1 @inproceedings{awangga2017colenak,
2   title={Colenak: GPS tracking model for post-stroke
3   rehabilitation program using AES-CBC URL encryption and QR-
4   Code},
5   author={Awangga, Rolly Maulana and Fathonah, Nuraini Siti and
6   Hasanudin, Trisna Irmayadi},
7   booktitle={Information Technology, Information Systems and
8   Electrical Engineering (ICITISEE), 2017 2nd International
   conferences on},
9   pages={255--260},
10  year={2017},
11  organization={IEEE}
12 }
```



**Gambar 4.1** Kecerdasan Buatan.

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

#### **4.1.1 Teori**

#### **4.1.2 Praktek**

#### **4.1.3 Penanganan Error**

#### **4.1.4 Bukti Tidak Plagiat**



**Gambar 4.2** Kecerdasan Buatan.

## **4.2 1174069 - Fanny Shafira Damayanti**

#### **4.2.1 Teori**

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

Klasifikasi teks merupakan sebuah model yang biasa digunakan untuk 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 4.3** Klasifikasi teks.

2. Jelaskan mengapa hal ini bisa terjadi, klasifikasi bunga tidak bisa digunakan untuk machine learning, sertakan ilustrasi gambar sendiri.

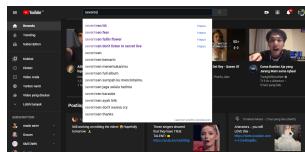
Karena machine learning tidak dapat menampilkan inputan sesuai dengan apa yang kita inputkan. Karena inputan tersebut serupa namun mesin memberikan output yang berbeda, biasanya output atau error ini disebut dengan istilah noise. Untuk contoh sederhananya misalkan kita inputkan salah satu label yang 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 4.4** Klasifikasi Bunga.

3. Jelaskan bagaimana yang dimaksud dengan teknik pembelajaran mesin pada teks yang digunakan dan sertakan ilustrasi buatan sendiri.

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 4.5** Klasifikasi teks Youtube.

4. Jelaskan apa yang dimaksud 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.

5. Jelaskan apa yang dimaksud dengan bag of words dengan ilustrasi sendiri.

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 4.6** Bag of words.

6. Jelaskan apa yang dimaksud dengan 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 4.7** TF-IDF.

#### 4.2.2 Praktek Program

1. Soal 1

```

1 #%% Soal 1
2 import pandas as pd #digunakan untuk mengimport library
# pandas dengan alias pd
3 pd = pd.read_csv("F:/Semester 6/Artificial Intelligence/
# Tugas 4/src/csv_fanny.csv") #membaca file csv

```

Kode di atas digunakan untuk buat aplikasi sederhana menggunakan pandas dengan format csv sebanyak 500 baris, hasilnya ialah sebagai berikut :

**Gambar 4.8** Hasil Soal 1.

## 2. Soal 2

```

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

```

Kode di atas digunakan untuk memecah dataframe tersebut menjadi dua bagian yaitu 450 row pertama dan 50 row kedua. Hasilnya adalah sebagai berikut :

**Gambar 4.9** Hasil Soal 2.

## 3. Soal 3

```

1 #%% Soal 3
2 import pandas as fanny #untuk import library pandas berguna
      untuk mengelola dataframe
3 fanny = fanny.read_csv("F://Semester 6/Artificial
      Intelligence/Tugas 4/src/Youtube03-LMFAO.csv") #membaca
      file dengan format csv
4
5 spam=fanny.query('CLASS == 1') #membagi tabel spam
6 nospam=fanny.query('CLASS == 0')#membagi tabel no spam
7
8 from sklearn.feature_extraction.text import CountVectorizer #
      untuk import countvectorizer berfungsi untuk memecah data
      tersebut menjadi sebuah kata yang lebih sederhana
9 vectorizer = CountVectorizer () #ntuk menjalankan fungsi
      tersebut , pada code ini tidak ada hasilnya dikarenakan
      spyder tidak mendukung hasil dari instasiasi.
10
11 dvec = vectorizer.fit_transform(fanny['CONTENT']) #untuk
      melakukan pemecahan data pada dataframe yang terdapat pada
      kolom konten

```

```
1 dvec #Untuk menampilkan hasil dari code sebelumnya
2
3 Daptarkata= vectorizer.get_feature_names()
4
5 dshuf = fanny.sample(frac=1)
6
7
8 d_train=dshuf[:300]
9 d_test=dshuf[300:]
10
11 d_train_att = vectorizer.fit_transform(d_train['CONTENT'])
12 d_train_att
13
14 d_train_label=d_train['CLASS']
15 d_test_label=d_test['CLASS']
```

Dengan menggunakan  $1174069 \bmod 4$  adalah 1, yang artinya menggunakan dataset LMFAO. Hasilnya adalah sebagai berikut :

**Gambar 4.10** Hasil Soal 3.

#### 4. Soal 4

```
1 #%% Soal 4
2
3 from sklearn import svm
4 clfsvm = svm.SVR(gamma = 'auto')
5 clfsvm.fit(d_train_att, d_train_label)
```

Klasifikasi dari data vektorisasi menggunakan klasifikasi Decision Tree. Hasilnya adalah sebagai berikut :

```
In [7]: from sklearn import svm
...: clfsvm = svm.SVR(gamma='auto')
...: clfsvm.fit(d_train_att, d_train_label)
Out[7]: SVR(C=10.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
gamma='auto',
kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
verbose=False)
```

Gambar 4.11 Hasil Soal 4.

## 5. Soal 5

```

1 #%%soal 5
2
3 from sklearn import tree
4 clftree = tree.DecisionTreeClassifier()
5 clftree.fit(d_train_att, d_train_label)

```

Plotlah confusion matrix dari praktik modul ini menggunakan matplotlib.  
Hasilnya adalah sebagai berikut :

```

In [8]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
Out[8]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
criterion='gini',
max_depth=None, max_features=None,
max_leaf_nodes=None,
min_impurity_decrease=0.0,
min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0,
presort='deprecated',
random_state=None, splitter='best')

```

**Gambar 4.12** Hasil Soal 5.

## 6. Soal 6

```

1 #%%soal 6/
2
3 from sklearn.metrics import confusion_matrix
4 pred_labels=clftree.predict(d_test)
5 cm=confusion_matrix(d_test_label, pred_labels)
6
7 #%%
8
9 import matplotlib.pyplot as plt
10
11 def plot_confusion_matrix(cm, classes,
12                           normalize=False,
13                           title='Confusion matrix',
14                           cmap=plt.cm.Blues):
15
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.
18         newaxis]
19         print("Normalized confusion matrix")
20     else:
21         print('Confusion matrix, without normalization')
22
23     print(cm)

```

Menjalankan program cross validation. Hasilnya adalah sebagai berikut :

```
In [11]: import matplotlib.pyplot as plt
...
...: def plot_confusion_matrix(cm, classes,
...:                         normalize=False,
...:                         title='Confusion matrix',
...:                         cmap=plt.cm.Blues):
...:     """
...:     If normalize is True, cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
...:     """
...:     if normalize:
...:         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
...:         print("Normalized confusion matrix")
...:     else:
...:         print('Confusion matrix, without normalization')
...:     print(cm)
...:     cm
```

**Gambar 4.13** Hasil Soal 6.

## 7. Soal 7

```
1 %%soal 7
2
3 from sklearn.model_selection import cross_val_score
4
5 scores=cross_val_score(clf, d_train_att, d_train_label, cv=5)
6
7 skor_rata2=scores.mean()
8 skoresd=scores.std()
```

Tree.export graphviz merupakan fungsi yang menghasilkan representasi Graphviz dari decision tree, yang kemudian ditulis ke outfile. Disini akan menyimpan classifiernya, akan meng ekspor file student performance jika salah akan mengembalikan nilai fail. Hasilnya adalah sebagai berikut :

```
In [12]: from sklearn.model_selection import cross_val_score
...
...: scores=cross_val_score(clf, d_train_att, d_train_label, cv=5)
...: skor_rata2=scores.mean()
...: skoresd=scores.std()
```

**Gambar 4.14** Hasil Soal 7.

## 8. Soal 8

```
1 %%soal 8 /
2
3 max_features_opts = range(5, 50, 5) #max_features_opts
4 n_estimators_opts = range(10, 200, 20) #n_estimators_opts
5 rf_params = fanny.empty((len(max_features_opts)*len(
6     n_estimators_opts),4), float) #rf_params sebagai variabel
7     untuk menjumlahkan yang sudah di tentukan sebelumnya
8 i = 0
9 for max_features in max_features_opts: #pengulangan
10    for n_estimators in n_estimators_opts: #pengulangan
11        clf = RandomForestClassifier(max_features=
12            max_features, n_estimators=n_estimators) #menampilkan
13            variabel csf
14            scores = cross_val_score(clf, df_train_att,
15            df_train_label, cv=5) #scores sebagai variabel training
```

```

11     rf_params[i,0] = max_features #index 0
12     rf_params[i,1] = n_estimators #index 1
13     rf_params[i,2] = scores.mean() #index 2
14     rf_params[i,3] = scores.std() * 2 #index 3
15     i += 1 #dengan ketentuan i += 1
16     print("Max features: %d, num estimators: %d, accuracy
17 : %0.2f (+- %0.2f)" %(max_features, n_estimators, scores.
mean(), scores.std() * 2))
# print hasil pengulangan yang sudah ditentukan

```

Buatlah program pengamatan komponen informasi. Jadi disini kita akan memprediksi nilai dari variabel test att dan test pass Hasilnya adalah sebagai berikut :

**Gambar 4.15** Hasil Soal 8.

### 4.2.3 Penanganan Error

## 1. ScreenShoot Error

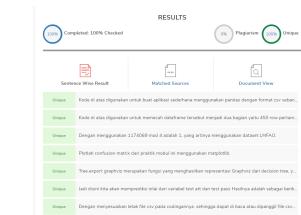
```
FileNotFoundException: [Errno 2] File b'F:/Semester 6/Artificial Intelligence/Tugas 4/src/fanny.csv' does not exist: b'F:/Semester 6/Artificial Intelligence/Tugas 4/src/fanny.csv'
```

## 2. Cara Penangan Error

- SyntaxError

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

#### 4.2.4 Bukti Tidak Plagiat



**Gambar 4.17** Bukti Tidak Melakukan Plagiat Chapter 4

#### 4.2.5 Link Youtube

<https://youtu.be/X-xd9Nb78Gs>

### 4.3 1174070 - Arrizal Furqona Gifary

#### 4.3.1 Teori

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

Klasifikasi teks merupakan sebuah model yang biasa digunakan untuk 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 4.18** Klasifikasi teks.

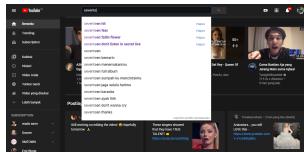
2. Jelaskan mengapa hal ini bisa terjadi, klasifikasi bunga tidak bisa digunakan untuk machine learning, sertakan ilustrasi gambar sendiri. Karena machine learning tidak dapat menampilkan inputan sesuai dengan apa yang kita inputkan. Karena inputan tersebut serupa namun mesin memberikan output yang berbeda, 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 4.19** Klasifikasi Bunga.

3. Jelaskan bagaimana yang dimaksud dengan teknik pembelajaran mesin pada teks yang digunakan dan sertakan ilustrasi buatan sendiri.

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 4.20** Klasifikasi teks Youtube.

4. Jelaskan apa yang dimaksud 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.

5. Jelaskan apa yang dimaksud dengan bag of words dengan ilustrasi sendiri.

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



**Gambar 4.21** Bag of words.

6. Jelaskan apa yang dimaksud dengan 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 4.22 TF-IDF.

#### 4.3.2 Praktek Program

##### 1. Soal 1

```

1 %% Soal 1
2 import pandas as pd #digunakan untuk mengimport library
# pandas dengan alias pd
3 pd = pd.read_csv("F:/Semester 6/Artificial Intelligence/
#Tugas 4/src/csv_izal.csv") #membaca file csv

```

Kode di atas digunakan untuk buat aplikasi sederhana menggunakan pandas dengan format csv sebanyak 500 baris, hasilnya ialah sebagai berikut :



Gambar 4.23 Hasil Soal 1.

##### 2. Soal 2

```

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

```

Kode di atas digunakan untuk memecah dataframe tersebut menjadi dua bagian yaitu 450 row pertama dan 50 row kedua. Hasilnya adalah sebagai berikut :



Gambar 4.24 Hasil Soal 2.

## 3. Soal 3

```

1 %% Soal 3
2 import pandas as izal #untuk import library pandas berguna
# untuk mengelola dataframe
3 izal = izal.read_csv("F:// Semester 6/Artificial Intelligence/
# Tugas 4/src/Youtube03-LMFAO.csv") #membaca file dengan
# format csv
4
5 spam=izal.query('CLASS == 1') #membagi tabel spam
6 nospam=izal.query('CLASS == 0')#membagi tabel no spam
7
8 from sklearn.feature_extraction.text import CountVectorizer #
# untuk import countvectorizer berfungsi untuk memecah data
# tersebut menjadi sebuah kata yang lebih sederhana
9 vectorizer = CountVectorizer () #ntuk menjalankan fungsi
# tersebut , pada code ini tidak ada hasilnya dikarenakan
# spyder tidak mendukung hasil dari instasiasi.
10
11 dvec = vectorizer.fit_transform(izal[ 'CONTENT' ]) #untuk
# melakukan pemecahan data pada dataframe yang terdapat pada
# kolom konten
12 dvec #Untuk menampilkan hasil dari code sebelumnya
13
14 Daptarkata= vectorizer.get_feature_names()
15
16 dshuf = izal.sample(frac=1)
17
18 d_train=dshuf[:300]
19 d_test=dshuf[300:]
20
21 d_train_att = vectorizer.fit_transform(d_train[ 'CONTENT' ])
22 d_train_att
23
24 d_train_label=d_train[ 'CLASS' ]
25 d_test_label=d_test[ 'CLASS' ]

```

Dengan menggunakan  $1174070 \bmod 4$  adalah 1, yang artinya menggunakan dataset LMFAO. Hasilnya adalah sebagai berikut :

Family - DataName					
Index	BOOK_ID	AUTHOR	DATE	CONTENT	CLASS
0	1	shane@shepd...	Cory Wilson	2015-05-29T0...	http://www...
1	12	steve@shepd...	Todd Giesing	2015-05-29T0...	http://www...
2	13	steve@shepd...	Luis Madero	2015-05-29T0...	http://www...
3	13	steve@shepd...	Cherry Fox	2015-05-29T0...	http://www...
4	14	steve@shepd...	PATRICK Z...	2015-05-29T0...	Party rock
5	15	steve@shepd...	Brian Bral...	2015-05-29T0...	Shuffle
6	16	steve@shepd...	Brian Bral...	2015-05-29T0...	Ong
7	17	steve@shepd...	Ale Defeo	2015-05-29T0...	This song i...
8	18	steve@shepd...	Giovanni	2015-05-29T0...	Ausssssssseeeeee...
9	19	steve@shepd...	Silvia Bassu	2015-05-29T0...	www!!!!!!!
10	20	steve@shepd...	Michael	2015-05-29T0...	I love this
11	21	steve@shepd...	merle	2015-05-29T0...	Merle
12	22	steve@shepd...	Alex Marin	2015-05-29T0...	I miss when
13	23	steve@shepd...	Alex Jansen	2015-05-29T0...	people dress
14	24	steve@shepd...	adam will	2015-05-29T0...	the last year of
15	25	steve@shepd...	Joe peplin	2015-05-29T0...	we ever!!!
16	26	steve@shepd...	leland	2015-05-29T0...	love music
17	27	steve@shepd...	luke kereo	2015-05-29T0...	Party rock
18	28	steve@shepd...	SoulJaCali	2015-05-29T0...	(8)
19	29	steve@shepd...	Nguyen Ngoc	2015-05-29T0...	and if you watched
20	30	steve@shepd...	Ferraro	2015-05-29T0...	you - lol.
21	31	steve@shepd...	steve	2015-05-27T1...	(increased)
22	32	steve@shepd...	steve	2015-05-27T1...	(decreased)

**Gambar 4.25** Hasil Soal 3.

#### 4. Soal 4

```
1 #%% Soal 4
2
3 from sklearn import svm
4 clfsvm = svm.SVR(gamma = 'auto')
5 clfsvm.fit(d_train_att, d_train_label)
```

Klasifikasi dari data vektorisasi menggunakan klasifikasi Decision Tree. Hasilnya adalah sebagai berikut :

```
In [7]: from sklearn import svm
...: clfsvm = svm.SVR(gamma = 'auto')
...: clfsvm.fit(dtrain_att, dtrain_label)
Out[7]:
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
gamma='auto',
kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
verbose=False)
```

**Gambar 4.26** Hasil Soal 4.

### 5. Soal 5

```
1 #%%soal 5
2
3 from sklearn import tree
4 clftree = tree.DecisionTreeClassifier()
5 clftree.fit(d_train_att, d_train_label)
```

Plotlah confusion matrix dari praktik modul ini menggunakan matplotlib. Hasilnya adalah sebagai berikut :

```
In [8]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(d_train_att, d_train_label)
Out[8]:
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
criterion='gini',
max_depth=None, max_features=None,
max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0,
presort='deprecated', random_state=None, splitter='best')
```

**Gambar 4.27** Hasil Soal 5.

## 6. Soal 6

```
1 #%%soal 6/
2
3 from sklearn.metrics import confusion_matrix
4 pred_labels=clftree.predict(d_test)
5 cm=confusion_matrix(d_test_label ,pred_labels)
6
7 #%%
8
9 import matplotlib.pyplot as plt
10
11 def plot_confusion_matrix(cm, classes,
12                           normalize=False,
13                           title='Confusion matrix',
14                           cmap=plt.cm.Blues):
15
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.
newaxis]
18         print("Normalized confusion matrix")
19     else:
20         print('Confusion matrix, without normalization')
21
22     print(cm)
```

Menjalankan program cross validation. Hasilnya adalah sebagai berikut :

```
In [11]: import matplotlib.pyplot as plt
...:
...: def plot_confusion_matrix(cm, classes,
...:                         normalize=False,
...:                         title='Confusion matrix',
...:                         cmap=plt.cm.Blues):
...:
...:     if normalize:
...:         cm = cm.astype('float') / cm.sum(axis=1)[:, np.
newaxis]
...:         print("Normalized confusion matrix")
...:     else:
...:         print('Confusion matrix, without normalization')
...:
...:     print(cm)
...:     cm
```

**Gambar 4.28** Hasil Soal 6.

## 7. Soal 7

```
1 #%%soal 7
2
```

```

3 from sklearn.model_selection import cross_val_score
4
5 scores=cross_val_score(clftree,d_train_att,d_train_label, cv
6 =5)
7
8 skor_rata2=scores.mean()
9 skoresd=scores.std()

```

Tree.export graphviz merupakan fungsi yang menghasilkan representasi Graphviz dari decision tree, yang kemudian ditulis ke outfile. Disini akan menyimpan classifiernya, akan meng ekspor file student performance jika salah akan mengembalikan nilai fail. Hasilnya adalah sebagai berikut :

```

In [12]: from sklearn.model_selection import cross_val_score
...:
scores=cross_val_score(clftree,d_train_att,d_train_label, cv=5)
...:
skor_rata2=scores.mean()
...:
skoresd=scores.std()

```

**Gambar 4.29** Hasil Soal 7.

## 8. Soal 8

```

1 %%soal 8 /
2
3 max_features_opts = range(5, 50, 5) #max_features_opts
4 sebagai variabel untuk membuat range 5,50,5
5 n_estimators_opts = range(10, 200, 20) #n_estimators_opts
6 sebagai variabel untuk membuat range 10,200,20
7 rf_params = izal.empty((len(max_features_opts)*len(
8     n_estimators_opts),4), float) #rf_params sebagai variabel
9 untuk menjumlahkan yang sudah di tentukan sebelumnya
10 i = 0
11 for max_features in max_features_opts: #pengulangan
12     for n_estimators in n_estimators_opts: #pengulangan
13         clftree = RandomForestClassifier(max_features=
14             max_features, n_estimators=n_estimators) #menampilkan
15         variabel csf
16         scores = cross_val_score(clf, df_train_att,
17             df_train_label, cv=5) #scores sebagai variabel training
18         rf_params[i,0] = max_features #index 0
19         rf_params[i,1] = n_estimators #index 1
20         rf_params[i,2] = scores.mean() #index 2
21         rf_params[i,3] = scores.std() * 2 #index 3
22         i += 1 #dengan ketentuan i += 1
23         print("Max features: %d, num estimators: %d, accuracy
24 : %0.2f (+/- %0.2f)" %(max_features, n_estimators, scores.
25 mean(), scores.std() * 2))
26         #print hasil pengulangan yang sudah ditentukan

```

Buatlah program pengamatan komponen informasi. Jadi disini kita akan memprediksi nilai dari variabel test att dan test pass Hasilnya adalah sebagai berikut :

**Gambar 4.30** Hasil Soal 8.

### 4.3.3 Penanganan Error

## 1. ScreenShoot Error

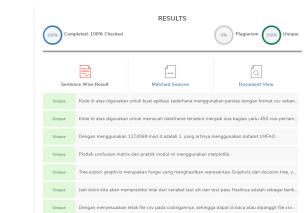
```
FileNotFoundException [Errno 2] File b'F://Semester 6/Artificial Intelligence/Tugas 4/src/fannyc.csv' does not exist: b'F://Semester 6/Artificial Intelligence/Tugas 4/src/fannyc.csv'
```

## 2. Cara Penangan Error

- **SyntaxError**

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

#### 4.3.4 Bukti Tidak Plagiat



**Gambar 4.32** Bukti Tidak Melakukan Plagiat Chapter 4

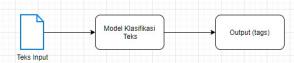
### 4.3.5 Link Youtube

## 4.4 1174066 - D.Irga B. Naufal Fakhri

### 4.4.1 Teori

#### 4.4.1.1 Klasifikasi Teks

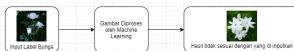
Klasifikasi teks merupakan sebuah model yang biasa digunakan untuk mengkategorikan sebuah teks ke dalam kelompok-kelompok yang lebih terorganisir. Jadi untuk setiap kalimat yang di masukan ke dalam mesin, mesin tersebut akan menjadikan setiap kata dari kalimat tersebut menjadi sebuah kolom. Untuk ilustrasinya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



**Gambar 4.33** Klasifikasi Teks.

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

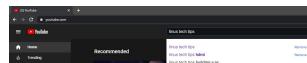
Karena machine learning tidak dapat menampilkan inputan sesuai dengan apa yang kita inputkan. Karena inputan tersebut serupa namun mesin memberikan output yang berbeda, biasanya output atau error ini disebut dengan istilah noise. Untuk contoh sederhananya misalkan kita inputkan salah satu label yang 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 4.34** Klasifikasi Bunga.

#### 4.4.1.3 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:



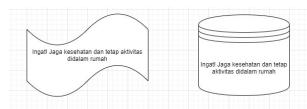
**Gambar 4.35** Klasifikasi Teks pada Youtube.

#### 4.4.1.4 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.

#### 4.4.1.5 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 4.36** Bag of Words.

#### 4.4.1.6 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 4.37** TF-IDF.

#### 4.4.2 Praktek Program

##### 4.4.2.1 Nomor 1

```

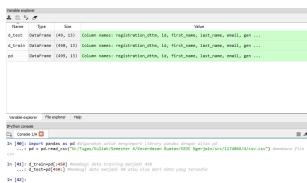
1 import pandas as pd #digunakan untuk mengimport library pandas
    dengan alias pd
2 pd = pd.read_csv("N:/Tugas/Kuliah/Semester 6/Kecerdasan Buatan/
    KB3C Ngerjain/src/1174066/4/csv.csv") #membaca file csv
  
```

```
In [4]: import pandas as pd #digunakan untuk mengimport library pandas dengan alias pd
```

**Gambar 4.38** Nomor 1

#### 4.4.2.2 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 4.39** Nomor 2

#### 4.4.2.3 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 Ngerjain/src/1174066/4/Youtube04-Eminem.csv") #Membaca file csv
3
4 from sklearn.feature_extraction.text import CountVectorizer # import fungsi countvectorize dari sklearn
5 vectorizer = CountVectorizer() #membuat instansi CountVectorizer
6
7 dvec = vectorizer.fit_transform(d[ 'CONTENT' ]) #Memasukkan data ke dvec
8 dvec #Melihat data yang dimasukkan ke dvec
9
10 daptarkata = vectorizer.get_feature_names() #Mendapatkan data dan memasukkannya ke daptarkata
11
12 dshuf = d.sample(frac=1) #Memasukkan sample kedalam variable dshuf
13
14 d_train = dshuf[:300] #Membuat data training
15 d_test = dshuf[300:] #Membuat data test
16
17 d_train_att = vectorizer.fit_transform(d_train[ 'CONTENT' ]) # Memasukkan data training dari vectorizer
18 d_train_att #Melihat data training
19
20 d_test_att = vectorizer.transform(d_test[ 'CONTENT' ]) #Memasukkan data test dari vectorizer
21 d_test_att #Melihat data training
```

```
22  
23 d_train_label = d_train[ 'CLASS' ] #Memberi label  
24 d_test_label = d_test[ 'CLASS' ] #Memberi Label
```

**Gambar 4.40** Nomor 3

#### 4.4.2.4 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 training
4 clfsvm.score(d_test_att, d_test_label) #Memunculkan clf sebagai
   testing yang sudah di training tadi
```

```
3n [4]: from sklearn import svm
...>   File "/usr/local/lib/python3.6/dist-packages/sklearn/base.py":139: FutureWarning: The default value of gamma will change
...>   from 'scale' to 'auto' or 'scale' to
...>   avoid this warning.
...>   warnings.warn("The default value of gamma will change
...>   from 'scale' to 'auto' or 'scale' to
...>   avoid this warning.", FutureWarning)
...>   gamma = _check_param(gamma, "gamma", "scale", "auto", "float")
...>   if gamma == "scale":
```

**Gambar 4.41** Nomor 4

#### 4.4.2.5 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 4.42** Nomor 5

#### 4.4.2.6 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,
9                           normalize=False,
10                           title='Confusion matrix',
11                           cmap=plt.cm.Blues):
12     """
13     This function prints and plots the confusion matrix.
14     Normalization can be applied by setting 'normalize=True'.
15     """
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
18         print("Normalized confusion matrix")
19     else:
20         print('Confusion matrix, without normalization')
21
22     print(cm)
23
24     plt.imshow(cm, interpolation='nearest', cmap=cmap)
25     plt.title(title)
26     #plt.colorbar()
27     tick_marks = np.arange(len(classes))
28     plt.xticks(tick_marks, classes, rotation=90)
29     plt.yticks(tick_marks, classes)
30
31     fmt = '.2f' if normalize else 'd'
32     thresh = cm.max() / 2.
33     #for i, j in itertools.product(range(cm.shape[0]), range(cm.
34     #shape[1])):
35     #    plt.text(j, i, format(cm[i, j], fmt),
36     #              horizontalalignment="center",
37     #              color="white" if cm[i, j] > thresh else "black"
38     #)
39
40     plt.tight_layout()
41     plt.ylabel('True label')
42     plt.xlabel('Predicted label')
43 types = pd.read_csv("N:/zz/KB3A-master/src/1174006/chapter4/
44   classes.txt", sep='\s+', header=None, usecols=[1], names=['type
45   '])
46 types = types['type']
47 types
48
49 import numpy as np
50 np.set_printoptions(precision=2)
51 plt.figure(figsize=(4,4), dpi=100)
52 plot_confusion_matrix(cmtree, classes=types, normalize=True)
53 plt.show()
```

**Gambar 4.43** Nomor 6

#### 4.4.2.7 Nomor 7

```

1 from sklearn.model_selection import cross_val_score
2
3 scorestree = cross_val_score(clftree, d_train_att, d_train_label,
4                             cv=5)
5 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
6                                         scorestree.std() * 2))
7
8 scoressvm = cross_val_score(clfsvm, d_train_att, d_train_label,
9                             cv=5)
10 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(),
11                                         scoressvm.std() * 2))
12
13 scores = cross_val_score(clf, d_train_att, d_train_label, cv=5)
14 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()
15                                         () * 2))

```

**Gambar 4.44** Nomor 7

#### 4.4.2.8 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(
4     n_estimators_opts),4), float)
5 i = 0
6 for max_features in max_features_opts:
7     for n_estimators in n_estimators_opts:
8         clf = RandomForestClassifier(max_features=max_features,
9             n_estimators=n_estimators)
10        scores = cross_val_score(clf, d_train_att, d_train_label,
11            cv=5)
12        rf_params[i,0] = max_features
```

```
10     rf_params[i,1] = n_estimators  
11     rf_params[i,2] = scores.mean()  
12     rf_params[i,3] = scores.std() * 2  
13     i += 1  
14     print("Max features: %d, num estimators: %d, accuracy:  
%0.2f (+/- %0.2f)" % (max_features, n_estimators, scores.mean()  
(), scores.std() * 2))
```

**Gambar 4.45** Nomor 8

#### 4.4.3 Penanganan Error

## 1. FileNotFoundError

```
| File "pandas\_libs\parsers.pyx", line 689, in
| pandas._libs.parsers.TextReader._setup_parser_source
|
| FileNotFoundError: [Errno 2] File 'B\N\Tugas\a\Semester 6\Kecerdasan Buatan\KB3C
| Ngerjan\src\11740664\csv.csv' does not exist: 'B\N\Tugas\a\Semester 6\Kecerdasan
| Buatan\KB3C\Ngerjan\src\11740664\csv.csv'
```

**Gambar 4.46** FileNotFoundErrors

## 2. Cara Penangan Error

- FileNotFoundException

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

#### 4.4.4 Bukti Tidak Plagiat



**Gambar 4.47** Bukti Tidak Melakukan Plagiat Chapter 4

#### 4.4.5 Link Youtube

<https://youtu.be/Lw0r-UAb8jY>

## 4.5 1174083 - Bakti Qilan Mufid

### Chapter-4 Klasifikasi Teks

#### 4.5.1 Teori

##### 4.5.1.1 Jelaskan apa itu klasifikasi teks, sertakan gambar iustrasi buatan sendiri

Klasifikasi teks merupakan salah satu tugas terpenting dalam Pemrosesan Bahasa Alami (Natural Language Processing). Ini adalah proses mengklasifikasikan string teks atau dokumen ke dalam kategori yang berbeda, tergantung pada konten string. Klasifikasi teks memiliki berbagai aplikasi, seperti mendeteksi sentimen pengguna dari tweet, mengklasifikasikan email sebagai spam atau ham, mengklasifikasikan posting blog ke dalam kategori yang berbeda, penandaan otomatis permintaan pelanggan, dan sebagainya. Berikut adalah contoh dari Klasifikasi Teks. Contohnya, misal kita ingin mencari kata dog, is, table, on, the . kemudian jika kata yang dimaksud sesuai maka akan menampilkan bilangan biner 1 dan jika salah 0. Seperti dibawah ini :

the dog is on the table



Gambar 4.48 Klasifikasi Teks

##### 4.5.1.2 Jelaskan mengapa klasifikasi bunga tidak bisa digunakan untuk machine learning, sertakan ilustrasi gambar sendiri

Dikarenakan tidak semua bunga memiliki ciri - ciri yang sama. Atau dalam kata lain terdapat data noise dalam klasifikasi bunga sehingga tidak bisa menggunakan machine learning. Contohnya Anggrek memiliki warna ungu, dengan jumlah kelopak 5. Kemudian ada bunga warna ungu dengan jumlah kelopak yang sama namun ternyata bukan anggrek dan kategorinya banyak sekali. Bahkan ada bunga yang tidak jelas apakah warnanya sesuai atau tidak, sehingga bisa menyebabkan data noise.



**Gambar 4.49** Klasifikasi Bunga

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

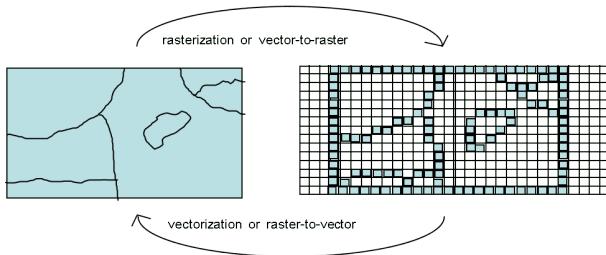
Menggunakan teknik bag-of-words pada klasifikasi berbasis text dan kata untuk mengklasifikasikan komentar yang ada di internet sebagai spam atau bukan. Misalkan pada kolom komentar dapat di cek seberapa sering suatu kata muncul dalam kalimat. Setiap kata dapat dijadikan baris dan kolomnya ini merupakan kategori kata terbut, apakah masuk kedalam spam atau tidak. dan contoh lainnya yaitu pada Caption. dimana akan muncul subtitle secara otomatis dari youtube menggunakan sensor suara yang disesuaikan dengan kata yang telah ditentukan. Contohnya seperti berikut :

CONTENT	CLASS
Huh, anyway check out this you[tube] channel: kobyoshi02	1
Hey guys check out my new channel and our first vid THIS IS US THE MONKEYS!!! I'm the monkey in the white shirt,please leave a like comment and please subscribe!!!!	1
just for test I have to say murdev.com	1
me shaking my sexy ass on my channel enjoy ^_^	1
watch?v=vtaRGgvGtW0 Check this out .	1
Hey, check out my new website!! This site is about kids stuff. kidsmediausa . com	1
Subscribe to my channel	1
i turned it on mute as soon as i came on i just wanted to check the views...	0
You should check my channel for Funny VIDEOS!!	1
and u shoud d check my channel and tell me what I should do next!	1

**Gambar 4.50** Klasifikasi Spam Comment di Youtube

**4.5.1.4** *Jelaskan apa yang dimaksud vektorisasi data.*

Vektorisasi adalah proses konversi data raster(gambar, pindaian) menjadi data vektor yang lebih umum disebut dengan istilah digitalisasi adapun aktifitasnya disebut digitasi. Wujud digitalisasi ini diklasifikasikan secara spesifik dalam tema-tema tertentu yang direpresentasikan oleh bentuk garis, poligon dan titik. Pada akhirnya proses vektorisasi ini menghasilkan suatu wujud topografi yang menggambarkan keadaan permukaan bumi atau bentang alam. Sifat data yang geometris menunjukkan ukuran dimensi yang sesungguhnya.



**Gambar 4.51** Vektorisasi dan Rasterisasi

**4.5.1.5** *Jelaskan apa itu bag of words dengan kata-kata yang sederhana dan ilustrasi sendiri.*

bag-of-words merupakan representasi teks yang menggambarkan kemunculan kata-kata dalam dokumen. Pengelompokan kata kata kedalam perhitungan, berapakah sebuah kata muncul dalam satu kalimat. Disebut "bag" kata-kata, karena informasi tentang susunan atau struktur kata dalam dokumen dibuang. Model ini hanya berkaitan dengan apakah kata-kata yang diketahui muncul dalam dokumen, bukan di mana dalam dokumen. Contohnya disini akan melihat kemunculan kata dari kalimat :

1. I Love Dogs
2. I hate dogs and knitting
3. Knitting is my hobby and passion.

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

**Gambar 4.52** Bag-Of-Words

**4.5.1.6** *Jelaskan apa itu TF-IDF, ilustrasikan dengan gambar sendiri.*

TF-IDF memberi kita frekuensi kata dalam setiap dokumen dalam korpus atau mengganti data jadi number. Ini adalah rasio berapa kali kata itu muncul dalam dokumen dibandingkan dengan jumlah total kata dalam dokumen itu. Itu meningkat seiring jumlah kemunculan kata itu di dalam dokumen meningkat. Setiap dokumen memiliki tf sendiri. Dalam ilustrasi disini saya akan mengganti contoh Bag of Words menjadi bentuk TF-IDF.

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	0.18	<b>0.48</b>	0.18							
Doc 2	0.18		0.18	<b>0.48</b>	0.18	0.18				
Doc 3					0.18	0.18	<b>0.48</b>	<b>0.95</b>	<b>0.48</b>	<b>0.48</b>

Gambar 4.53 Contoh TF-IDF

### 4.5.2 Praktek

4.5.2.1 buat aplikasi klasifikasi sederhana menggunakan pandas, buat data dummy format csv sebanyak 500 baris dan melakukan load ke dataframe panda.jelaskan arti setiap baris kode yang dibuat(harus beda dengan teman sekelas)

```
1 import pandas as pd #import package pandas, lalu dialiaskan
    menjadi pd.
2 marvel = pd.read_csv ('E:/backup/sem 6/Kecerdasan Buatan/KB3C –
    Copy/src/1174083/src4/Marvel.csv', sep=',') #membaca file csv
    dimana data pada file csv dipisahkan oleh koma, lalu
    ditampung di variable marvel.
```

Listing 4.1 kodingan praktek no. 1

Index	name	ID	ALIGN	EYE	HAIR
400	(Earth-616)	Identity	Characters	green eyes	black hair
487	Kenichio	Public	Neutral	Brown Eyes	Black Hair
488	Harada (Earth-616)	Identity	Characters		
489	Irene Adler (Earth-616)	Secret	Neutral	White Eyes	Silver Hair
490	Marrina (Earth-616)	Identity	Good	Black Eyes	Green Hair
491	Fred Davis Jr. (Earth-616)	Secret	Good	Blue Eyes	White Hair
492	Zelda DuBois (Earth-616)	Public	Bad	Green Eyes	Auburn Hair
493	Beyonder (Earth-616)	Identity	Neutral	Variable Eyes	Variable Hair
494	Roxanne Washington (Earth-616)	Secret	Characters	nan	nan
495	Bonita Juarez (Earth-616)	Identity	Good	Brown Eyes	Black Hair
496	Surtur (Earth-616)	No Dual Identity	Bad	Yellow Eyes	Red Hair
497	Stranger (Cosmic Being)	Secret	Neutral	White Eyes	White Hair
498	Brandy Clark (Earth-616)	Public	Good	nan	Brown Hair
499	Lillian Crawley (Earth-616)	Identity	Characters	Hazel Eyes	Brown Hair
	Ichabod (Earth-616)	Secret	Good	nan	White Hair

Gambar 4.54 hasil praktek soal no. 1

4.5.2.2 dari dataframe tersebut dipecah menjadi dua dataframe yaitu 450 row pertama dan 50 row sisanya(harus beda dengan teman sekelas)

```

1 marvel1 , marvel2 = marvel[:450] , marvel[450:] #membagi data
  menjadi dua bagian , variable marvel1 untuk menampung 450
  baris data pertama , variable marvel2 untuk menampung 50 baris
  data terakhir .

```

**Listing 4.2** kodingan praktek no. 2

Name	Type	Size	Value
marvel	DataFrame	(500, 7)	Column names: name, ID, ALIGN, EYE, HAIR, SEX, ALIVE
marvel1	DataFrame	(450, 7)	Column names: name, ID, ALIGN, EYE, HAIR, SEX, ALIVE
marvel2	DataFrame	(50, 7)	Column names: name, ID, ALIGN, EYE, HAIR, SEX, ALIVE

**Gambar 4.55** hasil praktek soal no. 2

4.5.2.3 praktekkan vektorisasi dan klasifikasi dari data(NPM mod 4, jika 0 maka ketty perry, 1 LMFAO, 2 Eminem, 3 Shakira) dengan Decission Tree. Tunjukkan keluaranya dari komputer sendiri dan artikan maksud luaran yang di dapatkan

```

1 print(1174083%4) #hasilnya 3 , maka Shakira

```

**Listing 4.3** 1174083 mod 4

```

1 # Vektorisasi Data
2 import pandas as pd
3 d = pd.read_csv("Youtube05-Shakira.csv")
4
5 from sklearn.feature_extraction.text import CountVectorizer
6 vectorizer = CountVectorizer()
7
8 dvec = vectorizer.fit_transform(d['CONTENT'])
9 dvec
10
11 daptarkata = vectorizer.get_feature_names()
12
13 dshuf = d.sample(frac=1)
14
15 d_train = dshuf[:300]
16 d_test = dshuf[300:]
17
18 d_train_att = vectorizer.fit_transform(d_train['CONTENT'])
19 d_train_att
20
21 d_test_att = vectorizer.transform(d_test['CONTENT'])
22 d_test_att
23
24 d_train_label = d_train['CLASS']
25 d_test_label = d_test['CLASS']

```

**Listing 4.4** kodingan praktek no. 3

```
In [7]: import pandas as pd
...: d = pd.read_csv("Youtube05-Shakira.csv")
...:
...: from sklearn.feature_extraction.text import CountVectorizer
...: vectorizer = CountVectorizer()
...:
...: dvec = vectorizer.fit_transform(d['CONTENT'])
...: dvec
...:
...: daptarkata = vectorizer.get_feature_names()
...:
...: 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
...:
...: d_test_att = vectorizer.transform(d_test['CONTENT'])
...: d_test_att
...:
...: d_train_label = d_train['CLASS']
...: d_test_label = d_test['CLASS']
```

**Gambar 4.56** hasil praktek soal no. 3(1)

Vektorisasi data content dari file Youtube04\_Shakira.CSV

Name	Type	Size	Value
d	DataFrame	(370, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS
d_test	DataFrame	(70, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS
d_test_label	Series	(70,)	Series object of pandas.core.series module
d_train	DataFrame	(300, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS
d_train_label	Series	(300,)	Series object of pandas.core.series module
daptarkata	list	1357	['00', '000', '0687119038', '08', '10', '100', '101721377578919894134' ...]
dshuf	DataFrame	(370, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS

**Gambar 4.57** hasil praktek soal no. 3(2)

**4.5.2.4 Cobalah klarifikasi dari data vektorisasi yang di tentukan di nomor sebelumnya dengan klasifikasi SVM. Tunjukkan keluaranya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan**

```
1 from sklearn import svm
2 clfsvm = svm.SVC()
3 clfsvm.fit(d_train_att, d_train_label)
4 clfsvm.score(d_test_att, d_test_label)
```

**Listing 4.5** kodingan praktek no. 4

```
In [9]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(d_train_att, d_train_label)
...: clfsvm.score(d_test_att, d_test_label)
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)
Out[9]: 0.6857142857142857
```

**Gambar 4.58** hasil praktek soal no. 4

**4.5.2.5 Cobalah klasifikasikan dari data vektorisasi yang ditentukan dari nomor sebelumnya dengan klasifikasi Decision Tree. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan**

```
1 from sklearn import tree
2 clftree = tree.DecisionTreeClassifier()
3 clftree.fit(d_train_att, d_train_label)
4 clftree.score(d_test_att, d_test_label)
```

**Listing 4.6** kodingan praktek no. 5

```
In [8]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(d_train_att, d_train_label)
...: clftree.score(d_test_att, d_test_label)
Out[8]: 0.9285714285714286
```

**Gambar 4.59** hasil praktek soal no. 5

**4.5.2.6 Plotlah confusion matrix dari praktek modul ini menggunakan matplotlib. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.**

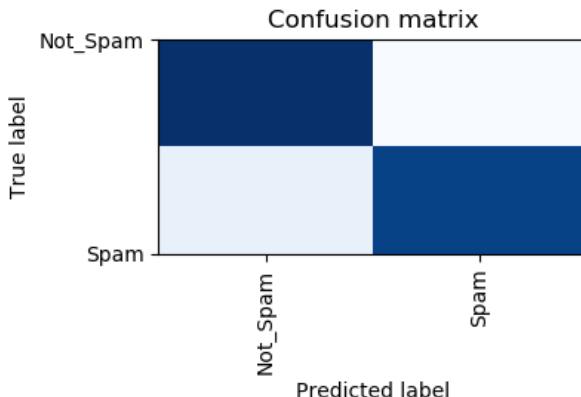
```
1 from sklearn.metrics import confusion_matrix
2 pred_labeltree = clftree.predict(d_test_att)
3 cmmtree = confusion_matrix(d_test_label, pred_labeltree)
4 cmmtree
5
6 import matplotlib.pyplot as plt
7 import itertools
8 def plot_confusion_matrix(cm, classes,
9                         normalize=False,
10                         title='Confusion matrix',
11                         cmap=plt.cm.Blues):
12     """
13     This function prints and plots the confusion matrix.
14     Normalization can be applied by setting `normalize=True`.
15     """
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
18         print("Normalized confusion matrix")
19     else:
```

```

20     print('Confusion matrix, without normalization')
21
22     print(cm)
23
24     plt.imshow(cm, interpolation='nearest', cmap=cmap)
25     plt.title(title)
26     #plt.colorbar()
27     tick_marks = np.arange(len(classes))
28     plt.xticks(tick_marks, classes, rotation=90)
29     plt.yticks(tick_marks, classes)
30
31     fmt = '.2f' if normalize else 'd'
32     thresh = cm.max() / 2.
33
34     plt.tight_layout()
35     plt.ylabel('True label')
36     plt.xlabel('Predicted label')
37
38 types = pd.read_csv("classes.txt", sep='\s+', header=None, usecols=[1], names=['type'])
39 types = types['type']
40 types
41
42 import numpy as np
43 np.set_printoptions(precision=2)
44 plt.figure(figsize=(4,4), dpi=100)
45 plot_confusion_matrix(cmtree, classes=types, normalize=True)
46 plt.show()

```

**Listing 4.7** kodingan praktek no. 6(1)



**Gambar 4.60** hasil praktek soal no. 6(1)

Plot confusion matrix dari klasifikasi Decission Tree.

```

1 from sklearn.metrics import confusion_matrix
2 pred_labelssvm = clfsvm.predict(d_test_att)

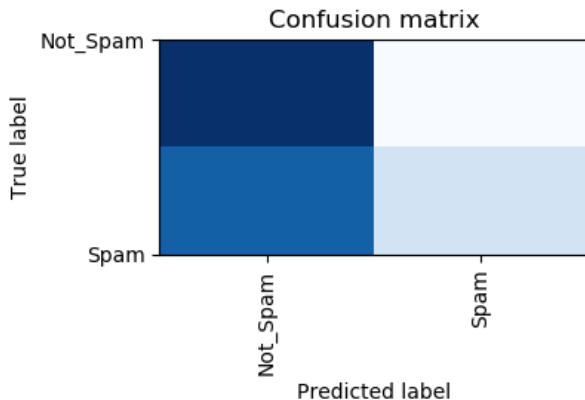
```

```

3 cmsvm = confusion_matrix(d_test_label , pred_labelssvm)
4 cmsvm
5
6 import matplotlib.pyplot as plt
7 import itertools
8 def plot_confusion_matrix(cm, classes,
9                           normalize=False,
10                           title='Confusion matrix',
11                           cmap=plt.cm.Blues):
12     """
13     This function prints and plots the confusion matrix.
14     Normalization can be applied by setting 'normalize=True'.
15     """
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
18         print("Normalized confusion matrix")
19     else:
20         print('Confusion matrix, without normalization')
21
22     print(cm)
23
24     plt.imshow(cm, interpolation='nearest', cmap=cmap)
25     plt.title(title)
26     #plt.colorbar()
27     tick_marks = np.arange(len(classes))
28     plt.xticks(tick_marks, classes, rotation=90)
29     plt.yticks(tick_marks, classes)
30
31     fmt = '.2f' if normalize else 'd'
32     thresh = cm.max() / 2.
33     #for i, j in itertools.product(range(cm.shape[0]), range(cm.
34     #shape[1])):
35     #    plt.text(j, i, format(cm[i, j], fmt),
36     #              horizontalalignment="center",
37     #              color="white" if cm[i, j] > thresh else "black"
38     )
39
40     plt.tight_layout()
41     plt.ylabel('True label')
42     plt.xlabel('Predicted label')
43
44 types = pd.read_csv("classes.txt",sep='\s+', header=None,usecols
45 = [1], names=['type'])
46 types = types['type']
47 types
48
49 import numpy as np
50 np.set_printoptions(precision=2)
51 plt.figure(figsize=(4,4), dpi=100)
52 plot_confusion_matrix(cmsvm, classes=types, normalize=True)
53 plt.show()

```

**Listing 4.8** kodingan praktik no. 6(2)



**Gambar 4.61** hasil praktek soal no. 6(2)

Plot confusion matrix dari klasifikasi SVM.

```

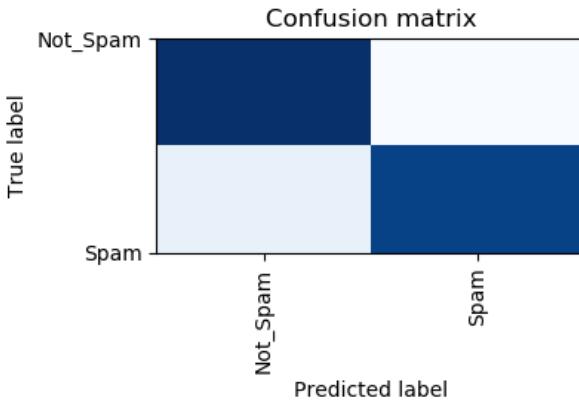
1 from sklearn.metrics import confusion_matrix
2 pred_labels = clf.predict(d_test_att)
3 cm = confusion_matrix(d_test_label, pred_labels)
4
5 import matplotlib.pyplot as plt
6 import itertools
7 def plot_confusion_matrix(cm, classes,
8                           normalize=False,
9                           title='Confusion matrix',
10                          cmap=plt.cm.Blues):
11     """
12     This function prints and plots the confusion matrix.
13     Normalization can be applied by setting 'normalize=True'.
14     """
15     if normalize:
16         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
17         print("Normalized confusion matrix")
18     else:
19         print('Confusion matrix, without normalization')
20
21     print(cm)
22
23     plt.imshow(cm, interpolation='nearest', cmap=cmap)
24     plt.title(title)
25     #plt.colorbar()
26     tick_marks = np.arange(len(classes))
27     plt.xticks(tick_marks, classes, rotation=90)
28     plt.yticks(tick_marks, classes)
29
30     fmt = '.2f' if normalize else 'd'
31     thresh = cm.max() / 2.
32     #for i, j in itertools.product(range(cm.shape[0]), range(cm.
33     #shape[1])):
34     #    plt.text(j, i, format(cm[i, j], fmt),
35

```

```

34     #           horizontalalignment="center",
35     #           color="white" if cm[i, j] > thresh else "black
36     ")
37     plt.tight_layout()
38     plt.ylabel('True label')
39     plt.xlabel('Predicted label')
40
41 types = pd.read_csv("classes.txt", sep='\s+', header=None, usecols
42     =[1], names=['type'])
43 types = types['type']
44 types
45 import numpy as np
46 np.set_printoptions(precision=2)
47 plt.figure(figsize=(4,4), dpi=100)
48 plot_confusion_matrix(cmtree, classes=types, normalize=True)
49 plt.show()

```

**Listing 4.9** kodingan praktek no. 6(3)**Gambar 4.62** hasil praktek soal no. 6(3)

Plot confusion matrix dari klasifikasi Random Forest.

*4.5.2.7 jalankan program cross validaiton pada bagian teori bab ini. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.*

```

1 from sklearn.model_selection import cross_val_score
2
3 scorestree = cross_val_score(clftree, d_train_att, d_train_label,
4     cv=5)
4 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
5     scorestree.std() * 2))

```

```

5
6 scoreSVM = cross_val_score(clfSVM, d_train_att, d_train_label,
7     cv=5)
7 print("Accuracy: %0.2f (+/- %0.2f)" % (scoreSVM.mean(),
8     scoreSVM.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()
10     () * 2))

```

**Listing 4.10** kodingan praktek no. 7

**Accuracy: 0.92 (+/- 0.05)**  
**Accuracy: 0.66 (+/- 0.05)**  
**Accuracy: 0.94 (+/- 0.03)**

**Gambar 4.63** hasil praktek soal no. 7

**4.5.2.8** Buatlah program pengamatan komponen informasi pada bagian teori bab ini. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

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(
4     n_estimators_opts),4), float)
4 i = 0
5 for max_features in max_features_opts:
6     for n_estimators in n_estimators_opts:
7         clf = RandomForestClassifier(max_features=max_features,
8             n_estimators=n_estimators)
9         scores = cross_val_score(clf, d_train_att, d_train_label,
10             cv=5)
11         rf_params[i,0] = max_features
12         rf_params[i,1] = n_estimators
13         rf_params[i,2] = scores.mean()
14         rf_params[i,3] = scores.std() * 2
15         i += 1
16     print("Max features: %d, num estimators: %d, accuracy:
16         %0.2f (+/- %0.2f)" % (max_features,
16             n_estimators, scores.mean(), scores.std() * 2))

```

**Listing 4.11** kodingan praktek no. 8

```
Max features: 5, num estimators: 10, accuracy: 0.89 (+/- 0.09)
Max features: 5, num estimators: 30, accuracy: 0.90 (+/- 0.07)
Max features: 5, num estimators: 50, accuracy: 0.91 (+/- 0.08)
Max features: 5, num estimators: 70, accuracy: 0.92 (+/- 0.07)
Max features: 5, num estimators: 90, accuracy: 0.93 (+/- 0.07)
Max features: 5, num estimators: 110, accuracy: 0.93 (+/- 0.07)
Max features: 5, num estimators: 130, accuracy: 0.94 (+/- 0.04)
Max features: 5, num estimators: 150, accuracy: 0.93 (+/- 0.05)
Max features: 5, num estimators: 170, accuracy: 0.93 (+/- 0.05)
Max features: 5, num estimators: 190, accuracy: 0.92 (+/- 0.06)
Max features: 10, num estimators: 10, accuracy: 0.92 (+/- 0.07)
Max features: 10, num estimators: 30, accuracy: 0.93 (+/- 0.04)
Max features: 10, num estimators: 50, accuracy: 0.94 (+/- 0.03)
Max features: 10, num estimators: 70, accuracy: 0.93 (+/- 0.05)
Max features: 10, num estimators: 90, accuracy: 0.93 (+/- 0.04)
Max features: 10, num estimators: 110, accuracy: 0.92 (+/- 0.06)
Max features: 10, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 10, num estimators: 150, accuracy: 0.94 (+/- 0.03)
Max features: 10, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 10, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 15, num estimators: 10, accuracy: 0.90 (+/- 0.05)
Max features: 15, num estimators: 30, accuracy: 0.93 (+/- 0.04)
Max features: 15, num estimators: 50, accuracy: 0.93 (+/- 0.05)
Max features: 15, num estimators: 70, accuracy: 0.92 (+/- 0.05)
Max features: 15, num estimators: 90, accuracy: 0.94 (+/- 0.04)
Max features: 15, num estimators: 110, accuracy: 0.94 (+/- 0.03)
Max features: 15, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 15, num estimators: 150, accuracy: 0.94 (+/- 0.03)
Max features: 15, num estimators: 170, accuracy: 0.94 (+/- 0.03)
Max features: 15, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 20, num estimators: 10, accuracy: 0.91 (+/- 0.05)
Max features: 20, num estimators: 30, accuracy: 0.91 (+/- 0.06)
Max features: 20, num estimators: 50, accuracy: 0.94 (+/- 0.04)
Max features: 20, num estimators: 70, accuracy: 0.92 (+/- 0.04)
Max features: 20, num estimators: 90, accuracy: 0.93 (+/- 0.04)
Max features: 20, num estimators: 110, accuracy: 0.93 (+/- 0.04)
Max features: 20, num estimators: 130, accuracy: 0.94 (+/- 0.03)
Max features: 20, num estimators: 150, accuracy: 0.93 (+/- 0.04)
Max features: 20, num estimators: 170, accuracy: 0.93 (+/- 0.04)
```

**Gambar 4.64** hasil praktek soal no. 8(1)

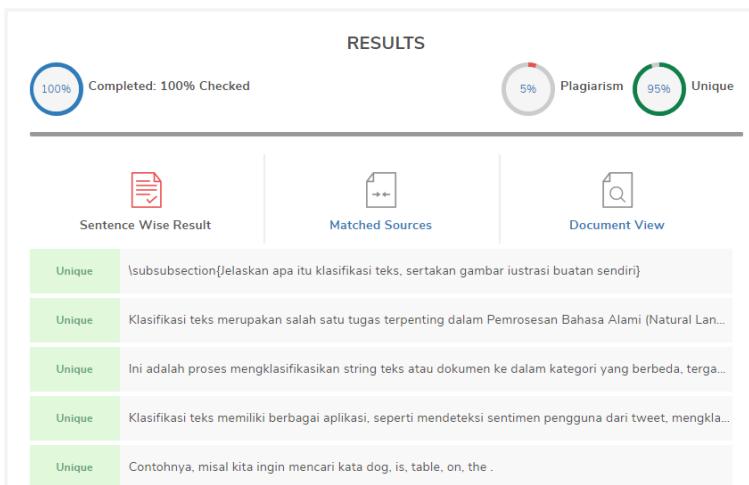
```
Max features: 25, num estimators: 150, accuracy: 0.93 (+/- 0.04)
Max features: 25, num estimators: 170, accuracy: 0.94 (+/- 0.03)
Max features: 25, num estimators: 190, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 10, accuracy: 0.93 (+/- 0.06)
Max features: 30, num estimators: 30, accuracy: 0.93 (+/- 0.05)
Max features: 30, num estimators: 50, accuracy: 0.95 (+/- 0.04)
Max features: 30, num estimators: 70, accuracy: 0.93 (+/- 0.04)
Max features: 30, num estimators: 90, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 110, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 130, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 150, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 30, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 10, accuracy: 0.91 (+/- 0.05)
Max features: 35, num estimators: 30, accuracy: 0.94 (+/- 0.05)
Max features: 35, num estimators: 50, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 70, accuracy: 0.94 (+/- 0.03)
Max features: 35, num estimators: 90, accuracy: 0.94 (+/- 0.03)
Max features: 35, num estimators: 110, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 150, accuracy: 0.94 (+/- 0.03)
Max features: 35, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 40, num estimators: 10, accuracy: 0.91 (+/- 0.06)
Max features: 40, num estimators: 30, accuracy: 0.93 (+/- 0.04)
Max features: 40, num estimators: 50, accuracy: 0.94 (+/- 0.03)
Max features: 40, num estimators: 70, accuracy: 0.94 (+/- 0.03)
Max features: 40, num estimators: 90, accuracy: 0.94 (+/- 0.03)
Max features: 40, num estimators: 110, accuracy: 0.93 (+/- 0.05)
Max features: 40, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 40, num estimators: 150, accuracy: 0.93 (+/- 0.05)
Max features: 40, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 40, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 10, accuracy: 0.91 (+/- 0.06)
Max features: 45, num estimators: 30, accuracy: 0.94 (+/- 0.04)
Max features: 45, num estimators: 50, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 70, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 90, accuracy: 0.94 (+/- 0.03)
Max features: 45, num estimators: 110, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 150, accuracy: 0.93 (+/- 0.05)
Max features: 45, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 190, accuracy: 0.93 (+/- 0.04)
```

**Gambar 4.65** hasil praktek soal no. 8(2)

#### 4.5.3 Penanganan Error

pada chapter 4 ini saya tidak menemukan error.

#### 4.5.4 Bukti Tidak Plagiat



**Gambar 4.66** Bukti tidak plagiat

#### 4.5.5 Link Youtube

<https://youtu.be/la8Jptbm3Mo>

### 4.6 Muhammad Reza Syachrani - 1174084

#### 4.6.1 Teori

1. Jelaskan apa itu klasifikasi teks, dan gambar ilustrasi  
klasifikasi teks adalah cara untuk mengklasifikasikan atau mengelompokkan teks berdasarkan parameter tertentu baik itu jenis teks atau jenis dari dokumen yang terdapat kumpulan teks didalamnya.



**Gambar 4.67** contoh klasifikasi teks

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

karna melakukan klasifikasi pada bunga tidak dapat menggunakan mesin learning karena terdapat banyak jenis-jenis bunga yang mirip hingga ada yang sama persis tetapi tidak sama. oleh karena itu klasifikasi bunga tida bisa dilakukan oleh mesin learning dikarenakan jika inputan ciri-ciri tidak sesuai maka bunga di inputkan kemungkinan jawaban dari mesin learning itu tidak tepat contoh menginputkan ciri-ciri bunga serupa tetapi pada mesin learning menjawab itu merupakan ciri-ciri bunga teratai.



**Gambar 4.68** contoh klasifikasi bunga

3. Jelaskan bagaimana teknik pembelajaran mesin pada teks pada kata-kata yang digunakan di youtube.

cara pembelajaran teks yang digunakan pada youtube yaitu dengan cara menyimpan inputan teks pada menu pencarian youtube. sehingga pada saat kita akan mencari data yang serupa maka youtube menyediakan rekomendasi-rekomendasi dari pencaharian. contoh pada menu pencarian kita menulis kata sa maka akan muncul banyak rekomendasi yang serupa yang sering di cari oleh orang dalam jangka waktu tertentu.

sa

satu hati sampai mati  
samudra cinta  
sara wijayanto rumah ruben onsu  
sara wijaya  
samudra cinta 22 maret 2020  
saaih halilintar  
sambel trasie happy asmara  
samudra cinta 21 maret 2020  
sapi  
sampek tuwek lirik  
satu nama tetap dihati  
sarah sheila kamalia  
sakit pinggang  
sakit dalam bercinta ipank lirik

Laporan prediksi penulisiran

**Gambar 4.69** contoh teknik pembelajaran mesin

4. Jelaskan apa yang dimaksud vektorisasi data

vektorisasi data merupakan pembagian data menjadi bagian-bagian yang lebih sederhana, contoh pada satu paragraf terdiri dari 200 kata kemudian dilakukan vektorisasi dengan cara membagi-bagi kata dalam paragraf tersebut ke dalam kalimat-kalimat yang terpisah kemudian dipecah lagi menjadi data dalam perkata selanjutnya kata-kata tersebut dijemahkan.

5. Jelaskan apa itu bag of words dan ilustrasi

bag of words merupakan representasi teks yang menggambarkan kemunculan kata-kata dalam dokumen. ePngelompokan kata-kata kedalam perhitungan, berapakah sebuah kata muncul dalam satu kalimat. Disebut "tas" kata-kata, karena informasi tentang susunan atau struktur kata dalam dokumen dibuang. Model ini hanya berkaitan dengan apakah kata-kata yang diketahui muncul dalam dokumen, bukan di mana dalam dokumen. Contohnya disini akan melihat kemunculan kata dari kalimat :

- Ariq suka menonton film. Alvan juga suka film.

- Alvan juga suka menonton anime.

	Arig	suka	menonton	film	Alvan	juga	anime
Doc1	1	2	1	2	1	1	
Doc2		1	1		1	1	1

**Gambar 4.70** contoh bag of words

#### 6. Jelaskan apa itu TF-IDF.

TF-IDF memberi frekuensi kata dalam setiap dokumen dalam mengganti data jadi number. Ini adalah rasio berapa kali kata itu muncul dalam dokumen dibandingkan dengan jumlah total kata dalam dokumen. Itu meningkat sejalan dengan jumlah kemunculan kata itu di dalam dokumen meningkat. Setiap dokumen memiliki tf sendiri. rumus TF-IDF :

$$W_{t,d} = TF_{t,d} * IDF_t \quad (1)$$

Keterangan :

$W_{t,d}$  = bobot dari  $t$  (*term*) dalam satu dokumen

$TF_{t,d}$  = frekuensi kemunculan  $t$  (*term*) dalam dokumen  $d$

$IDF_t$  = *Inverse document frequency*, dimana

$$IDF_t = \log\left(\frac{N}{n_t}\right) \quad (2)$$

Keterangan :

$N$  = jumlah semua dokumen

$n_t$  = jumlah dokumen yang mengandung *term t*

**Gambar 4.71** rumus Tf-IDF

#### 4.6.2 Praktek

##### 1. No. 1

```

1 # In [1]:
2 import pandas as pd #mengimport librari padas
3 us = pd.read_csv("D:/Git Kecerdasan Buatan/KB3C/src
        /1174084/4/us-500.csv") # variabel us untuk membaca file
        csv menggunakan fungsi read csv dari padas
4 print(len(us)) #melihat jumlah dari baris data yang telah di
        import
5 print(us.head()) #melihat lima baris pertama data yang telah
        di import
6 print(us.shape) #mengetahui banyak baris dan kolom dari data

```

```

500
   first_name ... web
0    James ... http://www.bentonjohnbjr.com
1 Josephine ... http://www.chanayjeffreyaesq.com
2      Art ... http://www.chemeljameslcpa.com
3     Lenna ... http://www.feltzprintingservice.com
4   Donette ... http://www.printingdimensions.com

[5 rows x 12 columns]
(500, 12)

```

**Gambar 4.72** aplikasi sederhana pandas

## 2. No. 2

```

1 # In [5]:
2 data_training = us[:450] #membuat data training sebanyak 450
  baris
3 data_testing = us[450:] #membuat data testing dari hasil
  pengurangan 500-450

```

data_testing	DataFrame	(50, 12)	Column names: first_name, last_name, company_name, address, city, coun ...
data_training	DataFrame	(450, 12)	Column names: first_name, last_name, company_name, address, city, coun ...

**Gambar 4.73** pecahan dataframe

## 3. No. 3

```

1 # In [1]:
2 import pandas as pd #Mengimport pandas
3 d=pd.read_csv("D:/Git Kecerdasan Buatan/KB3C/src/1174084/4/
  Youtube02-KatyPerry.csv") #Membuat variable d untuk
  membaca file csv dari dataset
4 # In [2]:
5 spam=d.query('CLASS == 1') #mengelompokkan komentar spam
6 nospam=d.query('CLASS == 0') #mengelompokkan komentar bukan
  spam
7 # In [3]: memanggil lib vektorisasi
8 #melakukan fungsi bag of word dengan cara menghitung semua
  kata
9 #yang terdapat dalam file
10 from sklearn.feature_extraction.text import CountVectorizer
11 vectorizer = CountVectorizer()
12 # In [3]:
13 dvec = vectorizer.fit_transform(d['CONTENT']) #melakukan bag
  of word pada dataframe pada colom CONTENT
14 # In [4]:
15 dvec #melihat isi vektorisasi
16 # In [5]:
17 print(d['CONTENT'][342]) #melihat isi data pada baris ke 342
18 # In [6]:

```

```
19 daptarkata=vectorizer.get_feature_names() #feature_names  
    merupakan digunakan untuk mengambil nama kolomnya ada apa  
    saja  
20 # In [7]:  
21 dshuf = d.sample(frac=1) #melakukan randomisasi pada datanya  
    supaya sempurna saat melakukan klasifikasi  
22 # In [8]:  
23 dk_train=dshuf[:300] #Data akan dibagi dari 300 row akhir  
    menjadi data training dan sisanya adalah data testing  
24 dk_test=dshuf[300:] #Data akan dibagi dari 300 row pertama  
    menjadi data training dan sisanya adalah data testing  
25 # In [9]:  
26 dk_train_att=vectorizer.fit_transform(dk_train['CONTENT']) #  
    melakukan training pada data training dan di vektorisasi  
27 print(dk_train_att)  
28 # In [10]:  
29 dk_test_att=vectorizer.transform(dk_test['CONTENT']) #  
    melakukan testing pada data testing dan di vektorisasi  
30 print(dk_test_att)  
31 # In [11]:  
32 dk_train_label=dk_train['CLASS'] #mengambil label spam dan  
    bukan spam  
33 print(dk_train_label)  
34 dk_test_label=dk_test['CLASS'] #mengambil label spam dan  
    bukan spam  
35 print(dk_test_label)
```

(0, 336)	1
(0, 242)	1
(0, 1561)	1
(0, 733)	1
(0, 1029)	1
(0, 1178)	1
(0, 1078)	1
(0, 811)	1
(1, 495)	1
(1, 1318)	1
(1, 816)	1
(1, 74)	1
(1, 479)	1
(1, 111)	1
(1, 113)	1
(1, 1256)	1
(1, 425)	1
(1, 1285)	1
(1, 180)	6
(1, 741)	1
(1, 399)	2
(1, 241)	2

Gambar 4.74 Vektorisai Dan Klasifikasi

## 4. No. 4

```

1 # In[12]:
2 from sklearn import svm #Mengimport svm
3 clfsvm = svm.SVC() #clfsvm sebagai variabel untuk mengatur
4         fungsi SVC
5 clfsvm.fit(dk_train_att, dk_train_label) #Mengatur data
6         training
7 clfsvm.predict(dk_test_att)
8 clfsvm.score(dk_test_att, dk_test_label) #Mengatur data
9         testing

```

```

In [62]: clfsvm.predict(dk_test_att)
Out[62]:
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0], dtype=int64)

In [63]: clfsvm.score(dk_test_att, dk_test_label) #Mengatur data testing
Out[63]: 0.5

```

**Gambar 4.75** Klasifikasi SVM

## 5. No. 5

```

1 # In[13]:
2 from sklearn import tree #Mengimport tree
3 clftree = tree.DecisionTreeClassifier() #clftree sebagai
4         variabel untuk decision tree
5 clftree.fit(dk_train_att, dk_train_label) #Mengatur data
6         training
7 # In[14]:
8 clftree.predict(dk_test_att)
9 # In[15]:
10 clftree.score(dk_test_att, dk_test_label) #Mengatur data
11         testing

```

```

In [60]: clftree.predict(dk_test_att)
Out[60]:
array([0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 1], dtype=int64)

In [61]: clftree.score(dk_test_att, dk_test_label) #Mengatur data testing
Out[61]: 0.92

```

**Gambar 4.76** Klasifikasi Desicon Tree

## 6. No. 6

```

1 # In[16]:
2 from sklearn.metrics import confusion_matrix #Mengimport
3         Confusion Matrix

```

```

3 pred_labels = clf.predict(dk_test_att) #Membuat variable
    pred_labels dari data testing
4 cm = confusion_matrix(dk_test_label, pred_labels) #cm sebagai
    variabel data label
5 cm

```

**Out[65]:**

```
array([[24,  0],
       [ 3, 23]], dtype=int64)
```

Gambar 4.77 confusion matrix

7. No. 7

```

1 # In[17]:
2 from sklearn.model_selection import cross_val_score #
    Mengimport cross_val_score
3 scores = cross_val_score(clf,dk_train_att,dk_train_label,cv
    =5) #Membuat variable scores sebagai variabel prediksi
        dari data training
4 scorerata2=scores.mean()
5 scorersd=scores.std()
6 # In[18]
7 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
    std() * 2)) #menampilkan data scores dengan ketentuan
        akurasi

```

```

In [66]: from sklearn.model_selection import cross_val_score #Mengimport
cross_val_score
...: scores = cross_val_score(clf,dk_train_att,dk_train_label,cv=5) #Membuat
variable scores sebagai variabel prediksi dari data training
...: scorerata2=scores.mean()
...: scorersd=scores.std()

In [67]: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
#menampilkan data scores dengan ketentuan akurasi
Accuracy: 0.95 (+/- 0.06)

```

Gambar 4.78 cross validaiton

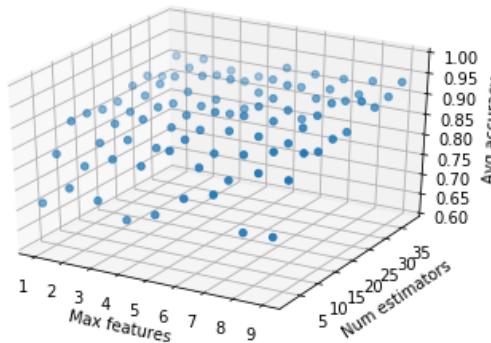
8. No. 8

```

1
2 # In[19]:
3 import numpy as np
4 max_features_opts = range(1, 10, 1) #Variable
    max_features_opts sebagai variabel untuk membuat range 1,
        10, 1

```

```
5 n_estimators_opts = range(2, 40, 4) #
  Variable n_estimators_opts sebagai variabel untuk membuat
  range 2, 40, 4
6 rf_params = np.empty((len(max_features_opts)*len(
  n_estimators_opts),4) , float) #Variable rf_params sebagai
  variabel untuk menjumlahkan yang sudah di tentukan
  sebelumnya
7 i = 0
8 for max_features in max_features_opts: #Perulangan
9   for n_estimators in n_estimators_opts: #Perulangan
10     clf = RandomForestClassifier(max_features=
11       max_features, n_estimators=n_estimators) #Menampilkan
12       variabel csf
13       scores = cross_val_score(clf , dk_train_att ,
14       dk_train_label , cv=5) #Variable scores sebagai variabel
15       training
16       rf_params[i,0] = max_features #index 0
17       rf_params[i,1] = n_estimators #index 1
18       rf_params[i,2] = scores.mean() #index 2
19       rf_params[i,3] = scores.std() * 2 #index 3
20       i += 1 #Dengan ketentuan i += 1
21       print("Max features: %d, num estimators: %d, accuracy
22 : %0.2f (+/- %0.2f)" %
23 (% (max_features , n_estimators , scores.mean() , scores.std() *
24 2))) #Print hasil pengulangan yang sudah ditentukan
25
26 # In [20]:
27 import matplotlib.pyplot as plt #Mengimport library
28   matplotlib sebagai plt
29 from mpl_toolkits.mplot3d import Axes3D #Mengimport axes3D
30   untuk menampilkan plot 3 dimensi
31 from matplotlib import cm #Memanggil data cm yang sudah
32 tersedia
33 fig = plt.figure() #Menghasilkan plot sebagai figure
34 fig.clf() #Figure di ambil dari clf
35 ax = fig.gca(projection='3d') #ax sebagai projection 3d
36 x = rf_params[:,0] #x sebagai index 0
37 y = rf_params[:,1] #y sebagai index 1
38 z = rf_params[:,2] #z sebagai index 2
39 ax.scatter(x, y, z) #Membuat plot scatter x y z
40 ax.set_zlim(0.6, 1) #Set zlim dengan ketentuan yang ada
41 ax.set_xlabel('Max features') #Memberikan nama label x
42 ax.set_ylabel('Num estimators') #Memberikan nama label y
43 ax.set_zlabel('Avg accuracy') #Memberikan nama label z
```



**Gambar 4.79** pengamatan komponen informasi

#### 4.6.3 Penanganan Error

##### 1. Error

```
FileNotFoundException: [Errno 2] File b'D:/Git Kecerdasan Buatan/KB3C/src/11174084/4/Youtube02-KatyPerry.csv' does not exist: b'D:/Git Kecerdasan Buatan/KB3C/src/11174084/4/Youtube02-KatyPerry.csv'
```

**Gambar 4.80** File Not Found Error

##### 2. Tuliskan Kode Error dan Jenis Error

- FileNotFoundException

##### 3. Cara Penanganan Error

- FileNotFoundException

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.

#### 4.6.4 Bukti Tidak Plagiat



**Gambar 4.81** plagiarism

#### 4.6.5 Link Video Youtube

[https://youtu.be/v63ayQTw\\_MI](https://youtu.be/v63ayQTw_MI)

### 4.7 1174077 - Alvan Alvanzah

#### 4.7.1 Teori

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

Klasifikasi teks merupakan sebuah model yang biasa digunakan untuk 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 4.82** Klasifikasi teks.

2. Jelaskan mengapa hal ini bisa terjadi, klasifikasi bunga tidak bisa digunakan untuk machine learning, sertakan ilustrasi gambar sendiri.

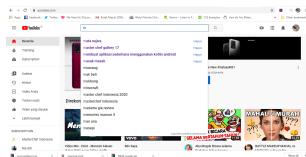
Karena machine learning tidak dapat menampilkan inputan sesuai dengan apa yang kita inputkan. Karena inputan tersebut serupa namun mesin memberikan output yang berbeda, biasanya output atau error ini disebut dengan istilah noise. Untuk contoh sederhananya misalkan kita inputkan salah satu label yang 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 4.83** Klasifikasi Bunga.

3. Jelaskan bagaimana yang dimaksud dengan teknik pembelajaran mesin pada teks yang digunakan dan sertakan ilustrasi buatan sendiri. 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 :



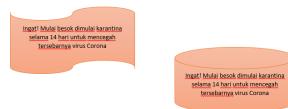
**Gambar 4.84** Klasifikasi teks Youtube.

4. Jelaskan apa yang dimaksud 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.

5. Jelaskan apa yang dimaksud dengan bag of words dengan ilustrasi sendiri.

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 4.85** Bag of words.

6. Jelaskan apa yang dimaksud dengan 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 :

	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 4.86** TF-IDF.

#### 4.7.2 Praktek Program

### 1. Soal 1

```
1 %% Soal 1
2 import pandas as pd #digunakan untuk mengimport library
3 pd = pd.read_csv("C:/Users/ASUS/Videos/zuplur/src/1174077/4/
4 alvan.csv") #membaca file csv
```

Kode di atas digunakan untuk buat aplikasi sederhana menggunakan pandas dengan format csv sebanyak 500 baris, hasilnya ialah sebagai berikut :

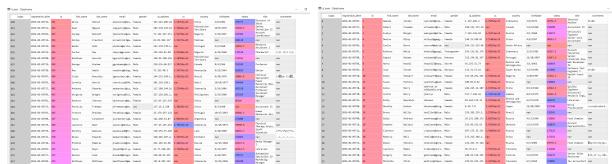


**Gambar 4.87** Hasil Soal 1.

## 2. Soal 2

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

Kode di atas digunakan untuk memecah dataframe tersebut menjadi dua bagian yaitu 450 row pertama dan 50 row kedua. Hasilnya adalah sebagai berikut :



Gambar 4.88 Hasil Soal 2.

## 3. Soal 3

```

1 %% Soal 3
2 import pandas as ps #untuk import library pandas berguna
    untuk mengelola dataframe
3 ps = ps.read_csv("C:/Users/ASUS/Videos/zuplur/src/1174077/4/
    Youtube03-LMFAO.csv") #membaca file dengan format csv
4
5 spam=ps.query( 'CLASS == 1') #membagi tabel spam
6 nospam=ps.query( 'CLASS == 0')#membagi tabel no spam
7
8 from sklearn.feature_extraction.text import CountVectorizer #
    untuk import countvectorizer berfungsi untuk memecah data
    tersebut menjadi sebuah kata yang lebih sederhana
9 vectorizer = CountVectorizer () #ntuk menjalankan fungsi
    tersebut, pada code ini tidak ada hasilnya dikarenakan
    spyder tidak mendukung hasil dari instasiasi.
10
11 dvec = vectorizer.fit_transform(ps[ 'CONTENT']) #untuk
    melakukan pemecahan data pada dataframe yang terdapat pada
    kolom konten
12 dvec #Untuk menampilkan hasil dari code sebelumnya
13
14 Daptarkata= vectorizer.get_feature_names()
15
16 dshuf = ps.sample(frac=1)
17
18 d_train=dshuf[:300]
19 d_test=dshuf[300:]
20
21 d_train_att = vectorizer.fit_transform(d_train[ 'CONTENT'])
22 d_train_att
23
24 d_train_label=d_train[ 'CLASS']
25 d_test_label=d_test[ 'CLASS']
```

Dengan menggunakan  $1174077 \bmod 4$  adalah 1, yang artinya menggunakan dataset LMFAO. Hasilnya adalah sebagai berikut :

**Gambar 4.89** Hasil Soal 3.

#### 4. Soal 4

```
1 %% Soal 4
2
3 from sklearn import svm
4 clfsvm = svm.SVR(gamma = 'auto')
5 clfsvm.fit(d_train_att, d_train_label)
```

Klasifikasi dari data vektorisasi menggunakan klasifikasi Decision Tree. Hasilnya adalah sebagai berikut :

```
In [7]: from sklearn import svm
...: clfsvm = svm.SVR(gamma = 'auto')
...: clfsvm.fit(d_train_attr, d_train_label)
Out[7]:
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
gamma='auto',
kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
verbose=False)
```

**Gambar 4.90** Hasil Soal 4.

### 5. Soal 5

```
1 #%%soal 5
2
3 from sklearn import tree
4 clftree = tree.DecisionTreeClassifier()
5 clftree.fit(d_train_att, d_train_label)
```

Plotlah confusion matrix dari praktik modul ini menggunakan matplotlib. Hasilnya adalah sebagai berikut :

```
In [8]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(d_train_att, d_train_label)
Out[8]:
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
criterion='gini',
max_depth=None, max_features=None,
max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0,
presort='deprecated', random_state=None, splitter='best')
```

**Gambar 4.91** Hasil Soal 5.

## 6. Soal 6

```
1 #%%soal 6/
2
3 from sklearn.metrics import confusion_matrix
4 pred_labels=clftree.predict(d_test)
5 cm=confusion_matrix(d_test_label ,pred_labels)
6
7 #%%
8
9 import matplotlib.pyplot as plt
10
11 def plot_confusion_matrix(cm, classes,
12                           normalize=False,
13                           title='Confusion matrix',
14                           cmap=plt.cm.Blues):
15
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.
newaxis]
18         print("Normalized confusion matrix")
19     else:
20         print('Confusion matrix, without normalization')
21
22     print(cm)
```

Menjalankan program cross validation. Hasilnya adalah sebagai berikut :

```
In [11]: import matplotlib.pyplot as plt
...:
...: def plot_confusion_matrix(cm, classes,
...:                         normalize=False,
...:                         title='Confusion matrix',
...:                         cmap=plt.cm.Blues):
...:
...:     if normalize:
...:         cm = cm.astype('float') / cm.sum(axis=1)[:, np.
newaxis]
...:         print("Normalized confusion matrix")
...:     else:
...:         print('Confusion matrix, without normalization')
...:
...:     print(cm)
...:     cm
```

**Gambar 4.92** Hasil Soal 6.

## 7. Soal 7

```
1 #%%soal 7
2
```

```

3 from sklearn.model_selection import cross_val_score
4
5 scores=cross_val_score(clftree,d_train_att,d_train_label, cv
6 =5)
7
8 skor_rata2=scores.mean()
9 skoresd=scores.std()

```

Tree.export graphviz merupakan fungsi yang menghasilkan representasi Graphviz dari decision tree, yang kemudian ditulis ke outfile. Disini akan menyimpan classifiernya, akan meng ekspor file student performance jika salah akan mengembalikan nilai fail. Hasilnya adalah sebagai berikut :

```

In [12]: from sklearn.model_selection import cross_val_score
...:
scores=cross_val_score(clftree,d_train_att,d_train_label, cv=5)
...:
skor_rata2=scores.mean()
...:
skoresd=scores.std()

```

**Gambar 4.93** Hasil Soal 7.

## 8. Soal 8

```

1 %%soal 8 /
2
3 max_features_opts = range(5, 50, 5) #max_features_opts
4 sebagai variabel untuk membuat range 5,50,5
5 n_estimators_opts = range(10, 200, 20) #n_estimators_opts
6 sebagai variabel untuk membuat range 10,200,20
7 rf_params = ps.empty((len(max_features_opts)*len(
8 n_estimators_opts),4), float) #rf_params sebagai variabel
9 untuk menjumlahkan yang sudah di tentukan sebelumnya
10 i = 0
11 for max_features in max_features_opts: #pengulangan
12     for n_estimators in n_estimators_opts: #pengulangan
13         clftree = RandomForestClassifier(max_features=
14             max_features, n_estimators=n_estimators) #menampilkan
15         variabel csf
16         scores = cross_val_score(clf, df_train_att,
17             df_train_label, cv=5) #scores sebagai variabel training
18         rf_params[i,0] = max_features #index 0
19         rf_params[i,1] = n_estimators #index 1
20         rf_params[i,2] = scores.mean() #index 2
21         rf_params[i,3] = scores.std() * 2 #index 3
22         i += 1 #dengan ketentuan i += 1
23         print("Max features: %d, num estimators: %d, accuracy
24 : %0.2f (+/- %0.2f)" %(max_features, n_estimators, scores.
25 mean(), scores.std() * 2))
26         #print hasil pengulangan yang sudah ditentukan

```

Buatlah program pengamatan komponen informasi. Jadi disini kita akan memprediksi nilai dari variabel test att dan test pass Hasilnya adalah sebagai berikut :

**Gambar 4.94** Hasil Soal 8.

### 4.7.3 Penanganan Error

## 1. ScreenShoot Error

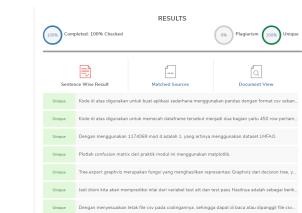
```
[FileNotFoundException: [Error 2] File 'b'C:/Users/KSUS/videose/raplur/src/1174877/4/alva.csv' does  
not exist: b'C:/Users/KSUS/videose/raplur/src/1174877/4/alva.csv'
```

## 2. Cara Penangan Error

- **SyntaxError**

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

#### **4.7.4 Bukti Tidak Plagiat**



**Gambar 4.96** Bukti Tidak Melakukan Plagiat Chapter 4

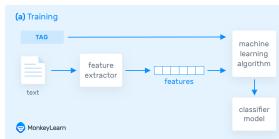
#### 4.7.5 Link Youtube

### 4.8 1174079 -Chandra Kirana Poetra

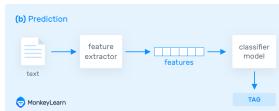
#### 4.8.1 Teori

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

Klasifikasi teks merupakan suatu proses dalam memberikan tag atau kategori pada suatu teks berdasarkan konten. Klasifikasi teks merupakan salah satu bagian dari natural language processing yang mencakup analisis, labeling, deteksi spam dan intent detection. Untuk ilustrasinya bisa dilihat pada gambar berikut :



**Gambar 4.97** Klasifikasi teks.



**Gambar 4.98** Klasifikasi teks.

2. Jelaskan mengapa hal ini bisa terjadi, klasifikasi bunga tidak bisa digunakan untuk machine learning, sertakan ilustrasi gambar sendiri. Karena meskipun bunga yang kita coba untuk klasifikasikan memiliki spesies yang sama, kebanyakan bunga sendiri memiliki bentuk ukuran yang berbeda beda sehingga akan membuat machine learning sulit diterapkan karena ukuran tidak bisa dijadikan sebagai salah satu parameter dalam machine learning.



**Gambar 4.99** Ukuran Bunga yang berbeda-beda.

3. Jelaskan bagaimana yang dimaksud dengan teknik pembelajaran mesin pada teks yang digunakan dan sertakan ilustrasi buatan sendiri.  
Youtube menggunakan istilah yang disebut sebagai keyword yang nantinya kita isi didalam form pencarian yang ada kemudian akan muncul video yang terkait



**Gambar 4.100** Klasifikasi teks Youtube.

4. Jelaskan apa yang dimaksud vektorisasi data.  
vektorisasi merupakan suatu proses untuk mengkonversi suatu algoritma yang beroperasi pada satu value dalam satu waktu menjadi kebeberapa value dalam satu waktu
5. Jelaskan apa yang dimaksud dengan bag of words dengan ilustrasi sendiri.

Merupakan model untuk menyederhanakan data dalam bidang natural language processing dan juga pengambilan informasi. di model ini text direpresentasikan sebagai suatu kantong kata, mengabaikan grammer dan susunan kata tapi menyimpan value pengulangan apabila ada kata yang diulang. model ini digunakan pada bidang komputer



**Gambar 4.101** Bag of words.

6. Jelaskan apa yang dimaksud dengan TF-IDF.

TF-IDF merupakan singkatan dari term frequency inverse document frequency yang merupakan statistik angka yang memiliki tujuan untuk memperlihatkan seberapa pentingnya suatu kata dalam suatu dokumen.



Gambar 4.102 TF-IDF.

## 4.8.2 Praktek Program

### 4.8.2.1 Nomor 1

```
1 %% Soal 1
2 import pandas as pd #digunakan untuk mengimport library pandas
   dengan alias pd
3 pd = pd.read_csv("F:/ Poltekpos/D4 TI 3C/Semester 6/Kecerdasan
   Buatan/Github/Upload 30 Maret 2020 github tugas 4/src
   /1174079/4/csv_chandra.csv") #membaca file csv
```

```
In [32]: import pandas as pd #digunakan untuk mengimport
library pandas dengan alias pd
... pd = pd.read_csv("F:/Poltekpos/D4 TI 3C/Semester 6/
Kecerdasan Buatan/Github/Upload 30 Maret 2020 github tugas 4/
src/1174079/4/csv_chandra.csv") #membaca file csv
```

Gambar 4.103 Nomor 1

### 4.8.2.2 Nomor 2

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

```
In [33]: d_train=pd[:450] #membagi data training menjadi 450
... d_test=pd[450:] #membagi data menjadi 50 atau sisa
dari data yang tersedia
```

Gambar 4.104 Nomor 2

### 4.8.2.3 Nomor 3

```
1 %% Soal 3
2 import pandas as pd #digunakan untuk mengimport library pandas
   dengan alias pd
```

```

3 d = pd.read_csv("F:/ Poltekpos/D4 TI 3C/Semester 6/Kecerdasan
    Buatan/Github/Upload 30 Maret 2020 github tugas 4/src
    /1174079/4/Youtube03-LMFAO.csv") #Membaca file csv
4
5 from sklearn.feature_extraction.text import CountVectorizer #
    import fungsi_countvectorize dari sklearn
6 vectorizer = CountVectorizer() #membuat instansi CountVectorizer
7
8 dvec = vectorizer.fit_transform(d['CONTENT']) #Memasukkan data ke
    dvec
9 dvec #Melihat data yang dimasukkan ke dvec
10
11 daptarkata = vectorizer.get_feature_names() #Mendapatkan data dan
    memasukkannya ke daptarkata
12
13 dshuf = d.sample(frac=1) #Memasukkan sample kedalam variable
    dshuf
14
15 d_train = dshuf[:300] #Membuat data training
16 d_test = dshuf[300:] #Membuat data test
17
18 d_train_att = vectorizer.fit_transform(d_train['CONTENT']) #
    Memasukkan data training dari vectorizer
19 d_train_att #Melihat data training
20
21 d_test_att = vectorizer.transform(d_test['CONTENT']) #Memasukkan
    data test dari vectorizer
22 d_test_att #Melihat data training
23
24 d_train_label = d_train['CLASS'] #Memberi label
25 d_test_label = d_test['CLASS'] #Memberi Label

```

```

rc(1174079/4/Youtube03-LMFAO.csv") #Membaca file csv
...
...     from sklearn.feature_extraction.text import
        CountVectorizer #import fungsi countvectorize dari sklearn
...     vectorizer = CountVectorizer() #membuat instansi
        CountVectorizer
...
...     dvec = vectorizer.fit_transform(d['CONTENT'])
        Memasukkan data ke dvec
...
...     dvec #Melihat data yang dimasukkan ke dvec
...
...     daptarkata = vectorizer.get_feature_names()
        Mendapatkan data dan memasukkannya ke daptarkata
...
...     dshuf = d.sample(frac=1) #Memasukkan sample kedalam
        variable dshuf
...
...     d_train = dshuf[:300] #Membuat data training
...     d_test = dshuf[300:] #Membuat data test
...
...     d_train_att =
        vectorizer.fit_transform(d_train['CONTENT']) #Memasukkan
        training dari vectorizer
...     d_train_att #Melihat data training
...
...     d_test_att = vectorizer.transform(d_test['CONTENT'])
        Memasukkan data test dari vectorizer
...     d_test_att #Melihat data training
...
...     d_train_label = d_train['CLASS'] #Memberi label
        d_train_label = d_train['CLASS'] #Memberi label

```

Gambar 4.105 Nomor 3

#### 4.8.2.4 Nomor 4

```

1 # SVM
2 from sklearn import svm #Mengimport svm dari sklearn
3 clfsvm = svm.SVC() #Membuat svc kedalam variable svm

```

```

4 clfsvm.fit(d_train_att, d_train_label) #Memprediksi data dari
    data training
5 clfsvm.score(d_test_att, d_test_label) #Memunculkan clf sebagai
    testing yang sudah di training tadi

```

```

In [38]: clfsvm.fit(d_train_att, d_train_label)
          C:\Users\acer\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
              unbalanced classes; specify gamma explicitly to 'auto' or 'scale'
              to avoid this warning.
              "avoid this warning.", FutureWarning)
Out[38]: 0.797104492753623

```

Gambar 4.106 Nomor 4

#### 4.8.2.5 Nomor 5

```

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

```

```

In [49]: from sklearn import tree #Mengimport tree dari
          sklearn
          ...
          clftree = tree.DecisionTreeClassifier() #Membuat
          decision tree
          ...
          clftree.fit(d_train_att, d_train_label) #Memprediksi
          data dari data training
          ...
          clftree.score(d_test_att, d_test_label) #Memunculkan
          clf sebagai testing yang sudah di training tadi
Out[49]: 0.9492753623188406

```

Gambar 4.107 Nomor 5

#### 4.8.2.6 Nomor 6

```

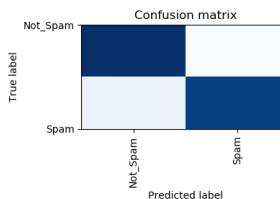
1 # Confusion Matrix – Decission Tree
2 from sklearn.metrics import confusion_matrix
3 pred_labeltree = clftree.predict(d_test_att)
4 cmtree = confusion_matrix(d_test_label, pred_labeltree)
5 cmtree
6
7 import matplotlib.pyplot as plt
8 import itertools
9 def plot_confusion_matrix(cm, classes,
                           normalize=False,
                           title='Confusion matrix',
                           cmap=plt.cm.Blues):
10
11     """
12     This function prints and plots the confusion matrix.
13     Normalization can be applied by setting 'normalize=True'.
14     """
15
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
18         print("Normalized confusion matrix")
19
20     else:

```

```

21     print('Confusion matrix, without normalization')
22
23     print(cm)
24
25     plt.imshow(cm, interpolation='nearest', cmap=cmap)
26     plt.title(title)
27     #plt.colorbar()
28     tick_marks = np.arange(len(classes))
29     plt.xticks(tick_marks, classes, rotation=90)
30     plt.yticks(tick_marks, classes)
31
32     fmt = '.2f' if normalize else 'd'
33     thresh = cm.max() / 2.
34     #for i, j in itertools.product(range(cm.shape[0]), range(cm.
35     #shape[1])):
36     #    plt.text(j, i, format(cm[i, j], fmt),
37     #              horizontalalignment="center",
38     #              color="white" if cm[i, j] > thresh else "black"
39     #)
40
41     plt.tight_layout()
42     plt.ylabel('True label')
43     plt.xlabel('Predicted label')
44
45 types = pd.read_csv("F:/Poltekpos/D4 TI 3C/Semester 6/Kecerdasan
46 Buatan/Github/Upload 30 Maret 2020 github tugas 4/src
47 /1174079/4/classes.txt", sep='|', header=None, usecols=[1],
48 names=['type'])
49 types = types['type']
50 types
51
52 import numpy as np
53 np.set_printoptions(precision=2)
54 plt.figure(figsize=(4,4), dpi=100)
55 plot_confusion_matrix(cmtree, classes=types, normalize=True)
56 plt.show()

```



**Gambar 4.108** Nomor 6

#### 4.8.2.7 Nomor 7

```

1 # Cross Validation
2 from sklearn.model_selection import cross_val_score
3

```

```

4 scorestree = cross_val_score(clftree, d_train_att, d_train_label,
5     cv=5)
6 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
7     scorestree.std() * 2))
8
9 scoressvm = cross_val_score(clfsvm, d_train_att, d_train_label,
10    cv=5)
11 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean(),
12     scoressvm.std() * 2))
13
14 scores = cross_val_score(clf, d_train_att, d_train_label, cv=5)
15 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()
16     () * 2))

```

```
C:\Users\acer\OneDrive\Udemy\stephanie\sklearn\svmbase.py:103: 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: 0.95 (+/- 0.00)
```

**Gambar 4.109** Nomor 7

#### 4.8.2.8 Nomor 8

```

1 # Komponen Informasi
2 max_features_opts = range(5, 50, 5)
3 n_estimators_opts = range(10, 200, 20)
4 rf_params = np.empty((len(max_features_opts)*len(
5     n_estimators_opts),4), float)
6 i = 0
7 for max_features in max_features_opts:
8     for n_estimators in n_estimators_opts:
9         clf = RandomForestClassifier(max_features=max_features,
10             n_estimators=n_estimators)
11         scores = cross_val_score(clf, d_train_att, d_train_label,
12             cv=5)
13         rf_params[i,0] = max_features
14         rf_params[i,1] = n_estimators
15         rf_params[i,2] = scores.mean()
16         rf_params[i,3] = scores.std() * 2
17         i += 1
18         print("Max features: %d, num estimators: %d, accuracy:
19             %0.2f (+/- %0.2f)" % (max_features, n_estimators, scores.mean(
20                 ), scores.std() * 2))      #print hasil pengulangan yang
21         sudah ditentukan

```

```

usage: [options] [files...]
Max features: 5, num estimators: 10, accuracy: 0.89 (+/- 0.05)
Max features: 5, num estimators: 30, accuracy: 0.92 (+/- 0.07)
Max features: 5, num estimators: 50, accuracy: 0.94 (+/- 0.04)
Max features: 5, num estimators: 70, accuracy: 0.94 (+/- 0.06)
Max features: 5, num estimators: 90, accuracy: 0.95 (+/- 0.05)
Max features: 5, num estimators: 110, accuracy: 0.95 (+/- 0.04)
Max features: 5, num estimators: 130, accuracy: 0.95 (+/- 0.04)
Max features: 5, num estimators: 150, accuracy: 0.94 (+/- 0.08)
Max features: 5, num estimators: 170, accuracy: 0.93 (+/- 0.05)
Max features: 5, num estimators: 190, accuracy: 0.94 (+/- 0.07)
Max features: 10, num estimators: 10, accuracy: 0.92 (+/- 0.03)
Max features: 10, num estimators: 30, accuracy: 0.95 (+/- 0.04)
Max features: 10, num estimators: 50, accuracy: 0.93 (+/- 0.07)
Max features: 10, num estimators: 70, accuracy: 0.93 (+/- 0.07)
Max features: 10, num estimators: 90, accuracy: 0.94 (+/- 0.06)
Max features: 10, num estimators: 110, accuracy: 0.93 (+/- 0.06)

```

**Gambar 4.110** Nomor 8

#### 4.8.3 Penanganan Error

##### 1. ScreenShoot Error

```

FileNotFoundException: [Error 3] File b'1174087/4/github/04 TI 3C/Semester 6/kecerdasan Buata/Github/upload/30 Maret 2020 github tugas 4/src/1174079/4/csv_chandra.csv' does not exist: b'F:\Poltekpos\04 TI 3C\Semester 6\kecerdasan Buata/Github\Upload\30 Maret 2020 github tugas 4/src/1174079/4/csv_chandra.csv'

```

**Gambar 4.111** SyntaxError

##### 2. Cara Penangan Error

- **SyntaxError**

Memperbaiki typo dan mengecek nama file serta direktorinya

#### 4.8.4 Bukti Tidak Plagiat

**Gambar 4.112** Bukti Tidak Melakukan Plagiat Chapter 4

#### 4.8.5 Link Youtube

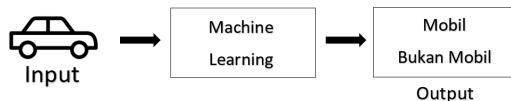
<https://youtu.be/8vkMpgUVTQs>

#### 4.9 1174087 - Ilham Muhammad Ariq

##### 4.9.1 Teori

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

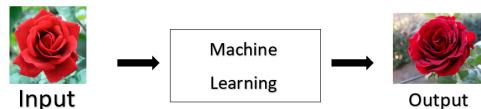
Klasifikasi teks merupakan sebuah model yang biasa digunakan untuk untuk mengkategorikan sebuah teks ke dalam kelompok-kelompok yang lebih terorganisir. Jadi untuk setiap kalimat yang di masukan ke dalam mesin, mesin tersebut akan menjadikan setiap kata dari kalimat tersebut menjadi sebuah kolom. Untuk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



**Gambar 4.113** Klasifikasi teks

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

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 bentuk dan ukurannya tidak sama. Untuk ilustrasinya bisa dilihat pada gambar berikut



**Gambar 4.114** Klasifikasi Bunga

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

Teknik machine learning yang dipakai pada teks yang digunakan di youtube bisa menggunakan bag of words dan random forest. Bag of words adalah proses mengubah teks menjadi vektor dengan panjang tetap dengan cara menghitung berapa kali setiap kata itu muncul. Random forest (RF) adalah suatu algoritma yang digunakan pada klasifikasi data dalam jumlah yang besar. Klasifikasi random forest dilakukan melalui penggabungan pohon (tree) dengan melakukan training pada sampel data yang dimiliki.

Atribut yang ada pada file csv Youtube01-Psy diantaranya, COMMENT\_ID, AUTHOR, DATE, CONTENT, CLASS.

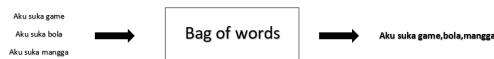
- COMMENT\_ID : merupakan key unik yang membedakan komen lainnya.
- AUTHOR : merupakan penulis dari komen tersebut.
- DATE : merupakan waktu dari komen tersebut dipublikasikan.
- CONTENT : merupakan isi komentarnya.
- CLASS : merupakan klasifikasi dari komennya (spam/notspam).

4. Jelaskan apa yang dimaksud 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.

5. Jelaskan apa yang dimaksud dengan bag of words dengan ilustrasi sendiri.

Bag of words adalah 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 4.115** Bag of words

6. Jelaskan apa yang dimaksud dengan 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

sentence 1	earth is the third planet from the sun		TF*IDF (sentence 1)	TF*IDF (Sentence 2)
sentence 2	Jupiter is the largest planet			
Word	TF (Sentence 1)	TF (Sentence 2)	TF*IDF (sentence 1)	TF*IDF (Sentence 2)
earth	1/8	0 log(2/1)=0	0.0375	0
is	1/8	1/5 log(2/2)=0	0	0
the	2/8	1/5 log(2/2)=0	0	0
third	1/8	0 log(2/1)=0.3	0.0375	0
planet	1/8	1/5 log(2/2)=0	0	0
from	0	0 log(2/1)=0.3	0	0
sun	1/8	0 log(2/1)=0.3	0.0375	0
largest	0	1/5 log(2/1)=0.3	0	0.06
Jupiter	0	1/5 log(2/1)=0.3	0	0.06

Gambar 4.116 TF-IDF

## 4.9.2 Praktek Program

### 1. Soal 1

```
1 import pandas as pd #digunakan untuk mengimport library
    pandas dengan alias pd
2 pd = pd.read_csv("E:/Kecerdasan Buatan/KB3C/src/1174087/4/
    csv_ariq.csv") #membaca file csv
```

Kode di atas digunakan untuk buat aplikasi sederhana menggunakan pandas dengan format csv sebanyak 500 baris, hasilnya ialah sebagai berikut

```
In [146]: import pandas as pd #digunakan untuk mengimport library pandas dengan
alias pd
...: pd = pd.read_csv("E:/Kecerdasan Buatan/KB3C/src/1174087/4/csv_ariq.csv")
membaca file csv
```

Gambar 4.117 Soal 1

### 2. Soal 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
```

Kode di atas digunakan untuk memecah dataframe tersebut menjadi dua bagian yaitu 450 row pertama dan 50 row kedua. Hasilnya adalah sebagai berikut

```
In [147]: d_train=pd[:450] membagi data training menjadi 450
...: d_test=pd[450:] membagi data menjadi 50 atau sisa dari data yang
tersedia
```

Gambar 4.118 Soal 2

### 3. Soal 3

```
1 import pandas as pd #untuk import library pandas berguna
    untuk mengelola dataframe
2 ariq = pd.read_csv("E:/Kecerdasan Buatan/KB3C/src/1174087/4/
    Youtube05-Shakira.csv") #membaca file dengan format csv
```

```

3
4 from sklearn.feature_extraction.text import CountVectorizer #
    untuk import countvectorizer berfungsi untuk memecah data
    tersebut menjadi sebuah kata yang lebih sederhana
5 vectorizer = CountVectorizer () #ntuk menjalankan fungsi
    tersebut, pada code ini tidak ada hasilnya dikarenakan
    spyder tidak mendukung hasil dari instasiasi.
6
7 dvec = vectorizer.fit_transform(ariq[ 'CONTENT' ]) #untuk
    melakukan pemecahan data pada dataframe yang terdapat pada
    kolom konten
8 dvec #Untuk menampilkan hasil dari code sebelumnya
9
10 Daptarkata= vectorizer.get_feature_names()
11
12 dshuf = ariq.sample(frac=1)
13
14 d_train=dshuf[:300]
15 d_test=dshuf[300:]
16
17 d_train_att = vectorizer.fit_transform(d_train[ 'CONTENT' ])
d_train_att
18
19 d_test_att = vectorizer.transform(d_test[ 'CONTENT' ])
d_test_att
20
21 d_train_label=d_train[ 'CLASS' ]
22 d_test_label=d_test[ 'CLASS' ]

```

Dengan menggunakan  $1174087 \bmod 4$  adalah 3, yang artinya menggunakan dataset shakira. Hasilnya adalah sebagai berikut

```

In [148]: Import pandas as pd #untuk import library pandas berguna untuk mengelola
          data
...: ariq = pd.read_csv("E:/Kecerdasan Buatan/X83C/src/1174087/4/youtube05-
Shakira.csv") #memuat file dengan format csv
...:
...: from sklearn.feature_extraction.text import CountVectorizer #untuk import
countvectorizer fungsi untuk memecah data tersebut menjadi sebuah kata yang
lebih sederhana
...: vectorizer = CountVectorizer () #ntuk menjalankan fungsi tersebut, pada
code ini tidak ada hasilnya dikarenakan spyder tidak mendukung hasil dari
instasiasi.
...:
...: dvec = vectorizer.fit_transform(ariq[ 'CONTENT' ]) #untuk melakukan
pemecahan data pada dataframe yang terdapat pada kolom konten
...: dvec #Untuk menampilkan hasil dari code sebelumnya
...:
...: Daptarkata= vectorizer.get_feature_names()
...:
...: dshuf = ariq.sample(frac=1)
...:
...: d_train=dshuf[:300]
...: d_test=dshuf[300:]
...:
...: d_train_att = vectorizer.fit_transform(d_train[ 'CONTENT' ])
...: d_train_att
...:
...: d_test_att = vectorizer.transform(d_test[ 'CONTENT' ])
...: d_test_att
...:
...: d_train_label=d_train[ 'CLASS' ]
...: d_test_label=d_test[ 'CLASS' ]

```

**Gambar 4.119 Soal 3**

#### 4. Soal 4

```

1 from sklearn import svm
2 clfsvm = svm.SVC()
3 clfsvm.fit(d_train_att , d_train_label)

```

Klasifikasi dari data vektorisasi menggunakan klasifikasi SVM. Hasilnya adalah sebagai berikut

```
In [150]: from sklearn import svm
.....
Out[150]:
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
kernel='rbf', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001, verbose=False)
```

**Gambar 4.120** Soal 4

## 5. Soal 5

```
1 from sklearn import tree
2 clftree = tree.DecisionTreeClassifier()
3 clftree.fit(d_train_att, d_train_label)
```

Klasifikasi dari data vektorisasi menggunakan klasifikasi Decision Tree. Hasilnya adalah sebagai berikut

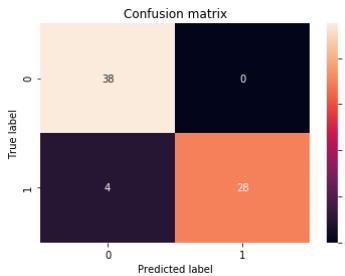
```
In [151]: from sklearn import tree
.....
Out[151]:
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, presort=False,
random_state=None, splitter='best')
```

**Gambar 4.121** Soal 5

## 6. Soal 6

```
1 from sklearn.ensemble import RandomForestClassifier
2 clf = RandomForestClassifier(n_estimators=80)
3 clf.fit(d_train_att, d_train_label)
4 clf.predict(d_test_att)
5
6 #%%soal 6
7
8 from sklearn.metrics import confusion_matrix
9 pred_labels=clf.predict(d_test_att)
10 cm=confusion_matrix(d_test_label, pred_labels)
11
12 import matplotlib.pyplot as plt
13 import seaborn as sns
14 sns.heatmap(cm, annot=True)
15 plt.title('Confusion matrix')
16 plt.ylabel('True label')
17 plt.xlabel('Predicted label')
18 plt.show()
```

Plot confusion matrix dari praktik modul ini menggunakan matplotlib. Hasilnya adalah sebagai berikut



Gambar 4.122 Soal 6

## 7. Soal 7

```

1 from sklearn.model_selection import cross_val_score
2 scores=cross_val_score(clf,d_train_att,d_train_label, cv=5)
3 skor_rata2=scores.mean()
4 skoresd=scores.std()
5 print("Accuracy: %0.2f (+/- %0.2f)" % (skor_rata2, skoresd * 2))

```

Menjalankan program cross validation. Hasilnya adalah sebagai berikut

```

In [154]: from sklearn.model_selection import cross_val_score
...: scores=cross_val_score(clf,d_train_att,d_train_label, cv=5)
...: skor_rata2=scores.mean()
...: skoresd=scores.std()
...: print("Accuracy: %0.2f (+/- %0.2f)" % (skor_rata2, skoresd * 2))
Accuracy: 0.95 (+/- 0.05)

```

Gambar 4.123 Soal 7

## 8. Soal 8

```

1 import numpy as np
2 max_features_opts = range(5, 50, 5) #max_features_opts
3 n_estimators_opts = range(10, 200, 20) #n_estimators_opts
4 rf_params = np.empty((len(max_features_opts)*len(
5     n_estimators_opts),4), float) #rf_params sebagai variabel
6     untuk menjumlahkan yang sudah di tentukan sebelumnya
7 i = 0
8 for max_features in max_features_opts: #pengulangan
9     for n_estimators in n_estimators_opts: #pengulangan
10         clf = RandomForestClassifier(max_features=
11             max_features, n_estimators=n_estimators) #menampilkan
12             variabel csf
13             scores = cross_val_score(clf, d_train_att,
14             d_train_label, cv=5) #scores sebagai variabel training
15             rf_params[i,0] = max_features #index 0
16             rf_params[i,1] = n_estimators #index 1
17             rf_params[i,2] = scores.mean() #index 2

```

```

13     rf_params[i,3] = scores.std() * 2 #index 3
14     i += 1 #dengan ketentuan i += 1
15     print("Max features: %d, num estimators: %d, accuracy
16       : %0.2f (+/- %0.2f)" %(max_features, n_estimators, scores.
mean(), scores.std() * 2))
#print hasil pengulangan yang sudah ditentukan

```

Program pengamatan komponen informasi. Jadi disini kita akan memprediksi nilai dari variabel test att dan test pass Hasilnya adalah sebagai berikut

```

Max features: 40, num estimators: 90, accuracy: 0.95 (+/- 0.05)
Max features: 40, num estimators: 110, accuracy: 0.94 (+/- 0.06)
Max features: 40, num estimators: 130, accuracy: 0.94 (+/- 0.06)
Max features: 40, num estimators: 150, accuracy: 0.94 (+/- 0.06)
Max features: 40, num estimators: 170, accuracy: 0.94 (+/- 0.06)
Max features: 40, num estimators: 190, accuracy: 0.94 (+/- 0.07)
Max features: 45, num estimators: 10, accuracy: 0.91 (+/- 0.08)
Max features: 45, num estimators: 30, accuracy: 0.92 (+/- 0.04)
Max features: 45, num estimators: 50, accuracy: 0.94 (+/- 0.07)
Max features: 45, num estimators: 70, accuracy: 0.93 (+/- 0.07)
Max features: 45, num estimators: 90, accuracy: 0.93 (+/- 0.06)
Max features: 45, num estimators: 110, accuracy: 0.94 (+/- 0.06)
Max features: 45, num estimators: 130, accuracy: 0.94 (+/- 0.05)
Max features: 45, num estimators: 150, accuracy: 0.94 (+/- 0.05)
Max features: 45, num estimators: 170, accuracy: 0.94 (+/- 0.06)
Max features: 45, num estimators: 190, accuracy: 0.94 (+/- 0.06)

```

**Gambar 4.124** Soal 8

### 4.9.3 Penanganan Error

#### 1. ScreenShoot Error

```
FileNotFoundException: [Errno 2] File b'csv_ariq.csv' does not exist: b'csv_ariq.csv'
```

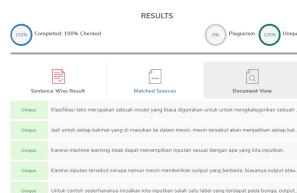
**Gambar 4.125** FileNotFoundError

#### 2. Cara Penangan Error

- **FileNotFoundException**

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

### 4.9.4 Bukti Tidak Plagiat



**Gambar 4.126** Bukti Tidak Melakukan Plagiat Chapter 4

#### 4.9.5 Link Youtube

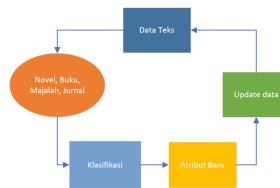
<https://youtu.be/lLxUddy2kW4>

#### 4.10 1174086 - Tia Nur candida

##### 4.10.1 Teori

###### 1. Klasifikasi Text

Klasifikasi teks merupakan cara dalam memilah milah data teks berdasarkan parameter tertentu dengan cara data yang bersifat dokumen ataupun teks yang memiliki kumpulan teks didalamnya, serta teks itu sendiri bertipe char atau string yang mudah diolah.



**Gambar 4.127** Klasifikasi Text

###### 2. Mengapa Klasifikasi Bunga tidak dapat menggunakan Machine Learning

Karena klasifikasinya menggunakan tipe data yang dimana attributnya memiliki nilai data berupa vektor dengan perbandingan masing-masing data yang dimiliki memiliki sedikit perbedaan, sehingga program atau sistem tidak dapat membedakan dengan tepat antara gambar 1 dan 2 dikarenakan memiliki perbedaan yang hampir tidak dapat dilihat dengan beberapa contoh gambar. untuk ilustrasi dapat dilihat pada gambar



**Gambar 4.128** Klasifikasi Bunga

3. Teknik Pembelajaran machine learning pada teks kata-kata di youtube  
Contohnya pada saat kita membuka sebuah video maka di sebelah kanan ada list "berikutnya", pada saat itu mesin melakukan ujicoba dan apabila anda menekan salah satu dari video tersebut maka hal tersebut akan direkam dan disimpan oleh mesin tersebut dan kemudian akan memutar yang ada pada list selanjutnya.



**Gambar 4.129** Machine Learning Youtube

4. Jelaskan apa yang dimaksud vektorisasi data.

Pemecahan data menjadi bagian-bagian yang lebih sederhana contoh ada sebuah paragraf, dari paragraf tersebut akan dibagi-bagi menjadi kalimat, yang nantinya akan dibagi-bagi kembali menjadi perkata.

5. Jelaskan apa itu bag of words dengan kata-kata yang sederhana dan ilustrasi sendiri

5. Model bag-of-words adalah representasi penyederhanaan yang digunakan dalam pemrosesan bahasa alami dan pengambilan informasi (IR). Dalam model ini, sebuah teks (seperti kalimat atau dokumen) direpresentasikan sebagai tas (multiset) dari kata-katanya, mengabaikan tata bahasa dan bahkan urutan kata tetapi menjaga multiplisitas. Model bag-of-words juga telah digunakan untuk visi komputer. Model bag-of-words umumnya digunakan dalam metode klasifikasi dokumen di mana (frekuensi) kemunculan setiap kata digunakan sebagai fitur untuk melatih classifier

	Document 1	Document 2
Term	Document 1	Document 2
aid	0   1	
all	1   0	
back	1   0	
brown	1   0	
come	1   0	
good	1   0	
fox	1   0	
men	1   0	
lazy	1   0	
the	1   0	
time	0   1	
to		1   0
of		1   0
for		1   0
party		1   0
their		1   0
over		1   0
had		1   0

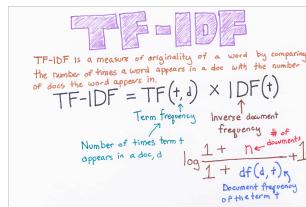
Stopword List

**Gambar 4.130** Bag of Words

6. Jelaskan apa itu TF-IDF, ilustrasikan dengan gambar sendiri.

metode algoritma yang berguna untuk menghitung bobot setiap kata yang umum digunakan. Metode ini juga terkenal efisien, mudah dan

memiliki hasil yang akurat. Metode ini akan menghitung nilai Term Frequency (TF) dan Inverse Document Frequency (IDF) pada setiap token (kata) di setiap dokumen dalam korpus. Secara sederhana, metode TF-IDF digunakan untuk mengetahui berapa sering suatu kata muncul di dalam dokumen.



Gambar 4.131 TF-IDF

#### 4.10.2 Praktek

1. import data pandas dan 500 baris data dumy kemudian di jelaskan tiap barisnya.

```
1 import pandas as pd #mengimport library pandas dan menamainya
pd
2 #%%
3 tia = pd.read_csv("D:/TI/SMT 6/AI/Chapter4/New folder/src
    /1174086.csv") #membuat variable bernama tia dan
    mengisinya dengan data dari dataset dummy yang telah
    dibuat
4 a = tia.head() #untuk melihat 5 baris pertama dari data tia
5 tia.shape #untuk mengetahui berapa banyak baris data
6 print(a) #menampilkan isi dari varibale a pada console
```

The screenshot shows a Jupyter Notebook interface. The code cell contains the following:

```
In [3]: tia = pd.read_csv("D:/TI/SMT 6/AI/Chapter4/New folder/src/1174086.csv")
#membuat variable bernama tia dan mengisinya dengan data dari dataset dummy yang telah
dibuat
a = tia.head() #untuk melihat 5 baris pertama dari data tia
tia.shape #untuk mengetahui berapa banyak baris data
print(a) #menampilkan isi dari varibale a pada console
```

The output cell shows the first 5 rows of the 'tia' DataFrame:

	id	first_name	last_name	email	gender
1	1	Angela	Bentley	abentley@ipgjaido.com	Female
2	2	Karen	Tokan	ktokan@bigipgl.de	Female
3	3	Osvaldo	Gortez	ogortez@outsystems.com	Female
4	5	Callie	Pyle	cpyle@netwises.com	Female

Gambar 4.132 Data Dummy

2. memecah data prame menjadi dua yag pertama 450 dan kedua sisanya

```
1 #%%
```

```

2 dtra = tia[:450] #memasukkan 450 data pertama ke dalam
   variable dtra
3 dtes = tia[450:] #memasukkan 50 data terakhir kedalam
   variable dtes

```

```

dtes DataFrame (50, 5) [Column names: id, first_name, last_name, email, gender]
dtra DataFrame (450, 5) [Column names: id, first_name, last_name, email, gender]

```

Gambar 4.133 Pisah Data

### 3. praktik vektorisasi

```

1 #%% memasukkan data dari file csv tersebut ke dalam variable
   data
2 data=pd.read_csv("D:/TI/SMT 6/AI/Chapter4/New folder/src
   /1174086.csv")
3 spam=data.query('CLASS == 1')
4 nospam=data.query('CLASS == 0')
5 #%% melakukan fungsi bag of word dengan cara menghitung semua
   kata
6 from sklearn.feature_extraction.text import CountVectorizer
7 vectorizer = CountVectorizer()
8 #%% melakukan bag of word pada dataframe pada colom CONTENT
9 data_vektorisasi = vectorizer.fit_transform(data['CONTENT'])
10 #%% melihat isi vektorisasi
11 data_vektorisasi
12 #%% melihat isi data pada baris ke 345
13 print(data['CONTENT'][345])
14 #%% untuk mengambil apa saja nama kolom yang tersedia
15 dk=vectorizer.get_feature_names()
16 #%%: melakukan randomisasi agar hasil sempurna pada saat
   klasifikasi
17 dshuf = data.sample(frac=1)
18 #%%: membuat data traning dan testing
19 dk_train=dshuf[:300]
20 dk_test=dshuf[300:]
21 #%%: melakukan training pada data training dan di vektorisasi
22 dk_train_att=vectorizer.fit_transform(dk_train['CONTENT'])
23 print(dk_train_att)
24 #%% melakukan testing pada data testing dan di vektorisasi
25 dk_test_att=vectorizer.transform(dk_test['CONTENT'])
26 print(dk_test_att)
27 #%%: Dimana akan mengambil label spam dan bukan spam
28 dk_train_label=dk_train['CLASS']
29 print(dk_train_label)
30 dk_test_label=dk_test['CLASS']
31 print(dk_test_label)

```

lakukan import library pandas yang diinisialisasi menjadi pd setelah itu ada dibuat variabel data dengan method read\_csv untuk membaca file berekstensikan csv yang dimasukan alamatnya pada kurung, lakukan klasifikasi atau pemilihan komentar yang berisi spam atau bukan spam

dengan parameter class samadengan 1 merupakan spam dan class samadengan 0 bukan spam setelah itu masukan librari CountVektorizer yang digunakan untuk vektorisasi data kemudian dilanjutkan pada bagian In[103] dibuat variabel yang berisi vektorisasi dari data pada data di field content setelah itu variabel tersebut di running hasilnya menunjukan 350 baris di kali 1738 kolom selanjutnya dicoba untuk memunculkan isi recod pada baris ke 345 maka akan muncul isian dari baris tersebut. selanjutnya dibuat variabel dk atau daftar yang berisi data hasil vektorisasi setelah yang terdiri dari variabel dshuf yang berisi data komen yang di dalamnya di buat random yang nantinya akan dibut data training dan data testing dengan ketentuan data training 300 dan data testing sebanyak 50 setelah itu data training di lakukan vektorisasi dan data testing juga dilakukan vektorisasi setelah itu kedua data training dan testing tersebut dibuat label dengan parameter field CLASS pada tabel.

```
...: dk_train,dk_test,dk_train['CLASS']
...: print(dk_train_label)
...: dk_test_label=dk_test['CLASS']
...: print(dk_test_label)
who is going to reach the billion first : katy or taylor ?
...
#> [1] "katy"
#> [2] "katy"
#> [3] "taylor"
#> [4] "taylor"
#> [5] "taylor"
#> [6] "taylor"
#> [7] "taylor"
#> [8] "taylor"
#> [9] "taylor"
#> [10] "taylor"
#> [11] "taylor"
#> [12] "taylor"
#> [13] "taylor"
#> [14] "taylor"
#> [15] "taylor"
#> [16] "taylor"
#> [17] "taylor"
#> [18] "taylor"
#> [19] "taylor"
#> [20] "taylor"
#> [21] "taylor"
#> [22] "taylor"
#> [23] "taylor"
#> [24] "taylor"
#> [25] "taylor"
#> [26] "taylor"
#> [27] "taylor"
#> [28] "taylor"
#> [29] "taylor"
#> [30] "taylor"
#> [31] "taylor"
#> [32] "taylor"
#> [33] "taylor"
#> [34] "taylor"
#> [35] "taylor"
#> [36] "taylor"
#> [37] "taylor"
#> [38] "taylor"
#> [39] "taylor"
#> [40] "taylor"
#> [41] "taylor"
#> [42] "taylor"
#> [43] "taylor"
#> [44] "taylor"
#> [45] "taylor"
#> [46] "taylor"
#> [47] "taylor"
#> [48] "taylor"
#> [49] "taylor"
#> [50] "taylor"
#> [51] "taylor"
#> [52] "taylor"
#> [53] "taylor"
#> [54] "taylor"
#> [55] "taylor"
#> [56] "taylor"
#> [57] "taylor"
#> [58] "taylor"
#> [59] "taylor"
#> [60] "taylor"
#> [61] "taylor"
#> [62] "taylor"
#> [63] "taylor"
#> [64] "taylor"
#> [65] "taylor"
#> [66] "taylor"
#> [67] "taylor"
#> [68] "taylor"
#> [69] "taylor"
#> [70] "taylor"
#> [71] "taylor"
#> [72] "taylor"
#> [73] "taylor"
#> [74] "taylor"
#> [75] "taylor"
#> [76] "taylor"
#> [77] "taylor"
#> [78] "taylor"
#> [79] "taylor"
#> [80] "taylor"
#> [81] "taylor"
#> [82] "taylor"
#> [83] "taylor"
#> [84] "taylor"
#> [85] "taylor"
#> [86] "taylor"
#> [87] "taylor"
#> [88] "taylor"
#> [89] "taylor"
#> [90] "taylor"
#> [91] "taylor"
#> [92] "taylor"
#> [93] "taylor"
#> [94] "taylor"
#> [95] "taylor"
#> [96] "taylor"
#> [97] "taylor"
#> [98] "taylor"
#> [99] "taylor"
#> [100] "taylor"
#> [101] "taylor"
#> [102] "taylor"
#> [103] "taylor"
#> [104] "taylor"
#> [105] "taylor"
#> [106] "taylor"
#> [107] "taylor"
#> [108] "taylor"
#> [109] "taylor"
#> [110] "taylor"
#> [111] "taylor"
#> [112] "taylor"
#> [113] "taylor"
#> [114] "taylor"
#> [115] "taylor"
#> [116] "taylor"
#> [117] "taylor"
#> [118] "taylor"
#> [119] "taylor"
#> [120] "taylor"
#> [121] "taylor"
#> [122] "taylor"
#> [123] "taylor"
#> [124] "taylor"
#> [125] "taylor"
#> [126] "taylor"
#> [127] "taylor"
#> [128] "taylor"
#> [129] "taylor"
#> [130] "taylor"
#> [131] "taylor"
#> [132] "taylor"
#> [133] "taylor"
#> [134] "taylor"
#> [135] "taylor"
#> [136] "taylor"
#> [137] "taylor"
#> [138] "taylor"
#> [139] "taylor"
#> [140] "taylor"
#> [141] "taylor"
#> [142] "taylor"
#> [143] "taylor"
#> [144] "taylor"
#> [145] "taylor"
#> [146] "taylor"
#> [147] "taylor"
#> [148] "taylor"
#> [149] "taylor"
#> [150] "taylor"
#> [151] "taylor"
#> [152] "taylor"
#> [153] "taylor"
#> [154] "taylor"
#> [155] "taylor"
#> [156] "taylor"
#> [157] "taylor"
#> [158] "taylor"
#> [159] "taylor"
#> [160] "taylor"
#> [161] "taylor"
#> [162] "taylor"
#> [163] "taylor"
#> [164] "taylor"
#> [165] "taylor"
#> [166] "taylor"
#> [167] "taylor"
#> [168] "taylor"
#> [169] "taylor"
#> [170] "taylor"
#> [171] "taylor"
#> [172] "taylor"
#> [173] "taylor"
#> [174] "taylor"
#> [175] "taylor"
#> [176] "taylor"
#> [177] "taylor"
#> [178] "taylor"
#> [179] "taylor"
#> [180] "taylor"
#> [181] "taylor"
#> [182] "taylor"
#> [183] "taylor"
#> [184] "taylor"
#> [185] "taylor"
#> [186] "taylor"
#> [187] "taylor"
#> [188] "taylor"
#> [189] "taylor"
#> [190] "taylor"
#> [191] "taylor"
#> [192] "taylor"
#> [193] "taylor"
#> [194] "taylor"
#> [195] "taylor"
#> [196] "taylor"
#> [197] "taylor"
#> [198] "taylor"
#> [199] "taylor"
#> [200] "taylor"
#> [201] "taylor"
#> [202] "taylor"
#> [203] "taylor"
#> [204] "taylor"
#> [205] "taylor"
#> [206] "taylor"
#> [207] "taylor"
#> [208] "taylor"
#> [209] "taylor"
#> [210] "taylor"
#> [211] "taylor"
#> [212] "taylor"
#> [213] "taylor"
#> [214] "taylor"
#> [215] "taylor"
#> [216] "taylor"
#> [217] "taylor"
#> [218] "taylor"
#> [219] "taylor"
#> [220] "taylor"
#> [221] "taylor"
#> [222] "taylor"
#> [223] "taylor"
#> [224] "taylor"
#> [225] "taylor"
#> [226] "taylor"
#> [227] "taylor"
#> [228] "taylor"
#> [229] "taylor"
#> [230] "taylor"
#> [231] "taylor"
#> [232] "taylor"
#> [233] "taylor"
#> [234] "taylor"
#> [235] "taylor"
#> [236] "taylor"
#> [237] "taylor"
#> [238] "taylor"
#> [239] "taylor"
#> [240] "taylor"
#> [241] "taylor"
#> [242] "taylor"
#> [243] "taylor"
#> [244] "taylor"
#> [245] "taylor"
#> [246] "taylor"
#> [247] "taylor"
#> [248] "taylor"
#> [249] "taylor"
#> [250] "taylor"
#> [251] "taylor"
#> [252] "taylor"
#> [253] "taylor"
#> [254] "taylor"
#> [255] "taylor"
#> [256] "taylor"
#> [257] "taylor"
#> [258] "taylor"
#> [259] "taylor"
#> [260] "taylor"
#> [261] "taylor"
#> [262] "taylor"
#> [263] "taylor"
#> [264] "taylor"
#> [265] "taylor"
#> [266] "taylor"
#> [267] "taylor"
#> [268] "taylor"
#> [269] "taylor"
#> [270] "taylor"
#> [271] "taylor"
#> [272] "taylor"
#> [273] "taylor"
#> [274] "taylor"
#> [275] "taylor"
#> [276] "taylor"
#> [277] "taylor"
#> [278] "taylor"
#> [279] "taylor"
#> [280] "taylor"
#> [281] "taylor"
#> [282] "taylor"
#> [283] "taylor"
#> [284] "taylor"
#> [285] "taylor"
#> [286] "taylor"
#> [287] "taylor"
#> [288] "taylor"
#> [289] "taylor"
#> [290] "taylor"
#> [291] "taylor"
#> [292] "taylor"
#> [293] "taylor"
#> [294] "taylor"
#> [295] "taylor"
#> [296] "taylor"
#> [297] "taylor"
#> [298] "taylor"
#> [299] "taylor"
#> [300] "taylor"
#> [301] "taylor"
#> [302] "taylor"
#> [303] "taylor"
#> [304] "taylor"
#> [305] "taylor"
#> [306] "taylor"
#> [307] "taylor"
#> [308] "taylor"
#> [309] "taylor"
#> [310] "taylor"
#> [311] "taylor"
#> [312] "taylor"
#> [313] "taylor"
#> [314] "taylor"
#> [315] "taylor"
#> [316] "taylor"
#> [317] "taylor"
#> [318] "taylor"
#> [319] "taylor"
#> [320] "taylor"
#> [321] "taylor"
#> [322] "taylor"
#> [323] "taylor"
#> [324] "taylor"
#> [325] "taylor"
#> [326] "taylor"
#> [327] "taylor"
#> [328] "taylor"
#> [329] "taylor"
#> [330] "taylor"
#> [331] "taylor"
#> [332] "taylor"
#> [333] "taylor"
#> [334] "taylor"
#> [335] "taylor"
#> [336] "taylor"
#> [337] "taylor"
#> [338] "taylor"
#> [339] "taylor"
#> [340] "taylor"
#> [341] "taylor"
#> [342] "taylor"
#> [343] "taylor"
#> [344] "taylor"
#> [345] "taylor"
#> [346] "taylor"
#> [347] "taylor"
#> [348] "taylor"
#> [349] "taylor"
#> [350] "taylor"
```

Gambar 4.134 Vektorisasi

#### 4. klasifikasi SVM

```
1 from sklearn import svm
2 clfsvm = svm.SVC()
3 clfsvm . fit (dk_train_att , dk_train_label)
4 clfsvm . score (dk_test_att , dk_test_label)
```

import librari svm dari sklearn kemudian membuat variabel clfsvm berisikan method svc setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

```
In [66]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(dk_train_att, dk_train_label)
...: clfsvm.score(dk_test_att, dk_test_label)
Out[66]: 0.94
```

Gambar 4.135 SVM

## 5. klasifikasi decision tree

```
1 from sklearn import tree  
2 clftree = tree.DecisionTreeClassifier()  
3 clftree.fit(dk_train_att, dk_train_label)  
4 clftree.score(dk_test_att, dk_test_label)
```

import librari tree dari sklearn kemudian membuat variabel clftree berisikan method DecisionTreeClasifier setelah itu variabel tersebut di berikan method fit dengan isian data train vektorisasi dan data training label yang berguna untuk melatih data tersebut agar dapat digunakan pada codingan selanjutnya setelah itu di coba untuk memunculkan score atau akurasi dari data tersebut menggunakan data testing vektorisasi dan data testing label.

```
In [67]: from sklearn import tree
      ...: clftree = tree.DecisionTreeClassifier()
      ...: clftree.fit(dk_train_att, dk_train_label)
      ...: clftree.score(dk_test_att, dk_test_label)
Out[67]: 0.96
```

Gambar 4.136 Desicion Tree

## 6. plot comfusion matrix

```
1 from sklearn.metrics import confusion_matrix  
2 pred_labels = clftree.predict(dk_test_att)  
3 cm = confusion_matrix(dk_test_label, pred_labels)  
4 cm
```

import library comfusion matrix selanjutnya dilakukan prediksi pada pada data tes nya kemudian data tersebut di masukan kedalam variabel cm dengan method confusion matrix yang di dalamnya terdapat data dari variabel perd label dan dk test label setelah itu variabel cm tersebut di running maka akan memunculkan nilai matrixnya.

```
In [80]: from sklearn.metrics import confusion_matrix
...: pred_labels = clftree.predict(dk_test_att)
...: cm = confusion_matrix(dk_test_label, pred_labels)
...: cm
Out[80]:
array([[25,  1],
       [ 1, 23]], dtype=int64)
```

Gambar 4.137 Confusion Matrix

## 7. cross validation

```
1 #%%  
2 from sklearn.model_selection import cross_val_score  
3 scores = cross_val_score(clftree, dk_train_att, dk_train_label,  
    cv=5)  
4 scorerata2=scores.mean()
```

```

5 scorersd=scores . std ()
6 #%%:
7 from sklearn.model_selection import cross_val_score
8 scores = cross_val_score(clftree , dk_train_att ,
9 dk_train_label , cv=5)
# show average score and +/- two standard deviations away (
10 # covering 95
11 #%% of scores)
12 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean() ,
13 scores.std() * 2))
14 #%%:
15 scorestree = cross_val_score(clftree , dk_train_att ,
16 dk_train_label , cv=5)
17 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean() ,
18 scorestree.std() * 2))
19 #%%:
20 scoressvm = cross_val_score(clfsvm , dk_train_att ,
21 dk_train_label , cv=5)
22 print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean() ,
23 scoressvm.std() * 2))

```

memunculkan nilai akurasi dari tiga metode yaitu random forest, decision tree, dan klasifikasi svm (suport vector machine) diamana akan di bandingkan tingkat akurasi dari semua hasil akurasi mana yang terbaik dan lebih akurat pada hasilnya data yang paling akurat yaitu random forest.

```

In [82]: from sklearn.model_selection import cross_val_score
...: ...: scores = cross_val_score(clftree , dk_train_att , dk_train_label , cv=5)
...: ...: scorestree = scores.mean()
...: ...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean() ,
...: ...: scores.std() * 2))
...: ...: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean() ,
...: ...: scorestree.std() * 2))
Accuracy: 0.82 (+/- 0.07)

In [83]: from sklearn.model_selection import cross_val_score
...: ...: scores = cross_val_score(clftree , dk_train_att , dk_train_label , cv=5)
...: ...: # show average score and +/- two standard deviations away (covering 95
...: ...: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean() ,
...: ...: scores.std() * 2))
...: ...: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean() ,
...: ...: scorestree.std() * 2))
Accuracy: 0.82 (+/- 0.07)

In [84]: scorestree = cross_val_score(clftree , dk_train_att , dk_train_label , cv=5)
...: ...: print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean() ,
...: ...: scorestree.std() * 2))
Accuracy: 0.82 (+/- 0.07)

In [85]: scoressvm = cross_val_score(clfsvm , dk_train_att , dk_train_label , cv=5)
...: ...: print("Accuracy: %0.2f (+/- %0.2f)" % (scoressvm.mean() ,
...: ...: scoressvm.std() * 2))
Accuracy: 0.91 (+/- 0.08)

```

Gambar 4.138 Cross Validation

## 8. Pengamatan program

```

1 #%%
2 import numpy as np
3 max_features_opts = range(1, 10, 1)
4 n_estimators_opts = range(2, 40, 4)
5 rf_params = np.empty((len(max_features_opts)*len(
6 n_estimators_opts),4) , float)
7 i = 0
8 for max_features in max_features_opts:
9     for n_estimators in n_estimators_opts:
10         clf = RandomForestClassifier(max_features=
11             max_features , n_estimators=n_estimators)
12         scores = cross_val_score(clf , dk_train_att ,
13 dk_train_label , cv=5)
14         rf_params[i,0] = max_features
15         rf_params[i,1] = n_estimators

```

```

13         rf_params[i,2] = scores.mean()
14         rf_params[i,3] = scores.std() * 2
15         i += 1
16         print("Max features: %d, num estimators: %d, accuracy"
17 : "%0.2f (+/- %0.2f)" % (max_features, n_estimators, scores.mean(), scores.std() *
2))

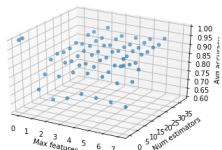
```

terdapat grafik data yang terdapat dari grafik tersebut di dapat dari codingan dengan cara pengulangan data masing masing 10 kali setelah itu di eksekusi menjadi grafik berbentuk 3D pada gambar tersebut menunjukkan rasio dari yang terrendah yaitu data SVM kemudian data decision tree dan hasil random forest.

### Gambar 4.139 Pengamatan Program

Berikut adalah untuk grafik

```
1 import matplotlib.pyplot as plt
2 from mpl_toolkits.mplot3d import Axes3D
3 from matplotlib import cm
4 fig = plt.figure()
5 fig.clf()
6 ax = fig.gca(projection='3d')
7 x = rf_params[:,0]
8 y = rf_params[:,1]
9 z = rf_params[:,2]
10 ax.scatter(x, y, z)
11 ax.set_zlim(0.6, 1)
12 ax.set_xlabel('Max features')
13 ax.set_ylabel('Num estimators')
14 ax.set_zlabel('Avg accuracy')
15 plt.show()
```



Gambar 4.140 Grafik

### 4.10.3 Penanganan Error

#### 4.10.3.1 Screenshoot Error

##### 1. Error 1

```
r1_params = np.empty((len(max_features_opts)*len(n_estimators_opts),4), float)
NameError: name 'np' is not defined
```

**Gambar 4.141** Error1

##### 2. Error 2

```
file "pandas\_libs\hashtable_class_helper.pxi", line 1492, in
pandas\_libs\hashtable\PyObjectHashTable.get_item
file "pandas\_libs\hashtable_class_helper.pxi", line 1500, in
pandas\_libs\hashtable\PyObjectHashTable.get_item
KeyError: 'CONTENTz'
```

**Gambar 4.142** Error2

##### 3. Error 3

```
file "pandas\_libs\parsers.pyx", line 695, in
pandas\_libs\parsers\TextHeader._setup_parser_source
FileNotFoundError: File b'D:\SemesterV\ATI\Chapter40d0b0.csv' does not exist
```

**Gambar 4.143** Error3

##### 4. Error 4

```
file "pandas\_libs\hashtable_class_helper.pxi", line 964, in
pandas\_libs\hashtable\Int64HashTable.get_item
KeyError: 5000
```

**Gambar 4.144** Error4

#### 4.10.3.2 Kode Error dan Jenisnya

##### 1. Kode Error 1 jenis Name Error

```
max_features_opts = np.empty(39, 1)
n_estimators_opts = np.empty(40, 4)
r1_params = np.empty((len(max_features_opts)*len(n_estimators_opts),4), float)
i = 0
```

**Gambar 4.145** Error1

##### 2. Kode Error 2 jenis Key Error

```
data_vektorisasi = vectorizer.fit_transform(data['CONTENTz'])
```

**Gambar 4.146** Error2

### 3. Kode Error 3 jenis FileNotFoundError

```
#%%
tia = pd.read_csv("D:/TI/SMT 6/AI/Chapter4/New folder/src/1174886.csv")
a = tia.head() #untuk melihat 5 baris pertama dari data tia
tia.shape #untuk mengetahui berapa banyak baris data
print(a) #menampilkan isi dari variabel a pada console
```

**Gambar 4.147** Error3

### 4. Kode Error 4 Key Error

```
#%%
print(data['CONTENT'][5000])
```

**Gambar 4.148** Error4

#### 4.10.3.3 Solusi

##### 1. Mengimport library numpy sebagai np

```
import numpy as np
max_features_opts = range(1, 10, 1)
n_estimators_opts = range(2, 40, 4)
rf_params = np.empty((len(max_features_opts)*len(n_estimators_opts),4), float)
```

**Gambar 4.149** Solusi 1

##### 2. Mengganti CONTENTz menjadi CONTENT

```
data_vektorisasi = vectorizer.fit_transform(data['CONTENT'])
```

**Gambar 4.150** Solusi 2

##### 3. Merubah backslash menjadi slash biasa

```
#%%
tia = tia.head() #untuk melihat 5 baris pertama dari data tia
tia.shape #untuk mengetahui berapa banyak baris data
print(a) #menampilkan isi dari variabel a pada console
```

**Gambar 4.151** Solusi 3

##### 4. Merubah data menjadi dibawah 350

```
#%%
print(data['CONTENT'][349])
```

**Gambar 4.152** Solusi 4

#### 4.10.4 Link Youtube

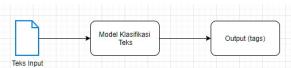
<https://youtu.be/Zsk8IXYg7Gk>

## 4.11 1174071 - Muhammad Abdul Gani Wijaya

### 4.11.1 Teori

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

Klasifikasi teks merupakan sebuah model yang biasa digunakan untuk 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 4.153** Klasifikasi teks.

2. Jelaskan mengapa hal ini bisa terjadi, klasifikasi bunga tidak bisa digunakan untuk machine learning, sertakan ilustrasi gambar sendiri.

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 4.154** Klasifikasi Bunga.

3. Jelaskan bagaimana yang dimaksud dengan teknik pembelajaran mesin pada teks yang digunakan dan sertakan ilustrasi buatan sendiri.

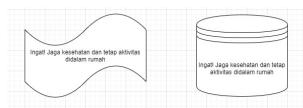
Teknik yang digunakan pada youtube salah satunya ialah keywords. Dengan keywords tersebut mesin dapat memberikan video sesuai dengan keyword yang kita inputkan pada kolom pencarian. Teknik pembelajarannya tergantung user memberikan input teks seperti apa, karena pada youtube itu sendiri akan menyesuaikan 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 4.155** Klasifikasi teks Youtube.

4. Jelaskan apa yang dimaksud 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.
5. Jelaskan apa yang dimaksud dengan bag of words dengan ilustrasi sendiri.

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 4.156** Bag of words.

6. Jelaskan apa yang dimaksud dengan 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 4.157** TF-IDF.

#### 4.11.2 Praktek Program

1. Soal 1

```

1 #%% Soal 1
2 import pandas as pd
3 #digunakan untuk mengimport library pandas dengan alias pd

```

Kode di atas digunakan untuk buat aplikasi sederhana menggunakan pandas dengan format csv sebanyak 500 baris, hasilnya ialah sebagai berikut :

**Gambar 4.158** Hasil Soal 1.

## 2. Soal 2

```

1 #membaca file csv
2
3 #%% Soal 2

```

Kode di atas digunakan untuk memecah dataframe tersebut menjadi dua bagian yaitu 450 row pertama dan 50 row kedua. Hasilnya adalah sebagai berikut :

**Gambar 4.159** Hasil Soal 2.

## 3. Soal 3

```

1 #untuk membagi data training menjadi 450
2 d_test=pd[450:]
3 #untuk membagi data menjadi 50 atau sisa dari data yang tersedia
4
5 #%% Soal 3
6 import pandas as gani
7 #untuk import library pandas berguna untuk mengelola dataframe
8 gani = gani.read_csv("C:/Users/muham/Downloads/Compressed/KB3C-master/KB3C-master/src/1174071/4/Youtube03-Shakira.csv")

```

```

9 #untukmembaca file dengan format csv
10
11 spam=gani.query( 'CLASS == 1' )
12 #untuk membagi tabel spam
13 nospam=gani.query( 'CLASS == 0' )
14 #untuk membagi tabel no spam
15
16 from sklearn.feature_extraction.text import CountVectorizer
17 #import countvectorizer berfungsi untuk memecah data tersebut
18     menjadi sebuah kata yang lebih sederhana
19 vectorizer = CountVectorizer()
20 #menjalankan fungsi tersebut , pada code ini tidak ada
21     hasilnya dikarenakan spyder tidak mendukung hasil dari
22     instasiasi .
23
24 dvec = vectorizer.fit_transform(gani[ 'CONTENT' ])
25 #melakukan pemecahan data pada dataframe yang terdapat pada
26     kolom konten
27 dvec #menampilkan hasil dari code sebelumnya
28
29 Daptarkata= vectorizer.get_feature_names()

```

Dengan menggunakan  $1174069 \bmod 4$  adalah 1, yang artinya menggunakan dataset LMFAO. Hasilnya adalah sebagai berikut :

ferry - DataFrame					
Index	COMMENT_ID	AUTHOR	DATE	CONTENT	CLASS
0	r1234567890..	Cory Wilson	2015-05-28T21..	re	0
1	r1234567890..	Iain Gating	2015-05-28T21..	re	0
2	r1234567890..	Led Public	2015-05-28T21..	I'm trying to ha..	1
3	r1234567890..	Cheryl Fox	2015-05-28T21..	Rock...lol..	0
4	r1234567890..	PATICK_TW	2015-05-28T21..	Party rock	0
5	r1234567890..	Brian Bral	2015-05-28T0..	Shuffle	0
6	r1234567890..	Brian Bral	2015-05-28T0..	Omg	0
7	r1234567890..	Alex Derev	2015-05-28T0..	Just really ..	0
8	r1234567890..	Simeon	2015-05-28T0..	Assassinate />	0
9	r1234567890..	Silvia Bauer	2015-05-28T0..	Incredible so ..	0
10	r1234567890..	Uniteide	2015-05-28T0..	I love you so much	0
11	r1234567890..	gentle maria	2015-05-28T0..	THIS LIEEE	0
12	r1234567890..		2015-05-28T0..	I miss you so ..	0
13	r1234567890..	Alex Jansen	2015-05-28T0..	OMG THE PRESS	0
14	r1234567890..	Aliden Hill	2015-05-27T2..	Best song of ..	0
15	r1234567890..	joe paulin	2015-05-27T2..	super nice,	0
16	r1234567890..	silvia Bauer	2015-05-27T2..	so many	0
17	r1234567890..	bilal kerno	2015-05-27T2..	unbelievable	0
18	r1234567890..	SoulInADrain..	2015-05-27T1..	PARTY ROCK	0
19	r1234567890..	Phuong Lin	2015-05-27T1..	Thums up if ..	0
20	r1234567890..	Gontalo	2015-05-27T1..	THIS IS ..	0
	r1234567890..	Umar TQN	2015-05-27T1..	so cool!!!!!!	0

Gambar 4.160    Hasil Soal 3.

#### 4. Soal 4

```

1 dshuf = gani.sample( frac=1)
2
3 d_train=dshuf[:300]
4 d_test=dshuf[300:]

```

Klasifikasi dari data vektorisasi menggunakan klasifikasi Decision Tree. Hasilnya adalah sebagai berikut :

```
In [7]: from sklearn import svm
... clfsvm = svm.SVR(gamma = 'auto')
... clfsvm.fit(d_train_att, d_train_label)
Out[7]:
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
     gamma='auto',
     kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
     verbose=False)
```

Gambar 4.161 Hasil Soal 4.

## 5. Soal 5

```
1 d_train_att
2
3 d_train_label=d_train[ 'CLASS' ]
4 d_test_label=d_test[ 'CLASS' ]
```

Plotlah confusion matrix dari praktik modul ini menggunakan matplotlib. Hasilnya adalah sebagai berikut :

```
In [8]: from sklearn import tree
... clftree = tree.DecisionTreeClassifier()
... clftree.fit(d_train_att, d_train_label)
Out[8]:
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
criterion='gini',
max_depth=None, max_features=None,
max_leaf_nodes=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0,
presort='deprecated', random_state=None, splitter='best')
```

Gambar 4.162 Hasil Soal 5.

## 6. Soal 6

```
1
2 from sklearn import svm
3 clfsvm = svm.SVR(gamma = 'auto')
4 clfsvm . fit (d_train_att , d_train_label)
5
6 #%%soal 5
7
8 from sklearn import tree
9 clftree = tree.DecisionTreeClassifier()
10 clftree . fit (d_train_att , d_train_label)
11
12 #%%soal 6/
13
14 from sklearn.metrics import confusion_matrix
15 pred_labels=clftree.predict(d_test)
16 cm=confusion_matrix(d_test_label , pred_labels)
17
18 #%%
19
20 import matplotlib.pyplot as plt
21
22 def plot_confusion_matrix(cm, classes,
```

Menjalankan program cross validation. Hasilnya adalah sebagai berikut :

```
In [11]: import matplotlib.pyplot as plt
...
... def plot_confusion_matrix(cm, classes,
...                          normalize=False,
...                          title='Confusion matrix',
...                          cmap=plt.cm.Blues):
...
...     if normalize:
...         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
...         print("Normalized confusion matrix")
...     else:
...         print('Confusion matrix, without normalization')
...
...     print(cm)
...
...     plt.imshow(cm,
```

**Gambar 4.163** Hasil Soal 6.

## 7. Soal 7

```
1             title='Confusion matrix' ,
2             cmap=plt.cm.Blues):
3
4     if normalize:
5         cm = cm.astype('float') / cm.sum(axis=1)[:, np.
newaxis]
6         print("Normalized confusion matrix")
7     else:
8         print('Confusion matrix, without normalization')
```

Tree.export graphviz merupakan fungsi yang menghasilkan representasi Graphviz dari decision tree, yang kemudian ditulis ke outfile. Disini akan menyimpan classifiernya, akan meng ekspor file student performance jika salah akan mengembalikan nilai fail. Hasilnya adalah sebagai berikut :

```
In [12]: from sklearn.model_selection import cross_val_score
...
... scores=cross_val_score(clftree,d_train_att,d_train_label,cv=5)
...
... skor_rata2=scores.mean()
... skoresd=scores.std()
```

**Gambar 4.164** Hasil Soal 7.

## 8. Soal 8

```
1     print(cm)
2
3 #%%soal 7
4
5 from sklearn.model_selection import cross_val_score
6
7 scores=cross_val_score(clftree , d_train_att , d_train_label , cv
=5)
8
9 skor_rata2=scores .mean()
10 skoresd=scores .std()
11
12 #%%soal 8 /
```

```
13  
14 max_features_opts = range(5, 50, 5)  
15 #membuat max_features_opts sebagai variabel untuk membuat  
    range 5,50,5  
16 n_estimators_opts = range(10, 200, 20)  
17 #membuat n_estimators_opts sebagai variabel untuk membuat  
    range 10,200,20
```

Buatlah program pengamatan komponen informasi. Jadi disini kita akan memprediksi nilai dari variabel test att dan test pass Hasilnya adalah sebagai berikut :

**Gambar 4.165** Hasil Soal 8.

### 4.11.3 Penanganan Error

## 1. ScreenShoot Error

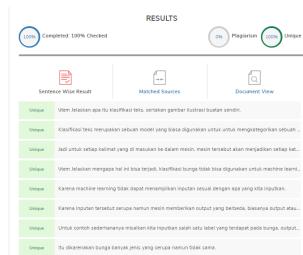
```
FileNotFoundException: [Errno 2] File b'F:/Semester 6/Artificial Intelligence/Tugas 4/src/fanny.csv' does not exist b'F:/Semester 6/Artificial Intelligence/Tugas 4/src/fanny.csv'
```

## 2. Cara Penangan Error

- **SyntaxError**

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

#### 4.11.4 Bukti Tidak Plagiat



Gambar 4.167 Bukti Tidak Melakukan Plagiat Chapter 4

#### 4.11.5 Link Youtube

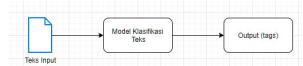
<https://youtu.be/X-xd9Nb78Gs>

### 4.12 1174071 - Muhammad Abdul Gani Wijaya

#### 4.12.1 Teori

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

Klasifikasi teks merupakan sebuah model yang biasa digunakan untuk 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 4.168 Klasifikasi teks.

2. Jelaskan mengapa hal ini bisa terjadi, klasifikasi bunga tidak bisa digunakan untuk machine learning, sertakan ilustrasi gambar sendiri. 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 4.169** Klasifikasi Bunga.

3. Jelaskan bagaimana yang dimaksud dengan teknik pembelajaran mesin pada teks yang digunakan dan sertakan ilustrasi buatan sendiri.

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 :



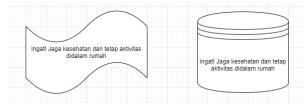
**Gambar 4.170** Klasifikasi teks Youtube.

4. Jelaskan apa yang dimaksud 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.

5. Jelaskan apa yang dimaksud dengan bag of words dengan ilustrasi sendiri.

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 4.171** Bag of words.

6. Jelaskan apa yang dimaksud dengan 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 4.172 TF-IDF.

#### 4.12.2 Praktek Program

##### 1. Soal 1

```
1 %% Soal 1
2 import pandas as pd
3 #digunakan untuk mengimport library pandas dengan alias pd
```

Kode di atas digunakan untuk buat aplikasi sederhana menggunakan pandas dengan format csv sebanyak 500 baris, hasilnya ialah sebagai berikut :

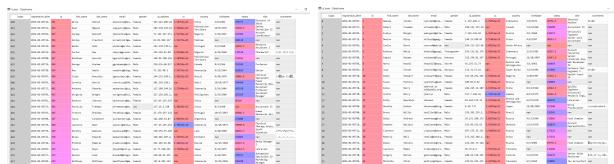


Gambar 4.173 Hasil Soal 1.

##### 2. Soal 2

```
1 #membaca file csv
2
3 %% Soal 2
```

Kode di atas digunakan untuk memecah dataframe tersebut menjadi dua bagian yaitu 450 row pertama dan 50 row kedua. Hasilnya adalah sebagai berikut :



Gambar 4.174 Hasil Soal 2.

## 3. Soal 3

```

1 #untuk membagi data training menjadi 450
2 d_test=pd[450:]
3 #untuk membagi data menjadi 50 atau sisa dari data yang tersedia
4
5 %% Soal 3
6 import pandas as gani
7 #untuk import library pandas berguna untuk mengelola dataframe
8 gani = gani.read_csv("C:/Users/muham/Downloads/Compressed/
KB3C-master/KB3C-master/src/1174071/4/Youtube03-Shakira .
csv")
9 #untuk membaca file dengan format csv
10
11 spam=gani.query('CLASS == 1')
12 #untuk membagi tabel spam
13 nospam=gani.query('CLASS == 0')
14 #untuk membagi tabel no spam
15
16 from sklearn.feature_extraction.text import CountVectorizer
17 #import countvectorizer berfungsi untuk memecah data tersebut menjadi sebuah kata yang lebih sederhana
18 vectorizer = CountVectorizer()
19 #menjalankan fungsi tersebut, pada code ini tidak ada hasilnya dikarenakan spyder tidak mendukung hasil dari instasiasi.
20
21 dvec = vectorizer.fit_transform(gani['CONTENT'])
22 #melakukan pemecahan data pada dataframe yang terdapat pada kolom konten
23 dvec #menampilkan hasil dari code sebelumnya
24
25 Daptarkata= vectorizer.get_feature_names()

```

Dengan menggunakan  $1174069 \bmod 4$  adalah 1, yang artinya menggunakan dataset LMFAO. Hasilnya adalah sebagai berikut :

Gambar 4.175 Hasil Soal 3.

#### 4. Soal 4

```
1 dshuf = gani.sample( frac=1)
2
3 d_train=dshuf[:300]
4 d_test=dshuf[300:]
```

Klasifikasi dari data vektorisasi menggunakan klasifikasi Decision Tree. Hasilnya adalah sebagai berikut :

```
In [7]: from sklearn import svm
...: clfsvm = svm.SVR(gamma = 'auto')
...: clfsvm.fit(d_train_att, d_train_label)
Out[7]:
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
gamma='auto',
kernel='rbf', max_iter=-1, shrinking=True, tol=0.001,
verbose=False)
```

**Gambar 4.176** Hasil Soal 4.

### 5. Soal 5

```
1 d_train_att  
2  
3 d_train_label=d_train['CLASS']  
4 d_test_label=d_test['CLASS']
```

Plotlah confusion matrix dari praktik modul ini menggunakan matplotlib. Hasilnya adalah sebagai berikut :

```
In [8]: from sklearn import tree
...: ctree = tree.DecisionTreeClassifier()
...: ctree.fit(d_train_att, d_train_label)
Out[8]:
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
criterion='gini',
max_depth=None, max_features=None,
max_leaf_nodes=None,
min_impurity_decrease=0.0,
min_impurity_split=None,
min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0,
presort='deprecated',
random_state=None, splitter='best')
```

Gambar 4.177 Hasil Soal 5.

## 6. Soal 6

```

1 from sklearn import svm
2 clfsvm = svm.SVR(gamma = 'auto')
3 clfsvm.fit(d_train_att , d_train_label)
4
5 #%%soal 5
6
7 from sklearn import tree
8 clftree = tree.DecisionTreeClassifier()
9 clftree.fit(d_train_att , d_train_label)
10
11 #%%soal 6/
12
13 from sklearn.metrics import confusion_matrix
14 pred_labels=clftree.predict(d_test)
15 cm=confusion_matrix(d_test_label , pred_labels)
16
17 #%%
18
19 import matplotlib.pyplot as plt
20
21 def plot_confusion_matrix(cm,
22

```

Menjalankan program cross validation. Hasilnya adalah sebagai berikut :

```

In [11]: import matplotlib.pyplot as plt
...:
...: def plot_confusion_matrix(cm, classes,
...:                         normalize=False,
...:                         title='Confusion matrix',
...:                         cmap=plt.cm.Blues):
...:
...:     if normalize:
...:         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
...:         print("Normalized confusion matrix")
...:     else:
...:         print('Confusion matrix, without normalization')
...:
...:     print(cm)
...:     cm

```

**Gambar 4.178** Hasil Soal 6.

## 7. Soal 7

```

1 title='Confusion matrix' ,
2 cmap=plt.cm.Blues):
3
4 if normalize:
5     cm = cm.astype('float') / cm.sum( axis=1)[:, np.
newaxis]
6     print("Normalized confusion matrix")
7 else:
8     print('Confusion matrix, without normalization')

```

Tree.export graphviz merupakan fungsi yang menghasilkan representasi Graphviz dari decision tree, yang kemudian ditulis ke outfile. Disini akan

menyimpan classifiernya, akan meng ekspor file student performance jika salah akan mengembalikan nilai fail. Hasilnya adalah sebagai berikut :

```
In [12]: from sklearn.model_selection import cross_val_score
...:
...:
scores=cross_val_score(clftree,d_train_att,d_train_label,cv=5
...:
...:
...:
skor_rata2=scores.mean()
...:
skorstd=scores.std()
```

**Gambar 4.179** Hasil Soal 7.

### 8. Soal 8

```
1     print(cm)
2
3 #%%soal 7
4
5 from sklearn.model_selection import cross_val_score
6
7 scores=cross_val_score(clftree,d_train_att,d_train_label,cv
8 =5)
9 skor_rata2=scores.mean()
10 skoresd=scores.std()
11
12 #%%soal 8 /
13
14 max_features_opts = range(5, 50, 5)
15 #membuat max_features_opts sebagai variabel untuk membuat
16 #range 5,50,5
17 n_estimators_opts = range(10, 200, 20)
18 #membuat n_estimators_opts sebagai variabel untuk membuat
19 #range 10,200,20
```

Buatlah program pengamatan komponen informasi. Jadi disini kita akan memprediksi nilai dari variabel test att dan test pass Hasilnya adalah sebagai berikut :

Gambar 4.180 Hasil Soal 8

### **4.12.3 Penanganan Error**

## 1. ScreenShoot Error

```
FileNotFoundException: [Errno 2] File b'F://Semester 6/Artificial Intelligence/Tugas 4/src/fanny.csv' does not exist: b'F://Semester 6/Artificial Intelligence/Tugas 4/src/fanny.csv'
```

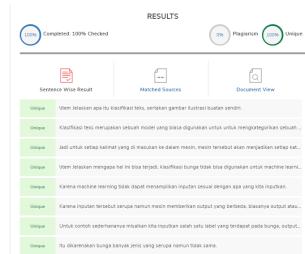
### Gambar 4.181 SyntaxError

## 2. Cara Penangan Error

- **SyntaxError**

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

#### 4.12.4 Bukti Tidak Plagiat



**Gambar 4.182** Bukti Tidak Melakukan Plagiat Chapter 4

#### 4.12.5 Link Youtube

<https://youtu.be/WA8JwD2lzPw>

## 4.13 1174067 - Kaka Kamaludin

#### 4.13.1 Teori

#### 4.13.1.1 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 4.183** Klasifikasi Teks.

#### 4.13.1.2 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 adalah label yang lain. Itu dikarenakan bunga banyak jenis yang serupa namun tidak sama. Untuk ilustrasinya bisa dilihat pada gambar berikut:



**Gambar 4.184** Klasifikasi Bunga.

#### 4.13.1.3 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 4.185** Klasifikasi Teks pada Youtube.

#### 4.13.1.4 Vektorisasi Data

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

#### 4.13.1.5 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 4.186** Bag of Words.

#### 4.13.1.6 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 4.187** TF-IDF.

#### 4.13.2 Praktek

4.13.2.1 buat aplikasi klasifikasi sederhana menggunakan pandas, buat data dummy format csv sebanyak 500 baris dan melakukan load ke dataframe panda.jelaskan arti setiap baris kode yang dibuat(harus beda dengan teman sekelas)

```

1 import pandas as pd #import package pandas, lalu dialiaskan
    menjadi pd.
2 csv = pd.read_csv('D:/OneDrive - Hybi.god/KULIAH/Semester 6/AI/
    KB3C/src/1174067/4/csv.csv', sep=',') #membaca file csv
    dimana data pada file csv dipisahkan oleh koma, lalu
    ditampung di variable csv.

```

**Listing 4.12** kodingan praktek no. 1

Index	registration_dttr	id	first_name	last_name	email
0	2016-02-03T0...	1	Amanda	Jordan	ajordan0@...
1	2016-02-03T0...	2	Albert	Freeman	afreeman1...
2	2016-02-03T0...	3	Evelyn	Morgan	emorgan2@...
3	2016-02-03T0...	4	Denise	Riley	driley3@...
4	2016-02-03T0...	5	Carlos	Burns	cburns4@...
5	2016-02-03T0...	6	Kathryn	White	kwhite5@...
6	2016-02-03T0...	7	Samuel	Holmes	sholmes6@...
7	2016-02-03T0...	8	Harry	Howell	hhowell7@...
8	2016-02-03T0...	9	Jose	Foster	jfoster8@...
9	2016-02-03T1...	10	Emily	Stewart	estewart9@...
10	2016-02-03T1...	11	Susan	Perkins	sperkins10@...
11	2016-02-03T1...	12	Alice	Berry	aberryb11@...
12	2016-02-03T1...	13	Justin	Berry	jberry12@...
13	2016-02-03T2...	14	Kathy	Reynolds	Unknown

Gambar 4.188 hasil praktek soal no. 1

4.13.2.2 dari dataframe tersebut dipecah menjadi dua dataframe yaitu 450 row pertama dan 50 row sisanya(harus beda dengan teman sekelas)

```
1 csv1, csv2 = csv [:450], csv [450:] #membagi data menjadi dua bagian, variabel csv1 untuk menampung 450 baris data pertama, variabel csv2 untuk menampung 50 baris data terakhir.
```

Listing 4.13 kodingan praktek no. 2

Name	Type	Size	Value
csv	DataFrame	(499, 13)	Column names: registration_dttr, id, first_name, last_name, email, gen...
csv1	DataFrame	(450, 13)	Column names: registration_dttr, id, first_name, last_name, email, gen...
csv2	DataFrame	(49, 13)	Column names: registration_dttr, id, first_name, last_name, email, gen...

Gambar 4.189 hasil praktek soal no. 2

4.13.2.3 praktekkan vektorisasi dan klasifikasi dari data(NPM mod 4, jika 0 maka ketty perry, 1 LMFAO, 2 Eminem, 3 Shakira) dengan Decission Tree. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud luaran yang dapatkan

```
1 print(1174067%4) #hasilnya 3, Shakira
```

Listing 4.14 1174067 mod 4

```

1 # Vektorisasi Data
2 import pandas as pd
3 d = pd.read_csv("Youtube05-Shakira.csv")
4
5 from sklearn.feature_extraction.text import CountVectorizer
6 vectorizer = CountVectorizer()
7
8 dvec = vectorizer.fit_transform(d[ 'CONTENT' ])
9 dvec
10
11 daptarkata = vectorizer.get_feature_names()
12
13 dshuf = d.sample(frac=1)
14
15 d_train = dshuf[:300]
16 d_test = dshuf[300:]
17
18 d_train_att = vectorizer.fit_transform(d_train[ 'CONTENT' ])
19 d_train_att
20
21 d_test_att = vectorizer.transform(d_test[ 'CONTENT' ])
22 d_test_att
23
24 d_train_label = d_train[ 'CLASS' ]
25 d_test_label = d_test[ 'CLASS' ]

```

**Listing 4.15** kodingan praktek no. 3

```

In [9]: import pandas as pd
...: d = pd.read_csv("Youtube05-Shakira.csv")
...:
...: from sklearn.feature_extraction.text import CountVectorizer
...: vectorizer = CountVectorizer()
...:
...: dvec = vectorizer.fit_transform(d[ 'CONTENT' ])
...: dvec
...:
...: daptarkata = vectorizer.get_feature_names()
...:
...: 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
...:
...: d_test_att = vectorizer.transform(d_test[ 'CONTENT' ])
...: d_test_att
...:
...: d_train_label = d_train[ 'CLASS' ]
...: d_test_label = d_test[ 'CLASS' ]

```

In [10]: |

**Gambar 4.190** hasil praktek soal no. 3(1)

Name	Type	Size	Value
d	DataFrame	(370, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS
d_test	DataFrame	(70, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS
d_test_label	Series	(70,)	Series object of pandas.core.series module
d_train	DataFrame	(300, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS
d_train_label	Series	(300,)	Series object of pandas.core.series module
daptarkata	list	1357	['00', '000', '0687119038', '08', '10', '100', '10171377578919894134' ...]
dshuf	DataFrame	(370, 5)	Column names: COMMENT_ID, AUTHOR, DATE, CONTENT, CLASS

Gambar 4.191 hasil praktek soal no. 3(2)

4.13.2.4 Cobalah klarifikasi dari data vektorisasi yang ditentukan di nomor sebelumnya dengan klasifikasi SVM. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

```
1 from sklearn import svm
2 clfsvm = svm.SVC()
3 clfsvm.fit(d_train_att, d_train_label)
4 clfsvm.score(d_test_att, d_test_label)
```

Listing 4.16 kodingan praktek no. 4

```
In [10]: from sklearn import svm
...: clfsvm = svm.SVC()
...: clfsvm.fit(d_train_att, d_train_label)
...: clfsvm.score(d_test_att, d_test_label)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:103: 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)
Out[10]: 0.6142857142857143
```

In [11]:

Gambar 4.192 hasil praktek soal no. 4

4.13.2.5 Cobalah klasifikasi dari data vektorisasi yang ditentukan dari nomor sebelumnya dengan klasifikasi Decision Tree. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan

```
1 from sklearn import tree
2 clftree = tree.DecisionTreeClassifier()
3 clftree.fit(d_train_att, d_train_label)
4 clftree.score(d_test_att, d_test_label)
```

Listing 4.17 kodingan praktek no. 5

```
In [11]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(d_train_att, d_train_label)
...: clftree.score(d_test_att, d_test_label)
Out[11]: 0.9142857142857143
```

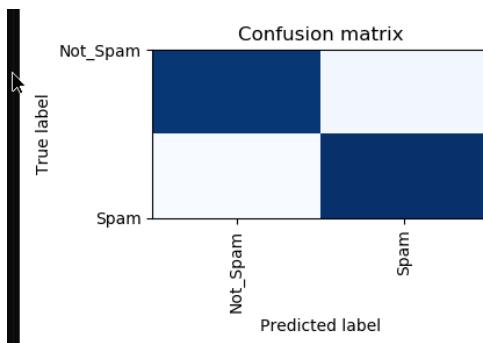
Gambar 4.193 hasil praktek soal no. 5

4.13.2.6 Plotlah confusion matrix dari praktek modul ini menggunakan *matplotlib*. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1 from sklearn.metrics import confusion_matrix
2 pred_labelstree = clftree.predict(d_test_att)
3 cmmtree = confusion_matrix(d_test_label , pred_labelstree)
4 cmmtree
5
6 import matplotlib.pyplot as plt
7 import itertools
8 def plot_confusion_matrix(cm, classes,
9                           normalize=False,
10                           title='Confusion matrix',
11                           cmap=plt.cm.Blues):
12     """
13     This function prints and plots the confusion matrix.
14     Normalization can be applied by setting 'normalize=True'.
15     """
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
18         print("Normalized confusion matrix")
19     else:
20         print('Confusion matrix, without normalization')
21
22     print(cm)
23
24     plt.imshow(cm, interpolation='nearest', cmap=cmap)
25     plt.title(title)
26     #plt.colorbar()
27     tick_marks = np.arange(len(classes))
28     plt.xticks(tick_marks, classes, rotation=90)
29     plt.yticks(tick_marks, classes)
30
31     fmt = '.2f' if normalize else 'd'
32     thresh = cm.max() / 2.
33
34     plt.tight_layout()
35     plt.ylabel('True label')
36     plt.xlabel('Predicted label')
37
38 types = pd.read_csv("classes.txt",sep='\s+', header=None,usecols
39                   =[1], names=['type'])
40 types = types['type']
41 types
42
43 import numpy as np
44 np.set_printoptions(precision=2)
45 plt.figure(figsize=(4,4), dpi=100)
46 plot_confusion_matrix(cmmtree, classes=types, normalize=True)
47 plt.show()
```

**Listing 4.18** kodingan praktek no. 6(1)



Gambar 4.194 hasil praktek soal no. 6(1)

Plot confusion matrix dari klasifikasi Decission Tree.

```

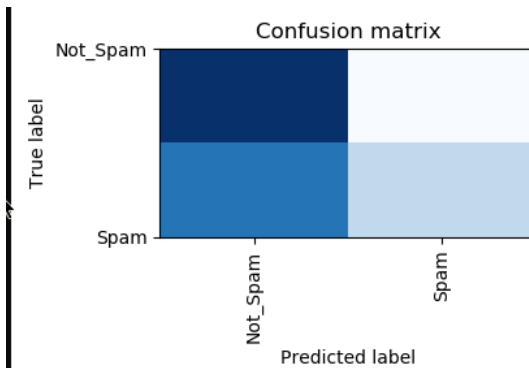
1 from sklearn.metrics import confusion_matrix
2 pred_labelssvm = clfsvm.predict(d_test_att)
3 cmsvm = confusion_matrix(d_test_label, pred_labelssvm)
4 cmsvm
5
6 import matplotlib.pyplot as plt
7 import itertools
8 def plot_confusion_matrix(cm, classes,
9                           normalize=False,
10                          title='Confusion matrix',
11                           cmap=plt.cm.Blues):
12 """
13 This function prints and plots the confusion matrix.
14 Normalization can be applied by setting 'normalize=True'.
15 """
16 if normalize:
17     cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
18     print("Normalized confusion matrix")
19 else:
20     print('Confusion matrix, without normalization')
21
22 print(cm)
23
24 plt.imshow(cm, interpolation='nearest', cmap=cmap)
25 plt.title(title)
26 #plt.colorbar()
27 tick_marks = np.arange(len(classes))
28 plt.xticks(tick_marks, classes, rotation=90)
29 plt.yticks(tick_marks, classes)
30
31 fmt = '.2f' if normalize else 'd'
32 thresh = cm.max() / 2.
33 #for i, j in itertools.product(range(cm.shape[0]), range(cm.
34 #shape[1])):
35 #    plt.text(j, i, format(cm[i, j], fmt),
36 #              horizontalalignment="center",
37 #              color="white" if cm[i, j] > thresh else "black"
38 #)

```

```

37
38     plt.tight_layout()
39     plt.ylabel('True label')
40     plt.xlabel('Predicted label')
41
42 types = pd.read_csv("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(cmsvm, classes=types, normalize=True)
50 plt.show()

```

**Listing 4.19** kodingan praktek no. 6(2)**Gambar 4.195** hasil praktek soal no. 6(2)

Plot confusion matrix dari klasifikasi SVM.

```

1 from sklearn.metrics import confusion_matrix
2 pred_labels = clf.predict(d_test_att)
3 cm = confusion_matrix(d_test_label, pred_labels)
4
5 import matplotlib.pyplot as plt
6 import itertools
7 def plot_confusion_matrix(cm, classes,
8                           normalize=False,
9                           title='Confusion matrix',
10                           cmap=plt.cm.Blues):
11     """
12     This function prints and plots the confusion matrix.
13     Normalization can be applied by setting `normalize=True`.
14     """
15     if normalize:
16         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
17         print("Normalized confusion matrix")
18     else:

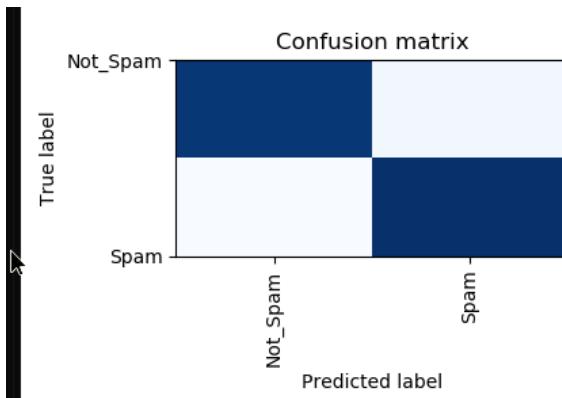
```

```

19     print('Confusion matrix, without normalization')
20
21     print(cm)
22
23     plt.imshow(cm, interpolation='nearest', cmap=cmap)
24     plt.title(title)
25     #plt.colorbar()
26     tick_marks = np.arange(len(classes))
27     plt.xticks(tick_marks, classes, rotation=90)
28     plt.yticks(tick_marks, classes)
29
30     fmt = '.2f' if normalize else 'd'
31     thresh = cm.max() / 2.
32     #for i, j in itertools.product(range(cm.shape[0]), range(cm.
33     #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
39     plt.tight_layout()
40     plt.ylabel('True label')
41     plt.xlabel('Predicted label')
42
43 types = pd.read_csv("classes.txt", sep='\s+', header=None, usecols
44     =[1], names=['type'])
45 types = types['type']
46 types
47
48 import numpy as np
49 np.set_printoptions(precision=2)
50 plt.figure(figsize=(4,4), dpi=100)
51 plot_confusion_matrix(cmtree, classes=types, normalize=True)
52 plt.show()

```

**Listing 4.20** kodingan praktek no. 6(3)



**Gambar 4.196** hasil praktek soal no. 6(3)

Plot confusion matrix dari klasifikasi Random Forest.

4.13.2.7 jalankan program cross validaiton pada bagian teori bab ini. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

1 from sklearn.model_selection import cross_val_score
2
3 scorestree = cross_val_score(clftree, d_train_att, d_train_label,
4     cv=5)
5 print("Accuracy: %0.2f (+/- %0.2f)" % (scorestree.mean(),
6     scorestree.std() * 2))
7
8 scoresvm = cross_val_score(clfsvm, d_train_att, d_train_label,
9     cv=5)
10 print("Accuracy: %0.2f (+/- %0.2f)" % (scoresvm.mean(),
11     scoresvm.std() * 2))
12
13 scores = cross_val_score(clf, d_train_att, d_train_label, cv=5)
14 print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()
15     () * 2))

```

**Listing 4.21** kodingan praktek no. 7

```

Accuracy: 0.92 (+/- 0.05)
Accuracy: 0.66 (+/- 0.05)
Accuracy: 0.94 (+/- 0.03)

```

**Gambar 4.197** hasil praktek soal no. 7

4.13.2.8 Buatlah program pengamatan komponen informasi pada bagian teori bab ini. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.

```

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(
4     n_estimators_opts),4), float)
5 i = 0
6 for max_features in max_features_opts:
7     for n_estimators in n_estimators_opts:
8         clf = RandomForestClassifier(max_features=max_features,
9             n_estimators=n_estimators)
10        scores = cross_val_score(clf, d_train_att, d_train_label,
11            cv=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

```

```
14     print("Max features: %d, num estimators: %d, accuracy: %.2f (+/- %.2f)" % (max_features, n_estimators, scores.mean(), scores.std() * 2))
```

**Listing 4.22** kodingan praktek no. 8

```
Max features: 5, num estimators: 10, accuracy: 0.89 (+/- 0.09)
Max features: 5, num estimators: 30, accuracy: 0.90 (+/- 0.07)
Max features: 5, num estimators: 50, accuracy: 0.91 (+/- 0.08)
Max features: 5, num estimators: 70, accuracy: 0.92 (+/- 0.07)
Max features: 5, num estimators: 90, accuracy: 0.93 (+/- 0.07)
Max features: 5, num estimators: 110, accuracy: 0.93 (+/- 0.07)
Max features: 5, num estimators: 130, accuracy: 0.94 (+/- 0.04)
Max features: 5, num estimators: 150, accuracy: 0.93 (+/- 0.05)
Max features: 5, num estimators: 170, accuracy: 0.93 (+/- 0.05)
Max features: 5, num estimators: 190, accuracy: 0.92 (+/- 0.06)
Max features: 10, num estimators: 10, accuracy: 0.92 (+/- 0.07)
Max features: 10, num estimators: 30, accuracy: 0.93 (+/- 0.04)
Max features: 10, num estimators: 50, accuracy: 0.94 (+/- 0.03)
Max features: 10, num estimators: 70, accuracy: 0.93 (+/- 0.05)
Max features: 10, num estimators: 90, accuracy: 0.93 (+/- 0.04)
Max features: 10, num estimators: 110, accuracy: 0.92 (+/- 0.06)
Max features: 10, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 10, num estimators: 150, accuracy: 0.94 (+/- 0.03)
Max features: 10, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 10, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 15, num estimators: 10, accuracy: 0.90 (+/- 0.05)
Max features: 15, num estimators: 30, accuracy: 0.93 (+/- 0.04)
Max features: 15, num estimators: 50, accuracy: 0.93 (+/- 0.05)
Max features: 15, num estimators: 70, accuracy: 0.92 (+/- 0.05)
Max features: 15, num estimators: 90, accuracy: 0.94 (+/- 0.04)
Max features: 15, num estimators: 110, accuracy: 0.94 (+/- 0.03)
Max features: 15, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 15, num estimators: 150, accuracy: 0.94 (+/- 0.03)
Max features: 15, num estimators: 170, accuracy: 0.94 (+/- 0.03)
Max features: 15, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 20, num estimators: 10, accuracy: 0.91 (+/- 0.05)
Max features: 20, num estimators: 30, accuracy: 0.91 (+/- 0.06)
Max features: 20, num estimators: 50, accuracy: 0.94 (+/- 0.04)
Max features: 20, num estimators: 70, accuracy: 0.92 (+/- 0.04)
Max features: 20, num estimators: 90, accuracy: 0.93 (+/- 0.04)
Max features: 20, num estimators: 110, accuracy: 0.93 (+/- 0.04)
Max features: 20, num estimators: 130, accuracy: 0.94 (+/- 0.03)
Max features: 20, num estimators: 150, accuracy: 0.93 (+/- 0.04)
Max features: 20, num estimators: 170, accuracy: 0.93 (+/- 0.04)
```

**Gambar 4.198** hasil praktek soal no. 8(1)

```

Max features: 25, num estimators: 150, accuracy: 0.93 (+/- 0.04)
Max features: 25, num estimators: 170, accuracy: 0.94 (+/- 0.03)
Max features: 25, num estimators: 190, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 10, accuracy: 0.93 (+/- 0.06)
Max features: 30, num estimators: 30, accuracy: 0.93 (+/- 0.05)
Max features: 30, num estimators: 50, accuracy: 0.95 (+/- 0.04)
Max features: 30, num estimators: 70, accuracy: 0.93 (+/- 0.04)
Max features: 30, num estimators: 90, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 110, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 130, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 150, accuracy: 0.94 (+/- 0.03)
Max features: 30, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 30, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 10, accuracy: 0.91 (+/- 0.05)
Max features: 35, num estimators: 30, accuracy: 0.94 (+/- 0.05)
Max features: 35, num estimators: 50, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 70, accuracy: 0.94 (+/- 0.03)
Max features: 35, num estimators: 90, accuracy: 0.94 (+/- 0.03)
Max features: 35, num estimators: 110, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 150, accuracy: 0.94 (+/- 0.03)
Max features: 35, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 35, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 40, num estimators: 10, accuracy: 0.91 (+/- 0.06)
Max features: 40, num estimators: 30, accuracy: 0.93 (+/- 0.04)
Max features: 40, num estimators: 50, accuracy: 0.94 (+/- 0.03)
Max features: 40, num estimators: 70, accuracy: 0.94 (+/- 0.03)
Max features: 40, num estimators: 90, accuracy: 0.94 (+/- 0.03)
Max features: 40, num estimators: 110, accuracy: 0.93 (+/- 0.05)
Max features: 40, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 40, num estimators: 150, accuracy: 0.93 (+/- 0.05)
Max features: 40, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 40, num estimators: 190, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 10, accuracy: 0.91 (+/- 0.06)
Max features: 45, num estimators: 30, accuracy: 0.94 (+/- 0.04)
Max features: 45, num estimators: 50, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 70, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 90, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 110, accuracy: 0.94 (+/- 0.03)
Max features: 45, num estimators: 130, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 150, accuracy: 0.93 (+/- 0.05)
Max features: 45, num estimators: 170, accuracy: 0.93 (+/- 0.04)
Max features: 45, num estimators: 190, accuracy: 0.93 (+/- 0.04)

```

**Gambar 4.199** hasil praktek soal no. 8(2)

#### 4.13.3 Penanganan Error

##### 1. FileNotFoundError

```

file "spamdata_1100/powers.csv", line 489, in
spamdata_1100/powers.loadheader_setattr_powers_csv"
[Errno 2] [Errno 2] File b'\\Desktop\\spamdata_1100\\powers\\powers.csv' does not exist: b'\\Desktop\\spamdata_1100\\powers\\powers.csv'

```

**Gambar 4.200** FileNotFoundError

##### 2. Cara Penangan Error

- FileNotFoundError

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

#### 4.13.4 Link Youtube

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

#### 4.14 Nurul Izza Hamka - 1174062

##### 4.14.1 Teori

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

Klasifikasi teks adalah sebuah proses dalam penentuan kategori suatu dokumen teks disertai dengan karakteristik teks itu sendiri.



**Gambar 4.201** Gambar Klasifikasi Data

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

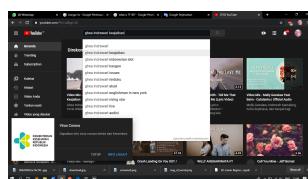
Karena Machine Learning tidak dapat menampilkan atau tidak dapat memberikan output sesuai dengan apa yang telah inputkan sebelumnya. hal ini terjadi karena mesin yang membeikan output atau yang biasa dikenal dengan noise. Contohnya adalah Ketika kita menginputkan sebuah label yang di dalamnya berupa Bungan, akan tetapi nantinya output yang di proses oleh mesin tersebut adalah labeh yang berbeda.



**Gambar 4.202** Gambar Proses Klasifikasi Bunga

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

Salah satu Teknik pembelajaran yang ada di youtube adalah dengan menggunakan keyword. Seperti halnya Ketika kita mau mencari sebuah video pada youtube, kita tinggal mengetikkan sebuah keyword maka selanjutnya akan di proses untuk menampilkan hasil sesuai dengan keyword yang kita inputkan tadi.



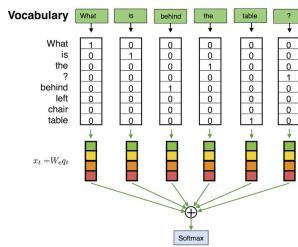
**Gambar 4.203** Gambar Machine Learning Di Youtube

4. Jelaskan apa itu vektorisasi data.

Vektorisasi adalah sebuah proses konversi data raster menjadi sebuah vector yang umum, ini disebut dengan digitalisasi sedangkan aktifitasnya ini disebut dengan istilah digitasi. Vektorisasi adalah proses mengubah algoritma dari operasi pada nilai tunggal pada suatu waktu untuk beroperasi pada satu set nilai pada satu waktu.

5. Jelaskan apa itu Bag of Words dengan kata-kata sederhana dan ilustrasi sendiri.

BOW adalah singkatan dari Bag of Words yang merupakan sebuah metode untuk mengekstrak sebuah fitur dari dokumen teks. Fitur ini dapat kita gunakan dalam pelatihan machine learning algorithm. Fitur yang menciptakan kosakata dari semua kata yang unik disemua dokumen.



**Gambar 4.204** Gambar Rumus Bag of Words

6. Jelaskan apa itu TF-IDF, ilustrasikan gambar sendiri.

TF-IDF adalah singkatan dari Term Frequency - Inverse Term Frequency. Bagian TF menghitung berapa kali suatu kata terjadi dalam korpus yang diberikan. Karena corpus terdiri dari banyak dokumen, setiap dokumen dan kata-katanya akan memiliki jumlah TF mereka sendiri. Bagian IDF menghitung seberapa jarang suatu kata muncul di dalam dokumen.

$$w_{i,j} = tf_{i,j} \times \log \left( \frac{N}{df_i} \right)$$

$tf_{ij}$  = number of occurrences of  $i$  in  $j$   
 $df_i$  = number of documents containing  $i$   
 $N$  = total number of documents

Gambar 4.205 Gambar TF-IDF

#### 4.14.2 Praktek Program

1. Aplikasi sederhana menggunakan pandas, membuat data dummy format csv sebanyak 500 baris dan melakukan load dataframe pandas.

```

1 #%% Soal 1
2
3 import pandas as pd #digunakan untuk mengimport library
4           pandas dengan alias pd
5 pd = pd.read_csv("D:/LEC/IT SMT VI/ARTIFICIAL INTELLIGENCE/
6 Chapter 4/src/csv_izza.csv") #membaca file csv

```

Gambar 4.206 Gambar Hasil Soal 1

2. Dari data frame No 1 dipecah menjadi dua dataframe yaitu 450 row pertama dan 50 row sisanya.

```

1 #%% Soal 2
2
3 d_train=pd[:450] #membagi data training menjadi 450
4 d_test=pd[450:] #membagi data menjadi 50 atau sisa dari data
5           yang tersedia

```

**Gambar 4.207** Gambar Hasil 450 Row Pertama

**Gambar 4.208** Gambar Hasil 50 Row Sisanya

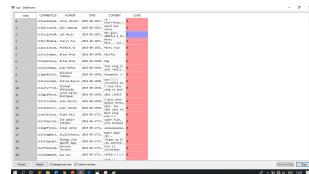
### 3. Vektorisasi dan klasifikasi data dengan Desicion Tree.

```
1 #%% Soal 3
2
3 import pandas as izza #untuk import library pandas berguna
4 untuk mengelola dataframe
5 izza = izza.read_csv("D:/LEC/IT SMT VI/ARTIFICIAL
6 INTELLIGENCE/Chapter 4/src/Youtube03-LMFAO.csv") #membaca
7 file dengan format csv
8
9 spam=izza.query('CLASS == 1') #membagi tabel spam
10 nospam=izza.query('CLASS == 0')#membagi tabel no spam
11
12 from sklearn.feature_extraction.text import CountVectorizer #
13 untuk import countvectorizer berfungsi untuk memecah data
14 tersebut menjadi sebuah kata yang lebih sederhana
15 vectorizer = CountVectorizer () #ntuk menjalankan fungsi
16 tersebut, pada code ini tidak ada hasilnya dikarenakan
17 spyder tidak mendukung hasil dari instasiasi.
18
19 dvec = vectorizer.fit_transform(izza[ 'CONTENT' ]) #untuk
20 melakukan pemecahan data pada dataframe yang terdapat pada
21 kolom konten
22 dvec #Untuk menampilkan hasil dari code sebelumnya
23
24 Daptarkata= vectorizer.get_feature_names()
25
26 dshuf = izza.sample(frac=1)
27
28 d_train=dshuf[:300]
29 d_test=dshuf[300:]
```

```

22 d_train_att = vectorizer.fit_transform(d_train[ 'CONTENT' ])
23 d_train_att
24
25 d_train_label=d_train[ 'CLASS' ]
26 d_test_label=d_test[ 'CLASS' ]

```



**Gambar 4.209** Gambar Hasil Soal 3

- Mengklasifikasikan data vektorisasi dengan klasifikasi SVM.

```

1 %% Soal 4
2
3 from sklearn import svm
4 clfsvm = svm.SVR(gamma = 'auto')
5 clfsvm.fit(d_train_att , d_train_label)

```

```

In [24]: from sklearn import svm
...: clfsvm = svm.SVR(gamma = 'auto')
...: clfsvm.fit(d_train_att , d_train_label)
Out[24]:
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='auto',
kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)

```

**Gambar 4.210** Gambar Hasil Soal 4

- Mengklasifikasi data vektorisasi dengan klasifikasi Desicion tree.

```

1 %%soal 5
2
3 from sklearn import tree
4 clftree = tree.DecisionTreeClassifier()
5 clftree.fit(d_train_att , d_train_label)

```

```

In [28]: from sklearn import tree
...: clftree = tree.DecisionTreeClassifier()
...: clftree.fit(d_train_att , d_train_label)
Out[28]:
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
min_samples_split=2, min_samples_leaf=1, min_impurity_decrease=0.0,
min_impurity_split=None, max_features=None, random_state=None,
min_weight_fraction_leaf=0.0, presort=False,
random_state=None, splitter='best')

```

**Gambar 4.211** Gambar Hasil Soal 5

- Plot confusion Matrix Menggunakan matplotlib.

```

1 #%%soal 6
2
3 from sklearn.metrics import confusion_matrix
4 pred_labels=clftree.predict(d_test)
5 cm=confusion_matrix(d_test_label ,pred_labels)
6
7 #%%
8
9 import matplotlib.pyplot as plt
10
11 def plot_confusion_matrix(cm, classes ,
12                           normalize=False ,
13                           title='Confusion matrix' ,
14                           cmap=plt.cm.Blues):
15
16     if normalize:
17         cm = cm.astype('float') / cm.sum(axis=1)[:, np.
18         newaxis]
19         print("Normalized confusion matrix")
20     else:
21         print('Confusion matrix, without normalization')
22
23     print(cm)

```

```

In [27]: import matplotlib.pyplot as plt
...:
...: def plot_confusion_matrix(cm, classes,
...:                          normalize=False,
...:                          title='Confusion matrix',
...:                          cmap=plt.cm.Blues):
...:
...:     if normalize:
...:         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
...:         print("Normalized confusion matrix")
...:     else:
...:         print('Confusion matrix, without normalization')
...:
...:     print(cm)

```

Gambar 4.212 Gambar Hasil Soal 6

## 7. Jalankan cross validation

```

1 #%%soal 7
2
3 from sklearn.model_selection import cross_val_score
4
5 scores=cross_val_score(clftree ,d_train_att ,d_train_label ,cv
6     =5)
7 skor_rata2=scores .mean()
8 skoresd=scores .std()

```

```

In [28]: from sklearn.model_selection import cross_val_score
...:
...: scores=cross_val_score(clftree ,d_train_att ,d_train_label ,cv=5)
...:
...: skor_rata2=scores.mean()
...: skoresd=scores.std()

```

Gambar 4.213 Gambar Hasil Soal Nomor 7

- Membuat program pengamatan komponen informasi .

```

1 #%%soal 8
2
3 max_features_opts = range(5, 50, 5) #max_features_opts
4     sebagai variabel untuk membuat range 5,50,5
5 n_estimators_opts = range(10, 200, 20) #n_estimators_opts
6     sebagai variabel untuk membuat range 10,200,20
7 rf_params = izza.empty((len(max_features_opts)*len(
8     n_estimators_opts),4), float) #rf_params sebagai variabel
9     untuk menjumlahkan yang sudah di tentukan sebelumnya
10 i = 0
11 for max_features in max_features_opts: #pengulangan
12     for n_estimators in n_estimators_opts: #pengulangan
13         clftree = RandomForestClassifier(max_features=
14             max_features, n_estimators=n_estimators) #menampilkan
15             variabel csf
16             scores = cross_val_score(clf, df_train_att,
17             df_train_label, cv=5) #scores sebagai variabel training
18             rf_params[i,0] = max_features #index 0
19             rf_params[i,1] = n_estimators #index 1
20             rf_params[i,2] = scores.mean() #index 2
21             rf_params[i,3] = scores.std() * 2 #index 3
22             i += 1 #dengan ketentuan i += 1
23             print("Max features: %d, num estimators: %d, accuracy
24 : %0.2f (+/- %0.2f)" %(max_features, n_estimators, scores.
25 mean(), scores.std() * 2))
26             #print hasil pengulangan yang sudah ditentukan
27

```

#### 4.14.3 Penanganan Error

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

FileNotFoundException: [Error 2] File 'F:/Semester 6/Artificial Intelligence/Tugas 4/src/fanny.csv' does not exist: 'F:/Semester 6/Artificial Intelligence/Tugas 4/src/fanny.csv'

**Gambar 4.214** Gambar Error

#### 4.14.4 Bukti Tidak Plagiat

- Berikut adalah gambar bukti tidak melakukan Plagiat



Gambar 4.215 Gambar Bukti Tidak Plagiat

#### 4.14.5 Link Youtube

1. Berikut adalah lampiran Link Youtube untuk Chapter 4  
<https://www.youtube.com/watch?v=zX3W99EB2ks>



## BAB 5

---

# CHAPTER 5

---

### 5.1 1174006 - Kadek Diva Krishna Murti

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

```
1 @inproceedings{awangga2017colenak,
2   title={Colenak: GPS tracking model for post-stroke
3   rehabilitation program using AES-CBC URL encryption and QR-
4   Code},
5   author={Awangga, Rolly Maulana and Fathonah, Nuraini Siti and
6   Hasanudin, Trisna Irmayadi},
7   booktitle={Information Technology, Information Systems and
8   Electrical Engineering (ICITISEE), 2017 2nd International
   conferences on},
9   pages={255--260},
10  year={2017},
11  organization={IEEE}
12 }
```



**Gambar 5.1** Kecerdasan Buatan.

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

### 5.1.1 Teori

### 5.1.2 Praktek

### 5.1.3 Penanganan Error

### 5.1.4 Bukti Tidak Plagiat



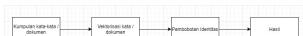
**Gambar 5.2** Kecerdasan Buatan.

## 5.2 1174066 - D.Irga B. Naufal Fakhri

### 5.2.1 Teori

#### 5.2.1.1 Jelaskan kenapa kata-kata harus di lakukan vektorisasi

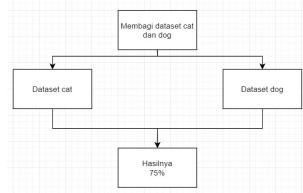
Kata kata harus dilakukan vektorisasi dikarenakan atau bertujuan utk mengukur nilai kemunculan suatau kata yang serupa dari sebuah kalimat sehingga kata-kata tersebut dapat di prediksi kemunculannya. atau juga di buatnya vektorisasi data digunakan untuk memprediksi bobot dari suatu kata misalkan ayam dan kucing sama-sama hewan maka akan dibuat prediksi apakah kata tersebut akan muncul pada kalimat yang kira-kira memiliki bobot yang sama.



**Gambar 5.3** Teori 1

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

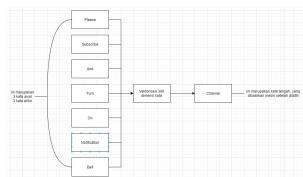
Karena setiap objek memiliki identitas khusus, misalnya sederhana. Dalam dataset Google ini memiliki 3 objek, kucing, anjing, dan ember yang disetuju. Kemudian dari masing-masing objek dibandingkan dataset antara kucing dan anjing kemudian kucing dan ember. Hasil yang diperoleh untuk kucing dan anjing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



**Gambar 5.4** Teori 2

### 5.2.1.3 Jelaskan konsep vektorisasi untuk kata

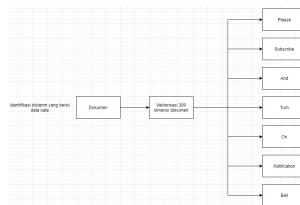
Konsep untuk vektorisasi kata sama dengan masukan atau input pada kata-kata di mesin pencari. Maka anggaplah itu akan dikeluarkan sebagai saran tentang kata tersebut. Jadi kata data diperoleh dari hasil yang diolah dalam kalimat sebelumnya yang telah diproses. Contoh sederhana dalam kalimat berikut (Please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



**Gambar 5.5** 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 5.6 Teori 4

### 5.2.1.5 Jelaskan apa mean dan standar deviasi

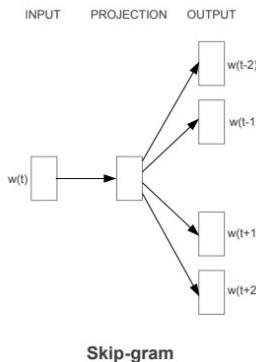
Mean adalah nilai rata-rata. Untuk mendapatkan makna ini, kita hanya perlu menambahkan data yang tersedia dan kemudian membaginya dengan jumlah data. Sedangkan standar deviasi adalah nilai statistik yang digunakan untuk menentukan bagaimana data didistribusikan dalam sampel, dan menentukan titik data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

<b>Mean</b> $\bar{X} = \frac{x_1 + x_2 + \dots + x_n}{n}$ atau $\bar{X} = \frac{\sum_{i=1}^n X_i}{n}$	$\bar{X}$ = rata-rata $\sum_{i=1}^n X_i$ = jumlah seluruh nilai data $n$ = 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 5.7 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 5.8** 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', 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:

```

[In 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)

```

**Gambar 5.9** 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:

**Gambar 5.10** Data love

```
[Out[5]: model_dir.get("fatin")]
array([ 0.00000000e+00, -1.04509100e-05,  0.19511500e-05,  0.13476052e-05,  0.14604840e-05,
       0.12896200e-05,  0.82457500e-05,  0.10351500e-05,  0.12207901e-05,  0.27748400e-05,
       0.06646025e-05,  0.18545112e-05, 0.16116502e-05,  0.21679688e-05,  0.27748400e-05,
       0.32801215e-05,  0.18492112e-05, 0.36132812e-05,  0.19351235e-05,  0.18104840e-05,
       0.15337125e-05,  0.18492112e-05, 0.18492112e-05,  0.18492112e-05,  0.18492112e-05,
       0.05957031e-05,  0.22949209e-05, 0.09804246e-05,  0.21811616e-05,  0.10582707e-05,
       0.13201215e-05,  0.18492112e-05, 0.18492112e-05,  0.18492112e-05,  0.18492112e-05,
       0.3125e-05,  0.06646025e-05, 0.18765468e-05,  0.17657501e-05, 0.01870884e-05,  0.16468240e-05,
       0.02583800e-05,  0.16380920e-05, 0.13183500e-05,  0.08087172e-05,  0.13859300e-05,
       0.27539062e-05, 0.06646025e-05, 0.10351502e-05, 0.20248444e-05, 0.08087172e-05,
       0.11133821e-05, 0.05858477e-05, 0.12225589e-05, 0.08914453e-05, 0.08087172e-05,
       0.02303154e-05, 0.04820315e-05, 0.16815625e-05, 0.04159389e-05, 0.08087172e-05,
       0.02008071e-05, 0.04199219e-05, 0.05034901e-05, 0.22734735e-05, 0.09138070e-05,
       0.17382612e-05, 0.22460913e-05, 0.03646679e-05, 0.19824219e-05, 0.08087172e-05,
       0.15337125e-05, 0.18492112e-05, 0.18492112e-05, 0.18492112e-05, 0.18492112e-05,
       0.13579421e-05, 0.30878125e-05, 0.06898539e-05, 0.08087172e-05, 0.09357500e-05,
       0.15247415e-05, 0.32603125e-05, 0.23820312e-05, 0.08087172e-05, 0.09357500e-05,
       0.20808071e-05, 0.34378125e-05, 0.23820312e-05, 0.08087172e-05, 0.09357500e-05,
       0.29296875e-05, 0.25595580e-05, 0.09849286e-05, 0.140625e-05, 0.08087172e-05,
       0.21502831e-05, 0.28177480e-05, 0.02929687e-05, 0.28051951e-05, 0.34969890e-05,
       0.25188487e-05, 0.28177480e-05, 0.02929687e-05, 0.28051951e-05, 0.34969890e-05,
       0.05658846e-05, 0.18498847e-05, 0.04878515e-05, 0.15936769e-05, 0.07748400e-05,
       0.04839081e-05, 0.22469128e-05, 0.0328125e-05, 0.15936769e-05, 0.05807187e-05,
       0.03673679e-05, 0.15234375e-05, 0.07519531e-05, 0.137429e-05, 0.04296487e-05,
       0.02655262e-05, 0.18492112e-05, 0.02225825e-05, 0.226125e-05, 0.04532296e-05,
       0.01888889e-05, 0.18492112e-05, 0.02225825e-05, 0.226125e-05, 0.04532296e-05])
```

**Gambar 5.11** Data faith

```
In [6]: model_dirge['fall']
Out[6]:
array([-0.04772461,  0.19742188, -0.09977344,  0.16894531, -0.12261527,
       -0.18693359,  0.84521289,  0.01964297,  0.14648438,  0.15830862,
      -0.08591486,  0.84492188,  0.015874,  0.88651406, -0.15824219,
      -0.18693359,  0.13971875,  0.01977344,  0.16894531, -0.12261527,
     -0.1815625,  0.13971875,  0.09228516, -0.12199375,  0.12895312,
     0.03417699,  0.21869373,  0.01977344,  0.16894531, -0.12261527,
     0.04199219, -0.29828212, -0.18554688,  0.08486894, -0.02087482,
     0.18261719,  0.15136719,  0.0445557,  0.04463904,  0.18498594,
     -0.19359378,  0.0546877,  0.0495895,  0.03737575, -0.32358989,
     0.18261719,  0.15136719, -0.0044557,  0.04463904,  0.27539862,
     -0.0255227,  0.06278828,  0.07080878, -0.07617183,  0.05642969,
     0.01672363,  0.04711514,  0.19628986, -0.08984375,  0.078125,
     0.05424869,  0.0813168,  0.12988211,  0.03279586,  0.14594781,
     0.08542486,  0.0813168,  0.12988211,  0.03279586,  0.14594781,
     0.08448889, -0.14108159,  0.0451816, -0.078125,  0.06988589,
     0.26757912,  0.02891953, -0.12895312, -0.04482812,  0.18945312,
     0.18261719,  0.15136719,  0.0445557,  0.04463904,  0.18498594,
     -0.14941686, -0.02331543, -0.03958978, -0.10489391, -0.14161856,
     -0.18261719,  0.03876722,  0.04598844, -0.209625, -0.35340809,
     0.18261719,  0.15136719,  0.0445557,  0.04463904,  0.18498594,
     0.0429875,  0.09983281,  0.08056641, -0.06268832,  0.12255895,
     0.08087112,  0.08482178, -0.1646625, -0.03274844,  0.0701125,
     0.07958984, -0.1269625,  0.0187983, -0.17773434,  0.061203945,
     -0.0255227,  0.06278828,  0.07080878, -0.07617183,  0.05642969,
     -0.14453125, -0.1885855, -0.0853775,  0.18269875, -0.03987475,
     0.04853394,  0.08135514, -0.15828312, -0.09130859, -0.12255895,
     -0.0255227,  0.06278828,  0.07080878,  0.11797579,  0.12895312,
```

Gambar 5.12 Data fall

```
In [7]: model_dirge['sick']
Out[7]:
array([-0.82171886e-01,  1.49414862e-01, -4.05273439e-02,  1.64862500e-01,
       -5.5976525e-01,  3.2238525e-01, -1.73828125e-01, -1.47469518e-01,
       -0.12261527,  0.12988211,  0.03279586,  0.14594781,  0.06988589,
       0.26757912,  0.02891953, -0.12895312, -0.04482812,  0.18945312,
       0.18261719,  0.15136719,  0.0445557,  0.04463904,  0.18498594,
       -0.14941686, -0.02331543, -0.03958978, -0.10489391, -0.14161856,
       -0.18261719,  0.03876722,  0.04598844, -0.209625, -0.35340809,
       0.18261719,  0.15136719,  0.0445557,  0.04463904,  0.18498594,
       0.0429875,  0.09983281,  0.08056641, -0.06268832,  0.12255895,
       0.08087112,  0.08482178, -0.1646625, -0.03274844,  0.0701125,
       0.07958984, -0.1269625,  0.0187983, -0.17773434,  0.061203945,
       -0.0255227,  0.06278828,  0.07080878, -0.07617183,  0.05642969,
       -0.14453125, -0.1885855, -0.0853775,  0.18269875, -0.03987475,
       0.04853394,  0.08135514, -0.15828312, -0.09130859, -0.12255895,
     -0.0255227,  0.06278828,  0.07080878,  0.11797579,  0.12895312,
```

Gambar 5.13 Data sick

```
In [8]: model_dirge['clear']
Out[8]:
array([-0.4148025e-04, -1.43695915e-01, -1.48418052e-01, -4.24846588e-02,
       -0.17890508,  0.84688179e-01,  1.76739121e-01, -1.46468375e-01,
       2.26562380e-01,  9.76562380e-01,  2.67578125e-01, -1.2983215e-01,
      1.24511739e-01,  2.23632812e-01, -2.13987188e-01,  3.19855354e-02,
      2.32256525e-01,  3.14453125e-01,  1.18116406e-01,  8.0071259e-02,
      2.7652125e-02,  1.13959358e-01, -1.78717938e-01, -1.46468375e-01,
      5.6679875e-02, -6.91339868e-01, -1.76719358e-01, -1.46468375e-01,
      9.1915125e-02,  1.07915156e-01,  1.10151562e-01, -0.3466938e-02,
      1.4082175e-02,  1.07915156e-01,  1.10151562e-01, -0.3466938e-02,
      2.55599375e-01, -7.12098025e-02,  2.83293125e-01, -2.75598055e-01,
      -4.069075e-02,  2.00087512e-01, -2.00087512e-01, -5.0571094e-02,
      2.7734375e-02,  1.32012598e-01,  1.32012598e-01, -0.3466938e-02,
      -4.94146602e-02,  3.67769531e-02,  1.91692508e-01,  2.7734375e-01,
      4.66380594e-02, -5.1778125e-02,  1.6389538e-01,  1.74780459e-02,
      2.01171875e-02, -2.01171875e-01,  1.50750552e-02,  2.61718750e-01,
      1.1585244e-01,  1.2946075e-01, -0.63627344e-01,  4.68754000e-01,
      1.1585244e-01,  1.2946075e-01, -0.63627344e-01,  4.68754000e-01,
      5.76913281e-02, -1.04988469e-01,  1.6985512e-01, -0.80731250e-02,
      -0.81171875e-01,  1.06248632e-01,  1.06248632e-01, -0.81171875e-01,
      -0.81171875e-01,  1.06248632e-01,  1.06248632e-01, -0.81171875e-01,
      7.1299025e-02,  4.37911719e-02,  2.05978215e-01,  5.71219902e-02,
      8.4087112e-02,  1.53129312e-01,  1.53129312e-01, -0.81171875e-01,
      2.4884675e-01, -6.5429875e-02, -2.02836712e-02,  1.52534779e-01,
      -5.7421675e-01, -0.82124803e-02, -2.00087512e-01, -5.0571094e-02,
      2.7734375e-02,  1.32012598e-01,  1.32012598e-01, -0.3466938e-02,
      -4.94146602e-02,  3.67769531e-02,  1.91692508e-01,  2.7734375e-01,
      4.66380594e-02, -5.1778125e-02,  1.6389538e-01,  1.74780459e-02,
      2.01171875e-02, -2.01171875e-01,  1.50750552e-02,  2.61718750e-01,
      1.1585244e-01,  1.2946075e-01, -0.63627344e-01,  4.68754000e-01,
      1.1585244e-01,  1.2946075e-01, -0.63627344e-01,  4.68754000e-01,
      5.76913281e-02, -1.04988469e-01,  1.6985512e-01, -0.80731250e-02,
      -0.81171875e-01,  1.06248632e-01,  1.06248632e-01, -0.81171875e-01,
      7.1299025e-02,  4.37911719e-02,  2.05978215e-01,  5.71219902e-02,
      8.4087112e-02,  1.53129312e-01,  1.53129312e-01, -0.81171875e-01,
      -2.22256525e-01, -1.43594688e-01,  8.30878125e-03, -9.81453112e-02,
      3.55599375e-01, -1.54312598e-01,  3.45915012e-02, -1.04988469e-01,
      1.1585244e-02, -2.7734375e-01,  2.60970254e-02,  3.02734375e-01,
```

Gambar 5.14 Data clear

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

```
Out[9]: array([-0.13482344,  0.25976562, -0.15017969, -0.27734375,  0.30273435,
       0.09969838,  0.39257812, -0.22549219, -0.18539575,  0.3671875,
       -0.18582734,  0.13671875,  0.2598625,  0.87128986,  0.28539862,
       0.24121894,  0.13671875,  0.13671875,  0.13671875,  0.13671875,
       -0.85883789,  0.0612793, -0.01549818,  0.87617188,  0.05128359,
       0.26031531,  0.38889538,  0.08162586, -0.056922597,  0.14648458,
       0.24121894,  0.13671875,  0.13671875,  0.13671875,  0.13671875,
       -0.24121894, -0.18945132, -0.15234375, -0.05493164,  0.81434326,
       0.24023486,  0.36132062, -0.12648057,  0.13671875,  0.13671875,
       -0.0246556,  0.32226562,  0.11579655,  0.18164602,  0.04931644,
       -0.14257812,  0.13671875,  0.13671875,  0.13671875,  0.13671875,
       -0.01567188,  0.26989458,  0.07801328, -0.0859375,  0.05955988,
       -0.14257812, -0.149625,  0.08927344, -0.14455125,  0.359375 ,
       0.24121894,  0.13671875,  0.13671875,  0.13671875,  0.13671875,
       0.04768742,  0.08537391, -0.04722461,  0.05908283,  0.72128986,
       0.0153775, -0.11621954,  0.07128986,  0.01468899, -0.18644531,
       0.0153775, -0.11621954,  0.07128986,  0.01468899,  0.0153775,
       0.3359975, -0.1875,  0.14589781,  0.08843867,  0.87617188,
       -0.09521484,  0.08843867,  0.26117188,  0.11238469,  0.33984775,
       0.0153775,  0.11621954,  0.08843867,  0.08843867,  0.0153775,
       -0.22949219,  0.14944486, -0.1953125,  0.08496904,  0.80753784,
       0.0153775,  0.11621954,  0.08843867,  0.08843867,  0.12618888,
       -0.08740344,  0.18683089, -0.1147648,  0.06516485,  0.12618888,
       0.11238469,  0.08808109,  0.08591406,  0.03988594,  0.0802935 ,
       0.08823975, -0.15234375,  0.19042369,  0.04988469,  0.25 ,
       -0.007171661,  0.04589844,  0.05182359,  0.04952754,  0.05491111,
       0.08823975, -0.15234375,  0.19042369,  0.04988469,  0.05491111,
       0.02091416,  0.26367188, -0.28718938,  0.08963281, -0.99802831,
       -0.03125, -0.13769531, -0.05737305,  0.38867188, -0.421875,
```

Gambar 5.15 Data shine

```
In [11]: model_dirgs['car']
```

```
Out[11]: array([-0.13482343,  0.00942358,  0.03344727, -0.95631769,  0.04839396,
       0.1257775, -0.04931641, -0.16945351, -0.20894855,  0.11962051,
       0.1866446, -0.25 , -0.16945351, -0.19742188, -0.01879883,
       0.05208193, -0.06210757,  0.06445312,  0.14453125, -0.04518016,
       0.11767578, -0.04492675,  0.17285155,  0.04394531, -0.23867575,
       0.0153775,  0.11621954,  0.08843867,  0.08843867,  0.0153775,
       0.84467773, -0.15527544,  0.2598625,  0.33984375,  0.00756836,
       -0.09521484,  0.08843867,  0.08843867,  0.08843867,  0.09521484,
       -0.89912189,  0.15698396,  0.08847861, -0.19495112,  0.0232031,
       0.0534643, -0.03963965,  0.11639664,  0.24121894, -0.234375 ,
       0.11238469,  0.08808109,  0.08591406,  0.03988594,  0.0802935 ,
       0.26010531,  0.37695312, -0.12558599,  0.11429781,  0.17675781,
       0.18000766,  0.0039304,  0.26757812,  0.20171754,  0.07109038,
       0.0153775,  0.11621954,  0.08843867,  0.08843867,  0.0153775,
       0.26171875, -0.08542578,  0.02583801, -0.05834961,  0.08787354,
       0.11767578, -0.04492675,  0.17285155,  0.04394531, -0.23867575,
       0.0153775,  0.11621954,  0.08843867,  0.08843867,  0.0153775,
       0.32631788, -0.04492188, -0.11429781,  0.22851562,  0.01647949,
       0.11238469,  0.08808109,  0.08591406,  0.03988594,  0.0802935 ,
       0.1815625,  0.08831771,  0.18793815, -0.24689375, -0.189375 ,
       -0.09375, -0.01623355,  0.20214544,  0.23144531,  0.05444336,
       0.08823975, -0.02779996,  0.20102156,  0.02329278,  0.03125 ,
       -0.17578125, -0.12558599, -0.05493164, -0.17582122,  0.28515625,
       0.11238469,  0.08808109,  0.08591406,  0.03988594,  0.0802935 ,
       0.00769843,  0.20597912, -0.01789894, -0.12988281,  0.04711914,
       0.0153775,  0.11621954,  0.08843867,  0.08843867,  0.0153775,
       0.84477773, -0.15527544,  0.2598625,  0.33984375,  0.00756836,
       -0.09521484,  0.08843867,  0.08843867,  0.08843867,  0.09521484,
       -0.09326172,  0.15828112, -0.16598396, -0.06654688,  0.19435594,
       -0.08823975, -0.02779996,  0.20102156,  0.02329278,  0.03125 ,
       -0.07324219,  0.16895156,  0.04588398, -0.17675781, -0.37709862,
       0.22558594,  0.16380594,  0.05102539, -0.08251593,  0.07958964,
```

Gambar 5.16 Data bag

```
In [12]: model_dirgs['wash']
```

```
Out[12]: array([ 4.0694422e-03, -0.141081562e-01, -5.46879890e-02,  1.34765625e-01,
       -3.3828159e-01,  3.24218759e-01, -8.44728562e-02, -1.29832612e-01,
       1.29832612e-01, -0.129832612e-01,  0.129832612e-01, -0.129832612e-01,
       -2.79541810e-02,  2.80987812e-01, -4.27246894e-02, -0.95468758e-01,
       -7.42187500e-02,  3.04867590e-01,  2.11014682e-01, -8.8867175e-02,
       2.11014682e-01, -0.211014682e-01,  0.211014682e-01, -0.211014682e-01,
       -0.29826112e-02,  0.62189755e-01,  1.9359975e-01,  0.217285156e-02,
       -0.29826112e-02, -0.80871550e-02,  1.0000000e+00,  0.217285156e-02,
       -0.80871550e-02, -3.80899755e-02,  1.0000000e+00, -1.398485156e-01,
       1.74804688e-01,  0.78749735e-02, 1.11238125e-01,  1.05998902e-01,
       -1.05998902e-01,  0.105998902e-01,  0.105998902e-01, -0.105998902e-01,
       -0.87712500e-02, -3.80899755e-02, 1.0000000e+00, -1.398485156e-01,
       1.74804688e-01,  0.78749735e-02, 1.11238125e-01,  1.05998902e-01,
       -1.05998902e-01,  0.105998902e-01,  0.105998902e-01, -0.105998902e-01,
       -0.43750000e-02, -1.02895781e-01,  3.80899755e-02, -5.05179104e-02,
       2.84423282e-02, -2.29492188e-01, -0.21586488e-01,  4.21875000e-02,
       -5.6445312e-02,  1.77734755e-01,  0.127285156e-01,  0.171939758e-02,
       -0.171939758e-02,  0.171939758e-01,  0.171939758e-01, -0.171939758e-01,
       4.21875000e-01,  5.32226562e-02, -3.9278125e-01,  1.74804688e-01,
       1.0591053e-02, -2.05978125e-02,  2.21797886e-01,  3.1859975e-01,
       -0.10591053e-01,  0.10591053e-01,  0.10591053e-01, -0.10591053e-01,
       1.5886713e-02,  8.39078125e-02,  1.6894312e-01,  2.79541816e-02,
       1.4092175e-02, -2.16984312e-01,  0.14092175e-01, -0.14092175e-01,
       1.69677734e-02, 1.69921757e-01, -1.46484756e-02,  2.65625000e-01,
       2.17205356e-02, 1.12984060e-02, -1.14257812e-01,  7.22652590e-02,
       -0.14257812e-01,  0.14257812e-01,  0.14257812e-01, -0.14257812e-01,
       -0.98144531e-02, 5.44433994e-02, 3.79468750e-01, 5.62500000e-01,
       -5.62500000e-01, 0.56250000e-01, 0.56250000e-01, -0.56250000e-01,
       -5.3320312e-01, -4.37580000e-01, 2.5975525e-01, -1.49414682e-01,
       5.66480258e-02, 2.13867188e-02, -2.86865324e-02, -1.70859843e-01,
       3.20734750e-02, -2.05978125e-02, 2.21797886e-01, 3.1859975e-01,
       -0.17743754e-02, 2.41692108e-02, -1.77734750e-01, 2.71484375e-01,
```

Gambar 5.17 Data car

**Gambar 5.18** Data wash

**Gambar 5.19** Data motor

```
In [15]: model_dirga.similarity('wash', 'clear')
Out[15]: 0.09019176
```

**Gambar 5.20** 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 5.21** Data wash dan clear

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

**Gambar 5.22** Data bag dan love

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

**Gambar 5.23** Data sick dan faith

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

**Gambar 5.24** 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 5.25** 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 Brajmusti, adalah nama aku"
.... print("Faktanya: " + test_string)
.... res = re.findall(r'\w+', test_string)
.... print("The list of words is : " + str(res))
Faktanya: Dirga Brajmusti, adalah nama aku
The list of words is : ['Dirga', 'Brajmusti', 'adalah', 'nama', 'aku']
```

**Gambar 5.26** Nomor 2-1

### 5.2.2.3 Nomor 3

```
1 from gensim.models.doc2vec import Doc2Vec, TaggedDocument
2 ## Example 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 [8]: from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from collections import deque
import random
import numpy as np
import string
import re
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')

# doc = ["I love machine learning",
#        "I love coding in python",
#        "This is a good pc",
#        "This is a good net",
#        "Hello, my name is Gagan"]
# tokenized_doc = []
# tokenized_doc.append([word for word in doc])

# print(tokenized_doc)

# [I love machine learning', 'I love coding in python', 'This is a good pc', 'This is a good net', 'Hello, my name is Gagan']

# print(deque(tokenized_doc))

# deque([['I love machine learning', 'I love coding in python', 'This is a good pc', 'This is a good net', 'Hello, my name is Gagan'], ['I love machine learning', 'I love coding in python', 'This is a good pc', 'This is a good net', 'Hello, my name is Gagan'], ['I love machine learning', 'I love coding in python', 'This is a good pc', 'This is a good net', 'Hello, my name is Gagan'], ['I love machine learning', 'I love coding in python', 'This is a good pc', 'This is a good net', 'Hello, my name is Gagan'], ['I love machine learning', 'I love coding in python', 'This is a good pc', 'This is a good net', 'Hello, my name is Gagan']])
```

**Gambar 5.27** Nomor 3

```

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

```

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

# Create Doc2Vec documents
docs = ["I love machine learning",
        "I love this machine learning",
        "This is a good pc",
        "This is a good mac",
        "I don't like windows"]
tagged_docs = []
for i, doc in enumerate(docs):
    tagged_docs.append(TaggedDocument(doc, [i]))
print(tagged_docs)

# Train the 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 Doc2Vec model
print(model.infer_vector(["I", "love", "this", "mac"]))

# Print the first 5 words
print(model.wv.most_similar("love"))
```

**Gambar 5.28** 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
6 # from IMDB
7 for dirname in ["train/pos", "train/neg", "train/unsup", "test/
8 pos", "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, [
16                     dirname + "/" + 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 5.29 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)
...+
...+ @membership_data
...+ model.delete_temporary_training_data(keep_inference=True)
```

**Gambar 5.30** 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')
...+
...+ @memberships
...+ model.delete_temporary_training_data(keep_inference=True)
2020-04-05 21:20:22,528 : INFO : Saving DocVec object under dirga.d2v, separately None
2020-04-05 21:20:22,682 : INFO : saved dirga.d2v
```

**Gambar 5.31** 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([-0.82322483, -0.0122747, -0.00295202, -0.00356266, -0.02363259,
       -0.00529805,  0.00301624, -0.00417492, -0.00891156,  0.00759805,
       0.01598096, -0.00897159, -0.01558069,  0.01741202,  0.01701254,
       0.0122115,  0.00433158,  0.00034627,  0.01111711,  0.0147115],  
dtype='float32')
```

Gambar 5.32 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 5.33 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 0.08022311 0.03531764]
```

Gambar 5.34 Nomor 9

### 5.2.3 Penanganan Error

- Error

## 1. FileNotFoundException

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

Gambar 5.35 Error 1

- Solusi

## 1. FileNotFoundException

Perhatikan letak file ada dimana, pastikan path telah benar

### 5.2.4 Bukti Tidak Plagiat



Gambar 5.36 Bukti Tidak Plagiat

### 5.2.5 Link Youtube

<https://youtu.be/AtLNMwxCUSk>

## 5.3 1174083 - Bakti Qilan Mufid

Chapter 5 - Vektorisasi data dan dokumen

### 5.3.1 Teori

#### 5.3.1.1 Jelaskan kenapa kata-kata dilakukan vektorisasi. dilengkapi dengan ilustrasi atau gambar

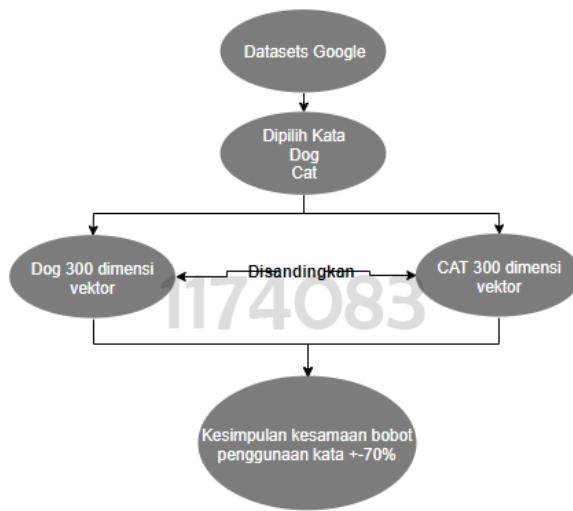
Kenapa kata-kata harus dilakukan vektorisasi karena bertujuan untuk melakukan perhitungan atau prediksi seberapa sering kata tersebut muncul. serta bertujuan untuk mengconvert data agar data tersebut dapat terbaca oleh machine learning, karena sistem tidak dapat membaca data secara langsung.



**Gambar 5.37** gambaran penjelasan no. 1

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

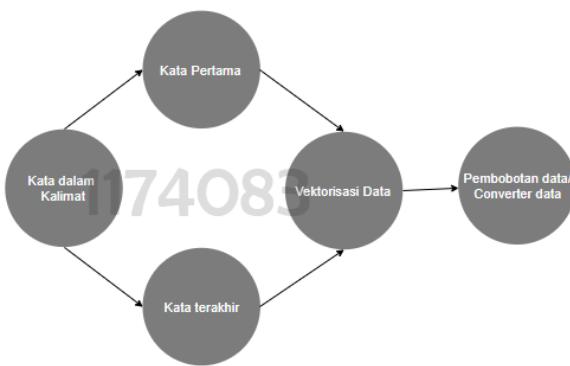
Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap kata, misalkan terdapat kata dog dan cat pada dataset google tersebut, lalu setiap kata tersebut dibuat dimensi vektor 300 untuk kata dog dan 300 dimensi vektor juga untuk kata cat kemudian kata tersebut dibandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70% kesamaan bobot dikarenakan kata dog dan cat sama-sama digunakan untuk hewan peliharaan.



**Gambar 5.38** gambaran penjelasan no. 2

### 5.3.1.3 Jelaskan konsep vektorisasi untuk kata dilengkapi dengan ilustrasi atau gambar

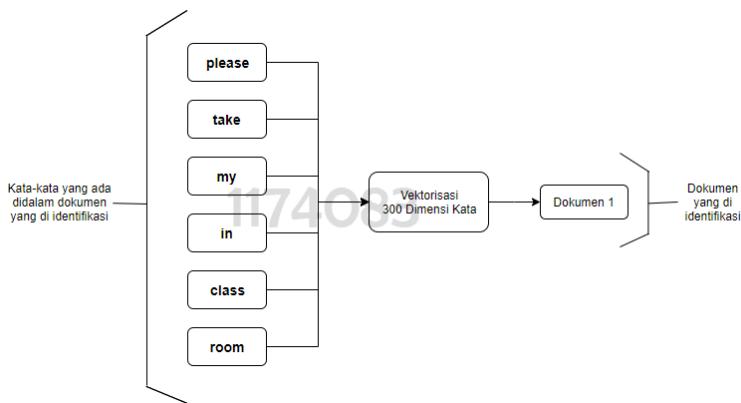
Vektorisasi untuk kata bertujuan untuk menganalisa data yang dimiliki, selain itu juga untuk mengetahui kata utama, kata tengah atau objek utama pada hal ini sangat berkaitan dengan dimensi vektor pada dataset google yang 300 tadi. Karena untuk mendapatkan nilai atau bobot dari kata tengah tersebut didapatkan dari proses dimensiasi dari kata tersebut.



**Gambar 5.39** gambaran penjelasan no. 3

#### 5.3.1.4 Jelaskan konsep vektorisasi untuk dokumen dilengkapi dengan ilustrasi atau gambar

Vektorisasi untuk dokumen hampir sama seperti vektorisasi untuk kata hanya saja berbeda pada pemilihan kata utama atau kata tengah yang terdapat pada satu dokumen. Jadi mesin akan membuat dimensi vektor 300 untuk dokumen dan nanti kata tengahnya akan di sandingkan pada dokumen yang terdapat pada dokumen tersebut.

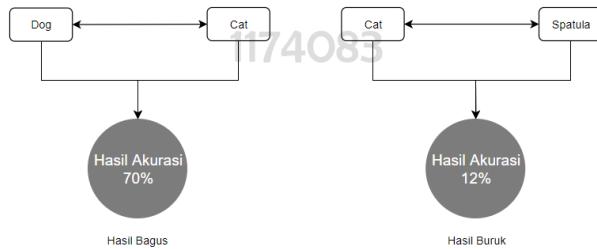


**Gambar 5.40** gambaran penjelasan no. 4

#### 5.3.1.5 Jelaskan apa mean dan standar deviasi dilengkapi dengan ilustrasi atau gambar

Mean merupakan nilai rata-rata dari beberapa data dimana ditentukan dengan membagi data tersebut. sedangkan standar defiation merupakan standar untuk menimbang kesalahan. Sehingga kesalahan tersebut di anggap wajar misalkan kita memperkirakan kedalaman dari dataset merupakan 2 atau 3

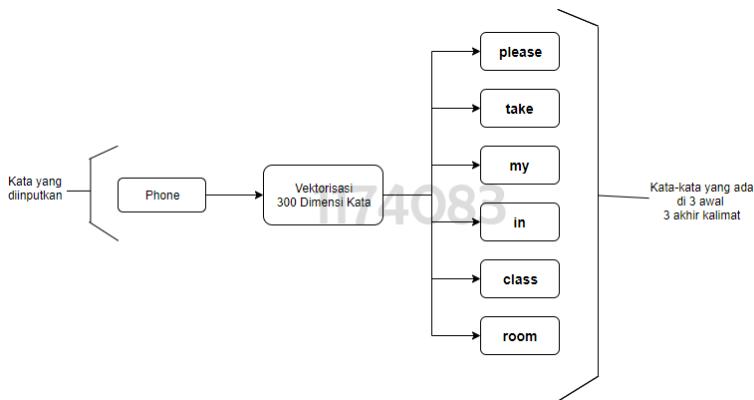
tapi pada kenyataanya merupakan 5 itu merupakan kesalahan tapi masih bisa dianggap wajar karena masih mendekati perkiraan awal.



**Gambar 5.41** gambaran penjelasan no. 5

#### 5.3.1.6 Jelaskan apa itu skip-gram dilengkapi dengan ilustrasi atau gambar

Skip-Gram merupakan teknik yang digunakan di area speech processing, dimana kebalikan dari konsep vektorisasi untuk kata. Dimana kata tengah menjadi acuan terhadap kata-kata pelengkap dalam suatu kalimat.



**Gambar 5.42** gambaran penjelasan no. 6

#### 5.3.2 Praktek

##### 5.3.2.1 Cobalah dataset google, dan jelaskan vektor dari kata love, faith, fall, sick, clear, shine, bag, car, wash, motor, cycle dan cobalah untuk melakukan perbandingan similaritas dari masing-masing kata tersebut. jelaskan arti dari outputan similitaritas dan setiap baris kode yang dibuat(harus beda dengan teman sekelas). (Nilai 5 untuk setiap perbandingan, disini ada 5 perbandingan similaritas)

Kode di atas 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
2
3 logging.basicConfig(format='%(asctime)s : %(levelname)s : %(
4     message)s', level=logging.INFO)
baktiqilan_model = gensim.models.KeyedVectors.
    load_word2vec_format('GoogleNews-vectors-negative300.bin',
        binary=True, limit=500000)
```

**Listing 5.1** kodingan praktek no. 1

```

In [3]: import gensim, logging
...:
...: logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
level=logging.INFO)
...: baktiqilan_model = gensim.models.KeyedVectors.load_word2vec_format('E:/
backup/sem 6/Kecerdasan Buatan/KB3C - Copy/src/1174083/src5/GoogleNews-vectors-
negative300.bin', binary=True, limit=500000)
2020-04-06 01:42:57,914 : INFO : loading projection weights from E:/backup/sem 6/
Kecerdasan Buatan/KB3C - Copy/src/1174083/src5/GoogleNews-vectors-negative300.bin
2020-04-06 01:43:25,755 : INFO : loaded (500000, 300) matrix from E:/backup/sem 6/
Kecerdasan Buatan/KB3C - Copy/src/1174083/src5/GoogleNews-vectors-negative300.bin
```

**Gambar 5.43** Hasil Praktek no 1

```

1 baktiqilan_model[ 'love' ]
```

ini untuk menampilkan data hasil vektorisasi data dari kata love, hasilnya ialah sebagai berikut :

```
In [1]: baktiqilan_model['love']
Out[1]:
array([ 0.10302734, -0.15234375,  0.02587891,  0.16503906, -0.16503906,
       0.06689218,  0.29298675, -0.26367188, -0.140625 ,  0.20117188,
       -0.02624512, -0.08283125, -0.02770996, -0.04394531, -0.23535156,
       0.16991218,  0.12898632,  0.15720556,  0.08658396, -0.06682422,
       0.12898632,  0.15720556,  0.08658396,  0.15720556,
       0.07515625,  0.05517578,  0.10693359,  0.11181641,  0.16308594,
       -0.11181641,  0.13964844,  0.01556396,  0.12792969,  0.15426688,
       0.07714844,  0.26171875,  0.08642578, -0.02514648,  0.33398438,
       0.18652344,  0.20996094,  0.07080878,  0.02608098, -0.10644531,
       0.10253966,  0.12384688,  0.04711914,  0.02269473,  0.03584961,
       0.10986328,  0.14941486, -0.18693359,  0.01556396,  0.08894375,
       0.12898632,  0.15720556,  0.08658396,  0.15720556,
       0.24315495,  0.08447266,  0.07080878,  0.18966496,  0.03515625,
       -0.09566796,  0.21972656, -0.09328064, -0.03198242,  0.18457031,
       0.28515625, -0.08593575, -0.11116141,  0.0213623 ,  0.30664662,
       -0.09225352, -0.18945312,  0.01513672,  0.18554688,  0.34375 ,
       -0.31054688,  0.22558594,  0.08748234, -0.22656125,  0.29494218,
       0.08825125, -0.38470588,  0.08748234,  0.08748234,  0.08748234,
       0.09003352,  0.12033012,  0.17071094,  0.12451253,  0.17071094,
       0.11767578,  0.19726562,  0.03466797, -0.10400391,  0.1640625 ,
       0.19726562,  0.19824219,  0.09521484,  0.09561523,  0.12597656,
       0.00073624, -0.0402832 ,  0.03663965,  0.01624565,
       -0.22167969,  0.171875 ,  0.12081179,  0.01965332,  0.4435125 ,
       0.06494414,  0.05932617,  0.01367188,  0.18495312,  0.06494414,
       0.08600625, -0.08593575,  0.09521484,  0.09561523,  0.04212875,
       0.131640625,  0.15344668,  0.02355957,  0.16992188,  0.06682422,
       0.140625 ,  0.13135394,  0.12729269,  0.12666547,  0.05883789,
       -0.00055695,  0.05761719, -0.08474266,  0.16992188,  0.13671875,
       0.09375 ,  0.08055641,  0.04080396, -0.03759766,  0.26367188,
       0.00662231, -0.01928711,  0.09423828, -0.13183594, -0.27929688,
       0.27734375,  0.10855894,  0.114245781, -0.27734375,
       0.11181641,  0.08622462,  0.08622462,  0.08622462,
       -0.07763672,  0.15102539,  0.08176758,  0.04421875, -0.08676719,
       0.15039062,  0.13671875,  0.15039062,  0.11914662,
       0.06000585, -0.03960625,  0.10839844,  0.02095273,  0.02331543,
       0.13183594,  0.01080977,  0.03149464, -0.12597656, -0.13671875,
       -0.30664662,  0.28515625,  0.09653281,  0.00564575,  0.08089438,
       -0.16016525,  0.14453125,  0.18261719,  0.16899766,  0.04345763,
       0.10464521,  0.18039896,  0.06258806,  0.1793125 ,  0.15282832,
       0.1129 ,  0.171579 ,  0.09505462,  0.09505325,  0.21210605],
```

Gambar 5.44 Hasil Praktek no 1.1

```
In [1]: baktiqilan_model['faith']
```

ini untuk menampilkan data hasil vektorisasi data dari kata faith, hasilnya ialah sebagai berikut :

```
In [3]: baktiqilan_model['faith']
Out[3]:
array([ 0.26367188, -0.04150391,  0.1953125 ,  0.13476562, -0.14648438,
       0.11962891,  0.03445703,  0.12081179,  0.13476562,
       0.06646025,  0.18945132, -0.16601562,  0.21679688, -0.21714843,
       0.3020125 ,  0.10449219,  0.36132812, -0.1953125 , -0.18164662,
       0.15333201,  0.10839844,  0.01367188,  0.23144531,
       0.09003352,  0.12033012,  0.08002424,  0.08002424,
       0.1320125 ,  0.22948475,  0.02355957,  0.02111815,  0.18554688,
       0.04125977,  0.17081179,  0.17488469,  0.01780894, -0.1640625 ,
       0.3125 ,  0.06646025,  0.02770996, -0.10265625, -0.140625 ,
       -0.02819824,  0.01257324, -0.09521484, -0.18066466, -0.140625 ,
       -0.02258301,  0.16308594, -0.13183594, -0.080087812,  0.13085938,
       0.27539062, -0.20605469,  0.05191562,  0.20214844, -0.1875 ,
       0.16902188,  0.13571875,  0.13769532,  0.16308594, -0.03806536,
       0.11181641,  0.04248475,  0.08391125,  0.16015625,  0.04150391,
       0.03331543,  0.04248475,  0.08391125,  0.16015625,  0.04150391,
       0.16016525,  0.13671875,  0.09613411,  0.32617168,  0.08215953,
       -0.20808731,  0.04199219,  0.05834961, -0.27734375,  0.09130859,
       -0.17382812,  0.03466797,  0.17842219, -0.08837891,
       0.1835975 ,  0.07324219,  0.11717875, -0.33984375,  0.16796875,
       0.13574219, -0.30078125,  0.00469971,  0.06005895, -0.29296875,
       0.15233201,  0.08296563,  0.08296563,  0.28320312,  0.08296563,
       0.20260563,  0.18039896,  0.03424668,  0.03322456,
       0.29296875,  0.15585938,  0.090430298,  0.148625 ,  0.05851047,
       0.212582031,  0.0291748 ,  0.02295688,  0.20019531,  0.34909938,
       0.10449219, -0.01940918,  0.04077148 ,  0.32226562, -0.1953125 ],
```

Gambar 5.45 Hasil Praktek no 1.2

```
In [1]: baktiqilan_model['fall']
```

ini untuk menampilkan data hasil vektorisasi data dari kata fall, hasilnya ialah sebagai berikut :

```
In [4]: baktiqilan_model['fall']
Out[4]:
array([-0.04272461,  0.10742188, -0.09277344,  0.16894531, -0.132815,
       -0.10693359,  0.04321289,  0.01904297,  0.14648438,  0.15839062,
       -0.08691496,  0.04492188,  0.01458574,  0.08691496, -0.19824219,
       -0.11139251,  0.13679259,  0.08356556,  0.012019375, -0.1269312,
       -0.03417969,  0.2109375,  0.01977529,  0.01544189,
       -0.01932512,  0.13679259,  0.08356556,  0.012019375, -0.1269312,
       -0.03417969,  0.2109375,  0.01977529,  0.01544189,
       -0.26953125, -0.0988877, -0.07763672, -0.15527344, -0.03393555,
       -0.04199219, -0.29882812, -0.18554686, -0.08496964, -0.02087402,
       -0.13574219, -0.22558954,  0.33789862, -0.03564453, -0.10839844,
       -0.19335938,  0.0546875,  0.04596055,  0.3671875, -0.03295898,
       -0.18269508, -0.15136719, -0.0445557,  0.04083968,  0.27539862,
       -0.0808625,  0.1484751,  0.01768984, -0.08789862,
       -0.0808625,  0.1484751,  0.01768984, -0.08789862,
       -0.0808625,  0.1484751,  0.01768984, -0.08789862,
       -0.0808625,  0.1484751,  0.01768984, -0.08789862,
       -0.0808625,  0.1484751,  0.01768984, -0.08789862,
       -0.18261719,  0.03076173,  0.04559844, -0.03540839,
       -0.12890625, -0.10595783,  0.17578125,  0.06683453,  0.34560938,
       -0.04296875,  0.09863281, -0.00505641, -0.02698028,  0.12255859,
       -0.02343575, -0.06494141,  0.06967969, -0.04589844,  0.04956055,
       -0.08086712, -0.04082178, -0.1648625,  0.0783125 ,
```

Gambar 5.46 Hasil Praktek no 1.3

```
1 baktiqilan_model[ 'sick' ]
```

ini untuk menampilkan data hasil vektorisasi data dari kata sick, hasilnya adalah sebagai berikut :

```
In [5]: baktiqilan_model['sick']
Out[5]:
array([ 1.8261718e-01,  1.49414062e-01, -4.05273438e-02,  1.64062500e-01,
       -2.59765625e-01,  3.22265625e-01,  1.73828125e-01, -1.47466938e-01,
       1.01074219e-01,  5.46875000e-02,  1.66992188e-01, -1.68945312e-01,
       2.2430419e-03,  9.66796785e-02, -1.66605625e-01, -1.12304688e-01,
       1.04862500e-01,  2.17968750e-01,  2.17968750e-01, -1.04862500e-01,
       8.74002434e-02,  2.56347852e-01,  3.417968750e-01, -2.56347852e-01,
       1.7871098e-01,  9.19321289e-04,  0.88671875e-02, -1.95311525e-01,
       1.81646625e-01, -2.65625000e-01,  1.98974609e-02, -6.39648438e-02,
       9.42362812e-02, -3.12500000e-02,  1.98974609e-02, -6.39648438e-02,
       1.18652344e-01,  1.23046675e-01,  0.03027344e-02,  4.68750000e-01,
       1.38806250e-01,  1.04904169e-01,  1.66992188e-01, -1.73828125e-01,
       5.79512312e-02,  1.04904169e-01,  1.66992188e-01, -0.08781250e-02,
       2.0117875e-01,  1.06201172e-01, -1.29882812e-01, -1.25976562e-01,
       5.56646625e-02,  3.14453125e-01,  5.61523438e-02, -1.20117188e-01,
       7.12896625e-02,  4.37011719e-02,  2.05078125e-01,  5.71289062e-02,
       8.44726562e-02,  2.15820312e-01, -1.26953125e-01,  8.78906250e-02,
       -2.40693750e-01,  4.03120938e-02,  1.73828125e-01, -1.68945312e-01,
       -3.57421375e-01,  3.03120938e-02,  2.08007812e-01, -0.03119048e-02,
       2.81984242e-02,  1.73828125e-01, -2.08007812e-01, -9.3261719e-02,
       -6.49414062e-01,  3.63769531e-02,  1.91462500e-01, -2.77343750e-01,
       3.54003966e-02,  1.56250000e-01, -2.03857422e-02,  2.26562500e-01,
       -4.66308594e-02,  5.15781250e-01, -2.17480464e-02,
       2.0117875e-01,  2.81171875e-01, -1.59756836e-01,  2.61718750e-01,
       1.11000000e-01,  1.59756836e-01, -2.17480464e-02,
       4.19921875e-01,  6.88476562e-02,  9.42362812e-02, -1.96289062e-01,
       -9.4238212e-02,  3.12500000e-02,  6.34765625e-02,  2.47882734e-02,
```

Gambar 5.47 Hasil Praktek no 1.4

```
1 baktiqilan_model[ 'clear' ]
```

ini untuk menampilkan data hasil vektorisasi data dari kata clear, hasilnya adalah sebagai berikut :

```
In [6]: baktiqilan_model['clear']
Out[6]:
array([-2.44148625e-04, -1.02050781e-01, -1.40414862e-01, -4.24806882e-02,
       -1.67068750e-01, -1.46484757e-01,  1.76757812e-01,  1.46484375e-01,
       2.26562500e-02,  9.76562500e-01, -2.67578125e-01, -1.29882812e-01,
      1.24511719e-01,  2.23628128e-01, -2.13867188e-01,  3.10858594e-02,
      2.00195312e-01, -6.11801562e-02,  6.76074219e-02, -1.21093750e-01,
      3.02025000e-01, -1.13180156e-02,  3.11801562e-01, -1.21093750e-01,
      2.75578125e-02, -6.64248847e-03,  7.03465862e-01, -7.73515625e-01,
      6.69768875e-02, -4.01333088e-03, -1.76710938e-01, -1.40388059e-03,
      7.91015265e-02,  0.07910156e-01, -8.34960938e-02,
      1.98074609e-02, -3.14311055e-03,  1.30615234e-02,  3.34472656e-02,
      2.55859375e-01,  7.12890625e-02,  2.83283125e-01, -2.75398625e-01,
      8.49946844e-02,  0.08499468e-01, -9.01601562e-02,  5.00488281e-02,
      3.49112094e-02,  7.32421875e-02, -1.17773438e-01,  5.00488281e-02,
      7.47078312e-02, -1.25000000e-01, -5.77370469e-02, -2.74658203e-02,
      -1.37329102e-02, -2.13867188e-01, -1.12729695e-01, -4.98468756e-02,
      1.92382812e-01,  1.32812500e-01,  5.00488281e-01, -1.66015265e-01,
      1.57470783e-02, -1.37695132e-01,  3.89183000e-02, -1.57226562e-01,
      1.52379062e-01, -1.09414862e-01, -2.01623000e-01, -3.01298625e-01,
      1.55778125e-01,  1.65639862e-01, -6.65992188e-01, -1.09414862e-02,
      2.85126930e-02,  2.69531250e-01, -2.01283125e-01, -2.41210938e-01,
      1.39160156e-02,  5.12695132e-02,  9.75562500e-02,  3.14941406e-02,
      2.51464844e-02, -2.85162500e-01, -6.88476562e-02,
      -5.29795156e-02,  2.06064688e-02,  0.07031250e-01, -1.60156250e-01,
      2.61236940e-02, -3.01513672e-02,  8.65699219e-03, -1.30859537e-01,
      3.88183594e-02,  0.68957038e-03,  9.31085859e-03, -6.05468750e-02,
      dtype=float32)
```

Gambar 5.48 Hasil Praktek no 1.5

```
In [7]: baktiqilan_model[ 'shine' ]
```

ini untuk menampilkan data hasil vektorisasi data dari kata shine, hasilnya adalah sebagai berikut :

```
In [7]: baktiqilan_model['shine']
Out[7]:
array([-0.12402344,  0.25976962, -0.15017969, -0.27723475,  0.30272348,
       0.09098093,  0.39327812, -0.22942210, -0.18359375,  0.36718775,
       -0.18360734,  0.13671875,  0.25398625,  0.07128986,  0.02539862,
       0.21777344,  0.24023475,  0.5234375,  0.12304688, -0.19335938,
       -0.05887389,  0.06127933, -0.01940918,  0.07617188,  0.05102539,
       0.28013789,  0.06127933, -0.01940918,  0.07617188,  0.05102539,
       -0.34765625,  0.028563477, -0.23925781, -0.04516602, -0.00479126,
       -0.24161094, -0.18195375, -0.05102539,  0.01877188,  0.00187126,
       0.39602375,  0.20379735,  0.14842812,  0.01877188,  0.00187126,
       0.24023475,  0.36123812, -0.12793699,  0.10595703,  0.09912189,
       -0.02465828,  0.11379653,  0.18164062,  0.04013144,
       -0.10253906, -0.00283813,  0.02882812, -0.1717875, -0.18945312,
       -0.01367188, -0.20898438, -0.07631322, -0.0859375,  0.05395986,
       -0.14257812, -0.140625,  0.03027344, -0.14453125,  0.359375,
       0.16113281,  0.22265625,  0.265625, -0.06347656, -0.02887617,
       0.04706781,  0.18164062,  0.09912188,  0.01877188,  0.00187126,
       0.1515075, -0.11621094,  0.07128986,  0.04044531,  0.00444531,
       0.08886719,  0.11523438,  0.09667969, -0.11083984,  0.16015625,
       0.3359375, -0.1875,  0.14558781,  0.00463867,  0.07617188,
       -0.09521484,  0.08474266,  0.19513203,  0.08496094, -0.00753784,
       -0.25390625,  0.05200195,  0.27539062, -0.08398438, -0.31054688,
       0.22942912,  0.14914406, -0.19513203,  0.08496094, -0.00753784,
       0.078125,  0.18598823,  0.02355957,  0.08347656,  0.32617188,
       -0.06740254,  0.10658594, -0.11474869, -0.18164062,  0.13378906,
       0.11230469, -0.080660109,  0.0891406,  0.058860594,  0.03086233,
       dtype=float32)
```

Gambar 5.49 Hasil Praktek no 1.6

```
In [8]: baktiqilan_model[ 'bag' ]
```

ini untuk menampilkan data hasil vektorisasi data dari kata bag, hasilnya adalah sebagai berikut :

```
In [8]: baktiqilan_model['bag']
Out[8]:
array([-0.03515625,  0.15234375, -0.12402344,  0.13378906, -0.11328125,
       -0.0133667, -0.16113281,  0.14648438, -0.06835938,  0.140625,
       -0.06008589, -0.3046875,  0.29969694, -0.04345763, -0.2109375,
       -0.05701703,  0.05953774,  0.05953774,  0.05953774, -0.08424285,
       -0.15849256,  0.15849256,  0.15849256,  0.15849256,  0.15849256,
       -0.1894512,  0.11323812,  0.27539862, -0.2877109,
       -0.01866641,  0.06689453,  0.257815,  0.03247097, -0.24699375,
       -0.05541992,  0.01013184,  0.24121094, -0.21875,  0.07568359,
       -0.09814458, -0.16113281,  0.16503908, -0.08521484, -0.16661562,
       -0.19824219,  0.05517578,  0.04268765,  0.18017423,  0.07342419,
       -0.12831125,  0.12831125,  0.12831125,  0.12831125,  0.12831125,
       -0.13941496,  0.1484375,  0.09619141,  0.21777344, -0.08544972,
       -0.02818284,  0.02539462,  0.03759765,  0.23242188,  0.19628906,
       -0.27539062,  0.00130859,  0.23730469,  0.09033203, -0.28515625,
       -0.05932617,  0.06591797,  0.01794434, -0.08055313, -0.1796875,
       -0.05615234, -0.12207681,  0.09863281, -0.05786133, -0.09373,
       -0.30274348, -0.0396484,  0.00744629, -0.17871094,  0.08544922,
       -0.2610046,  0.03780828,  0.08022085, -0.14451975, -0.14451975,
       -0.03231125,  0.12831125,  0.12831125,  0.12831125,  0.12831125,
       -0.1484375,  0.05566406,  0.23442338,  0.07515051,  0.21484375,
       -0.15722656,  0.3559375,  0.04736328, -0.04848584, -0.19726562,
       -0.27929688,  0.05566406,  0.10085594,  0.08611768, -0.26703125,
       -0.03295898, -0.14550781,  0.15917969,  0.16503966,  0.234375,
       -0.03588867,  0.04296875, -0.25,  0.1117875, -0.07714844,
       -0.00521851,  0.125,  0.08886719,  0.15527344, -0.02185069,
```

Gambar 5.50 Hasil Praktek no 1.7

```
1 baktiqilan_model[ 'bag' ]
```

ini untuk menampilkan data hasil vektorisasi data dari kata car, hasilnya ialah sebagai berikut :

```
In [9]: baktiqilan_model['car']
Out[9]:
array([ 0.13085938,  0.00842285,  0.03346727, -0.05883789,  0.04003906,
       -0.0133667, -0.16113281,  0.14648438, -0.06835938,  0.140625,
       -0.15865606, -0.35,  0.10480391,  0.10745958, -0.01030853,
       -0.05200195, -0.00216675,  0.06445312,  0.14453125, -0.04541916,
       -0.16113281, -0.01611328,  0.08838079,  0.08844726,  0.16210938,
       -0.04462773,  0.16503906,  0.03231125,  0.03231125,  0.03231125,
       -0.25585938, -0.01733398,  0.25598625,  0.33984375, -0.00756836,
       -0.09912199,  0.16503906,  0.03231125,  0.03231125,  0.03231125,
       -0.05346406, -0.03063965,  0.11883984,  0.24101094, -0.234375,
       -0.11767578,  0.04246257,  0.02258301, -0.05834961, -0.00767354,
       -0.20018531,  0.17652913,  0.12375899,  0.11204571,  0.17652913,
       -0.19809765,  0.0830365,  0.26757812,  0.20111768,  0.03710838,
       -0.11083984, -0.09814453, -0.3125,  0.03515625,  0.02832031,
       -0.26171875, -0.08642578, -0.02258301, -0.05834961, -0.00767354,
       -0.11767578, -0.04246257, -0.17285156,  0.04394531, -0.23046875,
       -0.1648625, -0.11474689, -0.06830273,  0.01196289, -0.24707931,
       -0.32617188, -0.04492188, -0.07407781,  0.07407781, -0.01047549,
       -0.15231662,  0.15231662,  0.15231662,  0.15231662,  0.15231662,
       -0.1015238,  0.08817871,  0.10791815,  0.24689375, -0.189375,
       -0.09375, -0.01623535,  0.20214844,  0.23144531, -0.05444336,
       -0.05541992, -0.20089438,  0.26757812,  0.27929688,  0.17068944,
       -0.17578125, -0.02770996,  0.204180156,  0.02392578,  0.03125,
       -0.25390625, -0.125,  -0.05491654, -0.17382812,  0.28515625,
       -0.23242188,  0.0234375,  0.020117186, -0.13476562,  0.26571188,
       -0.00769043,  0.08597812,  0.01708984, -0.12988281,  0.04711914,
       -0.22070312,  0.02099609,  0.29101562, -0.02893066,  0.17285156,
```

Gambar 5.51 Hasil Praktek no 1.8

```
1 baktiqilan_model[ 'car' ]
```

ini untuk menampilkan data hasil vektorisasi data dari kata wash, hasilnya ialah sebagai berikut :

```
In [10]: baktiqilan_model['wash']
Out[10]:
array([-9.4684922e-03, 1.416101562e-01, -5.46875000e-02, 1.34765625e-01,
       -2.38281250e-01, 3.42421875e-01, -8.44726562e-02, -1.29882812e-01,
       -1.07918156e-01, 2.53996259e-01, -1.13925391e-02, -1.66992188e-01,
       -2.79945125e-01, 2.04675825e-01, -4.19084686e-02, -1.11914862e-01,
       -7.43157812e-02, 5.84675825e-01, -1.08525391e-02, -1.55715782e-01,
       -2.67578125e-01, 2.12898625e-01, -2.02941895e-02, 1.39359375e-01,
       -6.29882812e-02, 1.62190375e-01, 1.93359375e-01, -2.17285156e-02,
       -2.6702800e-03, -9.13085938e-02, -1.00997656e-01, 2.3623812e-01,
       -8.00781250e-02, -3.08859375e-01, -1.39648438e-01,
       -1.74684453e-01, -2.40894453e-02, 1.06195312e-01, -5.03027244e-01,
       -3.4375900e-01, -1.28059781e-01, -3.88859375e-01, -5.05371994e-02,
       -5.07812500e-02, 2.15597812e-01, 2.81250000e-01, 7.03125000e-02,
       -2.84423282e-02, -2.2942188e-01, -5.81054688e-02, 4.51668156e-02,
       -3.56445312e-02, 1.77734375e-01, -1.2297312e-01, 3.71093750e-02,
       -1.06905125e-02, -2.52581250e-01, -1.08525391e-02, -1.74084580e-01,
       -1.21875800e-01, 5.32226562e-02, -3.175278125e-01, 1.74084580e-01,
       -1.77081953e-02, 2.05878125e-02, -1.26179686e-01, 3.18359375e-01,
       -1.08398438e-01, -4.30237852e-03, -2.45117188e-02, -2.08898437e-01,
       -3.588678125e-02, 8.30878125e-02, -1.68945312e-01, -2.79541810e-02,
       -1.04984469e-01, -3.47656259e-01, -5.26019531e-02, 2.24699375e-01,
       -1.07918156e-02, -2.40894453e-02, 1.06195312e-01, -5.03027244e-01,
       -2.17285156e-02, 1.13204688e-02, -1.14257812e-01, 7.22558259e-02,
       -4.32128900e-02, 1.11694336e-02, 3.0734736e-04, -7.91015625e-02,
       -5.98144531e-02, -5.44433594e-02, 3.73046875e-01, 5.62500000e-02,
       dtype=float32)
```

Gambar 5.52 Hasil Praktek no 1.9

```
1 baktiqilan_model[ 'wash' ]
```

ini untuk menampilkan data hasil vektorisasi data dari kata motor, hasilnya adalah sebagai berikut :

```
In [11]: baktiqilan_model['motor']
Out[11]:
array([ 5.73730469e-02, 1.503201562e-01, -1.32812500e-01,
       -2.59765625e-01, 1.77734375e-01, 3.68652344e-02, -4.37500000e-01,
       -2.34375000e-02, 2.57812500e-01, 1.74884686e-01, 2.44146625e-02,
       -2.51953125e-01, -5.76171875e-02, 8.15429688e-01, 1.86767578e-02,
       -3.83306781e-02, 1.98859375e-01, -2.11312500e-01, -2.11312500e-02,
       -1.23944785e-01, -2.46094688e-02, 1.13125000e-01, -2.11312500e-02,
       -1.55573438e-01, -1.55573438e-01, 4.50884375e-01,
       -3.02734375e-01, 1.53320312e-01, -1.69921075e-01, -1.01074219e-01,
       -3.26538086e-03, 2.28515625e-02, 8.98437500e-02, -7.12896256e-02,
       -1.54296875e-01, 8.88671875e-02, 5.61523430e-03,
       -4.46777312e-02, -3.06646886e-02, 7.42910156e-02, 5.58087500e-01,
       -1.30307354e-02, -1.30307354e-02, -3.14477375e-02, -1.30307354e-02,
       -3.10858594e-02, 6.58024414e-03, 4.02382812e-02,
       -2.78320312e-02, 1.97421875e-02, 1.47469938e-01, 2.80761719e-02,
       -1.50396625e-01, -1.37695125e-01, 9.96903750e-02, 1.28906250e-01,
       -3.34472656e-02, -1.08832227e-02, -2.14843750e-01, -9.52148438e-02,
       -6.30944686e-02, 7.39584375e-02, -3.06086250e-02, 2.10773250e-01,
       -2.21579062e-02, -3.3308430e-02, -9.05371875e-02, -1.37695125e-01,
       -1.53320312e-01, -7.12896256e-02, -3.14477375e-02, 7.66691562e-02,
       -8.00781250e-02, -1.14257812e-01, -9.71167950e-01, -2.61718750e-01,
       -3.84765625e-01, -1.87500000e-01, -1.10331562e-01, 1.00585938e-01,
       -1.08398438e-01, -1.87500000e-01, -1.91598391e-02,
       -2.40234375e-01, 8.34968938e-02, 4.19218750e-02, -1.91598391e-02,
       -9.73307354e-02, -2.18875000e-02, -4.08875000e-02, -1.91598391e-02,
       -8.20312500e-02, -3.32031250e-01, 3.27148438e-02, 5.71289062e-02,
       dtype=float32)
```

Gambar 5.53 Hasil Praktek no 1.10

```
1 baktiqilan_model[ 'motor' ]
```

ini untuk menampilkan data hasil vektorisasi data dari kata cycle, hasilnya adalah sebagai berikut :

```
In [12]: baktiqilan_model['cycle']
Out[12]:
array([ 0.04541016,  0.21679688,  0.02709961,  0.12353516, -0.20703135,
       -0.1328125 ,  0.26367188,  0.12898625, -0.125 ,  0.15332031,
      -0.18261719, -0.15820312,  0.06176758,  0.21972656, -0.15820312,
       0.02563477, -0.07568359,  0.0625 ,  0.04614258,  0.10354688,
      -0.13397996, -0.11666222,  0.33975776,  0.14941534,  0.08447266,
       0.07568359,  0.04614258,  0.10354688,  0.02563477,
      -0.16309734,  0.02172852,  0.10693359,  0.02490234, -0.10645453,
      -0.05541994, -0.29402188,  0.040039062,  0.06347656, -0.08447266,
      -0.17871094,  0.01165771,  0.01867777,  0.13677187, -0.1640625 ,
      0.11425781,  0.20880781,  0.06079182, -0.07275391,  0.15039862,
      0.18066496, -0.28515625,  0.044952734,  0.01806454,  0.03311116,
      0.00872803,  0.03564453,  0.29832812,  0.09969938, -0.1484375 ,
      -0.06910599,  0.05957831,  0.02709961, -0.07568359,  0.04614258,
      -0.03174037,  0.02172852,  0.1445375 ,  0.20214644,  0.03235355,
      0.08863281,  0.02038575,  0.08789082,  0.07226562, -0.09433828,
      -0.17080844,  0.1484375 ,  0.10546875,  0.06444531,  0.01031494,
      -0.02636719,  0.03686523,  0.125 ,  0.06787109,  0.14257812,
      0.37109375, -0.15722656,  0.09326172,  0.34969938, -0.00001553,
      -0.03613281,  0.16894531,  0.02856445,  0.10791016,  0.2421875 ,
      -0.14355469,  0.03173828,  0.07421875,  0.34179688,  0.14625 ,
      0.00872803,  0.03564453,  0.29832812,  0.09969938, -0.1484375 ,
      -0.2578125 ,  0.3671875 ,  0.01483154,  0.20703135,  0.08657989,
      -0.18351562, -0.31054688,  0.02844238,  0.184090391,  0.17773438,
      0.06684953, -0.1796875 ,  0.02783203,  0.15625 ,  0.02026367,
      0.0324707 , -0.13476562,  0.15527344,  0.11132812, -0.01055988,
      dtype=float32]
```

Gambar 5.54 Hasil Praktek no 1.11

```
In [13]: baktiqilan_model['wash']
Out[13]:
```

Ini merupakan persentase dari perbandingan kata wash dan clear, persentase yang di dapat ialah 7% Hasil tersebut tidak terlalu baik, ialah sebagai berikut :

```
In [13]: baktiqilan_model.similarity('wash', 'clear')
Out[13]: 0.09019175
```

Gambar 5.55 Hasil Praktek no 1.12

```
In [14]: baktiqilan_model.similarity('wash', 'clear')
```

Ini merupakan persentase dari perbandingan kata bag dan love, persentase yang di dapat ialah 7% Hasil tersebut tidak terlalu baik, ialah sebagai berikut :

```
In [14]: baktiqilan_model.similarity('bag', 'love')
Out[14]: 0.075360954
```

Gambar 5.56 Hasil Praktek no 1.13

```
In [15]: baktiqilan_model.similarity('bag', 'love')
```

Ini merupakan persentase dari perbandingan kata motor dan car, persentase yang di dapat ialah 7% Hasil tersebut tidak terlalu baik, ialah sebagai berikut :

```
In [15]: baktiqilan_model.similarity('motor', 'car')
Out[15]: 0.4810173
```

Gambar 5.57 Hasil Praktek no 1.14

```
1 baktiqilan_model.similarity('motor', 'car')
```

Ini merupakan persentase dari perbandingan kata sick dan faith, persentase yang di dapat ialah 48% Hasil tersebut cukup baik karena mesin dapat membedakan antara motor dan car, ialah sebagai berikut:

```
In [16]: baktiqilan_model.similarity('sick', 'faith')
Out[16]: 0.12307322
```

**Gambar 5.58** Hasil Praktek no 1.15

```
1 baktiqilan_model.similarity('sick', 'faith')
```

Ini merupakan persentase dari perbandingan kata cycle dan shine, persentase yang di dapat ialah 12% Hasil tersebut lumayan dibawah cukup, ialah sebagai berikut:

```
In [17]: baktiqilan_model.similarity('cycle', 'shine')
Out[17]: 0.06161793
```

**Gambar 5.59** Hasil Praktek no 1.16

### 5.3.2.2 jelaskan dengan kata dan ilustrasi fungsi dari extract words dan PermuteSentences (harus beda dengan teman sekelas)

```
1 import re
2
3 test_string = "Mawar itu merah, violet itu biru.!!!"
4 print ("Hasil : " + test_string)
5 res = re.findall(r'\w+', test_string)
6
7 print ("The list of words is : " + str(res))
```

**Listing 5.2** Kodingan Praktek no. 2

Kode tersebut berguna untuk membuat string memakai import re, dengan memakai test string sebagai string, dan membuat print untuk menambahkan kalimat sebelum test string, hasilnya sebagai berikut :

```
In [20]: import re
.....
.... test_string = "Mawar itu merah, violet itu biru.!!!"
.... print ("Hasil : " + test_string)
.... res = re.findall(r'\w+', test_string)
.....
.... print ("The list of words is : " + str(res))
Hasil : Mawar itu merah, violet itu biru.!!!
The list of words is : ['Mawar', 'itu', 'merah', 'violet', 'itu', 'biru']
```

**Gambar 5.60** Hasil Praktek no 2

```

1 import random
2
3 sent_matrix = [ ['Mawar', 'itu'],
4                 ['hitam', 'violet'],
5                 ['itu', 'merah'],
6                 ['kuning', 'hijau']
7             ]
8 result = ""
9 for elem in sent_matrix:
10     result += random.choice(elem) + " "
11
12 print (result)

```

**Listing 5.3** Kodingan Praktek no. 2

Kode tersebut berguna untuk membuat string memakai import random, dengan memakai sent matrix untuk membuat string, dan result sebagai print random yang akan diacak, hasilnya bisa berubah-ubah, sebagai berikut :

```

In [21]: import random
.....
.... sent_matrix = [ ['Mawar', 'itu'],
....                 ['hitam', 'violet'],
....                 ['itu', 'merah'],
....                 ['kuning', 'hijau']
....             ]
.... result = ""
.... for elem in sent_matrix:
....     result += random.choice(elem) + " "
....
....
.... print (result)
Mawar violet merah kuning

```

**Gambar 5.61** Hasil Praktek no 2.1

**5.3.2.3 Jelaskan fungsi dari librari gensim TaggedDocument dan Doc2Vec disertai praktek pemakaiannya. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.**

```

1 print (result)
2
3 # In [Praktek no. 3]
4
5 from gensim.models.doc2vec import Doc2Vec, TaggedDocument
6 ## Exapmple document (list of sentences)
7 doc = ["I love pdf",
8        "I love u",
9        "I love sleep",
10       "This is a good mouse",
11       "This is a good house",
12       "This is a good pause"]
13

```

```

14 tokenized_doc = [ 'love' ]
15 tokenized_doc
16
17 print(doc)
18 # In []
19 tagged_data = [ TaggedDocument(d, [ i ]) for i , d in enumerate(
20     tokenized_doc) ]
21 tagged_data
22 ## Train doc2vec model
23 model = Doc2Vec(tagged_data , vector_size=20, window=2, min_count
24     =1, workers=4, epochs = 100)
25 # Save trained doc2vec model
26 model.save("test_doc2vec.model")
27 ## Load saved doc2vec model
28 model= Doc2Vec.load("test_doc2vec.model")
29 ## Print model vocabulary
30 model.wv.vocab

```

**Listing 5.4** Kodingan Praktek no. 3

Fungsi dari library gensim untuk pemodelan topik tanpa pengawasan dan pemrosesan bahasa alami, atau bisa kita sebut dengan unsupervised. Fungsi dari doc2vec itu sendiri ialah untuk membandingkan bobot data yang terdapat pada dokumen yang lainnya, apakah kata-kata didalamnya 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. Hasilnya adalah sebagai berikut :

```

In[12]: from gensim.models.doc2vec import Doc2Vec, TaggedDocument
... ## Example document (list of sentences)
... doc = ["I love pdf",
...        "I love sleep",
...        "I love a good mouse",
...        "this is a good house",
...        "this is a good pause"]
... tokenized_doc = [ 'love' ]
... tokenized_doc
... print(doc)
['I love pdf', 'I love sleep', 'I love a good mouse', 'This is a good house', 'This is a good pause']

Out[13]:
{'I': <gensim.models.keyedvectors.Vocab at 0x1f63c05a7c8>,
 'o': <gensim.models.keyedvectors.Vocab at 0x1f63c05adc8>,
 'v': <gensim.models.keyedvectors.Vocab at 0x1f63c05a208>,
 'e': <gensim.models.keyedvectors.Vocab at 0x1f63c05a308>}

test_doc2vec.model 06/04/2020 2:36 MODEL File 8 KB

```

**Gambar 5.62** Hasil Praktek no 3

**5.3.2.4 Jelaskan dengan kata dan praktek cara menambahkan data training dari file yang di masukkan kepada variael dalam rangka melatih model doc2van. Tunjukan luaranya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.**

```

1 import re
2 import os
3 unsup_sentences = []
4
5 # source: http://ai.stanford.edu/~amaas/data/sentiment/, data
6 # from IMDB
7 for dirname in ["train/pos", "train/neg", "train/unsup", "test/
8     pos", "test/neg"]:
9     for fname in sorted(os.listdir("aclImdb/" + dirname)):
10         if fname[-4:] == '.txt':

```

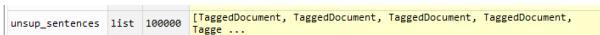
```

9         with open("aclImdb/" + dirname + "/" + fname ,
10            encoding='UTF-8') as f:
11                sent = f.read()
12                words = (sent)
13                unsup_sentences.append(TaggedDocument(words , [
14                  dirname + "/" + fname)))

```

**Listing 5.5** Kodingan Praktek no. 4

Disini kita memakai dataset dari aclImdb. Untuk menambahkan data training kita melakukan import library os, library os itu sendiri berfungsi untuk melakukan interaksi antara python dengan os laptop kita masing-masing, setelah itu kita buat variable unsup sentences. Selanjutnya pilih direktori tempat data kita disimpan. Selanjutnya itu untuk menyortir data yang terdapat pada folder aclImdb dan membaca file tersebut dengan ekstensi .txt. Hasil dari code pertama tersebut ialah terdapatnya data hasil running dari folder aclImdb. Hasilnya adalah sebagai berikut :


**Gambar 5.63** Hasil Praktek no 4

**5.3.2.5** *Jelaskan dengan kata dan praktek kenapa harus dilakukan pengocokan dan pembersihan data. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.*

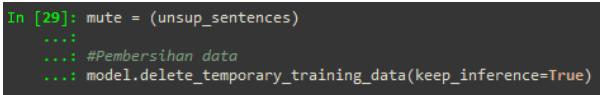
```

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

```

**Listing 5.6** Kodingan Praktek no. 5

Untuk bagian pengacakan data itu berguna untuk mengacak data supaya pada saat data di running bisa berjalan lebih baik dan hasil presentase akhirnya bisa lebih baik. Sedangkan untuk pembersihan data untuk memberikan ruang bagi ram laptop kita setelah melakukan running data sebanyak 3 juta lebih, agar lebih ringan saat proses selanjutnya. Hasilnya adalah sebagai berikut :


**Gambar 5.64** Hasil Praktek no 5

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

```

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

```

**Listing 5.7** Kodingan Praktek no. 6

Save data ini berfungsi untuk menyimpan file hasil dari proses pelatihan data sebelumnya, model tersebut dilakukan penyimpanan untuk memberikan keringanan pada ram agar saat kita akan melakukan pelatihan lagi, model tersebut tinggal di load saja tanpa harus melakukan pelatihan dari awal dan bisa menghemat waktu. Sedangkan untuk delete temporary training data ini berguna untuk menghapus data latihan yang sebelumnya sudah dilakukan dan disimpan, bertujuan untuk memberikan keringanan pada ram. Karena setelah melakukan proses pelatihan ram biasanya jadi tercekit sampai laptop jadi lag. Itulah fungsi dari delete temporary training data. Hasilnya adalah sebagai berikut :

```

In [31]: model.save('E:/backup/sem 6/Kecerdasan Buatan/KB3C - Copy/src/1174083/src5/baktiqilan.d2v')
...
...: #Delete temporary data
...: model.delete_temporary_training_data(keep_inference=True)
2020-04-06 02:59:05,738 : INFO : saving Doc2Vec object under E:/backup/sem 6/Kecerdasan Buatan/KB3C
- Copy/src/1174083/src5/baktiqilan.d2v, separately None
2020-04-06 02:59:05,746 : INFO : saved E:/backup/sem 6/Kecerdasan Buatan/KB3C - Copy/src/1174083/
src5/baktiqilan.d2v

```

**Gambar 5.65** Hasil Praktek no 6

*5.3.2.7 jalankan dengan kta dan praktek maksud dari infer code. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.*

```

1 model.infer_vector(["copas?", "disini aja"])

```

**Listing 5.8** Kodingan Praktek no. 7

Infer vector itu sendiri berguna untuk membandingkan kata yang tercantum dengan vektor yang mana pada dokumen yang sudah di load pada step sebelumnya. Selain itu infer vector juga untuk menghitung atau mengkalkulasikan vektor dari kata yang dicantumkan dari model yang telah kita buat. Alangkah baiknya kata yang dicantumkan itu lebih panjang lagi agar hasilnya bisa lebih baik lagi. Hasilnya adalah sebagai berikut :

```

In [36]: model.infer_vector(["copas?", "disini aja"])
Out[36]:
array([-0.00590723, -0.0159986,  0.01960215, -0.01713275, -0.01539539,
       -0.02306323,  0.00232481, -0.02290895,  0.01663994,  0.0067521,
       0.00234393, -0.01461942, -0.01461422,  0.00662783,  0.02411457,
       0.00014348, -0.01474553, -0.01883789,  0.0154572,  0.01344811],
      dtype=float32)

```

**Gambar 5.66** Hasil Praktek no 7

*5.3.2.8 Jelaskan dengan praktik dan kata maksud dari cosine similarity. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.*

```

1 from sklearn.metrics.pairwise import cosine_similarity
2 cosine_similarity(
3     [model.infer_vector(["copas?", "boleh"])],
4     [model.infer_vector(["plagiat?", "jangan"])])

```

**Listing 5.9** Kodingan Praktek no. 8

Cosine similarity ini 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 0.8% itu dikarenakan kata pertama dan kedua tidak memiliki kesamaan vektorisasi dan tidak terdapat pada salah satu dokumen. Hasilnya adalah sebagai berikut :

```

In [37]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity(
...:     [model.infer_vector(["copas?", "boleh"])],
...:     [model.infer_vector(["plagiat?", "jangan"])])
Out[37]: array([[0.29838568]], dtype=float32)

```

**Gambar 5.67** Hasil Praktek no 8

*5.3.2.9 Jelaskan dengan praktik score dari cross validation masing-masing metode. Tunjukkan keluarannya dari komputer sendiri dan artikan maksud setiap luaran yang didapatkan.*

```

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))

```

**Listing 5.10** Kodingan Praktek no. 9

Code tersebut akan melakukan perhitungan presentase dengan menggunakan cross validation dengan metode kneighborsClassifier. Memakai dataset iris, hasilnya adalah sebagai berikut :

```
In [38]: 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 0.08022311 0.03531764]
```

Gambar 5.68 Hasil Praktek no 9

### 5.3.3 Penanganan Error

#### 5.3.3.1 Terjadi error

```
In [1]: import gensim, logging
...
...: logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
level=logging.INFO)
...: baktiqilan_model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-
vectors-negative300.bin', binary=True, limit=500000)
Traceback (most recent call last):
File "<ipython-input-1-aade7c5d3e5d>", line 1, in <module>
    import gensim, logging
ModuleNotFoundError: No module named 'gensim'
```

Gambar 5.69 terjadi Error 1

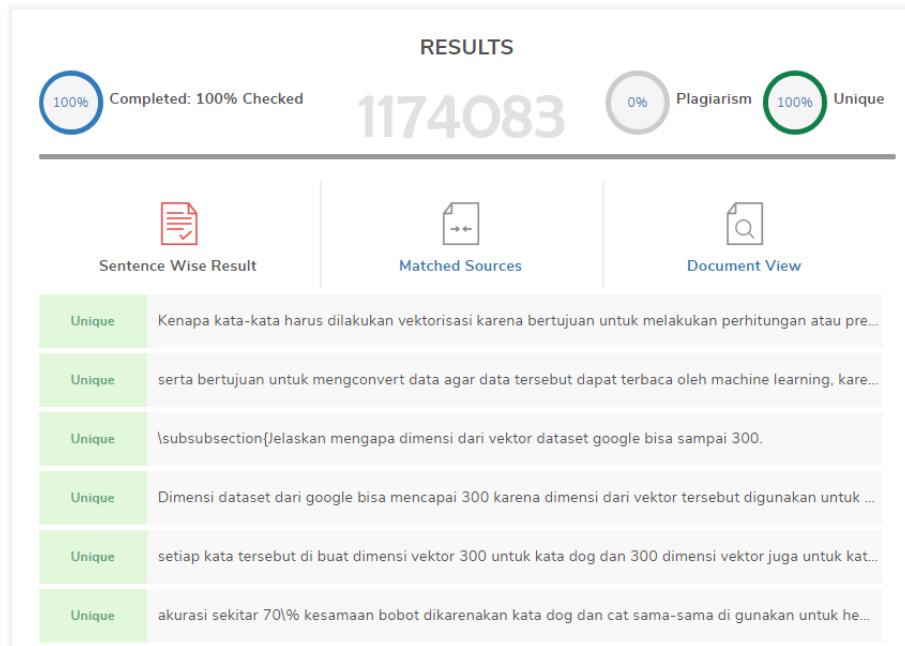
#### 5.3.3.2 Solusi

terjadi error karena tidak adanya modul gensim. dan solusinya cukup menginstallnya, seperti pada gambar berikut:

```
(base) C:\Users\Bakti Qilan>pip install gensim
```

Gambar 5.70 solusi atas error 1

### 5.3.4 Bukti Tidak Plagiat



**Gambar 5.71** Bukti tidak plagiat

### 5.3.5 Link Youtube

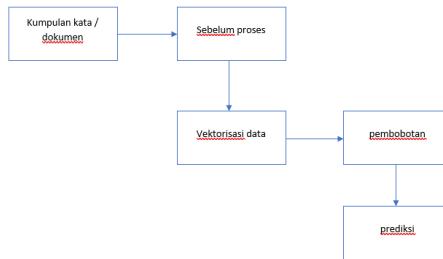
<https://youtu.be/EDs0epNKOU4>

## 5.4 1174087 - Ilham Muhammad Ariq

### 5.4.1 Teori

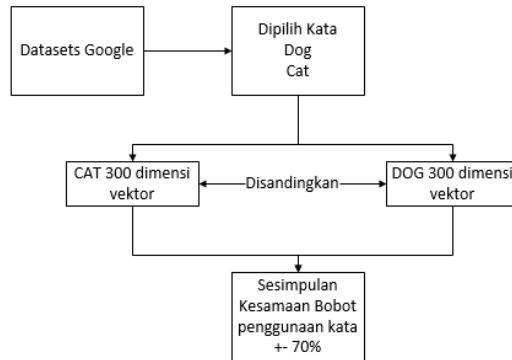
1. Jelaskan kenapa kata-kata harus di lakukan vektorisasi. Dilengkapi dengan ilustrasi atau gambar.

Kata kata harus dilakukan vektorisasi dikarenakan atau bertujuan utk mengukur nilai kemunculan suatau kata yang serupa dari sebuah kalimat sehingga kata-kata tersebut dapat di prediksi kemunculannya. atau juga di buatnya vektorisasi data digunakan untuk memprediksi bobot dari suatu kata misalkan ayam dan kucing sama-sama hewan maka akan dibuat prediksi apakah kata tersebut akan muncul pada kalimat yang kira-kira memiliki bobot yang sama.

**Gambar 5.72** Teori 1

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

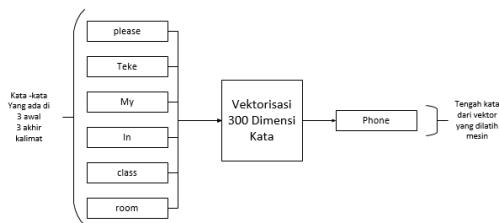
Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap kata, misalkan terdapat kata dog dan cat pada dataset google tersebut setiap kata tersebut dibuat dimensi vektor 300 untuk kata dog dan 300 dimensi vektor juga untuk kata cat kemudian kata tersebut dibandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70 persen kesamaan bobot dikarenakan kata dog dan cat sama-sama digunakan untuk hewan priharaan.

**Gambar 5.73** Teori 2

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

Vektorisasi untuk kata untuk mengetahui kata tengah dari suatu kalimat atau kata utama atau objek utama pada suatu kalimat contoh (Jangan lupa subscribe channel saya ya sekian terimakasih) kata tengah tersebut merupakan channel yang memiliki bobot sebagai kata tengah

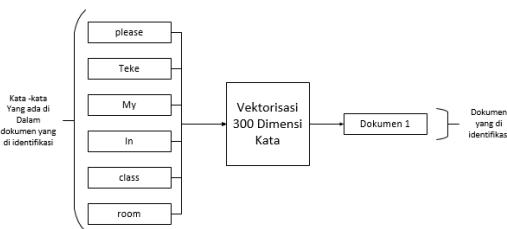
dari suatu kalimat atau bobot sebagai objek dari suatu kalimat. hal ini sangat berkaitan dengan dimensi vektor pada dataset google yang 300 tadi karena untuk mendapatkan nilai atau bobot dari kata tengah tersebut di dapatkan dari proses dimensiasi dari kata tersebut.



**Gambar 5.74** Teori 3

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

Vektorisasi untuk dokumen hampir sama seperti vektorisasi untuk kata hanya saja pemilihan kata utama atau kata tengah terdapat pada satu dokumen jadi mesin akan membuat dimensi vektor 300 untuk dokumen dan nanti kata tengahnya akan di sandingkan pada dokumen yang terdapat pada dokumen tersebut.

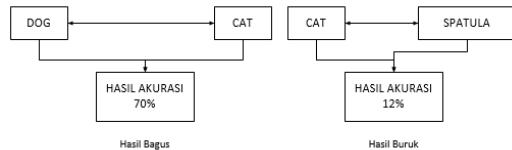


**Gambar 5.75** Teori 4

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

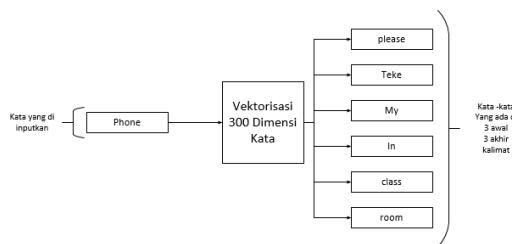
mean merupakan petunjuk terhadap kata-kata yang diolah jika kata-kata itu akurasinya tinggi berarti kata tersebut sering muncul begitu juga sebaliknya, sedangkan standar deviation merupakan standar untuk menimbang kesalahan. sehingga kesalahan tersebut dianggap wajar misalkan kita memperkirakan kedalaman dari dataset merupakan 2 atau 3 tapi pada kenyataannya merupakan 5 itu merupakan kesalahan tapi masih bisa

dianggap wajar karna masih mendekati perkiraan awal.



**Gambar 5.76** Teori 5

6. Jelaskan apa itu skip-gram,dilengkapi dengan ilustrasi atau gambar.  
Skip-Gram adalah kebalikan dari konsep vektorisasi untuk kata dimana kata tengah menjadi acuan terhadap kata-kata pelengkap dalam suatu kalimat.



**Gambar 5.77** Teori 6

#### 5.4.2 Praktek

- Cobalah dataset google, dan jelaskan vektor dari kata love, faith, fall, sick, clear,shine, bag, car, wash, motor, cycle dan cobalah untuk melakukan perbandingan similitud dari masing-masing kata tersebut.
  - berikut merupakan code import gensim digunakan untuk membuat data model atau rangcangan data yang akan dibuat. selanjutnya dibuat variabel gmodel yang berisi data vektor negatif. selanjutnya data tersebut di load agar data tersebut dapat ditampilkan dan diolah.

```

1 import gensim, logging
2 logging.basicConfig(format='%(asctime)s : %(levelname)s :
%(message)s', level=logging.INFO)
3 ariq_model = gensim.models.KeyedVectors.
load_word2vec_format('E:/Kecerdasan Buatan/chapter 5/
GoogleNews-vectors-negative300.bin', binary=True, limit
=500000)

```

```
In [73]: import gensim, logging
...: logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s',
...: level=logging.INFO)
...: ariq_model = gensim.models.KeyedVectors.load_word2vec_format('E:/Kecerdasan
Butan/chapter 5/GoogleNews-vectors-negative300.bin', binary=True, limit=500000)
2020-04-05 23:37:44,458 : INFO : loading projection weights from E:/Kecerdasan
Butan/chapter 5/GoogleNews-vectors-negative300.bin
2020-04-05 23:38:03,406 : INFO : loaded (500000, 300) matrix from E:/Kecerdasan
Butan/chapter 5/GoogleNews-vectors-negative300.bin
```

**Gambar 5.78** Praktek 1

- Pada gambar dapat dilihat bahwa vektor love memiliki array sebanyak 300 dimensi. Untuk identitas sektor satu adalah 0.10.

```
1 ariq_model[ 'love' ]
```

```
In [74]: ariq_model['love']
Out[74]:
array([-0.10302734, -0.15234375,  0.02587891,  0.16503906, -0.16503906,
       0.06689453,  0.29296875, -0.26367188, -0.140625,  0.20117188,
     -0.02624512, -0.08203125, -0.02770996, -0.04394531, -0.23535156,
     0.16992188,  0.12890625,  0.15722656,  0.00756836, -0.06982422,
    -0.03857422,  0.07958984,  0.22949219, -0.14355469,  0.16796875,
     0.02515625,  0.05517578,  0.10602350,  0.11181641, -0.15208504]
```

**Gambar 5.79** Praktek 1.1

- Vektor faith dapat dilihat memiliki nilai 0.26 , untuk similaritasnya cukup mendekati vektor love dimana faith dapat dikategorikan dalam satu kategori dengan love

```
1 ariq_model[ 'faith' ]
```

```
In [77]: ariq_model['faith']
Out[77]:
array([ 0.26367188, -0.04150391,  0.1953125,  0.13476562, -0.14648438,
       0.11962891,  0.04345703,  0.10851562,  0.12207031,  0.13476562,
     0.06640625,  0.18945372, -0.16601562,  0.21679688, -0.27148438,
     0.3203125,  0.10449219,  0.36132812, -0.1953125, -0.18164062,
    0.15332031, -0.1839844,  0.10253906, -0.01367188,  0.23144531,
```

**Gambar 5.80** Praktek 1.2

- Vektor fall hanya memiliki nilai minus yaitu -0.04 , dimana mesin memahami bahwa fall tidak terdapat dalam satu kategori yang sama dengan love dan faith.

```
1 ariq_model[ 'fall' ]
```

```
In [78]: ariq_model['fall']
Out[78]:
array([-0.04272461,  0.10742188, -0.09277344,  0.16894531, -0.1328125,
       -0.10693359,  0.04321289,  0.01094297,  0.14648438,  0.15839062,
     -0.08691406,  0.04492188,  0.0145874,  0.08691406, -0.19824219,
     -0.11035156,  0.01092529, -0.08300781, -0.0189209, -0.1953125,
     -0.1015625,  0.13671875,  0.09228516, -0.12109375,  0.12695312,
     0.03417969,  0.2109375,  0.01977539,  0.125,  0.01544189,
    -0.26953125, -0.0098877, -0.07763672, -0.15527344, -0.03393555,
```

**Gambar 5.81** Praktek 1.3

- Vektor sick memiliki nilai identitas 1.82 dimana tidak mendekati love, faith maupun fall.

```
1 ariq_model[ 'sick' ]
```

```
In [79]: ariq_model['sick']
Out[79]:
array([ 1.82617188e-01,  1.49414062e-01, -4.05273438e-02,  1.64062500e-01,
       -2.59765625e-01,  3.22265625e-01,  1.73828125e-01, -1.47460938e-01,
       1.01074219e-01,  5.46875000e-02,  1.66992185e-01, -1.68945312e-01,
       2.24304199e-03,  9.66796875e-03, -1.66015625e-01, -1.13304688e-01,
       1.66015625e-01,  1.79687500e-01,  5.92041016e-03,  2.45117188e-01,
       8.74023438e-02, -2.56347656e-02,  3.41796875e-01,  4.98046675e-02,
       1.78710938e-01, -9.91821289e-04,  8.88671875e-02, -1.95312500e-01,
       1.81640625e-01, -2.65625000e-01, -1.45507812e-01,  1.00505938e-01,
```

**Gambar 5.82** Praktek 1.4

- Vektor clear memiliki nilai identitas-2,44 dan tidak mendekati nilai dari vektor fall sehingga tidak dapat dijadikan dalam satu kategori.

```
1 ariq_model[ 'clear' ]
```

```
In [80]: ariq_model['clear']
Out[80]:
array([-2.44140625e-04, -1.02050781e-01, -1.49414062e-01, -4.24804688e-02,
       -1.67968750e-01, -1.46484375e-01,  1.76757812e-01,  1.46484375e-01,
       2.26562500e-01,  9.76562500e-02, -2.67578125e-01, -1.29882812e-01,
       1.24511719e-01,  2.23632812e-01, -2.13867188e-01,  3.10058594e-02,
       2.00195312e-01, -4.76074219e-02, -6.83593750e-02, -1.21093750e-01,
       3.22265625e-02,  3.14453125e-01, -1.11816406e-01,  8.00781250e-02,
       -2.75787890e-02, -6.04248047e-03, -7.37304688e-02, -1.72851562e-01,
       9.66796875e-02, -4.91333008e-03, -1.78710938e-01, -1.40380859e-03,
```

**Gambar 5.83** Praktek 1.5

- Untuk vektor shine -0.12 tidak mendekati vektor manapun.

```
1 ariq_model[ 'shine' ]
```

```
In [81]: ariq_model['shine']
Out[81]:
array([-0.12402344,  0.25976562, -0.159117969, -0.27734375,  0.30273438,
       -0.09966938,  0.39257812, -0.22949219, -0.18359375,  0.3671875 ,
       -0.10302734,  0.13671875,  0.25390625,  0.07128966,  0.02539062,
       0.21777344,  0.24023438,  0.5234375 ,  0.123804688, -0.19335938,
       -0.05883789,  0.0612793 , -0.01944918,  0.07617188,  0.05102539,
       0.20019531,  0.38085938,  0.00162506, -0.05029298,  0.14648438,
```

**Gambar 5.84** Praktek 1.6

- Vektor bag memiliki i=nilai identitas -0.03 yang mendekati dengan vektor fall. SEhingga mesin memahami bahwa mungkin saja kedua vektor tersebut berada dalam satu kategori.

```
1 ariq_model[ 'bag' ]
```

```
In [82]: ariq_model['bag']
Out[82]:
array([-0.03515625,  0.15234375, -0.12402344,  0.13378906, -0.11328125,
       -0.0133667 , -0.16113281,  0.14648438, -0.06835938,  0.140625 ,
       -0.06005859, -0.3046875 ,  0.20996094, -0.04345703, -0.2109375 ,
       -0.05957031, -0.05053711,  0.10253906,  0.19042969, -0.09423828,
       0.18847656, -0.07958984, -0.11035156, -0.07910156,  0.06347656,
```

**Gambar 5.85** Praktek 1.7

- Vektor car nilainya 0.13 mendekati vektor love dan faith sehingga mungkin dapat dikategorikan dalam satu kategori.

```
1 ariq_model[ 'car' ]
```

```
In [83]: ariq_model['car']
Out[83]:
array([ 0.13085938,  0.00842285,  0.03344727, -0.05883789,  0.04003906,
       -0.14257812,  0.04931641, -0.16894531,  0.20898438,  0.11962891,
       0.18066406, -0.25 , -0.18400391, -0.18742188, -0.01879883,
       0.05200195, -0.00216675 ,  0.06445312,  0.14453125, -0.04541016,
       0.16113281, -0.01611328, -0.03088379,  0.08447266,  0.16210938,
       0.04467773, -0.15527344,  0.25390625,  0.33984375,  0.00756836,
       -0.25585938, -0.01733398, -0.03295898,  0.16308594, -0.12597656,
```

**Gambar 5.86** Praktek 1.8

- Vektor wash memiliki nilai 9.46 jauh dari vektor vektor lainnya.

```
1 ariq_model[ 'wash' ]
```

```
In [84]: ariq_model['wash']
Out[84]:
array([ 9.4604922e-03,  1.41601562e-01, -5.46875000e-02,  1.34765625e-01,
       -2.38281250e-01,  3.24218750e-01, -8.44726562e-02, -1.29882812e-01,
       1.07910156e-01,  2.53906250e-01,  1.13525391e-02, -1.66992188e-01,
       -2.79541016e-02,  2.08007812e-01, -4.27246094e-02,  1.05468750e-01,
       -7.42187500e-02,  3.04687500e-01,  2.11914062e-01, -8.88671875e-02,
       2.67578125e-01,  2.12890625e-01,  1.74569547e-02,  2.02941895e-03,
       6.29882812e-02,  1.62109375e-01,  1.93359375e-01,  2.17285156e-02,
       -2.67028809e-03, -9.13085938e-02, -2.38281250e-01,  2.23632812e-01,
```

**Gambar 5.87** Praktek 1.9

- Vektor motor memiliki nilai identitas 5.73 yang bisa mendekati vektor wash. Dapat dikatakan bahwa motor dapat dicuci jika diarti dalam satu kategori yang sama.

```
1 ariq_model[ 'motor' ]
```

```
In [85]: ariq_model['motor']
Out[85]:
array([ 5.73730469e-02,  1.50390625e-01, -4.61425781e-02, -1.32812500e-01,
       -2.59765625e-01, -1.77734375e-01,  3.68652344e-02, -4.37500000e-01,
       -2.34375000e-02,  2.57812500e-01,  1.74804688e-01,  2.44140625e-02,
       -2.51953125e-01, -5.76171875e-02,  8.15429688e-02,  1.86767578e-02,
       -3.83300781e-02,  1.58203125e-01, -5.85937500e-02,  1.12304688e-01,
```

**Gambar 5.88** Praktek 1.10

- Vektor cycle memiliki nilai identitas 0.04

```
1 ariq_model['cycle']
```

```
In [86]: ariq_model['cycle']
Out[86]:
array([ 0.04541016,  0.21679688, -0.02709961,  0.12353516, -0.20703125,
       -0.1328152 ,  0.26367188, -0.12890625, -0.125 ,  0.15332031,
       -0.18261719, -0.15820312, -0.06176758,  0.21972656, -0.15820312,
       0.02563477, -0.07568359, -0.0675 ,  0.04614258, -0.31054688,
```

**Gambar 5.89** Praktek 1.11

- berikut merupakan hasil dari similaritas kata kata yang di olah menjadi matrix. Dapat disimpulkan bahwa:

- Untuk Love dan faith hasilnya adalah 37
- Untuk Love dan fall hasilnya adalah 11
- Untuk Love dan sick hasilnya adalah 26
- Untuk Love dan clear hasilnya adalah 6
- Untuk Love dan shine hasilnya adalah 20
- Untuk Love dan bag hasilnya adalah 7
- Untuk Love dan car hasilnya adalah 8
- Untuk Love dan wash hasilnya adalah 11
- Untuk Love dan motor hasilnya adalah 8
- Untuk Love dan wash hasilnya adalah 5

Artinya love dan faith memang dalam kategori yang sama misalnya dalam kategori percintaan/kepercayaan. Mesin suda mengetahui bahwa keduanya dapat dikategorikan sebagai percintaan/kepercayaan.

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

```
In [96]: ariq_model.similarity('love', 'faith')
Out[96]: 0.37053478

In [97]: ariq_model.similarity('love', 'fall')
Out[97]: 0.11425874

In [98]: ariq_model.similarity('love', 'sick')
Out[98]: 0.2665636

In [99]: ariq_model.similarity('love', 'clear')
Out[99]: 0.065844476

In [100]: ariq_model.similarity('love', 'shine')
Out[100]: 0.20605299

In [101]: ariq_model.similarity('love', 'bag')
Out[101]: 0.07536096

In [102]: ariq_model.similarity('love', 'car')
Out[102]: 0.0841786

In [103]: ariq_model.similarity('love', 'wash')
Out[103]: 0.11303361

In [104]: ariq_model.similarity('love', 'motor')
Out[104]: 0.08077487

In [105]: ariq_model.similarity('love', 'cycle')
Out[105]: 0.056645095
```

**Gambar 5.90** Praktek 1.12

2. Jelaskan dengan kata dan ilustrasi fungsi dari extract words dan PermuteSentences

ExtractWords merupakan function untuk menambahkan, menghilangkan atau menghapuskan, hal hal yang tidak penting atau tidak perlu di dalam teks. Dalam contoh dibawah ini. menggunakan function extract words untuk menghapus komen dengan python style , mencari data yang diinginkan, dan memberikan spasi pada teks.

```
In [106]: def extract_words(sent):
...:     sent = sent.lower()
...:     sent = re.sub(r'<[^>]+>', ' ', sent) # strip html tags
...:     sent = re.sub(r'(\w)\\'(\w)', '\1\2', sent) # remove apostrophes
...:     sent = re.sub(r'\W', ' ', sent) # remove punctuation
...:     sent = re.sub(r'\s+', ' ', sent) # remove repeated spaces
...:     sent = sent.strip()
...:     return sent.split()
```

**Gambar 5.91** Praktek 2

PermuteSentences merupakan class yang digunakan unutm melakukan pengocokan secara acak pada data yang ada. Digunakan cara ini agar tidak terjadi kelebihan memori pada saat dijalankan.

```
In [107]: import random
...: class PermuteSentences(object):
...:     def __init__(self, sents):
...:         self.sents = sents
...:
...:     def __iter__(self):
...:         shuffled= list(self.sents)
...:         random.choice(shuffled)
...:         for sent in shuffled:
...:             yield sent
```

Gambar 5.92 Praktek 2

3. Jelaskan fungsi dari librari gensim TaggedDocument dan Doc2Vec disertai praktek pemakaiannya.

Doc2vec adalah algoritma unsupervised untuk menghasilkan vektor untuk kalimat / paragraf / dokumen. Dan TaggedDocument merupakan function dari Doc2Vec untuk menampilkan tag kata atau kalimat yang diinginkan dari sebuah dokumen.

```
In [108]: from gensim.models.doc2vec import TaggedDocument
...: from gensim.models import Doc2Vec
```

Gambar 5.93 Praktek 3

4. Jelaskan dengan kata dan praktek cara menambahkan data training dari file yang dimasukkan kepada variabel dalam rangka melatih model doc2vac.

```
In [53]: import os
unsgp_sentences = []

In [54]: for dirname in ["train/pos","train/neg","train/unsp","test/pos","test/neg"]:
    for frame in sorted(os.listdir(dirname)):
        if frame[-4:] == ".txt":
            with open("scilimh/" + dirname + "/" + frame,encoding='UTF-8') as f:
                sent = f.read()
                words = extract_words(sent)
                unsgp_sentences.append(TaggedDocument(words,[dirname + ":" + frame]))
```

```
In [55]: for dirname in ["txt_sentokenpos","txt_sentokenneg"]:
    for frame in sorted(os.listdir(dirname)):
        if frame[-4:] == ".txt":
            with open("scilimh/" + dirname + "/" + frame,encoding='UTF-8') as f:
                for i, sent in enumerate(f):
                    words = extract_words(sent)
                    unsgp_sentences.append(TaggedDocument(words,[ "%s/%s-%d" % (dirname, frame, i)]))
```

```
In [56]: with open("stanfordSentimentTreebank/original_rt_snippets.txt", encoding='UTF-8') as f:
    for i, sent in enumerate(f):
        words = extract_words(sent)
        unsgp_sentences.append(TaggedDocument(words,['rt-%d' % i]))
```

Gambar 5.94 Praktek 4

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

Pengocokan dilakukan untuk mendapatkan hasil yang lebih akurat pada saat melakukan score, karena pengocokan mempengaruhi performa positif negatifnya dari scoring. Kemudian Pembersihan data dilakukan untuk membersihkan tag spasi ataupun data noisy yang tidak diperlukan dalam dokumen.

```
In [106]: def extract_words(sent):
...:     sent = sent.lower()
...:     sent = re.sub(r'<[^>]+>', ' ', sent) # strip html tags
...:     sent = re.sub(r'(\w)\\'\(\w)', '\1\2', sent) # remove apostrophes
...:     sent = re.sub(r'\W', ' ', sent) # remove punctuation
...:     sent = re.sub(r'\s+', ' ', sent) # remove repeated spaces
...:     sent = sent.strip()
...:     return sent.split()

In [107]: import random
...: class PermuteSentences(object):
...:     def __init__(self, sents):
...:         self.sents = sents
...:
...:     def __iter__(self):
...:         shuffled= list(self.sents)
...:         random.shuffle(shuffled)
...:         for sent in shuffled:
...:             yield sent
```

Gambar 5.95 Praktek 5

6. Jelaskan dengan kata dan praktek kenapa model harus di save dan kenapa temporari training harus dihapus.

Model disave untuk memudahkan kita dalam meng-edit file, kita tidak perlu mengetik ulang semua skrip dan tinggal membuka file tersebut jika ingin menguji ulang modelnya. Temporari training merupakan data training yang sebelumnya kita gunakan untuk mencoba skripnya, namun karena kita telah membuat modelnya dan untuk menghemat memori dilakukan penghapusan temporari training agar tidak terjadi lag ataupun hal lainnya.

```
1 model.delete_temporary_training_data(keep_inference=True)
2 model.save('reviews.d2v')
```

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

Infer vektor digunakan untuk mengkalkulasikan berapa vektor dari kata yang diberikan. Atau dengan kata lain mengubah kata yang diberikan menjadi bentuk vektor. Dari model yang telah dibuat

```
In [70]: model.infer_vector(extract_words("I will go home"))

Out[70]: array([-1.7533980e-01, 1.7654952e-01, 8.7283678e-02, 1.7998287e-02,
-6.2501184e-03, -3.2549590e-02, 1.6319191e-01, -1.3768603e-02,
5.1714227e-02, 4.6473891e-02, 1.6999269e-02, 2.6269551e-02,
-2.5963100e-02, 7.7627085e-02, -1.5499571e-02, 1.3232867e-01,
-4.3878130e-02, 4.8159737e-02, -5.1653433e-02, -7.2681673e-02,
-3.3085446e-02, 1.1083737e-02, -1.1083737e-02, 1.1083737e-02,
6.1815221e-03, 6.1555382e-02, -2.2586747e-01, 1.3859421e-01,
1.6294651e-02, -6.1351795e-02, 1.3722880e-01, -7.6827303e-02,
-1.4081554e-01, 4.0382754e-02, -1.2432394e-01, -1.1883363e-02,
6.4639559e-02, 7.7126230e-02, -1.1083737e-02, -5.9081519e-02,
1.0089800e-02, -6.0805818e-02, 6.5938474e-02, 3.7513527e-03,
5.5374558e-02, -1.3952439e-02, -3.5841893e-02, 4.9781232e-02,
-1.1787581e-01, 9.5303373e-02], dtype=float32)
```

Gambar 5.96 Praktek 7

8. Jelaskan dengan praktek dan kata maksud dari cosine similarity. Cosine similarity digunakan untuk melihat kesamaan atau kemiripan dari suatu kalimat/paragraf yang diinginkan. Apakah kalimat tersebut dapat dikategorikan dalam satu kategori atau tidak.

```
In [83]: from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity([model.infer_vector(extract_words("she going to school, after wash hand")),
                   [model.infer_vector(extract_words("Services sucks2."))]])
Out[83]: array([[0.04744839]], dtype=float32)

In [84]: from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity([model.infer_vector(extract_words("Dia pergi ke sekolah tadi siang")),
                   [model.infer_vector(extract_words("Services sucks2."))]])
Out[84]: array([[0.2095491]], dtype=float32)
```

Gambar 5.97 Praktek 8

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

```
1 from sklearn.neighbors import KNeighborsClassifier
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn.model_selection import cross_val_score
4 import numpy as np
5
6 clf = KNeighborsClassifier(n_neighbors=9)
7 clfrf = RandomForestClassifier()
8
9 scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
10 print((np.mean(scores), np.std(scores)))
11
12 scores = cross_val_score(clfrf, sentvecs, sentiments, cv=5)
13 print((np.mean(scores), np.std(scores)))
14
15 # bag-of-words comparison
16 from sklearn.pipeline import make_pipeline
17 from sklearn.feature_extraction.text import CountVectorizer,
    TfidfTransformer
18 pipeline = make_pipeline(CountVectorizer(), TfidfTransformer(),
                           RandomForestClassifier())
19
20 scores = cross_val_score(pipeline, sentences, sentiments, cv
                           =5)
21 print((np.mean(scores), np.std(scores)))
```

### 5.4.3 Penangan Error

1. FileNotFoundError

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

Gambar 5.98 FileNotFoundError

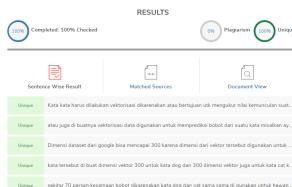
2. Jenis error

- FileNotFoundError

3. Cara Penanganan

Menyesuaikan tempat/direktori file berada

#### 5.4.4 Bukti Tidak Melakukan Plagiat



Gambar 5.99    Bukti Tidak Melakukan Plagiat

#### 5.4.5 Link Youtube

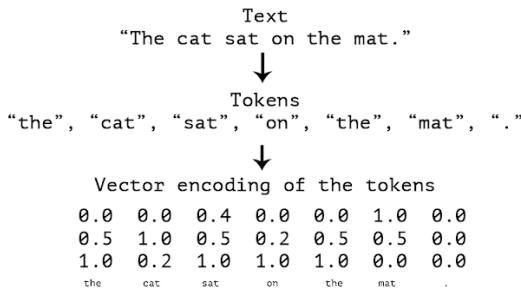
<https://youtu.be/vG9L4Ee1KYk>

### 5.5 1174079 - Chandra Kirana Poetra

#### 5.5.1 Teori

1. Jelaskan kenapa kata-kata harus di lakukan vektorisasi. Dilengkapi dengan ilustrasi atau gambar.

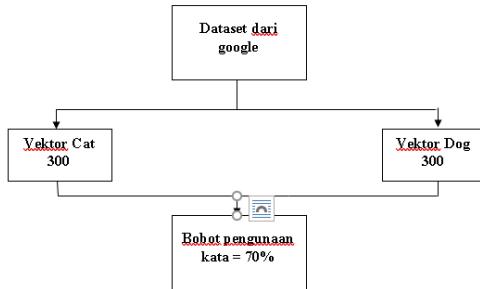
Karena machine learning tidak akan mengerti tentang data yang akan diproses yaitu kata kata, oleh karena itu proses vektorisasi diperlukan sebelumnya untuk menerjemahkan kata kata tadi ke suatu value yang mesin bisa mengerti agar nantinya dapat diproses oleh mesin



Gambar 5.100    Teori 1

2. Jelaskan mengapa dimensi dari vektor dataset google bisa sampai 300. Dilengkapi dengan ilustrasi atau gambar.  
Dataset yang didapatkan dari google bisa mencapai dimensi hingga 300

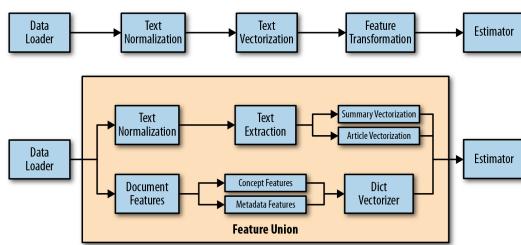
karena vektor yang ada digunakan sebagai perbandingan untuk memberi bobo dari suatu kata seperti misalkan yaitu kata dog dan juga cat yang ada pada dataset memiliki masing masing 300 dimensi vektor kemudian dibandingkan kesamaan katanya maka akan muncul hasil sekitar 70% kesamaan bobot karena keduanya sama sama kata yang digunakan untuk hewan peliharaan.



**Gambar 5.101** Teori 2

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

Vektorisasi kata merupakan suatu metodologi dalam natural language processing untuk melakukan pemetaan kata atau kalimat dari vocabulary menjadi suatu vektor dari angka yang digunakan untuk mencari prediksi kata atau sinonim

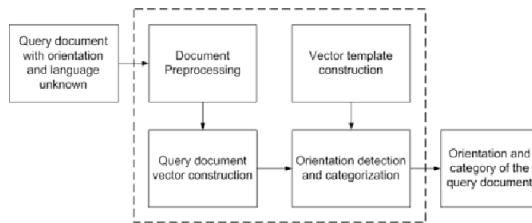


**Gambar 5.102** Teori 3

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

Tujuan utama dari vektorisasi dokumen adalah sama seperti vektorisasi data yaitu untuk merepresentasikan karakter unik dari suatu dokumen secara numerical sehingga komputer bisa memproses suatu data teks yang

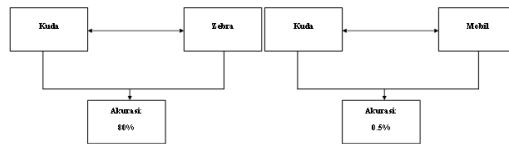
tidak terstruktur



**Gambar 5.103** Teori 4

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

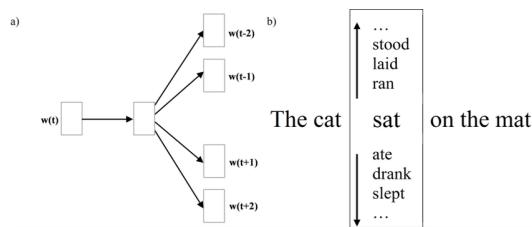
mean merupakan suatu hitungan yang menunjukkan seberapa sering suatu kata itu muncul . sedangkan standar deviasi merupakan suatu perhitungan pada statistik yang digunakan untuk menjelaskan homogenitas dari suatu data.



**Gambar 5.104** Teori 5

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

Skip gram digunakan untuk memprediksi konteks kata untuk suatu kata yang diberikan.



**Gambar 5.105** Teori 6

### 5.5.2 Praktek

1. Cobalah dataset google, dan jelaskan vektor dari kata love, faith, fall, sick, clear,shine, bag, car, wash, motor, cycle dan cobalah untuk melakukan perbandingan similitudi dari masing-masing kata tersebut.

- Import terlebih dahulu gensim serta logging, gensim untuk membuat data model sedangkan logging untuk melakukan logging kemudian load file bin google dengan memasukannya ke variable dengan limit 500000 dan juga binary true yang mengartikan bahwa file ini binary

```
1 import gensim, logging
2 logging.basicConfig(format='%(asctime)s : %(levelname)s :
3 %(%message)s', level=logging.INFO)
4 chandrakp_model = gensim.models.KeyedVectors.
5     load_word2vec_format('E:/Kecerdasan Buatan/chapter 5/
6     GoogleNews-vectors-negative300. bin', binary=True, limit
7     =500000)
```

```
2020-04-06 02:43:23,672 : INFO : loaded 4 pre-trained vectors from /content/drive/My Drive/googleNews-vectors-negative300.bin
7 into memory
8 see https://github.com/RaRe-Technologies/gensim/issues/1040 for details
9 & https://github.com/RaRe-Technologies/gensim/pull/1040 for more information
10
11 2020-04-06 02:43:35,758 : INFO : loaded (500000, 300) matrix from /content/drive/My Drive/googleNews-vectors-negative300.bin
```

**Gambar 5.106** Praktek 1

- pada gambar bisa dilihat bahwa vektor love ini memiliki perhitungan sebesar 0.10

```
1 chandrakp_model['love']
```

```
# In[]
chandrakp['love']

0.16992188, 0.12890625, 0.15722656, 0.00756836, -0.06982422,
-0.03857422, 0.07958984, 0.22949219, -0.14355469, 0.16796875,
-0.03515625, 0.05517578, 0.10693359, 0.1181641, -0.16308594,
```

**Gambar 5.107** Praktek 1.1

- Pada gambar ini Vektor faith memiliki nilai 0.26 , tidak jauh dengan vektor love yang berarti kata ini bisa satu kategori

```
1 chandrakp_model['faith']
```

```
# In[]
chandrakp['faith']

0.3203125, 0.10449219, 0.36132812, -0.1053125, -0.18164062,
0.15332831, -0.10839844, 0.10253906, -0.01367188, 0.23144531,
-0.05957031, -0.22949219, -0.00604248, 0.26171875, 0.10302734,
```

**Gambar 5.108** Praktek 1.2

- vektor fall sayangnya hanya mempunyai poin negatif 0.04 yang berarti tidak bisa satu kategori dengan love dan faith

```
1 chandrakp_model[ 'fall' ]
```

```
[ ] # In[]  
chandrakp['fall']  
  
[] -0.11835156, 0.01092529, -0.08300781, -0.0189289, -0.1953125,  
-0.1015625, 0.13671875, 0.09228516, -0.12189375, 0.12695312,  
0.03417969, 0.2109375, 0.01977539, 0.125, 0.01544189,
```

**Gambar 5.109** Praktek 1.3

- vektor sick memiliki nilai yang sangat jauh sekali dengan yang lainnya yaitu 1.8 yang berarti tidak satu kategori dengan love, faith maupun fall.

```
1 chandrakp_model[ 'sick' ]
```

```
[14] # In[]  
chandrakp['sick']  
# In[]  
  
[] 2.81982422e-02, 1.73828125e-01, -2.08007812e-01, -5.93261719e-02,  
-6.49414062e-02, 3.63769531e-02, 1.91406250e-01, 2.77343750e-01,
```

**Gambar 5.110** Praktek 1.4

- vektor clear memiliki nilai yang sangat jauh sekali dengan yang lainnya yaitu 2,44 sangat jauh dari vektor fall sehingga tidak bisa satu kategori.

```
1 chandrakp_model[ 'clear' ]
```

```
[] chandrakp['clear']  
# In[]  
  
[] 1.55273438e-01, 1.65039062e-01, -1.66992188e-01, 3.14941406e-02,  
2.85156250e-01, 2.69531250e-01, 3.32031250e-02, 2.41210930e-01,
```

**Gambar 5.111** Praktek 1.5

- vektor shine memiliki nilai -0.12 sayangnya tidak mendekati vektor manapun.

```
1 chandrakp_model[ 'shine' ]
```

```
chandrakp['shine']
# In[]

0.21777344, 0.24023438, 0.5234375, 0.12304688, -0.19335938,
-0.05883789, 0.0612793, -0.01940918, 0.07617188, 0.05102539,
```

**Gambar 5.112** Praktek 1.6

- Vektor bag mempunyai nilai identitas -0.03 yang dekat dengan salah satu vektor yaitu vektor fall. disini mesin mulai memahami bahwa kedua kategori ini bisa dijadikan satu kategori.

```
chandrakp_model['bag']
```

```
chandrakp['bag']
# In[]

-0.05957031, -0.05053711, 0.10253906, 0.19042969, -0.09423828,
0.18847656, -0.07958984, -0.11035156, -0.07910156, 0.06347656,
```

**Gambar 5.113** Praktek 1.7

- Vektor car mempunyai nilai 0.13 yang dekat dengan vektor love serta faith sehingga bisa dijadikan satu kategori.

```
chandrakp_model['car']
```

```
[19] chandrakp['car']
# In[]

0.05206195, -0.00216675, 0.06445312, 0.14453125, -0.04541016,
0.16113281, -0.01611328, -0.03088379, 0.08447266, 0.16210958,
```

**Gambar 5.114** Praktek 1.8

- Vektor wash mempunyai nilai sebesar 9.46 yang jauh sekali perbedaananya dengan vektor lain.

```
chandrakp_model['wash']
```

```
chandrakp['wash']
# In[]

1.08398438e-01, -4.30297852e-03, -2.451171888e-01, -2.08984375e-01,
3.58886719e-02, 0.30078125e-02, 1.68945312e-01, 2.79541016e-02,
```

**Gambar 5.115** Praktek 1.9

- Vektor motor mempunyai nilai identitas sebesar 5.73 yang dekat dengan nilai vektor wash.

```
1 chandrakp_model[ 'motor' ]
```

```
[21] chandrakp['motor']
# In[21]

Out[21]:
1.53320312e-01, -7.12898625e-02, -3.68652344e-02, 7.66601562e-02,
-8.08781250e-02, -1.14257812e-01, -9.71679688e-02, -2.61718750e-01,
3.84765625e-01, -1.08750000e-01, -1.10351562e-01, 1.00585938e-01,
```

**Gambar 5.116** Praktek 1.10

- Vektor cycle mempunyai nilai identitas sebesar 0.04

```
1 chandrakp_model[ 'cycle' ]
```

```
[22] chandrakp['cycle']
# In[22]

Out[22]:
0.02563477, -0.07568359, -0.0625, 0.04614258, -0.31854688,
-0.13378906, -0.11669922, -0.3359375, 0.078125, 0.08447266,
0.07226562, -0.06445312, 0.05517578, 0.14941406, 0.13671875,
```

**Gambar 5.117** Praktek 1.11

- berikut merupakan kesimpulan dari similaritas kata kata yang telah diolah menjadi matrix. Dapat disimpulkan bahwa:

- Untuk Love dan faith hasilnya adalah 37
- Untuk Love dan fall hasilnya adalah 11
- Untuk Love dan clear hasilnya adalah 6
- Untuk Love dan shine hasilnya adalah 20
- Untuk Love dan bag hasilnya adalah 7
- Untuk Love dan car hasilnya adalah 8
- Untuk Love dan wash hasilnya adalah 11
- Untuk Love dan motor hasilnya adalah 8
- Untuk Love dan wash hasilnya adalah 5

Data diatas menggambarkan bahwa mesin sudah mengerti bahwa kata faith dan love yang artinya cinta serta kepercayaan merupakan satu kategori

```
1 chandrakp_model.similarity( 'love' , 'faith' )
2 #%%%
3 chandrakp_model.similarity( 'love' , 'fall' )
4 #%%%
5 chandrakp_model.similarity( 'love' , 'sick' )
6 #%%%
7 chandrakp_model.similarity( 'love' , 'clear' )
```

```
8 #%%  
9 chandrakp_model.similarity('love', 'shine')  
10 #%%  
11 chandrakp_model.similarity('love', 'bag')  
12 #%%  
13 chandrakp_model.similarity('love', 'car')  
14 #%%  
15 chandrakp_model.similarity('love', 'wash')  
16 #%%  
17 chandrakp_model.similarity('love', 'motor')  
18 #%%  
19 chandrakp_model.similarity('love', 'cycle')
```

```
chandrakp.similarity('love', 'faith')  
↳ /usr/local/lib/python3.6/dist-packages  
    if np.issubdtype(vec.dtype, np.int):  
        0.3705348
```

```
< [30] chandrakp.similarity('love', 'fall')
```

```
↳ /usr/local/lib/python3.6/dist-packages  
    if np.issubdtype(vec.dtype, np.int):  
        0.114258744
```

```
< [31] chandrakp.similarity('love', 'clear')
```

```
↳ /usr/local/lib/python3.6/dist-packages  
    if np.issubdtype(vec.dtype, np.int):  
        0.06584449
```

```
< [32] chandrakp.similarity('love', 'shine')
```

```
↳ /usr/local/lib/python3.6/dist-packages  
    if np.issubdtype(vec.dtype, np.int):  
        0.20605297
```

```
[33] chandrakp.similarity('love', 'bag')
```

```
↳ /usr/local/lib/python3.6/dist-packages  
    if np.issubdtype(vec.dtype, np.int):  
        0.07536097
```

```
< [35] chandrakp.similarity('love', 'car')
```

```
↳ /usr/local/lib/python3.6/dist-packages  
    if np.issubdtype(vec.dtype, np.int):  
        0.0841786
```

```
< [36] chandrakp.similarity('love', 'wash')
```

```
↳ /usr/local/lib/python3.6/dist-packages  
    if np.issubdtype(vec.dtype, np.int):  
        0.11303361
```

```
< [37] chandrakp.similarity('love', 'cycle')
```

```
↳ /usr/local/lib/python3.6/dist-packages
```

2. Jelaskan dengan kata dan ilustrasi fungsi dari extract words dan PermuteSentences

ExtractWords adalah suatu function yang digunakan untuk memanipulasi data seperti menambahkan atau menghapuskan, berikut contohnya.

```
def extract_words(sent):
    sent = sent.lower()
    sent = re.sub(r'<[^>]+>', ' ', sent) # strip html tags
    sent = re.sub(r'(\w)' '(\w)', '\1\2', sent) # remove apostrophes
    sent = re.sub(r'\W', ' ', sent) # remove punctuation
    sent = re.sub(r'\s+', ' ', sent) # remove repeated spaces
    sent = sent.strip()
    return sent.split()
```

Gambar 5.119 Praktek 2

PermuteSentences adalah class yang dipakai untuk melakukan randomisasi data.

```
import random
class PermuteSentences(object):
    def __init__(self, sents):
        self.sents = sents

    def __iter__(self):
        shuffled= list(self.sents)
        random.shuffle(shuffled)
        for sent in shuffled:
            yield sent
```

Gambar 5.120 Praktek 2

3. Jelaskan fungsi dari librari gensim TaggedDocument dan Doc2Vec disertai praktek pemakaiannya.

Doc2vec adalah suatu algoritma unsupervised yang digunakan untuk generate data berupa vektor seperti dokumen,kalimat,paragraf. Lalu TaggedDocument itu merupakan suatu function Doc2Vec yang dipakai untuk menampilkan kata yang diinginkan.

```
from gensim.models.doc2vec import TaggedDocument
from gensim.models import Doc2Vec
```

Gambar 5.121 Praktek 3

4. Jelaskan dengan kata dan praktek cara menambahkan data training dari

file yang dimasukkan kepada variabel dalam rangka melatih model doc2vac.

```

import os
unsup_sentences = []

for dirname in ["train/pos", "train/neg", "train/unsup", "test/pos", "test/neg"]:
    for fname in sorted(os.listdir(dirname)):
        if fname[:4] == ".txt":
            with open("aclimdb/" + dirname + "/" + fname, encoding='UTF-8') as f:
                sent = f.read()
                words = extract_words(sent)
                unsup_sentences.append(TaggedDocument(words, [dirname + "/" + fname]))

for dirname in ["txt_sentoken/pos", "txt_sentoken/neg"]:
    for fname in sorted(os.listdir(dirname)):
        if fname[:4] == ".txt":
            with open("aclimdb/" + dirname + "/" + fname, encoding='UTF-8') as f:
                for i, sent in enumerate(f):
                    words = extract_words(sent)
                    unsup_sentences.append(TaggedDocument(words, ["%s-%d" % (dirname, fname, i)]))

with open("stanforSentimentBank/original_rt_snippets.txt", encoding='UTF-8') as f:
    for i, sent in enumerate(f):
        words = extract_words(sent)
        unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))

```

**Gambar 5.122** Praktek 4

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

Pengocokan perlu dilakukan supaya hasil yang didapat lebih akurat. Lalu Pembersihan data digunakan untuk menghapus data yang tidak diinginkan seperti spasi dan juga data noisy.

```

def extract_words(sent):
    sent = sent.lower()
    sent = re.sub(r'<[^>]+>', ' ', sent) # strip html tags
    sent = re.sub(r'(\w)\'(\w)', '\1\2', sent) # remove apostrophes
    sent = re.sub(r'\W', ' ', sent) # remove punctuation
    sent = re.sub(r'\s+', ' ', sent) # remove repeated spaces
    sent = sent.strip()
    return sent.split()

#%%
import random
class PermuteSentences(object):
    def __init__(self, sents):
        self.sents = sents

    def __iter__(self):
        shuffled= list(self.sents)
        random.choice(shuffled)
        for sent in shuffled:
            yield sent

```

**Gambar 5.123** Praktek 5

6. Jelaskan dengan kata dan praktek kenapa model harus di save dan kenapa temporari training harus dihapus.

Model disave agar nantinya kita mudah untuk mengedit atau menguji filenya kembali. Temporari training adalah data training yang telah kita jalankan sebelumnya, namun karena sudah kita buat modelnya dan juga untuk menghemat daya memori maka penghapusan data temporari

training dirasa perlu untuk menghindari lag atau kejadian yang tidak diinginkan.

```
1 model.delete_temporary_training_data(keep_inference=True)
2 model.save('reviews.d2v')
```

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

Infer vektor dipakai untuk menhitung vektor dari dokumen atau kata yang diberikan

```
model.infer_vector(extract_words("I will go home"))
array([-1.7533980e-01,  1.7654952e-01,  8.7283678e-02,  1.7998287e-02,
       6.2501184e-03, -3.2540500e-02,  1.6391091e-01, -1.3786003e-02,
```

**Gambar 5.124** Praktek 7

8. Jelaskan dengan praktek dan kata maksud dari cosine similarity. Cosine similarity dipakai untuk melihat nilai kesamaan dari suatu paragraf atau kalimat yang kita berikan. apakah satu kategori atau tidaknya.

```
from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity([
    [model.infer_vector(extract_words("she going to school, after wash hand"))],
    [model.infer_vector(extract_words("Services sucks2."))]])
array([[0.04744839]], dtype=float32)

from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity([
    [model.infer_vector(extract_words("Dia pergi ke sekolah tadi siang"))],
    [model.infer_vector(extract_words("Services sucks2."))]])
array([[0.2095491]], dtype=float32)
```

**Gambar 5.125** Praktek 8

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

```
1 from sklearn.neighbors import KNeighborsClassifier
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn.model_selection import cross_val_score
4 import numpy as np
5
6 clf = KNeighborsClassifier(n_neighbors=9)
7 clfrf = RandomForestClassifier()
8
9 scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
10 print((np.mean(scores), np.std(scores)))
11
12 scores = cross_val_score(clfrf, sentvecs, sentiments, cv=5)
13 print((np.mean(scores), np.std(scores)))
14
15 # bag-of-words comparison
16 from sklearn.pipeline import make_pipeline
17 from sklearn.feature_extraction.text import CountVectorizer,
      TfidfTransformer
```

```
18 pipeline = make_pipeline(CountVectorizer(), TfidfTransformer())
19
20 scores = cross_val_score(pipeline, sentences, sentiments, cv
21 =5)
22 print((np.mean(scores), np.std(scores)))
```

### 5.5.3 Penangan Error

1. word falla not in vocabulary

```
[50] # In[]
    chandrakp['falla']

[50]: -----
KeyError                                     T
<ipython-input-50-463c1d4c7b05> in <module>
----> 1 chandrakp['falla']

                                                2 frames
/usr/local/lib/python3.6/dist-packages/gens
    450             return result
    451         else:
--> 452             raise KeyError("word '%"
    453
    454     def get_vector(self, word):

KeyError: "word 'falla' not in vocabulary"
```

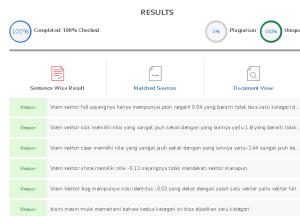
**Gambar 5.126** word falla not in vocabulary

2. Jenis error

- word falla not in vocabulary

3. Cara Penanganan  
typo, perbaiki

### 5.5.4 Bukti Tidak Melakukan Plagiat



**Gambar 5.127** Bukti Tidak Melakukan Plagiat

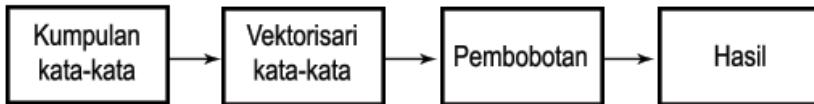
### 5.5.5 Link Youtube

<https://youtu.be/YWgD7jAeT8A>

## 5.6 Muhammad Reza Syachrani - 1174084

### 5.6.1 Teori

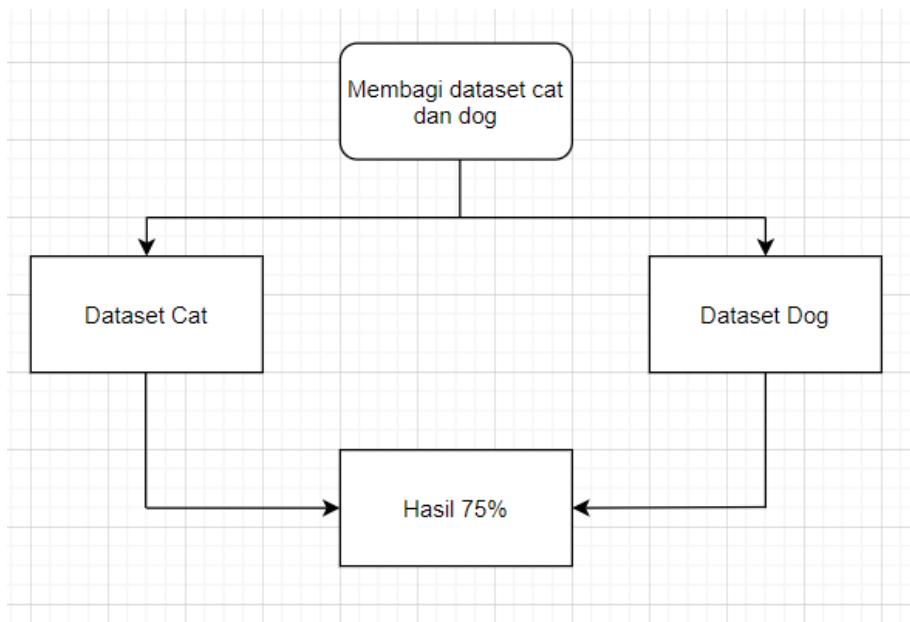
1. Jelaskan kenapa kata-kata harus di vektorisasi, dan gambar ilustrasi Kata-kata divektorisasi agar machine learning dapat memprosesnya karena machine learning tidak dapat membaca data text langsung sehingga perlu dilakukan vektorisasi untuk mengukur nilai dari suatu kata sehingga dapat mengetahui bobot dari setiap kata agar dapat dibaca dan melakukan prediksi.



**Gambar 5.128** Teori 1

2. Jelaskan mengapa dimensi dari vektor dataset google bisa sampai 300. Dilengkapidengan ilustrasi atau gambar.  
Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap kata, misalkan terdapat kata dog dan cat pada dataset google tersebut setiap kata tersebut dibuat dimensi vektor 300 untuk kata dog dan 300 dimensi vektor juga untuk kata cat kemudian kata tersebut dibandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70 persen kesamaan

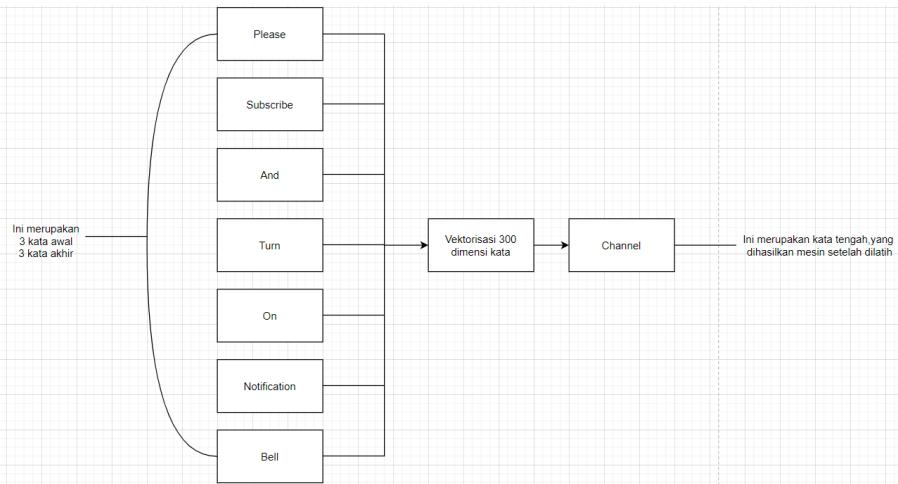
bobot dikarenakan kata dog dan cat sama sama di gunakan untuk hewan priharaan.



Gambar 5.129 Teori 2

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

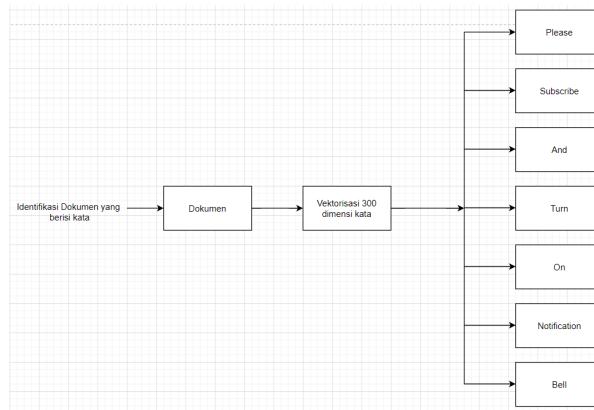
Vektorisasi untuk kata untuk mengetahui kata tengah dari suatu kalimat atau kata utama atau objek utama pada suatu kalimat contoh ( Jangan lupa subscribe channel saya ya sekian terimakasih ) kata tengah tersebut merupakan channel yang memiliki bobot sebagai kata tengah dari suatu kalimat atau bobot sebagai objek dari suatu kalimat. hal ini sangat berkaitan dengan dimensi vektor pada dataset google yang 300 tadi karena untuk mendapatkan nilai atau bobot dari kata tengah tersebut di dapatkan dari proses dimensiasi dari kata tersebut.



Gambar 5.130 Teori 3

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

Vektorisasi untuk dokumen hampir sama seperti vektorisasi untuk kata hanya saja pemilihan kata utama atau kata tengah terdapat pada satu dokumen jadi mesin akan membuat dimensi vektor 300 untuk dokumen dan nanti kata tengahnya akan di sandingkan pada dokumen yang terdapat pada dokumen tersebut.

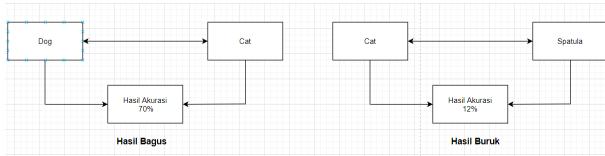


Gambar 5.131 Teori 4

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

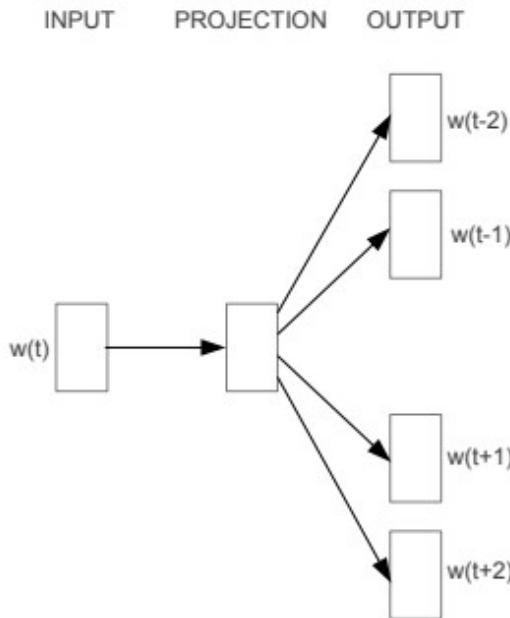
mean merupakan petunjuk terhadap kata-kata yang diolah jika kata

kata itu akurasinya tinggi berarti kata tersebut sering muncul begitu juga sebaliknya, sedangkan standar defiation merupakan standar untuk menimbang kesalahan. sehingga kesalahan tersebut di anggap wajar misalkan kita memperkirakan kedalaman dari dataset merupakan 2 atau 3 tapi pada kenyataanya merupakan 5 itu merupakan kesalahan tapi masih bisa dianggap wajar karna masih mendekati perkiraan awal. contoh pada gambar dibawah ini:



Gambar 5.132 Teori 5

6. Jelaskan apa itu skip-gram,dilengkapi dengan ilustrasi atau gambar  
Skip-Gram adalah kebalikan dari konsep vektorisasi untuk kata dimana kata tengah menjadi acuan terhadap kata-kata pelengkap dalam suatu kalimat.



## Skip-gram

Gambar 5.133 Teori 6

### 5.6.2 Praktek

1. mencoba dataset google dan penjelasan vektor dari kata love, faith, fall, sick, clear, shine, bag, car, wash, motor, dan cycle.
- berikut merupakan code import gensim digunakan untuk membuat data model atau rangcangan data yang akan di buat. selanjutnya dibuat variabel genmod yang berisi data vektor negativ. selanjutnya data tersebut di load agar data tersebut dapat di tampilkan dan di olah.

```
In [37]: import gensim
...: genmod = gensim.models.KeyedVectors.load_word2vec_format('D:/GoogleNews-
vectors-negative300.bin', binary=True)
```

Gambar 5.134 Praktek 1

- berikut merupakan hasil lpengolahan kata love pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [38]: genmod['love'].toarray():
array([ 0.10302734, -0.15234375,  0.02587891,  0.16503906, -0.16503906,
       0.06689453,  0.29296875, -0.26367188, -0.140625 ,  0.20117188,
      -0.02624512, -0.08203125, -0.02770996, -0.04394531, -0.23535156,
       0.16992188,  0.12890625,  0.15722656,  0.00756836, -0.06982422,
      -0.03857422,  0.07958984,  0.22949219, -0.14355469,  0.16796875,
      -0.03515625,  0.05517578,  0.10693359,  0.11181641, -0.16308594,
      -0.11181641,  0.13964844,  0.01556396,  0.12792969,  0.15429688,
      0.07714844,  0.26171875,  0.08642578, -0.02514648,  0.33398438,
      0.18652344, -0.20996094,  0.07080078,  0.02600098, -0.10644531,
     -0.10253906,  0.12304688,  0.04711914,  0.02209473,  0.05834961,
     -0.10986328,  0.14941406, -0.10693359,  0.01556396,  0.08984375,
      0.11230469, -0.04370117, -0.11376953, -0.0037384 , -0.01818848,
      0.24316406,  0.08447266, -0.07080078,  0.18066406,  0.03515625,
      -0.09667969, -0.21972656, -0.00328064, -0.03198242,  0.18457031,
      0.28515625, -0.0859375 , -0.11181641,  0.0213623 , -0.30664062,
     -0.09228516, -0.18945312,  0.01513672,  0.18554688,  0.34375 ,
     -0.31054688,  0.22558594,  0.08740234, -0.2265625 , -0.29492188,
      0.08251953, -0.38476562,  0.25390625,  0.26953125,  0.06298828,
     -0.00958252,  0.23632812, -0.17871094, -0.12451172, -0.17285156,
     -0.11767578,  0.19726562, -0.03466797, -0.10480391, -0.1640625 ,
     -0.19726562,  0.19824219,  0.09521484,  0.00561523,  0.12597656,
      0.00073624, -0.0402832 , -0.03063965,  0.01623535, -0.1640625 ,
     -0.22167969,  0.171875 ,  0.12011719, -0.01965332,  0.4453125 ,
      0.06494141,  0.05932617, -0.1640625 , -0.01367188,  0.18945312,
      0.05566406, -0.05004883, -0.01422119,  0.15917969,  0.07421875,
     -0.31640625, -0.0534668 , -0.02355957, -0.16992188,  0.0625 ,
     -0.140625 , -0.13183594, -0.12792969,  0.12060547,  0.05883789,
     -0.00055695,  0.05761719, -0.08447266,  0.16992188,  0.13671875,
     -0.09375 ,  0.08056641, -0.04003906, -0.03759766, -0.26367188,
```

**Gambar 5.135** Praktek 1

- berikut merupakan hasil lpengolahan kata faith pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [40]: genmod['faith']
Out[40]:
array([-0.26367188, -0.04150391,  0.1953125,  0.13476562, -0.14648438,
       0.11962891,  0.04345073,  0.18351562,  0.12207031,  0.13476562,
       0.06640625,  0.18945312, -0.16601562,  0.21679688, -0.27148438,
       0.3203125,  0.10449219,  0.36132812, -0.1953125, -0.18164062,
       0.15323031, -0.10839844,  0.10253906, -0.01367188,  0.23144531,
      -0.05957031, -0.22949219, -0.006604248,  0.26171875,  0.10302734,
     -0.1328125,  0.21484375,  0.01135254,  0.02111816,  0.18554688,
     0.04125977,  0.12011719,  0.17480469, -0.22167969, -0.13476562,
     0.3125,  0.06640625, -0.17675781, -0.01708984, -0.1640625,
     -0.02819824,  0.01257324, -0.09521484, -0.18066406, -0.140625,
     -0.02258301,  0.16308594, -0.13183594, -0.08007812,  0.13085938,
     0.27539062, -0.20605469,  0.10351562, -0.20214844, -0.1875,
     0.16992188,  0.13574219,  0.13769531,  0.16308594, -0.03881836,
     -0.11132812,  0.05688477,  0.12255859,  0.09814453, -0.04956055,
     -0.02331543, -0.04248047, -0.08283125,  0.16015625,  0.04150391,
     -0.16601562, -0.13671875,  0.09619141,  0.32617188,  0.08251953,
     -0.20800781,  0.04199219,  0.05834961, -0.27734375,  0.09130859,
     -0.17382812, -0.22460938,  0.03466797,  0.19824219, -0.08837891,
     0.18359375,  0.07324219,  0.1171875, -0.33984375,  0.16796875,
     -0.13574219, -0.30078125, -0.00469971,  0.06005859, -0.29296875,
     0.15234375,  0.02966309,  0.33203125,  0.28320312,  0.09375,
     -0.20605469, -0.09082031,  0.0534668,  0.05834961, -0.03222656,
     -0.29296875,  0.25585938,  0.00430298,  0.140625,  0.05810547,
     0.21582031,  0.02917478,  0.02929688,  0.20019531,  0.34960938,
     0.10449219, -0.01940918,  0.04077148,  0.32226562, -0.1953125,
     -0.05688477,  0.10498047, -0.04785156, -0.15136719, -0.07714844,
     -0.45898438, -0.29492188, -0.328125, -0.20996094,  0.38671875,
     0.0039978,  0.07373047, -0.01220703,  0.22460938,  0.14550781,
```

Gambar 5.136 Praktek 1

- berikut merupakan hasil pengolahan kata fall pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [41]: genmod['fall']
Out[41]:
array([-0.04272461,  0.10742188, -0.09277344,  0.16894531, -0.1328125,
       -0.10693359,  0.04321289,  0.01904297,  0.14648438,  0.15039062,
       -0.08691406,  0.04492188,  0.0145874,  0.08691406, -0.19824219,
       -0.11035156,  0.01092529, -0.08300781, -0.0189209,  0.1953125,
       -0.1015625,  0.13671875,  0.09228516, -0.12109375,  0.12695312,
       0.03417969,  0.2109375,  0.01977539,  0.125,  0.01544189,
       -0.26953125, -0.0098877, -0.07763672, -0.15527344, -0.03393555,
       0.04199219, -0.29882812, -0.18554688,  0.08496094, -0.02087402,
       0.13574219, -0.22558594,  0.33789062, -0.03564453, -0.10839844,
      -0.19335938,  0.0546875, -0.04956055,  0.3671875, -0.03295898,
       0.10205078, -0.15136719, -0.00445557,  0.04003906,  0.27539062,
       -0.06933594,  0.05834961,  0.01422119, -0.01397705, -0.05395508,
       -0.0255127,  0.06298828,  0.07080078, -0.07617188,  0.06542969,
       -0.01672363, -0.04711194,  0.19628906, -0.08984375,  0.078125,
       0.2109375,  0.0612793,  0.08789062,  0.19628906,  0.11376953,
       0.06542969,  0.03125,  0.12988281,  0.02270508,  0.14550781,
      -0.06225586, -0.37695312, -0.05737305, -0.06396484,  0.08984375,
       0.00448608, -0.14160156, -0.04541016, -0.0703125,  0.06005859,
       0.26757812,  0.02001953, -0.12695312, -0.04882812,  0.18945312,
       -0.03466797,  0.04638672,  0.1484375,  0.01708984, -0.08789062,
       -0.14941406, -0.02331543, -0.03955078, -0.10400391, -0.14160156,
       -0.18261719, -0.03076172,  0.04589844, -0.2890625, -0.03540039,
       0.12896025, -0.10595703,  0.17578125,  0.06689453,  0.34960938,
       0.04296875,  0.09863281, -0.08956641, -0.06298828,  0.12255859,
       0.0234375, -0.06494141,  0.09667969, -0.04589844,  0.04956055,
       0.08007812, -0.00482178, -0.1640625, -0.03271484,  0.0703125,
      -0.07958984, -0.12890625, -0.01879883, -0.17773438,  0.01293945,
     -0.20019531,  0.08886719, -0.18847656, -0.23828125,  0.02758789,
```

Gambar 5.137 Praktek 1

- berikut merupakan hasil lpengolahan kata sick pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [42]: genmod['sick']
Out[42]:
array([ 1.82617188e-01,  1.49414062e-01, -4.05273438e-02,  1.64062500e-01,
       -2.59765625e-01,  3.22265625e-01,  1.73828125e-01, -1.47460938e-01,
       1.01874219e-01,  5.46875000e-02,  1.66992188e-01, -1.68945312e-01,
      2.24304199e-03,  9.66796875e-02, -1.66015625e-01, -1.12304688e-01,
      1.66015625e-01,  1.79687500e-01,  5.92041016e-03,  2.45117188e-01,
      8.74023438e-02, -2.56347656e-02,  3.41796875e-01,  4.98046875e-02,
     1.78710938e-01, -9.91821289e-04,  8.88671875e-02, -1.95312500e-01,
     1.81640625e-01, -2.65625000e-01, -1.45507812e-01,  1.00585938e-01,
    9.42382812e-02, -3.12500000e-02,  1.98974609e-02, -6.39648438e-02,
    1.18652344e-01,  1.23046875e-01, -6.03027344e-02,  4.68750000e-01,
    9.13085938e-02, -3.12500000e-01,  1.845708312e-01, -1.51367188e-01,
    5.78613281e-02, -1.04980469e-01, -1.68945312e-01, -8.00781250e-02,
   -2.01171875e-01,  1.06201172e-02, -1.29882812e-01, -1.25976562e-01,
   -5.56640625e-02,  3.14453125e-01,  5.61523438e-02, -1.20117188e-01,
    7.12890625e-02,  4.37011719e-02,  2.05078125e-01,  5.71289062e-02,
    8.44726562e-02,  2.15820312e-01, -1.26953125e-01,  8.78906250e-02,
    2.48046875e-01,  6.54296875e-02, -2.02636719e-02,  1.52343750e-01,
   -3.57421875e-01,  3.02124023e-03, -2.08807812e-01, -5.05371094e-02,
    2.81982422e-02,  1.73828125e-01, -2.08807812e-01, -5.93261719e-02,
   -6.49414062e-02,  3.63769531e-02,  1.91406250e-01,  2.77343750e-01,
    3.54003906e-02,  1.56250000e-01, -2.03857422e-02,  2.26562500e-01,
   -4.66308594e-02, -5.17578125e-02, -1.63085938e-01,  4.17480469e-02,
   2.01171875e-01, -2.01171875e-01, -1.50756836e-02,  2.61718750e-01,
   -1.10839844e-01, -4.21875000e-01,  2.22167969e-02,  1.46484375e-01,
    4.19921875e-01, -6.88476562e-02,  9.42382812e-02, -1.96289062e-01,
   -9.42382812e-02, -3.12500000e-02,  6.34765625e-02,  2.47802734e-02,
   -1.61132812e-01, -1.53320312e-01,  1.31835938e-01, -1.81640625e-01,
   -3.22265625e-02, -1.43554688e-01,  8.30078125e-03, -9.81445312e-02,
```

**Gambar 5.138** Praktek 1

- berikut merupakan hasil lpengolahan kata clear pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [43]: genmod['clear']
Out[43]:
array([-2.44140625e-04, -1.02050781e-01, -1.49414062e-01, -4.24804688e-02,
       -1.67968750e-01, -1.46484375e-01, 1.76757812e-01, 1.46484375e-01,
       2.26562500e-01, 9.76562500e-02, -2.67578125e-01, -1.29882812e-01,
       1.24511719e-01, 2.23632812e-01, -2.13867188e-01, 3.10058594e-02,
       2.00195312e-01, -4.76074219e-02, -6.83593750e-02, -1.21093750e-01,
       3.22265625e-02, 3.14453125e-01, -1.11816406e-01, 8.00781250e-02,
       -2.75878906e-02, -6.04248047e-03, -7.37304688e-02, -1.72851562e-01,
       9.66796875e-03, -4.91333008e-03, -1.78710938e-01, -1.40388059e-03,
       7.91015625e-02, 1.07910156e-01, -1.10351562e-01, -8.34960938e-02,
       1.98974609e-02, -3.14331855e-03, 1.30615234e-02, 3.34472656e-02,
       2.55859375e-01, -7.12890625e-02, 2.83203125e-01, -2.75390625e-01,
       -8.49609375e-02, -3.73535156e-02, -9.17968750e-02, -1.30859375e-01,
       3.49121094e-02, 7.32421875e-02, 2.17773438e-01, 5.00488281e-02,
       7.47070312e-02, -1.25000000e-01, -5.73730469e-02, -2.74658203e-02,
       -1.37329102e-02, -2.13867188e-01, -1.12792969e-01, -4.98046875e-02,
       -1.92382812e-01, 1.32812500e-01, -5.00488281e-02, -1.66015625e-01,
       -1.57470703e-01, -1.37695312e-01, 3.88183594e-02, 1.57226562e-01,
       -1.52343750e-01, -1.64794922e-02, -2.27539062e-01, 3.34472656e-02,
       1.55273438e-01, 1.65039062e-01, -1.66992188e-01, 3.14941406e-02,
       2.85156250e-01, 2.69531250e-01, 3.32031250e-02, 2.41210938e-01,
       1.39160156e-02, 5.12695312e-02, 9.76562500e-02, 3.14941406e-02,
       2.51464844e-02, -2.85156250e-01, -1.24023438e-01, -6.88476562e-02,
       -5.2978156e-02, 2.06054688e-01, -2.07031250e-01, -1.60156250e-01,
       -2.61230469e-02, -3.01513672e-02, 8.60599219e-03, -1.30859375e-01,
       3.88183594e-02, 8.60595703e-03, 5.31005859e-03, -6.05468750e-02,
       1.03759766e-02, 1.33789062e-01, -1.89971924e-03, -4.27246094e-02,
       -1.14746094e-01, -1.47705078e-02, 9.71679688e-02, -1.66992188e-01,
       5.63964844e-02, -2.76184082e-03, -1.05468750e-01, -1.03027344e-01,
```

Gambar 5.139 Praktek 1

- berikut merupakan hasil lpengolahan kata shine pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [44]: genmod['shine']
Out[44]:
array([-0.12402344, 0.25976562, -0.15917969, -0.27734375, 0.30273438,
       0.09960938, 0.39257812, -0.22949219, -0.18359375, 0.3671875 ,
       -0.10302734, 0.13671875, 0.25390625, 0.07128906, 0.02539062,
       0.21777344, 0.24023438, 0.5234375 , 0.12304688, -0.19335938,
       -0.05883789, 0.0612793 , -0.01940918, 0.07617188, 0.05102539,
       0.20019531, 0.38085938, 0.00162506, -0.05029297, 0.14648438,
       -0.34765625, 0.02563477, -0.23925781, -0.04516602, -0.00479126,
       -0.24121094, -0.18945312, -0.15234375, -0.05493164, 0.01434326,
       0.390625 , -0.2109375 , 0.1484375 , -0.13183594, 0.24511719,
       -0.24023438, -0.36132812, -0.12792969, 0.10595703, 0.09912109,
       -0.0246582 , 0.32226562, 0.11376953, 0.18164062, 0.04931641,
       -0.10253906, -0.00283813, -0.29882812, -0.171875 , -0.18945312,
       -0.01367188, -0.20898438, -0.07861328, -0.0859375 , 0.05395508,
       -0.14257812, -0.140625 , 0.03027344, -0.114453125, 0.359375 ,
       0.16113281, 0.22265625, 0.265625 , -0.06347656, -0.02807617,
       0.04760742, 0.08837891, -0.04272461, 0.05908203, 0.07128906,
       0.01519775 , -0.11621094, 0.07128906, 0.01403809, -0.10644531,
       0.08886719, 0.11523438, 0.09667969, -0.11083984, 0.16015625,
       0.3359375 , -0.1875 , 0.14550781, 0.00463867, 0.07617188,
       -0.09521484, 0.08447266, 0.20117188, 0.11230469, -0.33984375,
       -0.25390625, 0.05200195, 0.27539062, -0.08398438, -0.31054688,
       -0.22949219, 0.14941406, -0.1953125 , 0.08496694, -0.00753784,
       0.078125 , 0.05908203, 0.02355957, 0.06347656, 0.32617188,
       -0.08740234, 0.10058594, -0.11474609, -0.18164062, 0.13378906,
       0.11230469, -0.00080109, 0.08691406, 0.03808594, 0.0300293 ,
       -0.03417969, -0.008003078, 0.14160156, -0.09619141, -0.11328125,
       0.00823975, -0.15234375, 0.19042969, 0.04980469, 0.25 ,
       -0.00171661, 0.04589844, -0.05102539, -0.04052734, -0.03491211,
```

Gambar 5.140 Praktek 1

- berikut merupakan hasil lpengolahan kata bag pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [45]: genmod['bag']
Out[45]:
array([-0.03515625,  0.15234375, -0.12402344,  0.13378906, -0.11328125,
       -0.0133667, -0.16113281,  0.14648438, -0.06835938,  0.140625 ,
       -0.06005859, -0.3046875,  0.20996094, -0.04345703, -0.2109375 ,
       -0.05957031, -0.05053711,  0.10253906,  0.19042969, -0.09423828,
       0.18847656, -0.07958984, -0.11035156, -0.07910156,  0.06347656,
      -0.15527344, -0.18945312,  0.11132812,  0.27539062, -0.06787109,
       0.01806641,  0.06689453,  0.2578125,  0.0324707, -0.24609375,
      -0.05541992,  0.01013184,  0.24121094, -0.21875,  0.07568359,
      -0.09814453, -0.16113281,  0.16503906, -0.09521484, -0.16601562,
      -0.41796875,  0.0300293,  0.19433594,  0.2890625,  0.12695312,
      -0.19824219, -0.05517578,  0.04296875, -0.10107422,  0.07324219,
      -0.13378906,  0.265625, -0.00466919,  0.19628906, -0.10839844,
      0.14941406,  0.1484375,  0.09619141,  0.21777344, -0.08544922,
      -0.02819824,  0.02539062, -0.03759766,  0.23242188,  0.19628906,
      0.27539062,  0.09130859,  0.23730469,  0.09033203, -0.28515625,
      0.05932617,  0.06591797, -0.01794434, -0.00055313, -0.1796875 ,
      0.05615234, -0.12207031, -0.09863281, -0.05786133, -0.09375 ,
      -0.30273438, -0.06396484, -0.00744629, -0.17871094,  0.08544922,
      0.20410156,  0.33789062,  0.00228882, -0.39453125, -0.14453125,
      -0.328125, -0.12695312, -0.08544922,  0.15234375,  0.03662109,
      -0.1484375,  0.05566406,  0.02844238,  0.07519531, -0.21484375,
      -0.15722656,  0.3359375, -0.04736328, -0.000405884, -0.19726562,
      0.27929688,  0.05566406, -0.10058594, -0.00811768, -0.20703125,
      0.03295898, -0.14550781, -0.15917969,  0.16503906,  0.234375 ,
      0.03588867,  0.04296875, -0.25,  0.1171875, -0.07714844,
      0.00521851,  0.125,  0.08886719,  0.15527344, -0.02185059,
      -0.15234375, -0.12890625, -0.34765625, -0.13769531, -0.18164062,
      0.37695312,  0.14160156, -0.03051758,  0.08203125,  0.09423828,
```

**Gambar 5.141** Praktek 1

- berikut merupakan hasil lpengolahan kata car pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [46]: genmod['car']
Out[46]:
array([ 0.13085938,  0.00842285,  0.03344727, -0.05883789,  0.04003906,
       -0.14257812,  0.04931641, -0.16894531,  0.20898438,  0.11962891,
       0.18066406, -0.25,        -0.10400391, -0.10742188, -0.01879883,
       0.05200195, -0.0021675,  0.06445312,  0.14453125, -0.04541016,
       0.16113281, -0.01611328, -0.03088379,  0.08447266,  0.16210938,
       0.04467773, -0.15527344,  0.25390625,  0.33984375,  0.00756836,
       -0.2558938, -0.01733398, -0.03295898,  0.16308594, -0.12597656,
       -0.09912109,  0.16503906,  0.06884766, -0.18945312,  0.02832031,
       -0.0534668, -0.03063965,  0.11083984,  0.24121094, -0.234375,
       0.12353516, -0.00294495,  0.1484375,  0.33203125,  0.05249023,
       -0.20019531,  0.37695312,  0.12255859,  0.11425781, -0.17675781,
       0.10009766,  0.0030365,  0.26757812,  0.20117188,  0.03710938,
       0.11083984, -0.09814453, -0.3125,        0.03515625,  0.02832031,
       0.26171875, -0.08642578, -0.02258301, -0.05834961, -0.00787354,
       0.11767578, -0.04296875, -0.17285156,  0.04394531, -0.23046875,
       0.1640625, -0.11474609, -0.06030273,  0.01196289, -0.24707031,
       0.32617188, -0.04492188, -0.11425781,  0.22851562, -0.016477949,
       -0.15839062, -0.13183594,  0.12597656, -0.17480469,  0.02209473,
       -0.1015625,  0.00817871,  0.10791016, -0.24609375, -0.109375,
       -0.09375, -0.01623535, -0.20214844,  0.23144531, -0.05444336,
       -0.05541992, -0.20898438,  0.26757812,  0.27929688,  0.17089844,
       -0.17578125, -0.02770996, -0.20410156,  0.02392578,  0.03125,
       -0.25390625, -0.125,        -0.05493164, -0.17382812,  0.28515625,
       -0.23242188,  0.0234375,  -0.20117188, -0.13476562,  0.26367188,
       0.00769043,  0.20507812, -0.01708984, -0.12988281,  0.04711914,
       0.22070312,  0.02099609, -0.29101562, -0.02893066,  0.17285156,
       0.04272461, -0.19824219, -0.04003906, -0.16992188,  0.10058594,
       -0.09326172,  0.15820312, -0.16503906, -0.06054688,  0.19433594,
```

Gambar 5.142 Praktek 1

- berikut merupakan hasil pengolahan kata wash pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [47]: genmod['wash']
Out[47]:
array([ 9.46044922e-03,  1.41601562e-01, -5.46875000e-02,  1.34765625e-01,
       -2.38281250e-01,  3.24218750e-01, -8.44726562e-02, -1.29882812e-01,
       1.07910156e-01,  2.53906250e-01,  1.13525391e-02, -1.66992188e-01,
       -2.79541016e-02,  2.08007812e-01, -4.27246094e-02,  1.05468750e-01,
       -7.42187500e-02,  3.04687500e-01,  2.11914062e-01, -8.8671875e-02,
       2.67578125e-01,  1.21890625e-01,  1.74560547e-02,  2.02941895e-03,
       2.9882812e-02,  1.62109375e-01,  1.93359375e-01,  2.17285156e-02,
       -2.67028809e-03, -9.13085938e-02, -2.38281250e-01,  2.23632812e-01,
       -8.00781250e-02, -3.80859375e-02, -1.00007656e-01, -1.39648438e-01,
       1.74804688e-01,  6.78710938e-02,  1.11328125e-01,  1.65039062e-01,
       -1.05468750e-01,  2.30712891e-02,  2.00195312e-01, -6.03027344e-02,
       -3.43750000e-01, -1.020507081e-01, -3.80859375e-01, -5.05371094e-02,
       5.07812500e-02,  1.45507812e-01,  2.81250000e-01,  7.03125000e-02,
       2.84423828e-02, -2.29492188e-01, -5.81054688e-02,  4.51660156e-02,
       -3.56445312e-02,  1.77734375e-01,  1.22070312e-01,  3.71093750e-02,
       -1.10839844e-01,  6.83593750e-02, -2.52685547e-02, -1.27929688e-01,
       4.21875000e-01,  5.32226562e-02, -3.92578125e-01,  1.74804688e-01,
       1.77001953e-02, -2.05078125e-02,  2.21679688e-01,  3.18359375e-01,
       1.08398438e-01, -4.30297852e-03, -2.45117188e-01, -2.08984375e-01,
       3.58886719e-02,  8.30078125e-02,  1.68945312e-01,  2.79541016e-02,
       1.04980469e-01, -3.47656250e-01, -5.20019531e-02,  2.24609375e-01,
       1.69677734e-02,  1.69921875e-01, -1.46484375e-01,  2.65625000e-01,
       2.17285156e-02,  1.12304688e-02, -1.14257812e-01,  7.22656250e-02,
       4.32128906e-02,  1.11694336e-02,  5.07354736e-04, -7.91015625e-02,
       -5.98144531e-02, -5.44433594e-02,  3.73046875e-01,  5.62500000e-01,
       -2.26562500e-01, -5.39550781e-02,  1.13769531e-01, -5.83496094e-02,
       -1.53320312e-01, -4.37500000e-01,  2.59765625e-01, -1.49414062e-01,
       5.66406250e-02,  2.13867188e-01, -2.86865234e-02, -1.70898438e-01,
```

Gambar 5.143 Praktek 1

- berikut merupakan hasil lpengolahan kata motor pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [48]: genmod['motor']
Out[48]:
array([ 5.73730469e-02,  1.50390625e-01, -4.61425781e-02, -1.32812500e-01,
       -2.59765625e-01, -1.7734375e-01,  3.68652344e-02, -4.37500000e-01,
       2.34375000e-02,  2.57812500e-01,  1.74804688e-01,  2.44140625e-02,
      -2.51953125e-01, -5.76171875e-02,  8.15429688e-02,  1.86767578e-02,
     -3.83300781e-02,  1.58203125e-01, -5.85937500e-02,  1.12304688e-01,
      1.56250000e-01, -4.24884688e-02, -1.32812500e-01,  2.11914062e-01,
     1.23046875e-01,  1.69921875e-01, -1.55273438e-01,  4.58984375e-01,
      3.02734375e-01,  1.53320312e-01, -1.69921875e-01, -1.01074219e-01,
     -3.26538086e-03,  2.28515625e-01,  9.84375000e-02, -7.12890625e-02,
      1.54296875e-01, -8.88671875e-02, -2.36328125e-01,  5.61523438e-03,
     -4.46777344e-02, -3.06640625e-01,  7.42187500e-02,  5.585937500e-01,
     -1.30858593e-01,  1.00585938e-01, -3.34472656e-02,  2.10937500e-01,
     3.10058594e-02, -6.50024414e-03,  6.34765625e-02,  4.02832031e-02,
     -2.78320312e-02,  1.07421875e-02,  1.47460938e-01,  2.80761719e-02,
     -1.50390625e-01, -1.37695312e-01,  9.96093750e-02,  1.28906250e-01,
     -3.34472656e-02, -1.08032227e-02, -2.14843750e-01, -9.52148438e-02,
     -6.39648438e-02,  7.51953125e-02, -3.06640625e-01,  2.17773438e-01,
     -2.21679688e-01,  2.33398438e-01,  5.05371094e-02, -3.37890625e-01,
      1.53320312e-01, -7.12890625e-02, -3.68652344e-02,  7.66601562e-02,
     -8.00781250e-02, -1.14257812e-01, -9.71679688e-02, -2.61718750e-01,
     3.84765625e-01, -1.87500000e-01, -1.10351562e-01,  1.00585938e-01,
     1.08398438e-01,  9.57031250e-02, -8.20312500e-02,  1.54296875e-01,
     -2.40234375e-01,  8.34960938e-02,  4.19921875e-02, -1.91650391e-02,
     9.71679688e-02,  2.52685547e-02, -5.46875000e-02, -5.88378906e-02,
     8.20312500e-02, -3.32031250e-01,  3.27148438e-02,  5.71289062e-02,
     1.77734375e-01, -9.57031250e-02,  2.45117188e-01,  6.88476562e-02,
     2.63671875e-01, -8.15429688e-02,  1.25976562e-01,  1.20849609e-02,
     4.00390625e-01,  8.69140625e-02, -3.00781250e-01, -1.99218750e-01,
```

**Gambar 5.144** Praktek 1

- berikut merupakan hasil lpengolahan kata cycle pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [49]: genmod['cycle']
Out[49]:
array([-0.04541016,  0.21679688, -0.02709961,  0.12353516, -0.20703125,
       -0.1328125 ,  0.26367188, -0.12890625, -0.125 ,  0.15332031,
       -0.18261719, -0.15820312, -0.06176758,  0.21972656, -0.15820312,
       0.02563477, -0.07568359, -0.0625 ,  0.04614258, -0.31054688,
      -0.13378906, -0.11669922, -0.3359375 ,  0.078125 ,  0.08447266,
      0.07226562, -0.06445312,  0.05517578,  0.14941406,  0.13671875,
      0.10302734,  0.02172852, -0.10693359,  0.02490234, -0.10644531,
      -0.05541992, -0.29492188, -0.40039062,  0.06347656, -0.08447266,
      0.17871094,  0.01165771, -0.01696777,  0.13671875, -0.1640625 ,
      0.11425781,  0.20800781, -0.06079102, -0.07275391,  0.15039062,
      0.18066406, -0.28515623, -0.040552734,  0.01806641,  0.00331116,
      0.00872803,  0.03564453, -0.29882812,  0.09960938, -0.1484375 ,
      -0.06787109,  0.05957031, -0.05517578, -0.19628906,  0.2265625 ,
      0.03173828, -0.07080078,  0.1484375 , -0.20214844, -0.03393555,
      0.09863281, -0.02038574, -0.08789062, -0.07226562, -0.09423282,
      -0.17089844,  0.1484375 ,  0.10546875,  0.06445312,  0.01031494,
      -0.02636719, -0.03686523, -0.125 , -0.06787109,  0.14257812,
      0.37109375, -0.15722656,  0.09326172,  0.34960938, -0.00091553,
      -0.03613281,  0.16894531, -0.02856445,  0.10791016, -0.32421875,
      -0.14355469,  0.03173828, -0.07421875,  0.34179688,  0.140625 ,
      0.0043335 , -0.12890625, -0.34960938, -0.02929688, -0.19628906,
      -0.2578125 , -0.3671875 ,  0.01483154,  0.20703125,  0.09667969,
      -0.10351562, -0.31054688, -0.02844238,  0.10400391,  0.17773438,
      0.06689453, -0.1796875 ,  0.02783203,  0.15625 ,  0.02026367,
      0.0324707 , -0.13476562,  0.15527344,  0.11132812, -0.01055908,
      0.07958984,  0.01989746,  0.25585938,  0.13378906,  0.02539062,
      0.10986328, -0.20605469,  0.07275391, -0.35546875, -0.02746582,
      -0.20800781,  0.10498047, -0.0625 , -0.01177979,  0.17382812,
```

Gambar 5.145 Praktek 1

- berikut merupakan hasil dari similaritas kata kata yang diolah menjadi matrix tadi adapun persentase untuk perbandingan setiap katanya yaitu 9 persen untuk kata wash dan clear 7 persen untuk kata bag dan love 48 persen untuk kata motor dan car 12 persen untuk kata sick dan faith dan terakhir yaitu 6 persen untuk kata cycle dan shine.

```
In [50]: genmod.similarity('wash', 'clear')
Out[50]: 0.09019176
```

```
In [51]: genmod.similarity('bag', 'love')
Out[51]: 0.07536096
```

```
In [52]: genmod.similarity('motor', 'car')
Out[52]: 0.4810173
```

```
In [53]: genmod.similarity('sick', 'faith')
Out[53]: 0.123073205
```

```
In [54]: genmod.similarity('cycle', 'shine')
Out[54]: 0.061617922
```

Gambar 5.146 Praktek 1

- jelaskan dengan kata dan ilustrasi fungsi dari extract words dan Permute Sentences

pada code berikut merupakan hasil dari running code untuk ekstrak word dimana pada baris ke tiga dimasukan perintah untuk menghapus tag html yang terdapat dalam file tersebut selanjutnya pada baris ke 4 yaitu perintah untuk menghilangkan tanda kutip satu selanjutnya pada baris ke 5 yaitu perintah untuk menghapus tanda baca pada file tersebut dan yang terakhir yaitu perintah untuk menghapus spasi atau sepasi berurutan. setelah itu dibuat class baru dari random yang bertujuan untuk mengkocok data yang ada pada file tersebut kemudian class permute sentence tersebut akan digunakan untuk mengolah data selanjutnya. untuk lebih jelasnya dapat dilihat pada gambar

```
In [55]: import re
...: def extract_words(sent):
...:     sent = sent.lower()
...:     sent = re.sub(r'<[^>]+>', ' ', sent) #hapus tag html
...:     sent = re.sub(r'(\w)\'(\w)', ' ', sent) #hapus petik satu
...:     sent = re.sub(r'\W', ' ', sent) #hapus tanda baca
...:     sent = re.sub(r'\s+', ' ', sent) #hapus spasi yang berurutan
...:     return sent.split()

In [56]: import random
...: class PermuteSentences(object):
...:     def __init__(self, sents):
...:         self.sents = sents
...:
...:     def __iter__(self):
...:         shuffled = list(self.sents)
...:         random.shuffle(shuffled)
...:         for sent in shuffled:
...:             yield sent
```

**Gambar 5.147** Praktek 2

3. Jelaskan fungsi dari librari gensim TaggedDocument dan Doc2Vec disertai praktek pemakaianya

gensim merupakan library untuk memodelkan topik unsupervised atau memodelkan bahasa dengan model unsupervised. Sadangkan taget dokumen digunakan untuk TaggedDokumen berarti memasukan dokumen untuk diolah oleh mesin. Kemudian Doc2Vec digunakan untuk membandingkan dokumen apakah isi dari dokumen itu bobotnya sama dengan dokumen yang disandingkan. menunjukan di baris ke satu dilakukan dari library gensim dokumen mengimport method TaggedDocument lalu pada baris kedua dari library gensim model melakukan import metod Doc2Vec.

```
In [57]: from gensim.models.doc2vec import TaggedDocument
...: from gensim.models import Doc2Vec
```

**Gambar 5.148** Praktek 3

4. Jelaskan dengan kata dan praktek cara menambahkan data training dari file yang dimasukkan kepada variabel dalam rangka melatih model doc2vac

cara memasukan data traning file pertama tentukan terlebih dahulu tempat file dokumen tersebut disimpan kemudian import librari os setelah itu buat variabel unsup\_sentences yang berisikan array kosong, lalu tentukan file yang akan dimasukan setelah itu lakukan os.listdir pada data yang akan dimasukan kemudian semua data tersebut di inisialisasi menjadi f kemudian nama f tersebut dimasukan ke variabel unsup. pada codingan tersebut merupakan praktikum untuk memasukan data doc2vec

```
In [23]: import os
...: unsup_sentences = []

In [24]: for dirname in ["train/pos", "train/neg", "train/unsup", "test/pos", "test/neg"]:
...:     for fname in sorted(os.listdir("D:/aclImdb/"+dirname)):
...:         if fname[-4:] == '.txt':
...:             with open("D:/aclImdb/"+dirname+"/"+fname,encoding='UTF-8') as f:
...:                 sent = f.read()
...:                 words = extract_words(sent)
...:                 unsup_sentences.append(TaggedDocument(words,[dirname+"/"+fname]))

In [25]: for dirname in ["txt_sentoken/pos", "txt_sentoken/neg"]:
...:     for fname in sorted(os.listdir("D:/aclImdb/"+dirname)):
...:         if fname[-4:] == '.txt':
...:             with open("D:/aclImdb/"+dirname+"/"+fname,encoding='UTF-8') as f:
...:                 for i, send in enumerate(f):
...:                     words = extract_words(sent)
...:                     unsup_sentences.append(TaggedDocument(words,[ "%s/%s-%d" %
...: (dirname, fname, i)]))

In [26]: with open("D:/stanforSentimentTreebank/original_rt_snippets.txt",
encoding='UTF-8') as f:
...:     for i, sent in enumerate(f):
...:         words = extract_words(sent)
...:         unsup_sentences.append(TaggedDocument(words,[ "rt-%d" % i]))
```

Gambar 5.149 Praktek 4

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

hal ini harus dilakukan supaya data lebis gembang untuk di olah dan meningkatkan tingkat akurasi dari proses pengolahan data Doc2Vec. kemudian harus dilakukan pembersihan data agar memori pc atau laptop yang di gunakan untuk mengolah data menjadi ringan dan menambah peforma dari mesin itu sendiri untuk codingan pertama lakukan terlebih dahulu randomisasi. selanjutnya membuat variabel baru dengan nama mute yang di isi data class random dan data unsup\_sentence. kemudian setelah pengolahan data dilakukan pembersihan dengan melakukan code.

```
In [27]: import re
...: def extract_words(sent):
...:     sent = sent.lower()
...:     sent = re.sub(r'<[^>]+>', ' ', sent) #hapus tag html
...:     sent = re.sub(r'(\w)\\'(\w)', ' ', sent) #hapus petik satu
...:     sent = re.sub(r'\W', ' ', sent) #hapus tanda baca
...:     sent = re.sub(r'\s+', ' ', sent) #hapus spasi yang berurutan
...:     return sent.split()

...:
...:
...: import random
...: class PermuteSentences(object):
...:     def __init__(self, sents):
...:         self.sents = sents
...:
...:     def __iter__(self):
...:         shuffled = list(self.sents)
...:         random.shuffle(shuffled)
...:         for sent in shuffled:
...:             yield sent

In [28]: mute=PermuteSentences(unsup_sentences)

In [29]: model = Doc2Vec(mute, dm=0, hs=1, vector_size=50)

In [30]: model.delete_temporary_training_data(keep_inference=True)
```

**Gambar 5.150** Praktek 5

6. Jelaskan dengan kata dan praktek kenapa model harus di save dan kenapa temporari training harus dihapus suapaya dalam pengolahan data tidak perlu menjalankan kembali data vektorisasi serta untuk meringankan beban ram. kemudian temporary harus dihapus guna meningkatkan peforma komputer.

```
In [40]: model.delete_temporary_training_data(keep_inference=True)
...: model.save('reza.d2v')
```

**Gambar 5.151** Praktek 6

7. jalankan dengan kta dan praktek maksud dari infer code inver\_code digunakan untuk membandingkan data doc2vec yang telah di olah dengan kata yang baru atau data yang ada dalam perintah vector itu sendiri contoh membandingkan kata (i will go home) untuk lebih jelasnya dapat di lihat pada gambar. kemudian untuk hasil running code tersebut dapat di lihat pada gambar pada hasil gambar tersebut terdapat hasil vektor yang rata rata berada pada kisaran 0,2 an yang berarti kata yang dimasukan pada inter\_vec datanya ada pada doc2vec atau ada data yang bobotnya menyamai kata-kata di dalam dokumen tersebut.

```
In [32]: model.infer_vector(extract_words("i will go home"))
Out[32]:
array([ 0.11367247,  0.01627037, -0.02634542, -0.00233531,  0.11615192,
       0.03835578, -0.09637795, -0.08611638, -0.19874588,  0.19353004,
      -0.13504118,  0.16723248,  0.05294943,  0.11971013,  0.15065596,
       0.00120262,  0.01779701, -0.11713073,  0.08604088, -0.10586435,
      0.01452718,  0.0278196 , -0.10640432, -0.00128474, -0.13638276,
      0.00356512,  0.1320768 ,  0.08737933, -0.14568117, -0.21087158,
      0.1913678 ,  0.33775824, -0.12074329,  0.0652655 ,  0.07574126,
      0.17250456, -0.01840756, -0.13633436,  0.02667661, -0.1061701 ,
     -0.2589361 ,  0.05849078, -0.05619296,  0.03440403,  0.12758958,
    -0.17287007, -0.0190108 ,  0.05780749,  0.12068204, -0.10444836 ],
       dtype=float32)
```

**Gambar 5.152** Praktek 7

8. Jelaskan dengan praktek dan kata maksud dari cosine similarity cosine\_simirarity setelah melakukan pengolahan data doc2vec dilakukan consine simirarity yang bertujuan untuk membandingkan data dengan yang telah di olah tadi apakah hasilnya mirip atau tidak untuk caranya yaitu dengan cara mencoba codingan yang terdapat pada gambar berikut maka akan muncul hasilnya berapa persen dengan tulisan 0.03 sekian yang berarti tingkat kemiripan dokumen yang kita uji tadi, untuk hasilnya dapat dilihat pada gambar berikut.

```
In [39]: from sklearn.metrics.pairwise import cosine_similarity
...: cosine_similarity([[model.infer_vector(extract_words("Jangan lupa cuci tangan"))],
...:                   [model.infer_vector(extract_words("Pake masker juga"))]])
Out[39]: array([[0.03864504]], dtype=float32)
```

**Gambar 5.153** Praktek 8

9. Jelaskan dengan praktek score dari cross validation  
Melakukan perhitungan presentase dengan menggunakan cross\_validation dengan metode kneighborsClassifier. Menggunakan dataset iris

```
In [37]: 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 0.08022311 0.03531764]
[0.33150734 0.08022311 0.03531764]
```

**Gambar 5.154** Praktek 9

### 5.6.3 Penanganan Error

1. FileNotFoundError Error

```

File "<ipython-input-41-e08ae423c176a>", line 2, in <module>
    for fname in sorted(os.listdir("aclImdb/"+dirname)):
FileNotFoundError: [WinError 3] The system cannot find the path specified: 'aclImdb/train/
pos'

```

**Gambar 5.155** FileNotFound Error

## 2. Jenis error

- FileNotFoundException

## 3. Cara Penanganan

Menyesuaikan tempat/direktori file berada

### 5.6.4 Bukti Tidak Plagiat



**Gambar 5.156** plagiarism

### 5.6.5 Link Video Youtube

<https://youtu.be/0hiPlSo83jY>

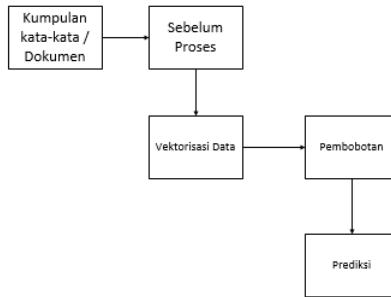
## 5.7 1174070 Arrizal Furqona Gifary

### 5.7.1 Teori

#### 1. Jelaskan Kenapa Kata-Kata harus dilakukan vektorisasi lengkapi dengan ilustrasi gambar.

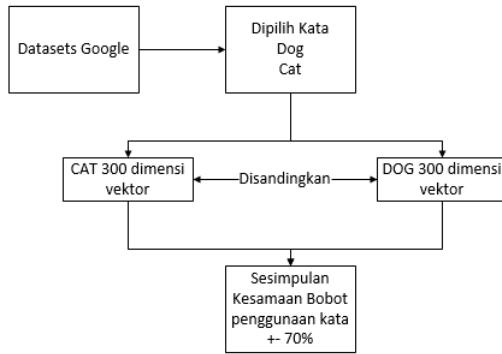
Kata kata harus dilakukan vektorisasi dikarenakan atau bertujuan utk mengukur nilai kemunculan suatu kata yang serupa dari sebuah kalimat sehingga kata-kata tersebut dapat di prediksi kemunculannya. atau juga di buatnya vektorisasi data digunakan untuk memprediksi bobot dari suatu kata misalkan ayam dan kucing sama-sama hewan maka akan dibuat prediksi apakah kata tersebut akan muncul pada kalimat yang kira-kira memiliki bobot yang sama.

#### 2. Jelaskan Mengapa dimensi dari vektor dataset google bisa mencapai 300 lengkapi dengan ilustrasi gambar.



Gambar 5.157 Teori 1

Dimensi dataset dari google bisa mencapai 300 karena dimensi dari vektor tersebut digunakan untuk membandingkan bobot dari setiap kata, misalkan terdapat kata dog dan cat pada dataset google tersebut setiap kata tersebut dibuat dimensi vektor 300 untuk kata dog dan 300 dimensi vektor juga untuk kata cat kemudian kata tersebut dibandingkan bobot kesamaan katanya maka akan muncul akurasi sekitar 70 persen kesamaan bobot dikarenakan kata dog dan cat sama-sama digunakan untuk hewan priharraaan.

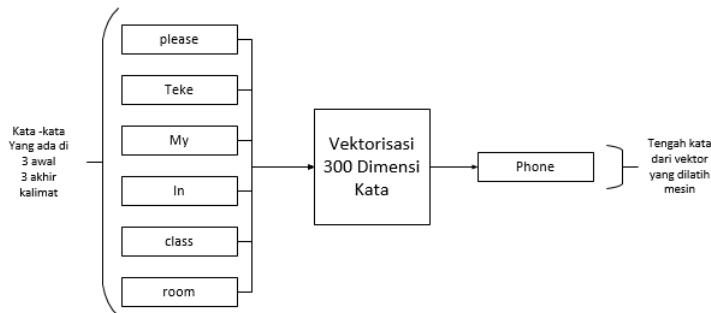


Gambar 5.158 Teori 2

3. Jelaskan Konsep vektorisasi untuk kata . dilengkapi dengan ilustrasi atau gambar.

Vektorisasi untuk kata untuk mengetahui kata tengah dari suatu kalimat atau kata utama atau objek utama pada suatu kalimat contoh ( Jangan lupa subscribe channel saya ya sekian terimakasih ) kata tengah tersebut merupakan channel yang memiliki bobot sebagai kata tengah dari suatu kalimat atau bobot sebagai objek dari suatu kalimat. hal ini san-

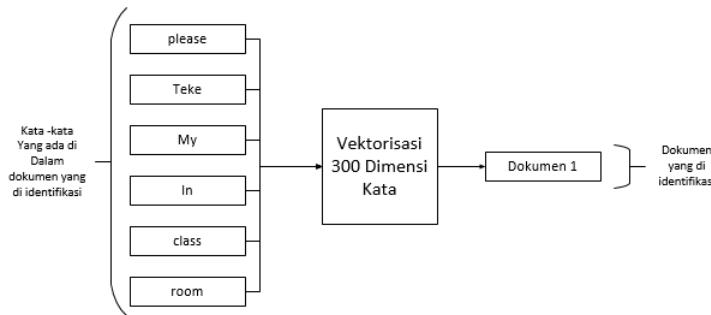
gat berkaitan dengan dimensi vektor pada dataset google yang 300 tadi karena untuk mendapatkan nilai atau bobot dari kata tengah tersebut di dapatkan dari proses dimensiasi dari kata tersebut.



### Gambar 5.159 Teori 3

4. Jelaskan Konsep vektorisasi untuk dokumen. dilengkapi dengan ilustrasi atau gambar.

Vektorisasi untuk dokumen hampir sama seperti vektorisasi untuk kata hanya saja pemilihan kata utama atau kata tengah terdapat pada satu dokumen jadi mesin akan membuat dimensi vektor 300 untuk dokumen dan nanti kata tengahnya akan di sandingkan pada dokumen yang terdapat pada dokumen tersebut.

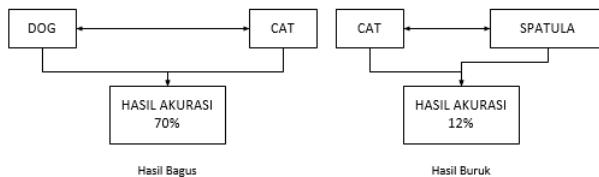


### Gambar 5.160 Teori 4

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

mean merupakan petunjuk terhadap kata-kata yang diolah jika kata-kata itu akurasinya tinggi berarti kata tersebut sering muncul begitu juga sebaliknya, sedangkan setandar defiasi merupakan standar untuk menim-

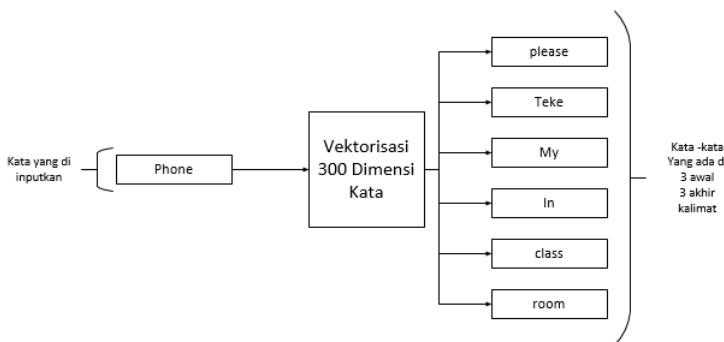
bang kesalahan. sehingga kesalahan tersebut di anggap wajar misalkan kita memperkirakan kedalaman dari dataset merupakan 2 atau 3 tapi pada kenyataannya merupakan 5 itu merupakan kesalahan tapi masih bisa dianggap wajar karna masih mendekati perkiraan awal.



**Gambar 5.161** Teori 5

6. Jelaskan Apa itu Skip-Gram sertakan contoh ilustrasi.

Skip-Gram adalah kebalikan dari konsep vektorisasi untuk kata dimana kata tengah menjadi acuan terhadap kata-kata pelengkap dalam suatu kalimat.



**Gambar 5.162** Teori 6

### 5.7.2 Praktikum

1. mencoba dataset google dan penjelasan vektor dari kata love, faith, fall, sick, clear, shine, bag, car, wash, motor, dan cycle.
  - berikut merupakan code import gensim digunakan untuk membuat data model atau rangcangan data yang akan dibuat. selanjutnya dibuat variabel gmodel yang berisi data vektor negatif. selanjutnya data tersebut di load agar data tersebut dapat ditampilkan dan diolah.

```
In [2]: import gensim
gmodel = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary = True)
```

**Gambar 5.163** Praktek 1

- berikut merupakan hasil lpengolahan kata love pada data google yang di load tadi. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [3]: gmodel['love']

Out[3]: array([ 0.10302734, -0.15234375,  0.02587891,  0.16503906, -0.16503906,
   0.06689453,  0.29296875, -0.26367188, -0.140625,  , 0.20117188,
  -0.02624512, -0.08203125, -0.02770996, -0.04394531, -0.23535156,
  0.16992188,  0.12898625,  0.15722656,  0.00756836, -0.06982422,
  -0.03857422, -0.07958984,  0.22949219, -0.14355469,  0.16796875,
  -0.03515625,  0.05517578,  0.10693359,  0.11181641, -0.16308594,
  -0.11181641,  0.13964844,  0.01556396,  0.12792969,  0.15429688,
  -0.07714844,  0.26171875,  0.08642578, -0.02514648,  0.33398438,
  0.18652344, -0.20996094,  0.07080078,  0.02600098, -0.10644531,
  -0.10253906,  0.12304688,  0.04711914,  0.02209473,  0.05834961,
  -0.10986328,  0.14941406, -0.10693359,  0.01556396,  0.08984375,
  0.11230469, -0.04378117, -0.11376953, -0.0037384 , -0.01818848,
  0.24316406,  0.08447266, -0.07080078,  0.18066406,  0.03515625,
  -0.09667969, -0.21972656, -0.00328064, -0.03198242,  0.18457031,
  0.28515625, -0.0859375 , -0.11181641,  0.0213623 , -0.30664062,
  -0.09228516, -0.18945312,  0.01513672,  0.18554688,  0.34375 ,
  -0.31054688,  0.22558594,  0.08740234, -0.2265625 , -0.29492188,
  0.08251953, -0.38476562,  0.25390625,  0.26953125,  0.06298828,
  -0.00958252,  0.23632812, -0.17871094, -0.12451172, -0.17285156,
  -0.11767578,  0.19726562, -0.03466797, -0.10400391, -0.1640625 ,
  -0.19726562,  0.19824219,  0.09521484,  0.00561523,  0.12597656,
  0.00073624, -0.0402832 , -0.030863965,  0.01623535, -0.1640625 ,
  -0.22167969,  0.171875 ,  0.12011719, -0.01965332,  0.4453125 ,
  0.06494141,  0.05932617, -0.1640625 , -0.01367188,  0.18945312,
  0.05566406, -0.05004883, -0.01422119,  0.15917969,  0.07421875,
  -0.31640625, -0.0534668 , -0.02355957, -0.16992188,  0.0625 ,
  -0.140625 , -0.13183594, -0.12792969,  0.12060547,  0.05883789,
```

**Gambar 5.164** Praktek 1

- berikut merupakan hasil lpengolahan kata faith pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata fall pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata sick pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata clear pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.

```
In [4]: gmodel['faith']
```

```
Out[4]: array([ 0.26367188, -0.04150391,  0.1953125 ,  0.13476562, -0.14648438,
   0.11962891,  0.04345703,  0.10351562,  0.12207031,  0.13476562,
   0.06640625,  0.18945312, -0.16601562,  0.21679688, -0.27148438,
   0.3203125 ,  0.10449219,  0.36132812, -0.1953125 , -0.18164062,
   0.15332031, -0.10839844,  0.10253906, -0.01367188,  0.23144531,
  -0.05957031, -0.22949219, -0.00604248,  0.26171875,  0.10302734,
  -0.1328125 ,  0.21484375,  0.01135254,  0.02111816,  0.18554688,
  0.04125977,  0.12011719,  0.17480469, -0.22167969, -0.13476562,
   0.3125 ,  0.06640625, -0.17675781, -0.01708984, -0.1640625 ,
  -0.02819824,  0.01257324, -0.09521484, -0.18066406, -0.140625 ,
  -0.02258301,  0.16308594, -0.13183594, -0.08007812,  0.13085938,
  0.27539062, -0.20605469,  0.10351562, -0.20214844, -0.1875 ,
  0.16992188,  0.13574219,  0.13769531,  0.16308594, -0.03881836,
  -0.11132812,  0.05688477,  0.12255859,  0.09814453, -0.04956055,
  -0.02331543, -0.04248047, -0.08203125,  0.16015625,  0.04150391,
  -0.16601562, -0.13671875,  0.09619141,  0.32617188,  0.08251953,
  -0.20800781,  0.04199219,  0.05834961, -0.27734375,  0.09130859,
  -0.17382812, -0.22460938,  0.03466797,  0.19824219, -0.08837891,
  0.18359375,  0.07324219,  0.1171875 , -0.33984375,  0.16796875,
  -0.13574219, -0.30078125, -0.00469971,  0.06005859, -0.29296875,
  0.15234375,  0.02966309,  0.33203125,  0.28320312,  0.09375 ,
  -0.20605469, -0.09082031,  0.0534668 ,  0.05834961, -0.03222656,
  -0.29296875,  0.25585938,  0.00430298,  0.140625 ,  0.05810547,
  0.21582031,  0.0291748 ,  0.02929688,  0.20019531,  0.34960938,
  0.10449219, -0.01940918,  0.04077148,  0.32226562, -0.1953125 ,
  -0.05688477,  0.10498847, -0.04785156, -0.15136719, -0.07714844,
  -0.45898438, -0.29492188, -0.328125 , -0.20996094,  0.38671875,
  0.0039978 .  0.07373047. -0.01220703.  0.22460938.  0.14550781.
```

**Gambar 5.165** Praktek 1

- berikut merupakan hasil lpengolahan kata shine pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata bag pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata car pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata wash pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata motor pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil lpengolahan kata cycle pada data google yang di load. Sehingga memunculkan hasil vektor 300 dimensi untuk kata tersebut.
- berikut merupakan hasil dari similaritas kata kata yang di olah menjadi matrix tadi adapun persentase untuk perbandingan setiap katanya

```
In [5]: gmodel['fall']
```

```
Out[5]: array([-0.04272461,  0.10742188, -0.09277344,  0.16894531, -0.1328125 ,
 -0.10693359,  0.04321289,  0.01904297,  0.14648438,  0.15039062,
 -0.08691406,  0.04492188,  0.0145874,   0.08691406, -0.19824219,
 -0.11035156,  0.01092529, -0.08300781, -0.0189209, -0.1953125,
 -0.1015625 ,  0.13671875,  0.09228516, -0.12109375,  0.12695312,
  0.03417969,  0.2109375,  0.01977539,  0.125,   0.01544189,
 -0.26953125, -0.0098877, -0.07763672, -0.15527344, -0.03393555,
  0.04199219, -0.29828212, -0.18554688,  0.08496094, -0.02087402,
  0.13574219, -0.22558594,  0.33789062, -0.03564453, -0.10839844,
 -0.19335938,  0.0546875, -0.04956055,  0.3671875, -0.03295898,
  0.10205078, -0.15136719, -0.00445557,  0.04003906,  0.27539062,
 -0.06935394,  0.05834961,  0.01422119, -0.01397705, -0.05395508,
 -0.0255127,  0.06298828,  0.07080078, -0.07617188,  0.06542969,
 -0.01672363, -0.047111914,  0.19628906, -0.08984375,  0.078125,
  0.2189375,  0.0612793,  0.08789062,  0.19628906,  0.11376953,
  0.06542969,  0.03125,  0.12982821,  0.02270508,  0.14550781,
 -0.06225586, -0.37695312, -0.05737305, -0.06396484,  0.08984375,
  0.00448608, -0.14160156, -0.04541016, -0.0703125,  0.06005859,
  0.26757812,  0.02001953, -0.12695312, -0.04882812,  0.18945312,
 -0.03466797,  0.04638672,  0.1484375,  0.01708984, -0.08789062,
 -0.14941406, -0.02331543, -0.03955078, -0.10400391, -0.14160156,
 -0.18261719, -0.03076172,  0.04589844, -0.2890625, -0.093540039,
  0.12890625, -0.10595783,  0.17578125,  0.06689453,  0.34960938,
  0.04296875,  0.09863281, -0.08956641, -0.06298828,  0.12255859,
  0.0234375, -0.06494141,  0.09667969, -0.04589844,  0.04956055,
  0.08007812, -0.00482178, -0.1646025, -0.03271484,  0.0703125,
 -0.07958984, -0.12890625, -0.01879883, -0.17773438,  0.01293945,
 -0.20019531,  0.08886719, -0.18847656, -0.23828125,  0.02758789,
 -0.14453125, -0.10058594, -0.0859375,  0.10205078, -0.00396729,
```

### Gambar 5.166 Praktek 1

yaitu 9 persen untuk kata wash dan clear 7 persen untuk kata bag dan love 48 persen untuk kata motor dan car 12 persen untuk kata sick dan faith dan terakhir yaitu 6 persen untuk kata cycle dan shine.

2. pada code berikut merupakan hasil dari running code untuk ekstrak word dimana pada baris ke tiga dimasukan perintah untuk menghapus tag html yang terdapat dalam file tersebut selanjutnya pada baris ke 4 yaitu perintah untuk menghilangkan tanda kutip satu selanjutnya pada baris ke 5 yaitu perintah untuk menghapus tanda baca pada file tersebut dan yang terakhir yaitu perintah untuk menghapus double sepsi atau sepsi berurutan. setelah itu dibuat class bari dari random yang bertujuan untuk mengkocok data yang ada pada file tersebut kemudian class permute sentence tersebut akan digunakan untuk mengolah data selanjutnya. untuk lebih jelasnya dapat dilihat pada gambar ??.
3. gensim merupakan library untuk memodelkan topik unsupervised atau memodelkan bahasa dengan model unsupervised. Saat ini taget dokumen digunakan untuk TaggetDokumen berarti memasukan dokumen untuk diolah oleh mesin. Kemudian Doc2Vec digunakan untuk membandingkan dokumen apakah isi dari dokumen itu bobotnya sama dengan dokumen yang di sandingkannya. menunjukkan di baris ke satu dilakukan dari librari gensim dokumen mengimport method taggedDocument lalu

```
In [6]: gmodel['sick']
```

```
Out[6]: array([-1.82617188e-01,  1.49414062e-01, -4.05273438e-02,  1.64062500e-01,
 -2.59765625e-01,  3.22265625e-01,  1.73828125e-01, -1.47460938e-01,
  1.01874219e-01,  5.46875000e-02,  1.66992188e-01, -1.68945312e-01,
  2.24304199e-03,  9.66796875e-02, -1.66015625e-01, -1.12304688e-01,
  1.66015625e-01,  1.79687500e-01,  5.92041016e-03,  2.45117188e-01,
  8.74023438e-02, -2.56347656e-02,  3.41796875e-01,  4.98046875e-02,
  1.78710938e-01, -9.91821289e-04,  8.88671875e-02, -1.95312500e-01,
  1.81640625e-01, -2.65625000e-01, -1.45597812e-01,  1.00585938e-01,
  9.42382812e-02, -3.12500000e-02,  1.98974609e-02, -6.39648438e-02,
  1.18652344e-01,  1.23046875e-01, -6.03027344e-02,  4.68750000e-01,
  9.13085938e-02, -3.12500000e-01,  1.84570312e-01, -1.51367188e-01,
  5.78613281e-02, -1.04980469e-01, -1.68945312e-01, -8.00781250e-02,
 -2.01171875e-01,  1.06201172e-02, -2.19882812e-01, -1.25976562e-01,
 -5.56640625e-02,  3.14453125e-01,  5.61523438e-02, -1.20117188e-01,
  7.12890625e-02,  4.37011719e-02,  2.05078125e-01,  5.71289062e-02,
  8.44726562e-02,  2.15820312e-01, -1.26953125e-01,  8.78906250e-02,
  2.48846875e-01, -6.54296875e-02, -2.02636719e-02,  1.52343750e-01,
 -3.57421875e-01,  3.02124023e-03, -2.08007812e-01, -5.05371094e-02,
  2.81982422e-02,  1.73828125e-01, -2.08007812e-01, -5.93261719e-02,
 -6.49414062e-02,  3.63769531e-02,  1.91406250e-01,  2.77343750e-01,
  3.54003906e-02,  1.56250000e-01, -1.03857422e-02,  2.26562500e-01,
 -4.66308594e-02, -5.17578125e-02, -1.63085938e-01,  4.17480469e-02,
  2.01171875e-01, -2.01171875e-01, -1.50756836e-02,  2.61718750e-01,
 -1.10839844e-01, -4.21875000e-01,  2.22167969e-02,  1.46484375e-01,
  4.19921875e-01, -6.88476562e-02,  9.42382812e-02, -1.96289062e-01,
 -9.42382812e-02, -3.12500000e-02,  6.34765625e-02,  2.47802734e-02,
 -1.61132812e-01, -1.53320312e-01,  1.31835938e-01, -1.81640625e-01,
```

**Gambar 5.167** Praktek 1

pada baris kedua dari librari gensim model melakukan import metod Doc2Vec.

4. cara memasukan data traning file pertama tentukan terlebih dahulu tempat file dokumen tersebut disimpan kemudian import librari os setelah itu buat variabel unsup\_sentences yang berisikan array kosong, lalu tentukan file yang akan dimasukan setelah itu lakukan os.listdir pada data yang akan dimasukan kemudian semua data tersebut di inisialisasi menjadi f kemudian nama f tersebut dimasukan ke variabel unsup. pada codingan tersebut merupakan praktikum untuk memasukan data doc2vec
5. kenapa harus dilakukan pengocokan data atau randomisasi ? hal ini harus dilakukan supaya data lebis gambang untuk di olah dan meningkatkan tingkat akurasi dari proses pengolahan data Doc2Vec. kemudian harus dilakukan pembersihan data agar memori pc atau laptop yang di gunakan untuk mengolah data menjadi ringan dan menambah peforma dari mesin itu sendiri untuk codingan pertama lakukan terlebih dahulu randomisasi. selanjutnya membuat variabel baru dengan nama mute yang di isi data class random dan data unsup\_sentence. kemudian setelah pengolahan data dilakukan pembersihan dengan melakukan code.
6. kenapa model harus di save ? suapaya dalam pengolahan data tidak perlu menjalankan kembali data vektorisasi serta untuk meringankan beban

```
In [7]: gmodel['clear']
```

```
Out[7]: array([-2.44140625e-04, -1.02050781e-01, -1.49414062e-01, -4.24804688e-02,
-1.67968750e-01, -1.46484375e-01, 1.76757812e-01, 1.46484375e-01,
2.26562500e-01, 9.76562500e-02, -2.67578125e-01, -1.29882812e-01,
1.24511719e-01, 2.23632812e-01, -2.13867188e-01, 3.10058594e-02,
2.00195312e-01, -4.76074219e-02, -6.83593750e-02, -1.21093750e-01,
3.22265625e-02, 3.14453125e-01, -1.11816406e-01, 8.00781250e-02,
-2.75878906e-02, -6.04248047e-03, -7.373046688e-02, -1.72851562e-01,
9.66796875e-02, -4.91333008e-03, -1.78710938e-01, -1.40380859e-03,
7.91015625e-02, 1.07910156e-01, -1.10351562e-01, -8.34960938e-02,
1.98974609e-02, -3.14331055e-03, 1.30615234e-02, 3.34472656e-02,
2.55859375e-01, -7.12890625e-02, 2.83203125e-01, -2.75390625e-01,
-8.49689375e-02, -3.73535156e-02, -9.17968750e-02, -1.30859375e-01,
3.49121094e-02, 7.32421875e-02, 2.17773438e-01, 5.00488281e-02,
7.47070312e-02, -1.25000000e-01, -5.73730469e-02, -2.74658203e-02,
-1.37329102e-02, -2.13867188e-01, -1.12792969e-01, -4.98046875e-02,
-1.92382812e-01, 1.32812500e-01, -5.00488281e-02, -1.66015625e-01,
-1.57470703e-02, -1.37695312e-01, 3.88183594e-02, 1.57226562e-01,
-1.52343750e-01, -1.64794922e-02, -2.27539062e-01, 3.34472656e-02,
1.55273438e-01, 1.65039062e-01, -1.66992188e-01, 3.14941406e-02,
2.85156250e-01, 2.69531250e-01, 3.32031250e-02, 2.41210938e-01,
1.39160156e-02, 5.12695312e-02, 9.76562500e-02, 3.14941406e-02,
2.51464844e-02, -2.85156250e-01, -1.24023438e-01, -6.88476562e-02,
-5.29785156e-02, 2.06054688e-01, -2.07031250e-01, -1.60156250e-01,
-2.61230469e-02, -3.01513672e-02, 8.66699219e-03, -1.30859375e-01,
3.88183594e-02, 8.60595703e-03, 5.31005859e-03, -6.05468750e-02,
1.03759766e-02, 1.33789062e-01, -1.89971924e-03, -4.27246094e-02,
-1.14746094e-01, -1.47705078e-02, 9.71679688e-02, -1.66992188e-01,
5.63964844e-02, -2.76184082e-03, -1.05468750e-01, -1.03027344e-01,
```

**Gambar 5.168** Praktek 1

ram. kemudian temporary harus dihapus guna meningkatkan peforma komputer.

7. inver\_code digunakan untuk membandingkan data doc2vec yang telah diolah dengan kata yang baru atau data yang ada dalam perintah vector itu sendiri contoh membandingkan kata (i will go home) untuk lebih jelasnya dapat di lihat pada gambar ???. kemudian untuk hasil running code tersebut dapat di lihat pada gambar ?? pada hasil gambar tersebut terdapat hasil vektor yang rata rata berada pada kisaran 0,2 an yang berarti kata yang dimasukan pada inter\_vec datanya ada pada doc2vec atau ada data yang bobotnya menyamai kata-kata di dalam dokumen tersebut.
8. consine\_simirarity setelah melakukan pengolahan data doc2vec dilakukan consine simirarity yang bertujuan untuk membandingkan data berisikan bahasa inggris dengan data yang telah di olah tadi apakah hasilnya mirip atau tidak untuk caranya yaitu dengan cara mencobacodingan yang terdapat pada gambar ?? berikut maka akan muncul hasilnya berapa persen dengan tulisan 0,4 sekian yang berarti tingkat kemiripan dokumen yang di uji tadi untuk hasilnya dapat dilihat pada gambar ?? berikut.
9. untuk melakukan cross validation pertama masukan terlebih dahulu metode KNeighborsClasifier dan RandomForestClasifier dari library sklearn ke-

```
In [8]: gmodel['shine']

Out[8]: array([-0.12402344,  0.25976562, -0.15917969, -0.27734375,  0.30273438,
   0.09960938,  0.39257812, -0.22949219, -0.18359375,  0.3671875 ,
  -0.10302734,  0.13671875,  0.25390625,  0.07128906,  0.02539062,
  0.21777344,  0.24023438,  0.5234375 ,  0.12304688, -0.19335938,
 -0.05883789,  0.0612793 , -0.01949918,  0.07617188,  0.05102539,
  0.20019531,  0.38085938,  0.00162506, -0.05029297,  0.14648438,
 -0.34765625,  0.02563477, -0.23925781, -0.04516682, -0.00479126,
 -0.24121094, -0.18945312, -0.15234375, -0.05493164,  0.01434326,
  0.390625 , -0.2109375 ,  0.1484375 , -0.13183594,  0.24511719,
 -0.24023438, -0.36132812, -0.12792969,  0.10595703,  0.09912109,
 -0.0246582 ,  0.32226562,  0.11376953,  0.18164062,  0.04931641,
 -0.10253906, -0.00283813,  0.29882812, -0.171875 , -0.18945312,
 -0.01367188, -0.20898438, -0.07861328, -0.0859375 ,  0.05395508,
 -0.14257812, -0.149625 ,  0.03027344, -0.14453125,  0.359375 ,
 0.16113281,  0.22265625,  0.265625 , -0.06347656, -0.02807617,
 0.04760742,  0.08837891, -0.04272461,  0.05988203,  0.07128906,
 0.01519775, -0.11621994,  0.07128906,  0.01403889, -0.10644531,
 0.08886719,  0.11523438,  0.09667969, -0.11083984,  0.16015625,
 0.3359375 , -0.1875 ,  0.14550781,  0.00463867,  0.07617188,
 -0.09521484,  0.08447266,  0.20117188,  0.11230469, -0.33984375,
 -0.25390625,  0.05200195,  0.27539062, -0.08398438, -0.31054688,
 -0.22949219,  0.14941406, -0.1953125 ,  0.08496094, -0.00753784,
 0.078125 ,  0.05908203,  0.02355957,  0.06347656,  0.32617188,
 -0.08740234,  0.10058594, -0.11474609, -0.18164062,  0.13378906,
 0.11230469, -0.00080109,  0.08691406,  0.03888594,  0.0300293 ,
 -0.03417969, -0.00830078,  0.14160156, -0.09619141, -0.11328125,
 0.00823975, -0.15234375,  0.19042969,  0.04980469,  0.25 ,
 -0.00171661.  0.04589844. -0.05102539. -0.04052734. -0.03491211.
```

Gambar 5.169 Praktek 1

mudian dilakukan cross validation setelah itu buat variabel clf dengan isi KNeighborsClasifier dan variabel clfrf dengan isi RandomForestClasifier kemudian di buat skor menggunakan cross validation dengan menggunakan variabel clf dan data sentvecs dan sentiments kemudian dengan numpy dibuat mean dari scores begitu pula untuk variabel clfrf selanjutnya melakukan import metode make\_pipeline yang dilakukan untuk membuat skor dari vektorisasi tfidf dan rf. untuk lebih jelasnya dapat di lihat pada gambar ?? maka akan muncul hasil rata-rata 0,76 sekian atau 76 persen untuk clf yang dapat dilihat pada gambar ?? dan untuk hasil clfrf menghasilkan hasil rata-rata di 71 persen yang dapat dilihat pada gambar ?? dan untuk hasil rata-rata keseluruhan cros validation sebesar 0,74 atau 74 persen yang dapat dilihat pada gambar ??.

```
1 from sklearn.neighbors import KNeighborsClassifier
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn.model_selection import cross_val_score
4 import numpy as np
5
6 clf = KNeighborsClassifier(n_neighbors=9)
7 clfrf = RandomForestClassifier()
8
9 scores = cross_val_score(clf, sentvecs, sentiments, cv=5)
10 print((np.mean(scores), np.std(scores)))
11
12 scores = cross_val_score(clfrf, sentvecs, sentiments, cv=5)
```

```
In [9]: gmodel['bag']

Out[9]: array([-0.03515625,  0.15234375, -0.12402344,  0.13378906, -0.11328125,
   -0.0133667, -0.16113281,  0.14648438, -0.06835938,  0.140625 ,
   -0.06005859, -0.3046875,  0.20996694, -0.04345703, -0.2109375,
   -0.05957031, -0.05053711,  0.10253906,  0.19042969, -0.09423828,
   0.18847656, -0.07958984, -0.11035156, -0.07910156,  0.06347656,
   -0.1527344, -0.18945312,  0.11132812,  0.27539062, -0.06787109,
   0.01806641,  0.06689453,  0.2578125,  0.0324707, -0.24609375,
   -0.05541992,  0.01813184,  0.24121094, -0.21875,  0.07568359,
   -0.09814453, -0.16113281,  0.16503906, -0.09521484, -0.16601562,
   -0.41796875,  0.0300293,  0.19433594,  0.2890625,  0.12695312,
   -0.19824219, -0.05517578,  0.04296875, -0.10107422,  0.07324219,
   -0.13378906,  0.265625, -0.00466919,  0.19628906, -0.10839844,
   0.14941406,  0.1484375,  0.09619141,  0.21777344, -0.08544922,
   -0.02819824,  0.02539662, -0.03759766,  0.23242188,  0.19628906,
   0.27539062,  0.09130859,  0.23738469,  0.09033203, -0.28515625,
   0.05932617,  0.06591797, -0.01794434, -0.00055313, -0.1796875 ,
   0.05615234, -0.12207831, -0.09863281, -0.05786133, -0.09375 ,
   -0.30273438, -0.06396484, -0.00744629, -0.17871094,  0.08544922,
   -0.20401056,  0.33789062,  0.00228882, -0.39453125, -0.14453125,
   -0.328125, -0.12695312, -0.08544922,  0.15234375,  0.03662109,
   -0.1484375,  0.05566406,  0.02844238,  0.07519531, -0.21484375,
   -0.15722656,  0.3359375, -0.04736328, -0.00405884, -0.19726562,
   0.27929688,  0.05566406, -0.10058594, -0.00811768, -0.20703125,
   0.03295898, -0.14550781, -0.15917969,  0.16503906,  0.234375 ,
   0.03588867,  0.04296875, -0.25,  0.1171875, -0.07714844,
   0.00521851,  0.125,  0.08886719,  0.15527344, -0.02185059,
   -0.15234375, -0.12890625, -0.34765625, -0.13769531, -0.18164062,
```

Gambar 5.170 Praktek 1

```
13 print((np.mean(scores), np.std(scores)))
14
15 # bag-of-words comparison
16 from sklearn.pipeline import make_pipeline
17 from sklearn.feature_extraction.text import CountVectorizer,
   TfIdfTransformer
18 pipeline = make_pipeline(CountVectorizer(), TfIdfTransformer()
   (), RandomForestClassifier())
19 scores = cross_val_score(pipeline, sentences, sentiments, cv
   =5)
20 print((np.mean(scores), np.std(scores)))
```

### 5.7.3 Penanganan Error

```
In [10]: gmodel['car']
```

```
Out[10]: array([ 0.13085938,  0.00842285,  0.03344727, -0.05883789,  0.04003906,
 -0.14257812,  0.04931641, -0.16894531,  0.20898438,  0.11962891,
  0.18066406, -0.25 ,   -0.10400391, -0.10742188, -0.01879883,
  0.05200195, -0.00216675,  0.06445312,  0.14453125, -0.04541016,
  0.16113281, -0.01611328, -0.03088379,  0.08447266,  0.16210938,
  0.04467773, -0.15527344,  0.25390625,  0.33984375,  0.00756836,
 -0.25585938, -0.01733398, -0.03295898,  0.16308594, -0.12597656,
 -0.09912109,  0.16503906,  0.06884766, -0.18945312,  0.02832031,
 -0.0534668 , -0.03063965,  0.11083984,  0.24121094, -0.234375 ,
  0.12353516, -0.02944495,  0.1484375 ,  0.33203125,  0.05249023,
 -0.20019531,  0.37695312,  0.12255859,  0.11425781, -0.17675781,
 0.10009766,  0.0030365 ,  0.26757812,  0.20117188,  0.03710938,
 0.11083984, -0.09814453, -0.3125 ,   0.03515625,  0.02832031,
 0.26171875, -0.08642578, -0.02258301, -0.05834961, -0.00787354,
 0.11767578, -0.04296875, -0.17285156,  0.04394531, -0.23046875,
 0.1640625 , -0.11474609, -0.06030273,  0.01196289, -0.24707031,
 0.32617188, -0.04492188, -0.11425781,  0.22851562, -0.01647949,
 -0.15039062, -0.13183594,  0.12597656, -0.17480469,  0.02209473,
 -0.1015625 ,  0.00817871,  0.10791016, -0.24609375, -0.109375 ,
 -0.09375 , -0.01623535, -0.20214844,  0.23144531, -0.05444336,
 -0.05541992, -0.20898438,  0.26757812,  0.27929688,  0.17089844,
 -0.17578125, -0.02770996, -0.20410156,  0.02392578,  0.03125 ,
 -0.25390625, -0.125 ,   -0.05493164, -0.17382812,  0.28515625,
 -0.23242188, -0.0234375 , -0.20117188, -0.13476562,  0.26367188,
 0.00769043,  0.20507812, -0.01708984, -0.12988281,  0.04711914,
 0.22070312,  0.02099609, -0.29101562, -0.02893066,  0.17285156,
 0.04272461, -0.19824219, -0.04003906, -0.16992188,  0.10058594,
 -0.09326172,  0.15820312, -0.16503906, -0.06054688,  0.19433594,
 -0.07080078, -0.06884766, -0.09619141, -0.07226562,  0.04882812,
```

Gambar 5.171 Praktek 1

```
In [11]: gmodel['wash']
```

```
Out[11]: array([ 9.46044922e-03,  1.41601562e-01, -5.46875000e-02,  1.34765625e-01,
 -2.38281250e-01,  3.24218750e-01, -8.44726562e-02, -1.29882812e-01,
 1.07910156e-01,  2.53906250e-01,  1.13525391e-02, -1.66992188e-01,
 -2.79541016e-02,  2.08007812e-01, -4.27246094e-02,  1.05468750e-01,
 -7.42187500e-02,  3.04687500e-01,  2.11914062e-01, -8.88671875e-02,
 2.67578125e-01,  2.12890625e-01,  1.74560547e-02,  2.02941895e-03,
 6.29882812e-02,  1.62109375e-01,  1.93359375e-01,  2.17285156e-02,
 -2.67028809e-03, -9.13085938e-02, -3.28821250e-01,  2.23632812e-01,
 -8.00781250e-02, -3.80859375e-02, -1.000097656e-01, -1.39648438e-01,
 1.74804688e-01,  6.78710938e-02,  1.11328125e-01,  1.65039062e-01,
 -1.05468750e-01,  2.30712891e-02,  2.00195312e-01, -6.03027344e-02,
 -3.43750000e-01, -1.02050781e-01, -3.80859375e-01, -5.05371094e-02,
 5.07812500e-02,  1.45507812e-01,  2.81250000e-01,  7.03125000e-02,
 2.84423828e-02, -2.29492188e-01, -5.81054688e-02,  4.51660156e-02,
 -3.56445312e-02,  1.77734375e-01,  1.22070312e-01,  3.71893750e-02,
 -1.10839844e-01,  6.83593750e-02, -2.52685547e-02, -1.27929688e-01,
 4.21875000e-01,  5.32226562e-02, -3.92578125e-01,  1.74804688e-01,
 1.77001953e-02, -2.05078125e-02,  2.21679688e-01,  3.18359375e-01,
 1.08398438e-01, -4.30297852e-03, -2.45117188e-01, -2.08984375e-01,
 3.58886719e-02,  8.30078125e-02,  1.68945312e-01,  2.79541016e-02,
 1.04980469e-01, -3.47656250e-01, -5.20019531e-02,  2.24609375e-01,
 1.69677734e-02,  1.69921875e-01, -1.46484375e-01,  2.65625000e-01,
 2.17285156e-02,  1.12304688e-02, -1.14257812e-01,  7.22656250e-02,
 4.32128906e-02,  1.11694336e-02,  5.07354736e-04, -7.91815625e-02,
 -5.98144531e-02, -5.44433594e-02,  3.73046875e-01,  5.62500000e-01,
 -2.26562500e-01, -5.39550781e-02,  1.13769531e-01, -5.83496094e-02,
 -1.53320312e-01, -4.37500000e-01,  2.59765625e-01, -1.49414062e-01,
 5.66406250e-02,  2.13867188e-01, -2.86865234e-02, -1.70898438e-01,
```

Gambar 5.172 Praktek 1

```
In [12]: gmodel['motor']
```

```
Out[12]: array([ 5.73730469e-02,  1.50390625e-01, -4.61425781e-02, -1.32812500e-01,
   -2.59765625e-01, -1.77734375e-01,  3.68652344e-02, -4.37500000e-01,
   2.34375000e-02,  2.57812500e-01,  1.74804688e-01,  2.44140625e-02,
   -2.51953125e-01, -5.76171875e-02,  8.15429688e-02,  1.86767578e-02,
   -3.83300781e-02,  1.58203125e-01, -5.85937500e-02,  1.12304688e-01,
   1.56250000e-01, -4.24884688e-02, -1.32812500e-01,  2.11914062e-01,
   1.23046875e-01,  1.69921875e-01, -1.55273438e-01,  4.58984375e-01,
   3.02734375e-01,  1.53320312e-01, -1.69921875e-01, -1.01074219e-01,
   -3.26538866e-03,  2.28515625e-01,  8.98437500e-02, -7.12890625e-02,
   1.54296875e-01, -8.88671875e-02, -2.36328125e-01,  5.61523438e-03,
   -4.4677344e-02, -3.06640625e-01,  7.42187500e-02,  5.58593750e-01,
   -1.30859375e-01,  1.00585938e-01, -3.34472656e-02,  2.10937500e-01,
   3.10058594e-02, -6.50024414e-03,  6.34765625e-02,  4.02832031e-02,
   -2.78320312e-02,  1.07421875e-02,  1.47468938e-01,  2.80761719e-02,
   -1.50390625e-01, -1.37695312e-01,  9.96093750e-02,  1.28906250e-01,
   -3.34472656e-02, -1.08032272e-02, -2.14843750e-01, -9.52148438e-02,
   -6.39648438e-02,  7.51953125e-02, -3.06640625e-01,  2.17773438e-01,
   -2.21679688e-01,  2.33398438e-01,  5.05371094e-02, -3.37890625e-01,
   1.53320312e-01, -7.12890625e-02, -3.68652344e-02,  7.66601562e-02,
   -8.00781250e-02, -1.14257812e-01, -9.71679688e-02, -2.61718750e-01,
   3.84765625e-01, -1.87500000e-01, -1.10351562e-01,  1.00585938e-01,
   1.08398438e-01,  9.57031250e-02, -8.20312500e-02,  1.54296875e-01,
   -2.40234375e-01,  8.34960938e-02,  4.19921875e-02, -1.91650391e-02,
   9.71679688e-02,  2.52685547e-02, -5.46875000e-02, -5.88378906e-02,
   8.20312500e-02, -3.32031250e-01,  3.27148438e-02,  5.71289062e-02,
   1.77734375e-01, -9.57031250e-02,  2.45117188e-01,  6.88476562e-02,
   2.63671875e-01, -8.15429688e-02,  1.25976562e-01,  1.20884960e-02,
   4.00390625e-01,  8.69140625e-02, -3.00781250e-01, -1.99218750e-01,
```

Gambar 5.173 Praktek 1

```
In [13]: gmodel['cycle']
```

```
Out[13]: array([ 0.04541016,  0.21679688, -0.02709961,  0.12353516, -0.20703125,
   -0.1328125 ,  0.26367188, -0.12890625, -0.125 ,  0.15332031,
   -0.18261719, -0.15820312, -0.06176758,  0.21972656, -0.15820312,
   0.02563477, -0.07568359, -0.0625 ,  0.04614258, -0.31054688,
   -0.13378906, -0.11669922, -0.3359375 ,  0.078125 ,  0.08447266,
   0.07226562, -0.06445312,  0.05517578,  0.14941406,  0.13671875,
   0.10302734,  0.02172852, -0.10693359,  0.02490234, -0.10644531,
   -0.05541992, -0.29492188, -0.40039662,  0.06347656, -0.08447266,
   0.17871094,  0.01165771, -0.01696777,  0.13671875, -0.1648625 ,
   0.11425781,  0.02800781, -0.06079102, -0.07275391,  0.15039062,
   0.18066406, -0.28515625, -0.04052734,  0.01806641,  0.00331116,
   0.00872803, -0.03564453, -0.29882812,  0.09960938, -0.1484375 ,
   -0.06787109,  0.095957031, -0.05517578, -0.19628906,  0.2265625 ,
   0.03173828, -0.07080078,  0.1484375 , -0.20214844, -0.03393555,
   0.09863281, -0.02038574, -0.08789662, -0.07226562, -0.09423828,
   -0.17088944,  0.1484375 ,  0.10546875,  0.06445312,  0.01031494,
   -0.02636719, -0.03686523, -0.125 , -0.06787109,  0.14257812,
   0.37109375, -0.15722656,  0.09326172,  0.34960938, -0.00091553,
   -0.03613281,  0.16894531, -0.02856445,  0.18791016, -0.32421875,
   -0.14355469,  0.03173828, -0.07421875,  0.34179688,  0.140625 ,
   0.0043335 , -0.12890625, -0.34960938, -0.02929688, -0.19628906,
   -0.2578125 , -0.3671875 ,  0.01483154,  0.20703125,  0.09667969,
   -0.10315162, -0.31054688,  0.02844238,  0.10400391,  0.17773438,
   0.06689453, -0.1796875 ,  0.02783203,  0.15625 ,  0.02026367,
   0.0324707 , -0.13476562,  0.15527344,  0.11132812, -0.01055908,
   0.07958984,  0.01989746,  0.25585938,  0.13378906,  0.02539062,
   0.10986328, -0.20605469,  0.07275391, -0.35546875, -0.02746582,
```

Gambar 5.174 Praktek 1

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

```
Out[15]: 0.09019176
```

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

```
Out[16]: 0.07536096
```

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

```
Out[17]: 0.4810173
```

```
In [18]: gmodel.similarity('sick', 'faith')
```

```
Out[18]: 0.123073205
```

```
In [19]: gmodel.similarity('cycle', 'shine')
```

```
Out[19]: 0.061617922
```

**Gambar 5.175** Praktek 1

```
In [20]: import re
def extract_words(sent):
    sent = sent.lower()
    sent = re.sub(r'<[^>]+>', ' ', sent) #hapus tag html
    sent = re.sub(r'(\w)\`(\w)', ' ', sent) #hapus petik satu
    sent = re.sub(r'\W', ' ', sent) #hapus tanda baca
    sent = re.sub(r'\s+', ' ', sent) #hapus spasi yang berurutan
    return sent.split()
```

```
In [21]: import random
class PermuteSentences(object):
    def __init__(self, sents):
        self.sents = sents

    def __iter__(self):
        shuffled = list(self.sents)
        random.shuffle(shuffled)
        for sent in shuffled:
            yield sent
```

**Gambar 5.176** Praktek 2

```
In [22]: from gensim.models.doc2vec import TaggedDocument
from gensim.models import Doc2Vec
```

```
In [ ]:
```

**Gambar 5.177** Praktek 3

```
In [53]: import os
unsup_sentences = []

In [54]: for dirname in ["train/pos", "train/neg", "train/unsup", "test/pos", "test/neg"]:
    for fname in sorted(os.listdir("aclimdb/" + dirname)):
        if fname[-4:] == '.txt':
            with open("aclimdb/" + dirname + "/" + fname, encoding='UTF-8') as f:
                sent = f.read()
                words = extract_words(sent)
            unsup_sentences.append(TaggedDocument(words, [dirname + "/" + fname]))

In [55]: for dirname in ["txt_sentoken/pos", "txt_sentoken/neg"]:
    for fname in sorted(os.listdir(dirname)):
        if fname[-4:] == '.txt':
            with open("aclimdb/" + dirname + "/" + fname, encoding='UTF-8') as f:
                for i, sent in enumerate(f):
                    words = extract_words(sent)
            unsup_sentences.append(TaggedDocument(words, ["%s/%s-%d" % (dirname, fname, i)]))

In [56]: with open("stanforSentimentTreebank/original_rt_snippets.txt", encoding='UTF-8') as f:
    for i, sent in enumerate(f):
        words = extract_words(sent)
    unsup_sentences.append(TaggedDocument(words, ["rt-%d" % i]))
```

Gambar 5.178 Praktek 4

```
In [77]: import re
def extract_words(sent):
    sent = sent.lower()
    sent = re.sub(r'<[^>]+>', ' ', sent) #hapus tag html
    sent = re.sub(r'(\w)\'(\w)', ' ', sent) #hapus petik satu
    sent = re.sub(r'\W', ' ', sent) #hapus tanda baca
    sent = re.sub(r'\s+', ' ', sent) #hapus spasi yang berurutan
    return sent.split()

import random
class PermuteSentences(object):
    def __init__(self, sents):
        self.sents = sents

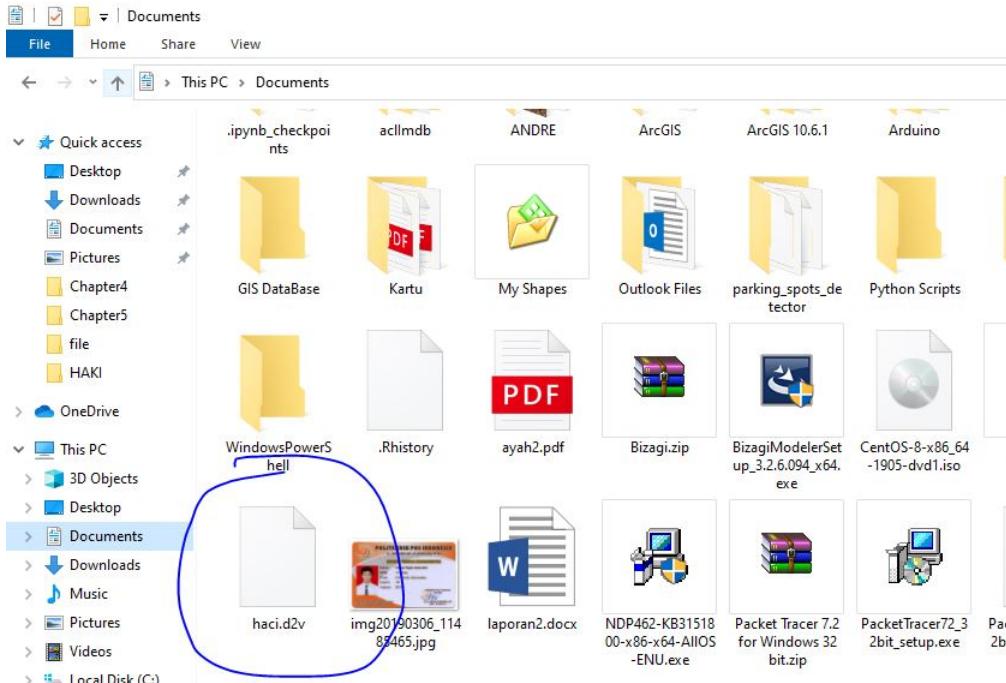
    def __iter__(self):
        shuffled = list(self.sents)
        random.shuffle(shuffled)
        for sent in shuffled:
            yield sent

mute=PermuteSentences(unsup_sentences)
model = Doc2Vec(mute, dm=0, hs=1, size=50)
model.delete_temporary_training_data(keep_inference=True)
```

Gambar 5.179 Praktek 5

```
In [78]: model.delete_temporary_training_data(keep_inference=True)
model.save('haci.d2v')
```

Gambar 5.180 Praktek 6



Gambar 5.181 Praktek 6

```
In [79]: model.infer_vector(extract_words("I will go home"))
```

```
Out[79]: array([-1.7533980e-01,  1.7654952e-01,   8.7283678e-02,   1.7998287e-02,
 6.2501184e-03, -3.2540500e-02,   1.6391091e-01,  -1.3786003e-02,
 5.1714227e-02,  4.6473891e-02,   1.6999269e-02,  2.6260551e-02,
 -2.5963100e-02,  7.7627085e-02,  -1.5409571e-02,  1.3232067e-01,
 -4.3878101e-02,  4.8159737e-02,  -5.1153481e-02, -7.2681673e-02,
 -3.1809844e-02, -2.4982829e-02,   1.1039837e-01,  7.9181477e-02,
 6.1015221e-03,  6.1555382e-02,  -2.2586747e-01,  1.3859421e-01,
 1.6294651e-02, -6.1351705e-02,   1.3722880e-01,  -7.6827303e-02,
 -1.4081554e-01,  4.0382754e-02,  -1.2432394e-01, -3.1883363e-02,
 6.4639568e-02,  7.1126200e-02,  -4.4149000e-02, -5.9015196e-02,
 1.0176090e-04,  1.0674181e-02,   6.5928474e-02,  3.7513527e-03,
 5.5374358e-02, -1.3092439e-02,  -3.5841893e-02,  4.9701292e-02,
 -1.1787581e-01,  9.5303327e-02], dtype=float32)
```

Gambar 5.182 Praktek 7

```
In [83]: from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(
    [model.infer_vector(extract_words("she going to school, after wash hand"))],
    [model.infer_vector(extract_words("Services sucks2."))])

Out[83]: array([[0.04744839]], dtype=float32)

In [84]: from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(
    [model.infer_vector(extract_words("Dia pergi ke sekolah tadi siang"))],
    [model.infer_vector(extract_words("Services sucks2."))])

Out[84]: array([[0.2095491]], dtype=float32)
```

Gambar 5.183 Praktek 8



## BAB 6

---

# CHAPTER 6

---

### 6.1 1174006 - Kadek Diva Krishna Murti

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

```
1 @inproceedings{awangga2017colenak,
2   title={Colenak: GPS tracking model for post-stroke
3   rehabilitation program using AES-CBC URL encryption and QR-
4   Code},
5   author={Awangga, Rolly Maulana and Fathonah, Nuraini Siti and
6   Hasanudin, Trisna Irmayadi},
7   booktitle={Information Technology, Information Systems and
8   Electrical Engineering (ICITISEE), 2017 2nd International
   conferences on},
9   pages={255--260},
10  year={2017},
11  organization={IEEE}
12 }
```



Gambar 6.1 Kecerdasan Buatan.

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

#### 6.1.1 Teori

#### 6.1.2 Praktek

#### 6.1.3 Penanganan Error

#### 6.1.4 Bukti Tidak Plagiat



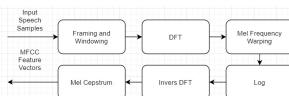
Gambar 6.2 Kecerdasan Buatan.

## 6.2 1174066 - D.Irga B. Naufal Fakhri

#### 6.2.1 Teori

##### 6.2.1.1 *Kenapa file suara harus di lakukan MFCC*

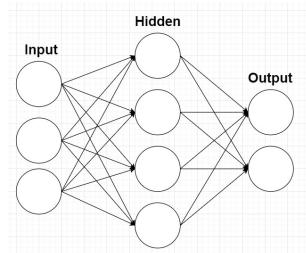
Karena MFCC adalah koefisien yang mewakili audio. Ekstraksi fitur dalam proses ini ditandai dengan konversi data suara menjadi gambar spektrum gelombang. File audio dilakukan oleh MFCC sehingga objek suara dapat dikonversi menjadi matriks. Suara akan menjadi vektor yang akan diproses sebagai output.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



**Gambar 6.3** Teori 1

#### 6.2.1.2 Konsep dasar neural network

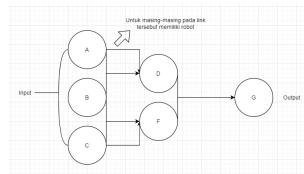
Konsep sederhana dari neural network atau jaringan saraf sederhana dengan proses pembelajaran pada anak-anak dengan memetakan pola-pola baru yang diperoleh dari input untuk membuat pola-pola baru pada output. Contoh sederhana ini menganalogikan kinerja otak manusia. Jaringan saraf itu sendiri terdiri 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 6.4** Teori 2

#### 6.2.1.3 Konsep pembobotan dalam neural network

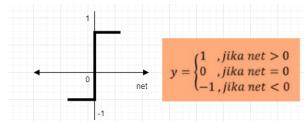
Pembobotan di dalam neural network juga akan menentukan penanda koneksi. Dalam proses neural network mulai dari input yang diterima oleh neuron bersama dengan nilai bobot masing-masing input. Setelah memasuki neuron, nilai input akan ditambahkan oleh fungsi penerima. Hasil penambahan ini akan diproses 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 6.5** Teori 3

#### 6.2.1.4 Konsep fungsi aktifasi dalam neural network

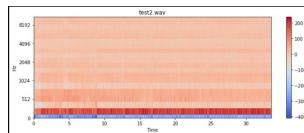
Fungsi aktivasi dalam neural network ialah merupakan suatu operasi matematik yang dikenakan pada sinyal output. Fungsi ini digunakan untuk mengaktifkan atau menonaktifkan neuron. Fungsi aktivasi ini terbagi setidaknya menjadi 6, adapun sebagai berikut:

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 6.6** 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 terseksi adalah kekuatan atau nilai tertinggi dibandingkan dengan warna canggih. Yang merah terdengar di bawah pendengaran manusia, sehingga tidak bisa didengar secara langsung.



**Gambar 6.7** 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.

```
label_and_ids = label_row_ids + np.arange(1, len(label), return_inverse=True, dtype='integer')
label_row_ids = label_row_ids.astype(np.int32, copy=False)
onehot_labels = to_categorical(label_and_ids, len(label.unique_ids))#one hot
onehot_labels = np.array(onehot_labels).T
```

**Gambar 6.8** Teori 6-1

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

→

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 6.9** 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
>>> x = np.array([[1,1],[1,1],[2,3],[4,5]])
>>> np.unique(x)
array([1, 2, 3, 4, 5])
```

**Gambar 6.10** 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','i','n','g','a'])
>>> print(var)
[ a, a, y, d, i, r, g ]  Length=8, categories: [a, d, g, i, r, s, y] 8 items
Categories (7, object): [a, d, g, i, r, s, y] 8 items
```

**Gambar 6.11** 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 6.12 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 import np_utils
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 glob
...: import numpy as np
...: import matplotlib.pyplot as plt
...: from keras.layers import Dense, Activation
...: from keras.models import Sequential
...: from keras.utils import np_utils
...:
...: def display_mfcc(song):
...:     y, _ = librosa.load(song)
...:     mfcc = librosa.feature.mfcc(y)
...:
...:     plt.figure(figsize=(10, 4))
...:     librosa.display.specshow(mfcc, x_axis='time', y_axis='mel')
...:     plt.colorbar()
...:     plt.title(song)
...:     plt.tight_layout()
...:     plt.show()
Using TensorFlow backend.
```

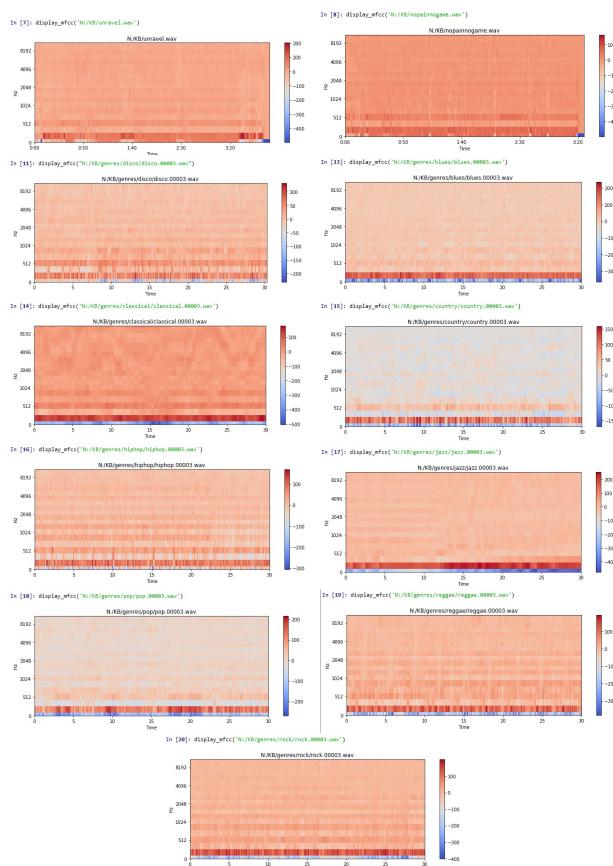
Gambar 6.13 Nomor 1

### 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 6.14 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 dua 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 between 0,1 (divide by max)
    ...
    mfcc /= np.max(np.abs(mfcc))
    ...
    return np.ndarray.flatten(mfcc)[:25000]
```

Gambar 6.15 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',
6               'jazz', '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(
10            sound_files), genre))
11         for f in sound_files:
12             features = extract_features_song(f)
13             all_features.append(features)
14             all_labels.append(genre)
15
16     # convert labels to one-hot encoding cth blues : 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(
22     label_uniq_ids)) #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():
    ...
    all_features = []
    all_labels = []
    ...
    genres = ['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae', 'rock']
    ...
    for genre in genres:
        sound_files = glob.glob('N:/KB/genres/' + genre + '/*.au')
        for f in sound_files:
            features = extract_features_song(f)
            all_features.append(features)
            all_labels.append(genre)
    ...
    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 6.16 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, -0.00000079, -0.00000001, ..., -0.00000004, 0.00000004]
labels	float32 [1000, 10]	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

**Gambar 6.17** 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.

The screenshot shows a Jupyter Notebook interface with several code cells and their outputs.

```

In [28]: training_split = 0.8

```

```

alldata  float32 (10000, 29500) [[-0.003160 ... -0.301305 ... -0.7460887 ... 0. 0.]
features  float32 (10000, 29500) [[-0.0030975 ... -0.3013435 ... -0.7334447 ... -0.4101034 ... 0.032724]
labels    float32 (10000, 10) [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
splitsize int64 3000
train    float32 (7000, 29500) [[-0.003160 ... -0.301305 ... -0.7460887 ... 0. 0.]
test     float32 (3000, 29500) [[-0.003160 ... -0.301305 ... -0.7460887 ... 0. 0.]
training_split float32 1.0

```

```

In [29]: np.random.shuffle(alldata)
...: np.random.shuffle(features)
...: np.random.shuffle(labels)
...: train, test = alldata[:splitsize], alldata[splitsize:]
...: train, test = train[:int(training_split)], train[int(training_split):]
...: train_labels = train[:, -10]
...: test_labels = test[:, -10]

```

```

train  float32 (7000, 29500) [[-0.003160 ... -0.301305 ... -0.7460887 ... 0. 0.]
test   float32 (3000, 29500) [[-0.003160 ... -0.301305 ... -0.7460887 ... 0. 0.]
train_labels float32 (7000, 10) [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
test_labels float32 (3000, 10) [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

```

```

In [31]: print(np.shape(train))
...: print(np.shape(test))
(800, 25010)
(200, 25010)

```

```

In [32]: train_input = train[:, :-10]
...: train_labels = train[:, -10]

```

```

In [34]: print(np.shape(train_input))
...: print(np.shape(train_labels))
(800, 25000)
(800, 10)

```

Gambar 6.18 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 6.19 Nomor 7

### 6.2.2.8 Nomor 8

```

1 model.compile(optimizer='adam',

```

```

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

```

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 parameteranya 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)           25000.00
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: 2,401,110
Non-trainable params: 0
None

```

**Gambar 6.20** 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 1/10
Train on 888 samples, validate on 222 samples
1/1 [00:00<00:00] - ETA: 0s - loss: 1.769 - acc: 0.769 - val_loss: 1.769 - val_accuracy: 0.769
Epoch 2/10
1/1 [00:00<00:00] - ETA: 0s - loss: 1.009 - accuracy: 0.848 - val_loss: 1.009 - val_accuracy: 0.848
Epoch 3/10
1/1 [00:00<00:00] - ETA: 0s - loss: 0.941 - accuracy: 0.865 - val_loss: 0.941 - val_accuracy: 0.865
Epoch 4/10
1/1 [00:00<00:00] - ETA: 0s - loss: 0.934 - accuracy: 0.874 - val_loss: 0.934 - val_accuracy: 0.874
Epoch 5/10
1/1 [00:00<00:00] - ETA: 0s - loss: 0.934 - accuracy: 0.874 - val_loss: 0.934 - val_accuracy: 0.874
Epoch 6/10
1/1 [00:00<00:00] - ETA: 0s - loss: 0.934 - accuracy: 0.874 - val_loss: 0.934 - val_accuracy: 0.874
Epoch 7/10
1/1 [00:00<00:00] - ETA: 0s - loss: 0.934 - accuracy: 0.874 - val_loss: 0.934 - val_accuracy: 0.874
Epoch 8/10
1/1 [00:00<00:00] - ETA: 0s - loss: 0.934 - accuracy: 0.874 - val_loss: 0.934 - val_accuracy: 0.874
Epoch 9/10
1/1 [00:00<00:00] - ETA: 0s - loss: 0.934 - accuracy: 0.874 - val_loss: 0.934 - val_accuracy: 0.874
Epoch 10/10
1/1 [00:00<00:00] - ETA: 0s - loss: 0.934 - accuracy: 0.874 - val_loss: 0.934 - val_accuracy: 0.874

```

**Gambar 6.21** Nomor 9

#### 6.2.2.10 Nomor 10

```

1 loss, acc = model.evaluate(test_input, test_labels, batch_size
2 =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("Done!")
... :    print("Loss: %.4f, accuracy: %.4f" % (loss, acc))
200/200 [=====] - 0s 459us/step
Done!
100%|██████████| 2/2 [00:00<00:00, 0.515s]
```

**Gambar 6.22** Nomor 10

#### 6.2.2.11 Nomor 11

```
1 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]:
```

Gambar 6.23 Nomor 11

### 6.2.3 Penanganan Error

### 6.2.3.1 Error

- #### ■ NoBackendError

**Gambar 6.24** NoBackendError

### 6.2.3.2 Solusi Error

- NoBackendError

Cek kembali file format audionya dipastikan menggunakan wav

#### 6.2.4 Bukti Tidak Plagiat



Gambar 6.25 Bukti tidak plagiat

#### 6.2.5 Link Youtube

[https://youtu.be/zA7U-zfvI\\_w](https://youtu.be/zA7U-zfvI_w)

## BAB 7

---

# CHAPTER 7

---

### 7.1 1174006 - Kadek Diva Krishna Murti

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

```
1 @inproceedings{awangga2017colenak,
2   title={Colenak: GPS tracking model for post-stroke
3   rehabilitation program using AES-CBC URL encryption and QR-
4   Code},
5   author={Awangga, Rolly Maulana and Fathonah, Nuraini Siti and
6   Hasanudin, Trisna Irmayadi},
7   booktitle={Information Technology, Information Systems and
8   Electrical Engineering (ICITISEE), 2017 2nd International
   conferences on},
9   pages={255--260},
10  year={2017},
11  organization={IEEE}
12 }
```



**Gambar 7.1** Kecerdasan Buatan.

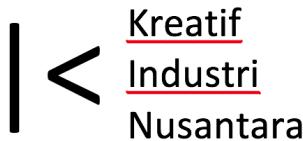
1. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

#### **7.1.1 Teori**

#### **7.1.2 Praktek**

#### **7.1.3 Penanganan Error**

#### **7.1.4 Bukti Tidak Plagiat**



**Gambar 7.2** Kecerdasan Buatan.