LAPORAN PROJECT UAS DATA MINING

"Text Mining Dengan Python"

Disusun Oleh:

Kelompok 4

1.	Afifa Witania	081911633001
2.	Nikita Lia Megawati	081911633015
3.	Ardina Dana Nugraha	081911633028
4.	Nadhif H. Risantosa	081911633070
5.	Ajyan Brava Bietrosula	081911633073
6.	Intalitha Fulka Hajar Amethys	081911633074

Dosen Pengampu:

Endah Purwanti, S.Si., M.Kom.

PRODI S1 SISTEM INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS AIRLANGGA

TAHUN AJARAN 2021/2022

DAFTAR ISI

PERCOBAAN

A. *Preprocessing* Dataset

Preprocessing data dilakukan dengan mengolah data mentah menjadi sekumpulan informasi yang lebih berkualitas, efisien, dan layak untuk digunakan. Tahapan *preprocessing data* terdiri dari *case folding*, perbaikan *term, tokenizing, remove stopword, stemming* dan pembobotan *term. Library* yang digunakan dalam *preprocessing* kasus ini adalah sebagai berikut.

```
#import library
import pandas as pd
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import re
import string
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
```

Gambar 1. Library yang digunakan untuk tahap preprocessing

Dataset yang digunakan adalah dataset dengna format excel (.xlsx) yang diberi nama "Data UAS Datamining.xlsx". Dataset yang digunakan terdiri dari 30 dokumen.

	ID	Teks
0	1	Selamat malam dok saya ingin bertanya tadi say...
1	2	Halo Dokter,\n\nSaya mau bertanya, dalam waktu...
2	3	Dok, saya Pria berusia 28 Tahun Ketika saya ce...
3	4	Halo dok, izin bertanya. Nenek saya punya peny...
4	5	Halo dok\n\nJadi malam itu, karena merasa haus...

Gambar 2. Gambaran singkat dataset yang digunakan

a) *Case folding*

Case folding dilakukan dengan mengubah seluruh *term* menjadi *lowercase* atau mengubah seluruh karakter huruf 'A-Z' menjadi huruf kecil 'a-z'. Berikut *syntax* yang digunakan dalam *case folding.*

```
#CASE FOLDING
df['Teks'] = df['Teks'].str.lower()
print("Case Folding".center(60,"-"))
print(df.head(5))
print("")
```

Gambar 3. Syntax pada tahap case folding

Hasil dari *case folding* yaitu sebagai berikut:

Case Folding

	ID	Teks
0	1	selamat malam dok saya ingin bertanya tadi say...
1	2	halo dokter,\n\nsaya mau bertanya, dalam waktu...
2	3	dok, saya pria berusia 28 tahun ketika saya ce...
3	4	halo dok, izin bertanya. nenek saya punya peny...
4	5	halo dok\n\njadi malam itu, karena merasa haus...

Gambar 4. Output dari tahap case folding

b) Perbaikan *term*

Perbaikan *term* dilakukan dengan menghilangkan *link,* angka, tanda baca dan *term* dengan satu karakter serta *whitespace.* Hal ini bertujuan untuk membersihkan data dari data-data yang tidak diperlukan. Berikut *syntax* yang digunakan dalam perbaikan term.

```
def hapus_tweet_special(text):
    #hapus tab, new line, ans back slice
    text = text.replace('\\t'," ").replace('\\n'," ").replace('\\u'," ").replace('\\',"")
    #hapus non ASCII (emoticon, chinese word, .etc)
    text = text.encode('ascii', 'replace').decode('ascii')
    #hapus mention, link, hashtag
    text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)"," ", text).split())
    #hapus incomplete URL
    return text.replace("http://", " ").replace("https://", " ")

df['Teks'] = df['Teks'].apply(hapus_tweet_special)

#menghilangkan angka
def hapus_number(text):
    return  re.sub(r"\d+", "", text)
df['Teks'] = df['Teks'].apply(hapus_number)

#menghilangkan tanda baca
def hapus_punctuation(text):
    return text.translate(str.maketrans("","",string.punctuation))
df['Teks'] = df['Teks'].apply(hapus_punctuation)

#hapus whitespace leading & trailing
def hapus_whitespace(text):
    return text.strip()
df['Teks'] = df['Teks'].apply(hapus_whitespace)

#white space dobel --> 1 whitespace
def hapus_whitespace_multiple(text):
    return re.sub('\s+',' ',text)
df['Teks'] = df['Teks'].apply(hapus_whitespace_multiple)

def hapus_single_char(text):
    return re.sub(r"\b[a-zA-Z]\b", "", text)
df['Teks'] = df['Teks'].apply(hapus_single_char)
```

Gambar 5. Syntax untuk tahap perbaikan term

Hasil dari pengolahan di atas yaitu sebagai berikut:

```
----------------------Perbaikan Term----------------------
0     selamat malam dok saya ingin bertanya tadi say...
1     halo dokter saya mau bertanya dalam waktu ming...
2     dok saya pria berusia tahun ketika saya cek hb...
3     halo dok izin bertanya nenek saya punya penyak...
4     halo dok jadi malam itu karena merasa haus sek...
5     asalamualaikum dok maaf ganggu saya mau nanya ...
6     hallo dok saya mau konsultasi ibu saya menderi...
7     pagi dok sudah dua bulan ini dermatitis seboro...
```

Gambar 6. Output dari tahap perbaikan term

c) *Tokenizing*

Tokenizing atau tokenisasi berarti *filtering* membagi suatu unit menjadi unit yang lebih kecil. Tokenisasi per kata diterapkan pada dataset dengan membagi kalimat menjadi kata agar proses analisis teks dapat dilakukan dengan baik. Dari *tokenizing* yang dilakukan diperoleh 428 token. Berikut *syntax* yang digunakan dalam tokenisasi.

```
#TOKENIZING
def word_tokenize_wrapper(text):
    return word_tokenize(text)
df['tokens'] = df['Teks'].apply(word_tokenize_wrapper)

print("Hasil Tokenisasi".center(60,"-"))
print(df['tokens'].head())
print("")
```

Gambar 7. Syntax untuk tahap tokenizing

Hasil dari tokenisasi yaitu sebagai berikut:

```
----------------------Hasil Tokenisasi----------------------
0    [selamat, malam, dok, saya, ingin, bertanya, t...
1    [halo, dokter, saya, mau, bertanya, dalam, wak...
2    [dok, saya, pria, berusia, tahun, ketika, saya...
3    [halo, dok, izin, bertanya, nenek, saya, punya...
4    [halo, dok, jadi, malam, itu, karena, merasa, ...
Name: tokens, dtype: object
```

Gambar 8. Output dari tahap tokenizing

d) *Remove stopword*

Stopword adalah kata yang sering muncul dan tidak memiliki makna penting. Untuk menghilangkan *stopword*, digunakan *library* Sastrawi. Selain itu, *list* dari *stopword* dapat ditambahkan pada variabel 'more_stopword'. Berikut *syntax* yang digunakan dalam menghilangkan *stopword.*

```
#REMOVE STOPWORD
stop_factory = StopWordRemoverFactory()
print(stop_factory.get_stop_words())
#more_stopword = ['ia','bahwa','oleh','alo','haloo','yg','udah','karna']
defaultStopword = ['a', 'ada', 'adalah', 'adanya', 'adapun', 'agak', 'agaknya',
dataStopword = stop_factory.get_stop_words()+defaultStopword#+more_stopword
stopword = stop_factory.create_stop_word_remover()
def remove_stopwords(words):
    return [word for word in words if word not in dataStopword]
print("Tanpa Stopword".center(60,"-"))
df['tokens_WSW'] = df['tokens'].apply(remove_stopwords)
print(df['tokens_WSW'].head())
print("")
```

Gambar 9. Syntax untuk tahap remove stopword

Hasil dari *remove stopword* yaitu sebagai berikut:

```
------------------------Tanpa Stopword-------------------------
0    [selamat, malam, dok, test, gula, darah, puasa...
1    [dokter, waktu, minggu, gampang, kencing, minu...
2    [dok, pria, berusia, tahun, cek, hbac, hasil, ...
3    [dok, izin, nenek, penyakit, diabetes, th, men...
4    [dok, malam, haus, sehabis, makan, minum, air,...
Name: tokens_WSW, dtype: object
```

Gambar 10. Output dari tahap remove stopword

e) *Stemming*

Stemming berarti mengembalikan suatu kata menjadi bentuk baku kata tersebut. Karena dataset menggunakan bahasa Indonesia, *library* yang digunakan untuk *stemming* adalah Sastrawi. Berikut *syntax* yang digunakan dalam *stemming*.

```
#STEMMING
factory = StemmerFactory()
stemmer = factory.create_stemmer()
def stemmed_wrapper(term):
    return stemmer.stem(term)
term_dict = {}
for document in df['tokens_WSW']:
    for term in document:
        if term not in term_dict:
            term_dict[term] = ' '
#print(len(term_dict))
for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)
    #print(term,":" ,term_dict[term])
#print(term_dict)

def get_stemmed_term(document):
    return [term_dict[term] for term in document]

print("Hasil Stemming".center(60,"-"))
df['tokens'] = df['tokens_WSW'].apply(get_stemmed_term)
print(df['tokens'])
print("")
```

Gambar 11. Syntax untuk tahap stemming

Hasil dari *stemming* yaitu sebagai berikut:

```
------------------------Hasil Stemming------------------------
0     [selamat, malam, dok, tanya, tadi, test, gula,...
1     [halo, dokter, mau, tanya, waktu, minggu, gamp...
2     [dok, pria, usia, tahun, cek, hbac, hasil, lal...
3     [halo, dok, izin, tanya, nenek, punya, sakit, ...
4     [halo, dok, jadi, malam, rasa, haus, sekali, h...
```

Gambar 12. Output dari tahap stemming

f) Pembobotan *term*

Pembobotan *term* dilakukan untuk menentukan tingkat kepentingan dari suatu *term*. Pembobotan *term* dilakukan menggunakan *TF-IDF*. *TF-IDF* (*term frequency-inverse document frequency*) adalah nilai yang menunjukkan betapa pentingnya sebuah kata (atau kelompok kata) bagi sebuah dokumen atau kumpulan teks. Skor TF-IDF akan melihat kata-kata apa yang muncul dan seberapa sering kata-kata itu muncul. Variabel X adalah *array* vektor yang akan digunakan untuk melatih model *K-Means*. *Syntax* yang digunakan yaitu sebagai berikut.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

#vectorizer = TfidfVectorizer(sublinear_tf=True, min_df=5, max_df=0.95)
vectorizer = TfidfVectorizer() #min_df = minimal muncul di df dokumen

X = vectorizer.fit_transform(output)
print(X.shape)
print(X)
```

Gambar 13. Tahap pembobotan term

Hasil dari pembobotan *term* adalah sebagai berikut.

```
(30, 428)
  (0, 194)      0.12938298817094646
  (0, 390)      0.15520982812721668
  (0, 402)      0.15520982812721668
  (0, 143)      0.20556027781098837
  (0, 285)      0.13672057812101962
  (0, 316)      0.366559697775978
  (0, 396)      0.20556027781098837
  (0, 13)       0.20556027781098837
  (0, 175)      0.4355736122039042
  (0, 244)      0.10270418852883481
  (0, 319)      0.31041965625443335
  (0, 77)       0.4284102369917884
  (0, 141)      0.3213076777438413
  (0, 397)      0.183279848887989
  (0, 97)       0.066770733623424
  (0, 247)      0.1229107751449687
  (0, 354)      0.13672057812101962
  (1, 391)      0.14358639937850956
  (1, 34)       0.14358639937850956
  (1, 304)      0.11698100934326912
  (1, 385)      0.14358639937850956
  (1, 403)      0.14358639937850956
  (1, 182)      0.14358639937850956
  (1, 293)      0.12802324389083358
  (1, 33)       0.12802324389083358
  :     :
  (27, 77)      0.11655194298057696
  (27, 141)     0.11655194298057696
  (27, 354)     0.1487830835913292
  (28, 393)     0.30276083663701886
  (28, 0)       0.2699449571626707
  (28, 43)      0.5398899143253414
  (28, 85)      0.49332344027998437
  (28, 415)     0.24666172013999219
  (28, 4)       0.22860183844124018
  (28, 97)      0.3933748949937397
```

```
[[0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.         0.03649509 ... 0.         0.03050619 0.        ]
 [0.         0.         0.         ... 0.01450685 0.         0.05004713]
 ...
 [0.         0.         0.         ... 0.         0.         0.        ]
 [0.         0.15969874 0.         ... 0.         0.         0.03060048]
 [0.         0.         0.         ... 0.         0.         0.        ]]
```

Gambar 14. Output dari tahap pembobotan term

B. *Clustering*

Setelah data sudah dibersihkan melalui tahapan *preprocessing* dataset, dilakukan pemeriksaan jumlah cluster pada data menggunakan *elbow method* dan *clustering* menggunakan K-Means. Akurasi hasil *clustering* kemudian dihitung.

a) Elbow Method

Elbow method adalah salah satu metode *k-means clustering* yang digunakan untuk menentukan jumlah cluster. Fungsi 'Sum_of_squared_distances' berguna untuk menjalankan teks melalui vectorizer TF-IDF sebelum menjalankan K-Means *clustering*. Fungsi ini kemudian akan memplot jumlah kuadrat jarak sebelum memilih banyaknya cluster. Berikut *syntax* yang digunakan dalam membuat *elbow method* ke dalam plot.

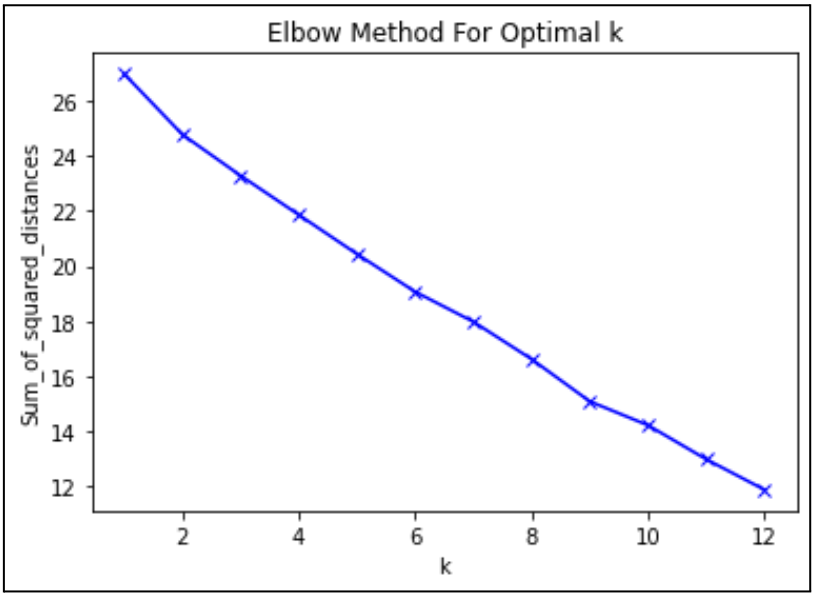
```
Sum_of_squared_distances = []
K = range(1,13)
for k in K:
   km = KMeans(n_clusters=k, init='k-means++', max_iter=200, n_init=10, random_state=36)
   km = km.fit(X)
   Sum_of_squared_distances.append(km.inertia_)
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```

Gambar 15. Syntax untuk membuat plot elbow method (versi lama)

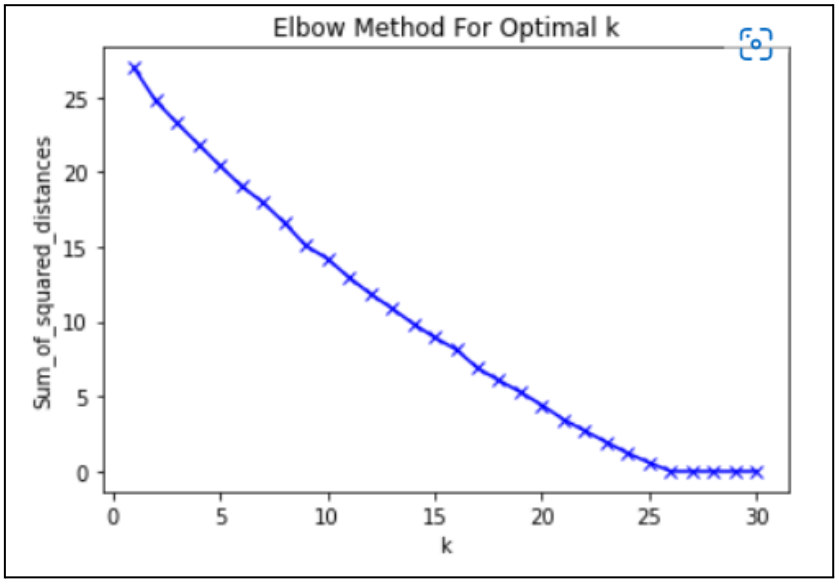
```
Sum_of_squared_distances = []
K = range(1,31)
for k in K:
   km = KMeans(n_clusters=k, init='k-means++', max_iter=200, n_init=10, random_state=36)
   km = km.fit(X)
   Sum_of_squared_distances.append(km.inertia_)
plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method For Optimal k')
plt.show()
```

Gambar 16. Syntax untuk membuat plot elbow method (versi baru)

Berikut ini adalah hasil dari plot jumlah kuadrat jarak.



Gambar 17. Hasil plot elbow method (versi lama)



Gambar 18. Hasil plot elbow method (versi baru)

Berdasarkan hasil *elbow method* pada gambar 17 (versi lama), dapat diketahui bahwa jumlah *cluster* optimal berada pada k= 9 (9 cluster) dan pada gambar 18 didapat jumlah cluster k=25 (25 *cluster*).

b) *K-Means Clustering*

K-Means adalah salah satu algoritma *unsupervised* untuk mengelompokkan sekumpulan data ke dalam sejumlah grup atau kelompok cluster yang ditentukan. Algoritma dibawah ini menginisialisasi *centroid* di bidang vektor dan menetapkan titik ke *centroid* terdekat. Berikut *syntax* yang digunakan dalam melakukan *K-Means Clustering.*

```
# initialize kmeans with 3 centroids
kmeans = KMeans(n_clusters=k, init='k-means++', max_iter=200, n_init=10, random_state=42)
# fit the model
kmeans.fit(X)
# store cluster labels in a variable
clusters = kmeans.labels_
```

Gambar 19. Syntax untuk implementasi K-Means (versi lama) (versi lama)

```
# initialize kmeans with 3 centroids
kmeans = KMeans(n_clusters=25, init='k-means++', max_iter=200, n_init=10, random_state=42)
# fit the model
kmeans.fit(X)
# store cluster labels in a variable
clusters = kmeans.labels_

cluster_center_tf=kmeans.cluster_centers_
print(cluster_center_tf)

listTF2 = cluster_center_tf.tolist()
#print(listTF2)
dataTF2 = pd.DataFrame(listTF2)
dataTF2.to_excel("TF-IDF2.xlsx")
```

Gambar 20. Syntax untuk implementasi K-Means (versi baru)

c) *Dimensional Reduction and Visualization*

Tahapan *Dimensional Reduction and Visualization* digunakan untuk memvisualisasikan hubungan antara teks dan grup berdasarkan X *(array* vektor) dari TF-IDF, model K-Means, dan cluster terkait. Grafik disajikan dalam bentuk 2 dimensi. Pengurangan dimensi dataset dilakukan menggunakan teknik PCA, karena PCA cenderung mempertahankan dimensi yang paling baik dalam meringkas variabilitas total data, dengan menghapus dimensi yang berkontribusi sedikit.

Sklearn.decomposition import PCA memeriksa dimensi array vektor x0 dan x1 untuk setiap teks. Tahap ini memungkinkan untuk membuat diagram sebar atau *scatter plot* untuk mempermudah proses analisis data. Berikut *syntax* yang digunakan dalam *dimensional reduction and visualization.*

```
from sklearn.decomposition import PCA

# initialize PCA with 2 components
pca = PCA(n_components=2, random_state=42)
# pass our X to the pca and store the reduced vectors into pca_vecs
pca_vecs = pca.fit_transform(X.toarray())
# save our two dimensions into x0 and x1
x0 = pca_vecs[:, 0]
x1 = pca_vecs[:, 1]
```

Gambar 21. Syntax untuk Dimensional Reduction and Visualization (versi lama)

```
from sklearn.decomposition import PCA

# PCA untuk mereduksi data
pca = PCA(n_components=2, random_state=42)
# pass our X to the pca and store the reduced vectors into pca_vecs
pca_vecs = pca.fit_transform(X.toarray())
# save our two dimensions into x0 and x1
x0 = pca_vecs[:, 0]
x1 = pca_vecs[:, 1]
```

Gambar 22. Syntax untuk Dimensional Reduction and Visualization (versi baru)

d) Visualisasi *Cluster*

Sebelum membuat bagan, melakukan pengaturan *dataframe* dengan membuat cluster kolom, x0, x1. Selanjutnya dilihat kata kunci mana yang paling relevan untuk setiap centroid sehingga nama setiap *cluster* dapat diganti dengan label yang lebih baik.

```
# assign clusters and pca vectors to our dataframe
df['cluster'] = clusters
df['x0'] = x0
df['x1'] = x1
```

Gambar 23. Syntax untuk penetapan cluster dan vektor PCA ke dataframe (versi lama)

```
# assign clusters and pca vectors to our dataframe
df['cluster'] = clusters
df['x0'] = x0
df['x1'] = x1
```

Gambar 24. Syntax untuk penetapan cluster dan vektor PCA ke dataframe (versi baru)

Fungsi dibawah ini mengembalikan kata kunci untuk setiap centroid dari K-Means, K-Means membuat 9 dan 25 *cluster* kemudian dilakukan pemetaan (*mapping*).

```
def get_top_keywords(n_terms):
    """This function returns the keywords for each centroid of the KMeans"""
    df = pd.DataFrame(X.todense()).groupby(clusters).mean() # groups the TF-IDF vector by cluster
    terms = vectorizer.get_feature_names_out() # access tf-idf terms
    for i,r in df.iterrows():
        print('\nCluster {}'.format(i))
        print(','.join([terms[t] for t in np.argsort(r)[-n_terms:]])) # for each row of the dataframe, find the n terms that have the highest tf idf

get_top_keywords(20)
```

Gambar 25. Syntax untuk penetapan cluster dan vektor pca ke dataframe (versi lama)

```
def get_top_keywords(n_terms):
    """This function returns the keywords for each centroid of the KMeans"""
    df = pd.DataFrame(X.todense()).groupby(clusters).mean() # groups the TF-IDF vector by cluster
    terms = vectorizer.get_feature_names_out() # access tf-idf terms
    for i,r in df.iterrows():
        print('\nCluster {}'.format(i))
        print(','.join([terms[t] for t in np.argsort(r)[-n_terms:]])) # for each row of the dataframe, find the n terms that have the highest tf idf

get_top_keywords(20)
```

Gambar 26. Syntax untuk penetapan cluster dan vektor pca ke dataframe (versi baru)

Berikut ini adalah hasil dari pemetaan untuk setiap *cluster*, dimana jumlah *cluster*-nya adalah 9 (optimal-versi lama) dan 25 (optimal-versi baru). *Cluster* dibagi berdasarkan *term* yang diperoleh yaitu sebagai berikut.

```
Cluster 0
nyenyak,pagidok,tama,begadang,pola,bahak,gds,gak,karna,bersendwa,sunti,sya,saudara,kurang,kembung,minum,gejala,insulin,tidur,haus

Cluster 1
test,mgdl,hexokinase,glukosa,kasih,suntik,terima,dampak,sandang,reduksi,militus,haloo,tinggi,pp,dok,ayah,jam,darah,puasa,gula

Cluster 2
pipis,intensitas,sakit,malam,coba,ga,pengaruh,kasih,opa,nenek,obatobatan,konsumsi,dok,dokter,diabetes,waktu,manis,minum,kencing,obat

Cluster 3
jahit,op,dibiarkna,dokkakek,takut,dok,kontrol,minggu,amputasi,sembuh,basah,nanah,muncul,kaki,hitam,luk,cabut,kulit,luka,jari

Cluster 4
gr,haloo,dokter,malam,sore,diabetes,ulkus,liat,dewasa,dengar,google,gangren,engga,terimaksih,anak,diabetik,abses,aja,dok,beda

Cluster 5
kolesterol,spy,bgmn,cegah,asam,beliau,putih,kepala,racun,tidur,jalan,mama,muncul,udah,dok,ngga,susah,sy,air,sakit

Cluster 6
derita,terimakasih,sembuh,periksa,infeksi,milik,potong,umur,bagi,an,gelang,belakang,korek,kaki,tutup,tahap,luka,kulit,daging,tumbuh

Cluster 7
metforminapakah,sakit,derita,dokter,hasil,sitokine,maag,nafsu,terimakasih,hbac,ambang,say,ac,hb,gula,darah,dok,obat,makan,cek

Cluster 8
hbac,habis,guladiabetes,gula,gt,gr,google,gmn,glukosa,gulah,hb,dok,makan,kalo,hari,mg,ramadan,glimipiride,minum,takjil
```

Gambar 27. Hasil pemetaan clustering (versi lama)

```
Cluster 0
konsumsi,coba,ulu,serta,mual,muntah,hati,glimepiride,dok,dokobat,omeprazole,konsultasi,metforminapakah,ganti,darah,gula,makan,obat,maag,nafsu

Cluster 1
hbac,habis,guladiabetes,gula,gt,gr,google,gmn,glukosa,gulah,hb,dok,makan,kalo,hari,mg,ramadan,glimipiride,minum,takjil

Cluster 2
akibat,serta,operasi,dok,laku,harus,sebab,telak,tengah,kelingking,rutin,tindak,kakek,luka,nanah,muncul,bln,lubang,kaki,jari

Cluster 3
diabetik,buka,basah,ulkus,sekrang,cair,biar,nacl,dokkakek,dibiarkna,kasa,takut,isi,jahit,op,luka,kulit,hitam,luk,cabut

Cluster 4
rumah,test,resep,didiagnosa,ambil,tes,gulah,prediabetes,rata,alat,acarbose,hormat,malam,normal,tinggi,pp,darah,jam,gula,puasa

Cluster 5
gulah,guladiabetes,gula,gt,hari,gds,dokter,sore,abses,beda,diabetes,dok,liat,dewasa,dengar,google,gangren,engga,aja,anak

Cluster 6
glimipiride,hati,diabetes,terimakasih,tidur,gejala,suntik,muntah,didiagnosa,kadang,karna,sya,bersendwa,kembung,sunti,kurang,saudara,minum,haus,ir


Cluster 7
guladiabetes,gt,gr,google,hcu,hd,gmn,glukosa,hasil,darah,gula,obat,terimakasih,dok,ambang,hb,ac,say,makan,cek

Cluster 8
malam,yg,selamat,makasih,kemarin,pas,dok,banyaksaya,bawa,doktapi,tampung,manisdok,terusdan,gk,rasakencing,minum,coba,kencing,pipis,manis

Cluster 9
busuk,gas,seriussekarang,dibikinmohon,rawat,dgn,dalamkejadian,gigil,bikin,sembarang,bikinin,kasih,tinggi,aja,gelas,operasi,tangan,kuku,ga,ayah

Cluster 10
derita,terimakasih,sembuh,periksa,infeksi,milik,potong,umur,bagi,an,gelang,belakang,korek,kaki,tutup,tahap,luka,kulit,daging,tumbuh

Cluster 11
hangat,gk,habis,yg,guladiabetes,gt,gr,google,gmn,glukosa,glimipiride,gulah,dok,diabetes,gula,haloo,sandang,militus,dampak,reduksi

Cluster 12
usia,anak,turun,ngalamin,tuh,tua,oma,perna,sd,betis,na,udah,waktu,luka,tu,nyilu,bener,aja,ga,opa

Cluster 13
sy,kadang,maag,hipertensi,asam,eh,obt,kolesterol,komplikasi,utk,sdh,cegah,kambuh,mnum,spy,kasi,urat,bgmn,muncul,sakit

Cluster 14
kmudian,seni,hobby,mending,istirahat,liter,banyak,antuk,nyaman,rasa,lelah,tidur,minum,dok,jalan,putih,racun,kepala,sy,air

Cluster 15
pikir,saat,sadar,bengkak,kelamin,mudah,seboroik,luka,minggu,sembuh,pagi,basah,infeksi,milik,dokter,sama,jamur,sulit,kulit,muncul

Cluster 16
gitu,karna,kasa,dokter,derita,kasih,selamat,terima,siang,saran,kaburoleh,matayg,beraktipitas,mobil,kembalimisal,kendara,diabets,doksuami,lihat,suntil

Cluster 17
normal,selamat,cek,gejala,haus,nanya,makasih,maaf,olahraga,gak,tama,begadang,lapar,gds,pagidok,bahak,nyenyak,jadwal,pola,tidur

Cluster 18
pengaruh,mual,badan,pusing,luar,resep,izin,mengcopy,sehat,ajak,dosis,kota,th,beli,dokter,sakit,konsumsi,obat,nenek,obatobatan

Cluster 19
ginjal,perut,atas,tenang,digerakin,pulang,tambah,gelisah,jerumus,angkatin,asa,gula,minggu,tinggi,sakit,beliau,udah,mama,susah,ngga

Cluster 20
pria,aman,gdp,koq,hnya,disiplin,tahun,diberika,guladiabetes,inidikatakn,banyakdan,detail,dokjuga,bhwa,cek,obat,dokter,hasil,sitokine,hbac

Cluster 21
dmn,bantu,putih,olahraga,gampang,senam,gel,jogging,rokok,tdk,bantuanya,uda,terimah,bau,tp,dokter,normal,minum,intensitas,kencing

Cluster 22
nanya,waktu,siang,hipertensi,jelas,maaf,metformin,glimepiride,sama,diabetes,trimakasih,asalamualaikum,batuk,sekian,ganggu,amlodipine,erdostein,kasih

Cluster 23
glimipiride,hangat,hari,habis,gmn,google,gr,haloo,gula,gt,guladiabetes,gulah,malam,aja,ulkus,abses,terimaksih,dok,diabetik,beda

Cluster 24
efek,gejala,sy,makasih,diagnosis,gmn,hasil,pp,jelas,test,gimana,bagus,yah,dm,gula,dok,darah,mgdl,glukosa,hexokinase
```

Gambar 28. Hasil pemetaan clustering (versi baru)

```python
# map clusters to appropriate labels
cluster_map = {0: "Cluster 0", 1: "Cluster 1", 2: "Cluster 2", 3:"Cluster 3", 4:"Cluster 4", 5: "Cluster 5", 6:"Cluster 6", 7:"Cluster 7", 8:"Cluster
# apply mapping
df['cluster'] = df['cluster'].map(cluster_map)
```

Gambar 29. Syntax pemetaan cluster ke label yang sesuai (versi lama)

```
# map clusters to appropriate labels
cluster_map = {0: "Cluster 0", 1: "Cluster 1", 2: "Cluster 2", 3:"Cluster 3", 4:"Cluster 4", 5: "Cluster 5", 6:"Cluster 6", 7:"Cluster 7", 8:"Cluster
# apply mapping
df['cluster'] = df['cluster'].map(cluster_map)
```

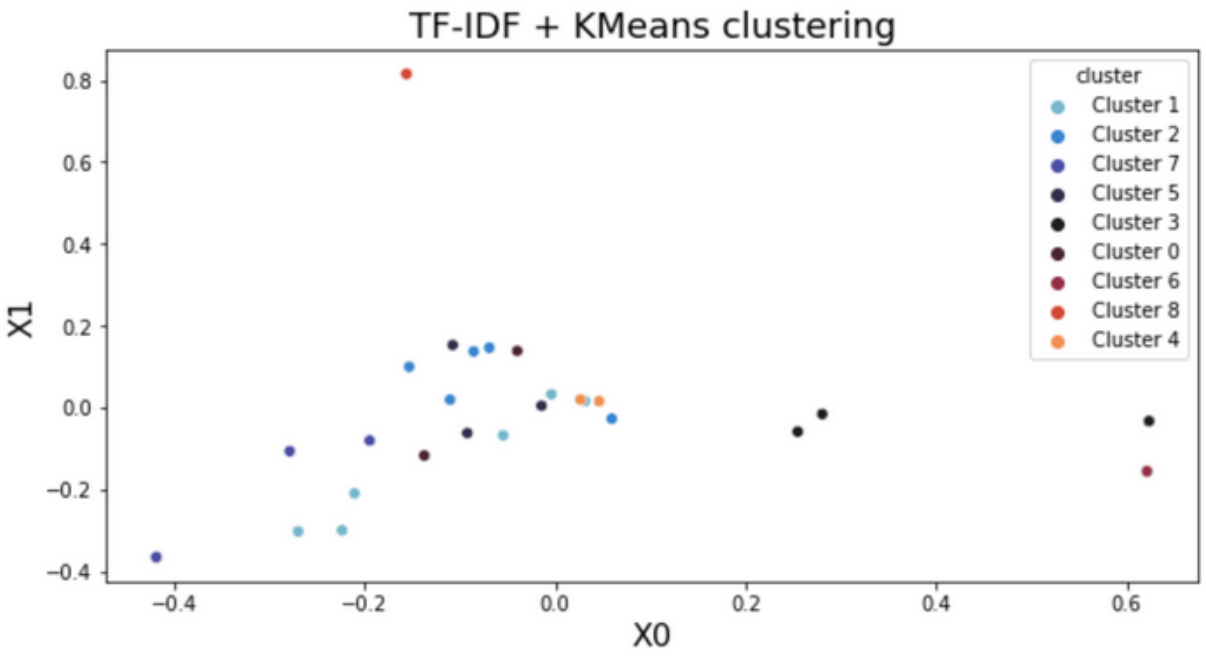
Gambar 30. Syntax pemetaan cluster ke label yang sesuai (versi baru)

Selanjutnya membuat *scatter plot* dengan *library seaborn*, di mana *hue* adalah kelas yang digunakan untuk mengelompokkan data. *Library Seaborn* ini digunakan untuk memvisualisasikan teks yang dikelompokkan dengan cara yang sangat sederhana.

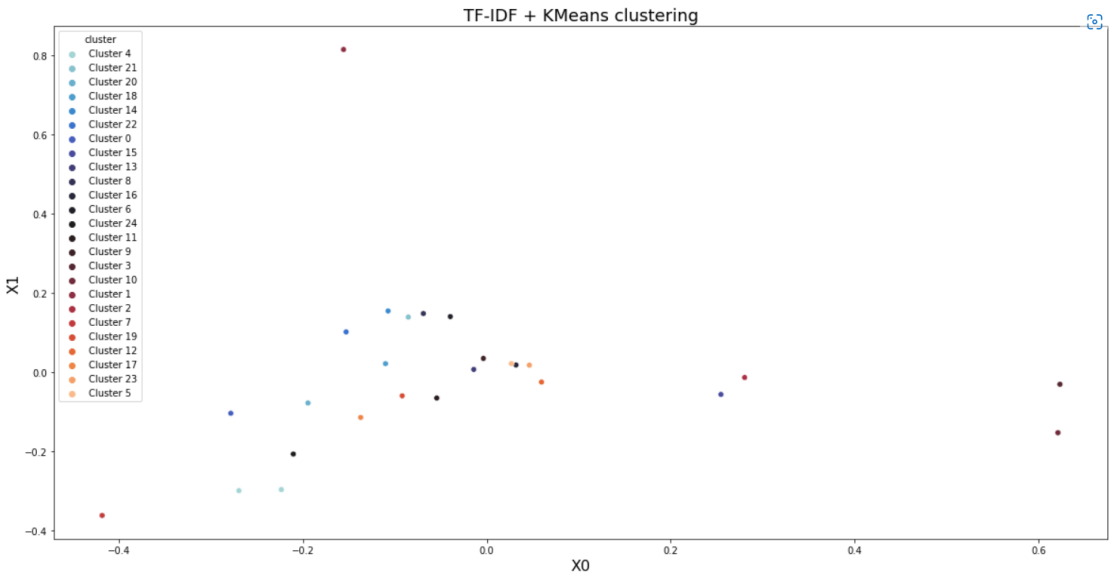
```
# set image size
plt.figure(figsize=(10, 5))
# set a title
plt.title("TF-IDF + KMeans clustering", fontdict={"fontsize": 18})
# set axes names
plt.xlabel("X0", fontdict={"fontsize": 16})
plt.ylabel("X1", fontdict={"fontsize": 16})
# create scatter plot with seaborn, where hue is the class used to group the data
sns.scatterplot(data=df, x='x0', y='x1', hue='cluster', palette="icefire")
plt.show()
```

Gambar 31. Syntax untuk membuat scatter plot

Hasil dari visualisasi *clustering* dalam *scatter plot* sebagai berikut.



Gambar 32. Hasil scatter plot TF IDF dan K-Means Clustering (versi lama)



Gambar 33. Hasil scatter plot TF IDF dan K-Means Clustering (versi baru)

```
from collections import Counter
clustered_docs = kmeans.fit_predict(X)
def purity(labels, clustered):
    # find the set of cluster ids
    cluster_ids = set(clustered)
    N = len(clustered)
    majority_sum = 0
    for cl in cluster_ids:
        # for this cluster, we compute the frequencies of the different human labels we encounter
        # the result will be something like { 'camera':1, 'books':5, 'software':3 } etc.
        labels_cl = Counter(l for l, c in zip(labels, clustered) if c == cl)
        # we select the *highest* score and add it to the total sum
        majority_sum += max(labels_cl.values())
    # the purity score is the sum of majority counts divided by the total number of items
    #print("Majority",majority_sum, "N", N)
    return majority_sum / N

skor = purity(df["Teks"],clustered_docs)
print('Akurasi: {:.2f}%'.format(skor*100))
```

Gambar 34. Syntax evaluasi hasil clustering

Hasil evaluasi *clustering* yang didapatkan akurasi sebesar 43,33% untuk versi lama dan 96.67% untuk versi baru.

```
Akurasi: 43.33%
```

Gambar 35. Hasil evaluasi clustering (versi lama)

```
Akurasi: 96.67%
```

Gambar 36. Hasil evaluasi clustering (versi baru)

C. Analisis Hasil *Clustering*

Sebelum melakukan *clustering*, dilakukan *preprocessing text mining* yang terdiri dari *case folding,* perbaikan *term, tokenizing, remove stopword, stemming,* dan pembobotan *term.* Tahap *case folding*, perbaikan *term, tokenizing*, *remove stopword,* dan *stemming* dilakukan untuk memperbaiki dataset agar informasi data yang ditampilkan berkualitas, layak, dan efisien untuk diolah lebih lanjut. Pada tahap pembobotan *term,* pembobotan kata dilakukan menggunakan *TF-IDF* untuk melihat betapa pentingnya sebuah kata (atau kelompok kata) bagi sebuah dokumen atau kumpulan teks. Selanjutnya dilakukan pemeriksaan jumlah *cluster* menggunakan *elbow method*, sehingga didapatkan jumlah *cluster* optimal k=25 (25 *cluster*). Banyaknya *cluster* karena *clustering* yang dilakukan berdasarkan *term* yang didapat (428 *term)* dan bukan dokumen (30 dokumen). Setelah didapatkan jumlah *cluster* optimal, dilakukan *mapping* ke label yang sesuai dan membuat visualisasi *clustering* dalam *scatter plot*. Tahapan terakhir yakni melakukan evaluasi *clustering* menggunakan *purity. Purity* digunakan untuk menghitung jumlah kata yang sering muncul dalam setiap *cluster*, sehingga didapatkan akurasi sebesar 43,33% untuk percobaan pertama dan 96,67% untuk percobaan kedua. Hasil akurasi diperoleh cukup rendah pada percobaan pertama (43,33%) karena terdapat beberapa kata yang memiliki makna sama pada tahap *preprocessing*, sehingga pengelompokan menjadi *cluster* tertentu menjadi kurang spesifik dan akurat. Nilai *k* percobaan pertama diperoleh dari *elbow method* yang memiliki *range* 1-11. Sementara itu, hasil akurasi diperoleh cukup tinggi pada percobaan kedua (96,67%) karena terdapat beberapa kata yang memiliki makna berbeda pada tahap *preprocessing*, sehingga pengelompokan menjadi *cluster* tertentu menjadi spesifik dan akurat. Nilai *k* percobaan kedua diperoleh dari *elbow method* yang memiliki *range* 1-31. Pada percobaan pertama variabel yang menjadi dasar untuk pembagian *cluster* kurang jelas. Selain itu, pada tahap *preprocessing*, masih banyak *stopword* yang seharusnya dihapus; contohnya *typo* pada “pagidok” dan “alo” (seharusnya termasuk *stopword* karena *term* “halo” termasuk dalam *stopword*).