Laporan Tugas Project Based Learning

Mata Kuliah Penambangan Data

Anggota Kelompok:

- Egi Shidqi Rabbani/1301190443

- Mohammad Rifqi Atiko/1301190470

Kelas: IF-43-PIL-DS01

Kode Dosen: DDR

Pernyataan :

Dalam pengerjaan tugas ini penulis tidak mengerjakan dengan cara yang melanggar aturan perkuliahan dan kode etik akademisi.

Program Studi Sarjana Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2022

Bagian 1: Data

- **Penjelasan Data**

Dataset yang digunakan pada tugas ini berasal dari:

Depression: Reddit Dataset (Cleaned) | Kaggle

Dataset tersebut adalah data yang menunjukkan depresi atau tidaknya seseorang berdasarkan komentar di media sosial Reddit. Dataset terdiri dari dua kolom, yaitu 'clean_text' yang berisi komentar seseorang di Reddit dan 'is_depression', label yang menunjukkan apakah seseorang tersebut depresi atau tidak. Label 0 pada 'is_depression' menunjukkan bahwa seseorang tersebut tidak depresi dan label 1 menunjukkan bahwa seseorang tersebut depresi. Dataset yang digunakan berisi 7731 data.

```
#Read Dataset
df = pd.read_csv("/content/drive/MyDrive/Penambangan Data/Tugas Project Based Learning/depression_dataset_reddit_cleaned.csv")
df.head()
```

	clean_text	is_depression
0	we understand that most people who reply immed...	1
1	welcome to r depression s check in post a plac...	1
2	anyone else instead of sleeping more when depr...	1
3	i ve kind of stuffed around a lot in my life d...	1
4	sleep is my greatest and most comforting escap...	1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7731 entries, 0 to 7730
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   clean_text     7731 non-null   object
 1   is_depression  7731 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 120.9+ KB
```

- **Eksplorasi Data**

Setelah dilakukan eksplorasi data, diketahui bahwa pada dataset terdapat 81 baris data yang duplikat.

```
duplicate = df[df.duplicated()]
duplicate
```

	clean_text	is_depression
259	hi i wa wondering if anyone ha this happen to ...	1
468	i want to be dead ive been suicidal for year i...	1
508	i want to be dead ive been suicidal for year i...	1
606	i want to be dead ive been suicidal for year i...	1
747	ha anyone been prescribed mirtazapine or other...	1
...	...	...
7184	just been given ma marching order got ta go do...	0
7191	just been given ma marching order got ta go do...	0
7198	doing homework	0
7574	i m not liking that new itunes pricing at all ...	0
7577	cant eat drink or breath properly thanks to th...	0

81 rows × 2 columns

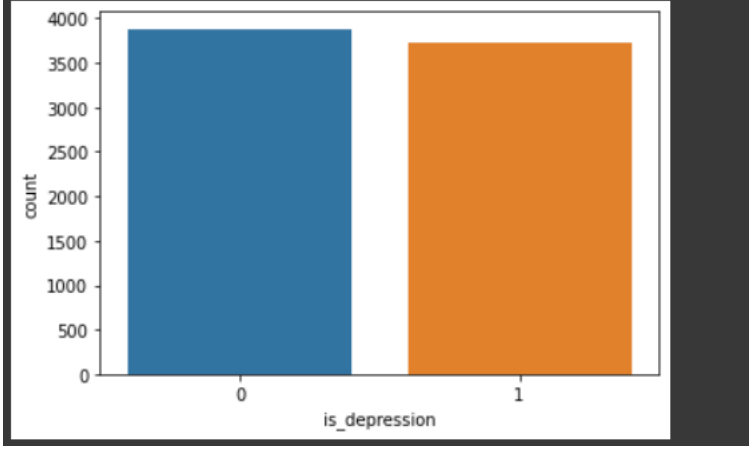
Selain itu, dataset juga tidak seimbang. Data yang berlabel 0 berjumlah 3879 sedangkan data yang berlabel 1 berjumlah 3718.

```
#Cek Jumlah label
df2['is_depression'].value_counts()
```

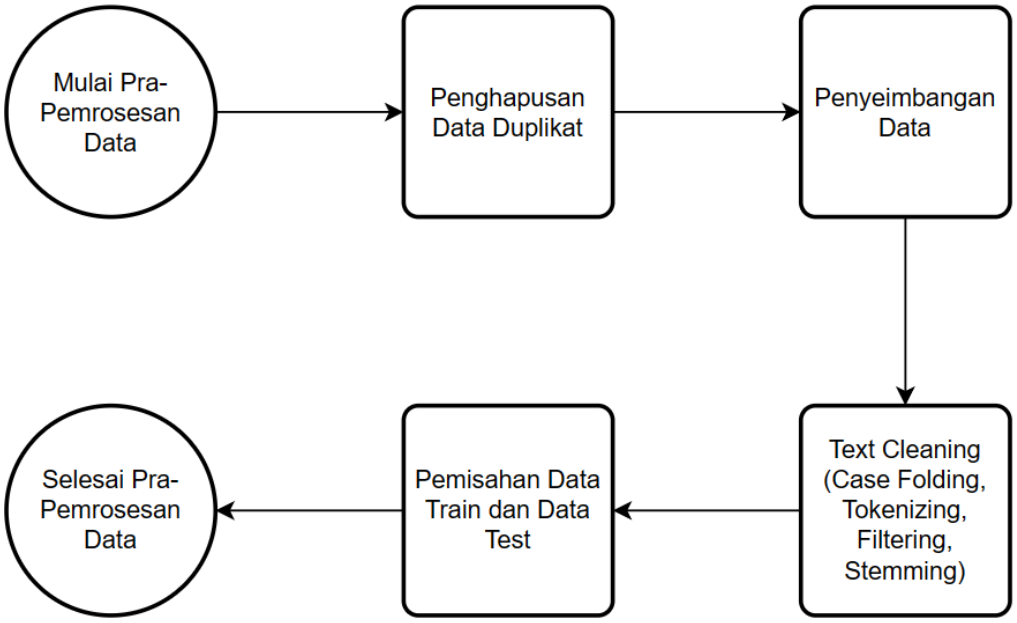
```
0    3879
1    3718
Name: is_depression, dtype: int64
```

```
sns.countplot(data=df2, x='is_depression')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd819d8b1c0>
```



Bagian 2: Pra-Pemrosesan Data



Gambar di atas adalah *flow chart* dari pra-pemrosesan data yang dilakukan oleh kelompok kami.

- **Hapus Data Duplikat**

Saat eksplorasi data, diketahui bahwa pada dataset terdapat beberapa data yang duplikat, oleh karena itu data-data yang duplikat tersebut dihapus untuk membersihkan data.

```
df2 = df.drop_duplicates(keep=False)
df2.head()
```

	clean_text	is_depression
0	we understand that most people who reply immed...	1
1	welcome to r depression s check in post a plac...	1
2	anyone else instead of sleeping more when depr...	1
3	i ve kind of stuffed around a lot in my life d...	1
4	sleep is my greatest and most comforting escap...	1

Setelah dilakukan penghapusan data duplikat dengan menggunakan fungsi drop_duplicates, tidak terdapat lagi data yang duplikat pada dataset dan tersisa 7597 data pada dataset.

```
duplicate = df2[df2.duplicated()]
duplicate
```

clean_text	is_depression

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7597 entries, 0 to 7730
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   clean_text     7597 non-null   object
 1   is_depression  7597 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 178.1+ KB
```

- **Menyeimbangkan Data**

Pada tahap eksplorasi data, diketahui bahwa data belum seimbang. Terdapat 3879 data yang berlabel 0 sedangkan hanya 3718 data yang

berlabel 1. Oleh karena itu dilakukan penyeimbangan data dengan menghapus sebagian data berlabel 0.

```
#Menghapus Sebagian Data berlabel "0" Agar Seimbang
df_balanced = (df2.groupby('is_depression', as_index=False)
               .apply(lambda x: x.sample(n=3718))
               .reset_index(drop=True))
df_balanced.head()
```

	clean_text	is_depression
0	ad not yet appeared google adsense team said i...	0
1	off to work	0
2	mrsaintnick hey i m leavin in the morning	0
3	kisluvkis oh that is very sad poor boy	0
4	gosh it is raining in summer cause of the glob...	0

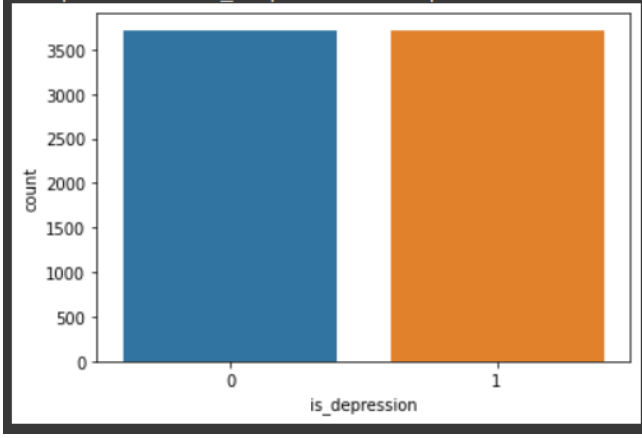
Dataset menjadi seimbang setelah dilakukan penyeimbangan data. Masing-masing data berlabel 0 dan 1 memiliki 3718 data. Sisa data pada dataset setelah dilakukan penyeimbangan data berjumlah 7436 data.

```
#Cek Jumlah label
df_balanced['is_depression'].value_counts()
```

```
0    3718
1    3718
Name: is_depression, dtype: int64
```

```
sns.countplot(data=df_balanced, x='is_depression')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd80a8b98e0>
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7436 entries, 0 to 7435
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   clean_text     7436 non-null   object
 1   is_depression  7436 non-null   int64
dtypes: int64(1), object(1)
memory usage: 116.3+ KB
```

- **Text Cleaning**

Sebelum mengimplementasikan model *machine learning* untuk dilakukan klasifikasi, dilakukan *text cleaning* terlebih dahulu. Tahap ini adalah tahap yang penting untuk klasifikasi teks.

```
#Fungsi untuk membersihkan teks
import string
def text_cleaning(a):
  cleaning = [char for char in a if char not in string.punctuation] #Mengambil teks tanpa punctuation (ex: "'", ",", "!", ".")
  cleaning = ''.join(cleaning) #Menyatukan teks setelah dibersihkan puctuation nya
  return [word for word in cleaning.split() if word.lower() not in stopwords.words('english')] #Mengembalikan teks per kata setelah dibersihkan stopword nya
```

Kode pada gambar di atas adalah fungsi untuk membersihkan teks. Yang pertama dilakukan pada fungsi tersebut adalah memisahkan setiap huruf pada kalimat dan menghapus *punctuation* (tanda baca) yang ada. Setelah itu huruf-hurf tersebut disatukan kembali menjadi kalimat yang utuh. Selanjutnya adalah membagi kalimat menjadi kata-kata lalu mengubah huruf besar menjadi huruf kecil serta dilakukan penghapusan *stopwords* (kata-kata yang tidak bermakna). Stopwords pada bahasa inggris di antaranya adalah *doesn't, it, on, do.*

```
#Menampilkan Stopwords pada bahasa inggris
from nltk.corpus import stopwords
stopWords = set(stopwords.words('english'))
print(stopWords)
```

```
{'doesn', 'who', 'now', 'again', 'was', 'but', 'by', 'had', 'hasn',
```

Contoh hasil dari kalimat yang telah diterapkan fungsi text_cleaning dapat di lihat pada gambar.

```
#Data setelah dilakukan text_cleaning
print(df_balanced.iloc[:,0].apply(text_cleaning))
```

```
0       [kel, marshall, tell, mortgage, quote, last, s...
1                 [damn, pc, ha, completely, given, suck]
2       [http, twitpic, com, e, wan, na, wear, doc, ma...
3       [mangaaa, hope, increase, capacity, fast, yest...
4       [honeymunchkin, anger, getting, bigger, every,...
                              ...
7431    [halitosis, also, associated, depression, symp...
7432    [new, article, obmintegrativeandcomplementarym...
7433    [alone, king, kch, din, baad, ye, bhi, chale, ...
7434    [need, go, routine, female, doctor, appointmen...
7435                                               [given]
Name: clean_text, Length: 7436, dtype: object
```

```
#CountVectorizer untuk mengubah setiap kata menjadi angka
from sklearn.feature_extraction.text import CountVectorizer
bow_transformer = CountVectorizer(analyzer=text_cleaning).fit(df_balanced['clean_text']) #Apply text_cleaning dan CountVectorizer

print(len(bow_transformer.vocabulary_))
bow_transformer.vocabulary_
```

Kode pada gambar di atas adalah penerapan fungsi text_cleaning serta pengubahan setiap kata ke dalam bentuk angka dengan menggunakan CountVectorizer.

```
'getting': 6706,
'bigger': 1691,
'every': 5501,
'minute': 10406,
'go': 6806,
'uglycomments': 16997,
'one': 11498,
'video': 17575,
'lyn': 9768,
'thanks': 16235,
'hun': 7721,
'didnt': 4332,
'even': 5489,
'pitty': 12263,
'couldnt': 3502,
'see': 14299,
'sing': 14724,
'reaaly': 13169,
'miss': 10435,
'john': 8693,
'mayer': 10067,
```

Gambar di atas adalah hasil pengubahan kata ke dalam bentuk angka, terlihat pada gambar bahwa kata *getting* diubah menjadi angka 6706, *bigger* menjadi 1691, dan seterusnya.

```
#Menujukkan setiap kata yang ada pada dataset
tokens = bow_transformer.get_feature_names()
tokens
```

```
 'absence',
 'absent',
 'absents',
 'absolut',
 'absolute',
 'absolutely',
 'absolutelybatty',
 'absolutly',
 'absorb',
 'absorbed',
 'absorbence',
 'absoutely',
 'abstinence',
 'abstract',
 'absurd',
 'absurdly',
 'abt',
 'abu',
 'abundant',
```

Kode pada gambar di atas adalah untuk melihat kata-kata apa saja yang ada pada dataset.

```
#Transformasi data
#Menunjukkan frekuensi kata pada setiap data
title_bow = bow_transformer.transform(df_balanced['clean_text'])
print(title_bow)
```

```
  (0, 0)        1
  (0, 6821)     1
  (0, 8927)     1
  (0, 9207)     1
  (0, 9982)     1
  (0, 10612)    1
  (0, 11038)    1
  (0, 13043)    1
  (0, 14100)    1
  (0, 14796)    1
  (0, 16129)    1
  (0, 18320)    1
  (1, 3169)     1
  (1, 3834)     1
  (1, 6750)     1
  (1, 7095)     1
```

Kode pada gambar di atas adalah untuk melihat frekuensi kata pada setiap data. Misalnya kata dengan angka 6821 pada baris data ke-0 muncul sebanyak 1 kali.

Selanjutnya, data-data yang telah ditransformasi tadi diubah ke dalam bentuk array.

```
#Menampilkan head dataset
#Frekuensi kemunculan kata diubah ke dalam bentuk array
print(df_balanced.head())
x = title_bow.toarray()
print(x)
x.shape
```

```
                                          clean_text  is_depression
0  kel marshall tell me about it had some mortgag...              0
1       damn my pc ha completely given out this suck              0
2  http twitpic com y e i wan na wear my doc mart...              0
3  mangaaa i hope they will increase the capacity...              0
4  honeymunchkin my anger is getting bigger for e...              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]]
(7436, 18485)
```

- **Splitting Data Train dan Test**

Tahap terakhir pada pra-pemrosesan data adalah memisahkan dataset menjadi dataset *train* dan dataset *test* dengan menggunakan train_test_split. Jumlah dataset adalah 20% dari total dataset, sedangkan 80% sisanya adalah dataset *train*.

```
#Memisahkan dataset menjadi dataset training dan dataset test
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, df_balanced.is_depression, test_size=0.2, random_state=50)
```

```
#Melihat ukuran data yang telah dipisah
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(5948, 18485)
(1488, 18485)
(5948,)
(1488,)
```

Bagian 3: Implementasi Algoritma KNN (K-Nearest Neighbour)

- ### Penjelasan Algoritma KNN

K-Nearest Neighbor atau K-NN merupakan algoritma *supervised learning classifier*, yang menggunakan kedekatan untuk membuat klasifikasi atau prediksi tentang pengelompokan titik data individual. Meskipun dapat digunakan untuk masalah regresi atau klasifikasi, K-NN biasanya digunakan sebagai algoritma klasifikasi

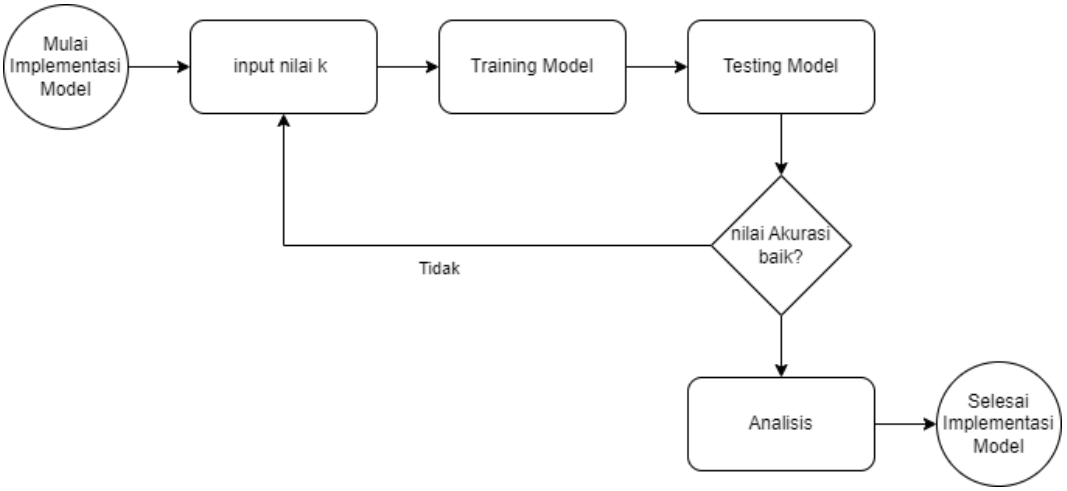
Terdapat tiga langkah utama pada algoritma KNN, yaitu:

- Langkah pertama, nilai K dipilih oleh pengguna untuk memberi tahu algoritma berapa banyak tetangga yang harus dipertimbangkan saat memberikan penilaian tentang grup tempat contoh target berada.
- Langkah kedua, model memeriksa jarak antara contoh target dan setiap contoh dalam kumpulan data.
- Langkah ketiga dimana jarak tersebut kemudian ditambahkan ke dalam *list* dan diurutkan. Setelah itu, daftar yang sudah terurut diperiksa dan label untuk elemen K teratas dikembalikan. Dengan kata lain, jika K diatur ke 5, model akan memeriksa label dari 5 titik data teratas yang terdekat dengan titik data target. Saat merender prediksi tentang titik data target, penting jika tugasnya adalah tugas regresi atau klasifikasi.

Pada tugas ini K-NN digunakan karena beberapa faktor yang memberikan keunggulan terhadap algoritma ini, yaitu sebagai berikut:

1. Kekuatan prediksi
2. Waktu perhitungan
3. Kemudahan pemahaman output

- **Model K-NN Classifier**



Pada Mocel K-NN Classifier diperlukan input nilai k (jumlah tetangga). Setelah itu dilakukan training terhadap model untuk melatih model tersebut terhadap data. Kemudian model dites dengan menggunakan data dari tes set dan dilihat nilai akurasinya, apabila belum baik maka dilakukan input nilai k lagi, jika sudah baik maka dilakukan analisis.

- **Implementasi Code**

Pertama yang dilakukan pada implementasi code di tugas ini adalah *import* library K-Nearest Neighbour Classifier dari Scikit Learn dan juga accuracy score dan confusion matrix sebagai parameter evaluasi performansi model.

```
2 from sklearn.neighbors import KNeighborsClassifier
3 from sklearn.metrics import accuracy_score, confusion_matrix
```

1. Inisialisasi nilai k

Jumlah tetangga dijadikan eksperimen pada model K-NN Classifier inim dimana terdapat beberapa nilai k yang akan digunakan dan dibandingkan hasilnya, contoh dibawah nilai k=9.

```
4 knn = KNeighborsClassifier(n_neighbors=9) #Jumlah K=9
5 knn.fit(x_train, y_train)
```

2. Training Model K-NN Classifier

Kemudian dilakukan training terhadap model K-NN classifier dengan menggunakan data train set dimana dimasukkan input berupa X_train (data fitur) dan y_train (label atau target).

```
4 knn = KNeighborsClassifier(n_neighbors=9) #Jumlah K=9
5 knn.fit(x_train, y_train)
```

3. Testing Model K-NN Classifier

Selanjutnya dilakukan prediksi terhadap model K-NN classifier, pertama menggunakan data train set dan kemudian dilakukan prediksi data terhadap test set.

```
1 train_pred = knn.predict(x_train)
2 test_pred = knn.predict(x_test)
```

Ditampilkan nilai akurasi (*accuracy score)* dan confusion matrix sebagai parameter evaluasi dari model yang telah diimplementasikan.

```
print("Train Set Accuracy: "+str(accuracy_score(y_train, knn.predict(x_train))*100))
print("Test Set Accuracy: "+str(accuracy_score(y_test,knn.predict(x_test))*100))
```

```
[ ]  1 #Confusion Matrix Train
     2 confusion_matrix(y_train, train_pred)

    array([[2944,   63],
           [1191, 1750]])

[ ]  1 #Confusion Matrix Test
     2 confusion_matrix(y_test, test_pred)

    array([[680,  31],
           [376, 401]])
```

- **Hasil yang Diharapkan**

  Keterbatasan utama saat menggunakan KNN adalah nilai k yang tidak tepat (jumlah tetangga yang salah untuk dipertimbangkan) dapat dipilih yang dapat mengembalikan prediksi yang tidak akurat. akan dilakukan eksperimen terhadap nilai k yang diharapkan dapat memberikan nilai hasil akurasi yang terbaik.

- **Link google colab**

  https://colab.research.google.com/drive/16gBlkt7QH56kFiXrFEp-IUasckespduW?usp=sharing

- **Referensi**

  https://youtu.be/oq68P8Kv7nE

Bagian 4: Evaluasi dan Kesimpulan

- **Evaluasi**

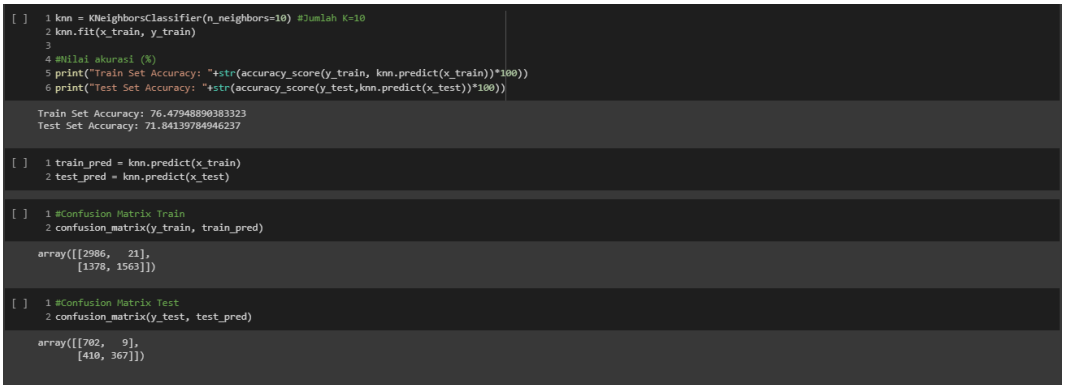
  Metode pengukuran performansi yang digunakan pada tugas ini adalah *confusion matrix* dengan menghitung *accuracy. Akurasi adalah metode evaluasi yang paling umum digunakan dalam pemodelan machine learning untuk mengukur seberapa baik model melakukan prediksi. Akurasi mengukur seberapa banyak prediksi yang dibuat oleh model yang sesuai dengan label aktual dari data yang digunakan untuk melatih model. Secara umum, metode ini digunakan untuk mengukur seberapa baik model dapat membedakan antara kategori yang berbeda dalam klasifikasi.*

  Dapat dilihat pada gambar dibawah, dengan nilai k = 9 nilai akurasi yang didapatkan adalah 78,91% untuk train set dan 72,65% untuk test set. Confusion matrix pada train set memiliki nilai 2944 *true positive* (orang depresi yang terprediksi depresi), 63 *false positive* (orang tidak depresi yang terprediksi depresi), 1191 *false negative* (orang depresi yang

terprediksi tidak depresi) dan 1750 *true negative* (orang tidak depresi yang terprediksi tidak depresi). Confuusion matrix pada test set memiliki nilai 680 *true positive* (orang depresi yang terprediksi depresi), 31 *false positive* (orang tidak depresi yang terprediksi depresi), 376 *false negative* (orang depresi yang terprediksi tidak depresi) dan 401 *true negative* (orang tidak depresi yang terprediksi tidak depresi).

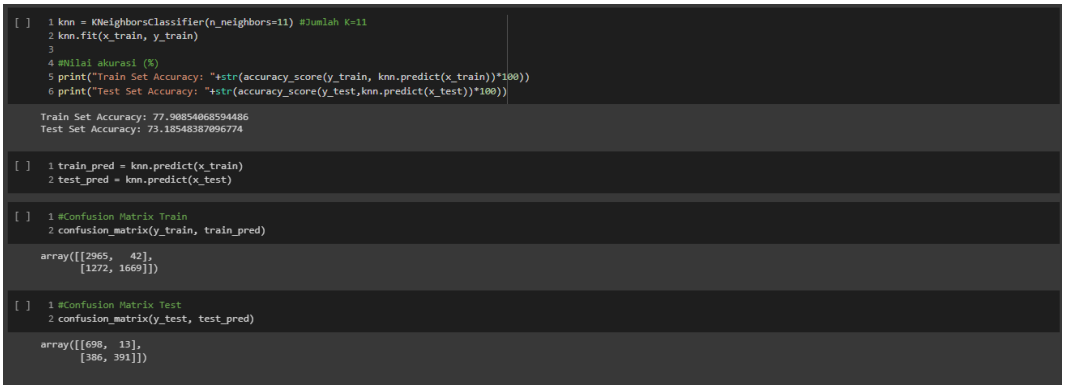
```
[ ]  1 #Klasifikasi dengan KNN
     2 from sklearn.neighbors import KNeighborsClassifier
     3 from sklearn.metrics import accuracy_score, confusion_matrix
     4 knn = KNeighborsClassifier(n_neighbors=9) #Jumlah K=9
     5 knn.fit(x_train, y_train)
     6
     7 #Nilai akurasi (%)
     8 print("Train Set Accuracy: "+str(accuracy_score(y_train, knn.predict(x_train))*100))
     9 print("Test Set Accuracy: "+str(accuracy_score(y_test,knn.predict(x_test))*100))

    Train Set Accuracy: 78.9172831203766
    Test Set Accuracy: 72.64784946236558

[ ]  1 train_pred = knn.predict(x_train)
     2 test_pred = knn.predict(x_test)

[ ]  1 #Confusion Matrix Train
     2 confusion_matrix(y_train, train_pred)

    array([[2944,   63],
           [1191, 1750]])

[ ]  1 #Confusion Matrix Test
     2 confusion_matrix(y_test, test_pred)

    array([[680,  31],
           [376, 401]])
```

Gambar dibawah adalah klasifikasi dengan nilai k=10, nilai akurasi yang didapatkan adalah 76,48% untuk train set dan 71,84% untuk test set. Confusion matrix pada train set memiliki nilai 2986 *true positive* (orang depresi yang terprediksi depresi), 21 *false positive* (orang tidak depresi yang terprediksi depresi), 1378 *false negative* (orang depresi yang terprediksi tidak depresi) dan 1563 *true negative* (orang tidak depresi yang terprediksi tidak depresi). Confuusion matrix pada test set memiliki nilai 702 *true positive* (orang depresi yang terprediksi depresi), 9 *false positive* (orang tidak depresi yang terprediksi depresi), 410 *false negative* (orang depresi yang terprediksi tidak depresi) dan 367 *true negative* (orang tidak depresi yang terprediksi tidak depresi).



```
[ ]  1 knn = KNeighborsClassifier(n_neighbors=10) #Jumlah K=10
     2 knn.fit(x_train, y_train)
     3
     4 #Nilai akurasi (%)
     5 print("Train Set Accuracy: "+str(accuracy_score(y_train, knn.predict(x_train))*100))
     6 print("Test Set Accuracy: "+str(accuracy_score(y_test,knn.predict(x_test))*100))

Train Set Accuracy: 76.4794889383323
Test Set Accuracy: 71.84139784946237

[ ]  1 train_pred = knn.predict(x_train)
     2 test_pred = knn.predict(x_test)

[ ]  1 #Confusion Matrix Train
     2 confusion_matrix(y_train, train_pred)

array([[2986,   21],
       [1378, 1563]])

[ ]  1 #Confusion Matrix Test
     2 confusion_matrix(y_test, test_pred)

array([[702,   9],
       [410, 367]])
```

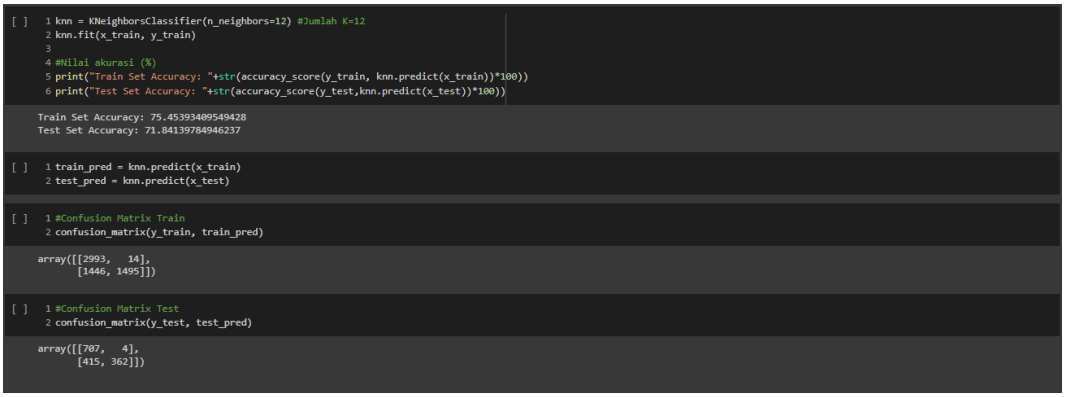
Gambar dibawah ini adalah klasifikasi dengan nilai k = 11, nilai akurasi yang didapatkan adalah 77,91% untuk train set dan 73,19% untuk test set. Confusion matrix pada train set memiliki nilai 2965 *true positive* (orang depresi yang terprediksi depresi), 42 *false positive* (orang tidak depresi yang terprediksi depresi), 1272 *false negative* (orang depresi yang terprediksi tidak depresi) dan 1669 *true negative* (orang tidak depresi yang terprediksi tidak depresi). Confuusion matrix pada test set memiliki nilai 698 *true positive* (orang depresi yang terprediksi depresi), 13 *false positive* (orang tidak depresi yang terprediksi depresi), 386 *false negative* (orang depresi yang terprediksi tidak depresi) dan 391 *true negative* (orang tidak depresi yang terprediksi tidak depresi).



```
[ ]  1 knn = KNeighborsClassifier(n_neighbors=11) #Jumlah K=11
     2 knn.fit(x_train, y_train)
     3
     4 #Nilai akurasi (%)
     5 print("Train Set Accuracy: "+str(accuracy_score(y_train, knn.predict(x_train))*100))
     6 print("Test Set Accuracy: "+str(accuracy_score(y_test,knn.predict(x_test))*100))

Train Set Accuracy: 77.90854068594486
Test Set Accuracy: 73.18548387096774

[ ]  1 train_pred = knn.predict(x_train)
     2 test_pred = knn.predict(x_test)

[ ]  1 #Confusion Matrix Train
     2 confusion_matrix(y_train, train_pred)

array([[2965,   42],
       [1272, 1669]])

[ ]  1 #Confusion Matrix Test
     2 confusion_matrix(y_test, test_pred)

array([[698,  13],
       [386, 391]])
```

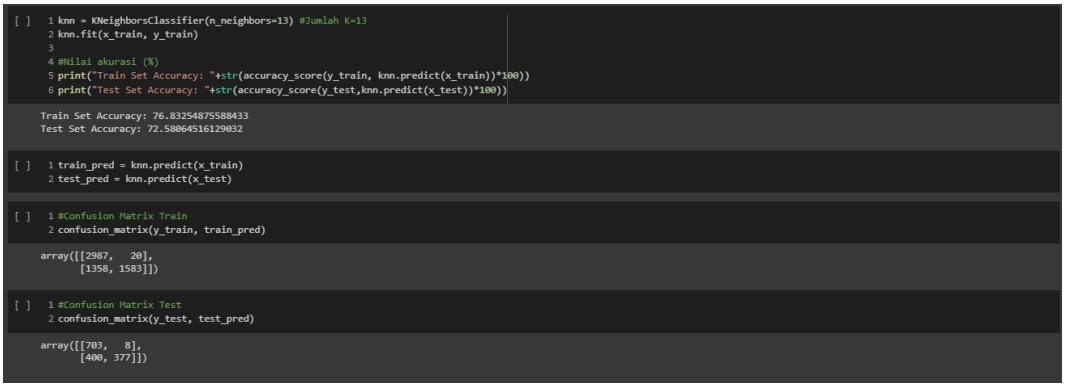
Gambar dibawah ini adalah klasifikasi dengan nilai k = 12, nilai akurasi yang didapatkan adalah 75,45% untuk train set dan 71,84% untuk test set. Confusion matrix pada train set memiliki nilai 2993 *true positive*

(orang depresi yang terprediksi depresi), 14 *false positive* (orang tidak depresi yang terprediksi depresi), 1446 *false negative* (orang depresi yang terprediksi tidak depresi) dan 1495 *true negative* (orang tidak depresi yang terprediksi tidak depresi). Confuusion matrix pada test set memiliki nilai 707 *true positive* (orang depresi yang terprediksi depresi), 4 *false positive* (orang tidak depresi yang terprediksi depresi), 415 *false negative* (orang depresi yang terprediksi tidak depresi) dan 362 *true negative* (orang tidak depresi yang terprediksi tidak depresi).



```
[ ]  1 knn = KNeighborsClassifier(n_neighbors=12) #Jumlah K=12
     2 knn.fit(x_train, y_train)
     3
     4 #Nilai akurasi (%)
     5 print("Train Set Accuracy: "+str(accuracy_score(y_train, knn.predict(x_train))*100))
     6 print("Test Set Accuracy: "+str(accuracy_score(y_test,knn.predict(x_test))*100))

Train Set Accuracy: 75.45393409549428
Test Set Accuracy: 71.84139784946237

[ ]  1 train_pred = knn.predict(x_train)
     2 test_pred = knn.predict(x_test)

[ ]  1 #Confusion Matrix Train
     2 confusion_matrix(y_train, train_pred)

array([[2993,   14],
       [1446, 1495]])

[ ]  1 #Confusion Matrix Test
     2 confusion_matrix(y_test, test_pred)

array([[707,   4],
       [415, 362]])
```

Gambar dibawah ini adalah klasifikasi dengan nilai k = 13, nilai akurasi yang didapatkan adalah 76,83% untuk train set dan 72,58% untuk test set. Confusion matrix pada train set memiliki nilai 2987 *true positive* (orang depresi yang terprediksi depresi), 20 *false positive* (orang tidak depresi yang terprediksi depresi), 1358 *false negative* (orang depresi yang terprediksi tidak depresi) dan 1583 *true negative* (orang tidak depresi yang terprediksi tidak depresi). Confuusion matrix pada test set memiliki nilai 703 *true positive* (orang depresi yang terprediksi depresi), 8 *false positive* (orang tidak depresi yang terprediksi depresi), 400 *false negative* (orang depresi yang terprediksi tidak depresi) dan 377 *true negative* (orang tidak depresi yang terprediksi tidak depresi).



```
[ ]  1 knn = KNeighborsClassifier(n_neighbors=13) #Jumlah K=13
     2 knn.fit(x_train, y_train)
     3
     4 #Nilai akurasi (%)
     5 print("Train Set Accuracy: "+str(accuracy_score(y_train, knn.predict(x_train))*100))
     6 print("Test Set Accuracy: "+str(accuracy_score(y_test,knn.predict(x_test))*100))

Train Set Accuracy: 76.83254875588433
Test Set Accuracy: 72.58064516129032

[ ]  1 train_pred = knn.predict(x_train)
     2 test_pred = knn.predict(x_test)

[ ]  1 #Confusion Matrix Train
     2 confusion_matrix(y_train, train_pred)

array([[2987,   20],
       [1358, 1583]])

[ ]  1 #Confusion Matrix Test
     2 confusion_matrix(y_test, test_pred)

array([[703,   8],
       [400, 377]])
```

Tabel di bawah adalah hasil evaluasi performansi menggunakan akurasi pada setiap nilai K yang diimplementasikan pada metode klasifikasi K-Nearest Neighbour.

Nilai K	Akurasi Model Pada Data Train	Akurasi Model Pada Data Test
9	**78,91%**	72,64%
10	76,47%	71,84%
11	77,90%	**73,18%**
12	75,45%	71,84%
13	76,83%	72,58%

- **Kesimpulan**

Berdasarkan hasil evaluasi diatas dapat disimpulkan bahwa K-NN dapat digunakan untuk melakukan klasifikasi depresi atau tidak-nya berdasarkan data komentar di media sosial reddit dengan cepat dan memiliki nilai akurasi yang baik. Dari beberapa nilai k, k=9, k=10, k=11, k=12, dan k=13. implementasi K-NN dengan nilai k=11 memiliki hasil terbaik dengan akurasi 77,91% untuk train set dan 73,19% untuk test set .Nilai confusion matrix pada train set, 2965 *true positive* (orang depresi

yang terprediksi depresi), 42 *false positive* (orang tidak depresi yang terprediksi depresi), 1272 *false negative* (orang depresi yang terprediksi tidak depresi) dan 1669 *true negative* (orang tidak depresi yang terprediksi tidak depresi). Confuusion matrix pada test set memiliki nilai 698 *true positive* (orang depresi yang terprediksi depresi), 13 *false positive* (orang tidak depresi yang terprediksi depresi), 386 *false negative* (orang depresi yang terprediksi tidak depresi) dan 391 *true negative* (orang tidak depresi yang terprediksi tidak depresi).

Bagian 5: Link Video Presentasi

Video presentasi dari kelompok kami dapat dilihat melalui:

https://drive.google.com/drive/folders/1d4DLPwaA6Y9PmvvHck97kbk0p7DTVh6z?usp=sharing