PENGANTAR PEMBELAJARAN MESIN

Perbandingan Performansi Metode Klasifikasi dengan Sebelumnya

Dosen Pengampu : Lailil Muflikhah, Dr., S.Kom., M.Sc.

Kelas Pengantar Pembelajaran Mesin - A

Disusun oleh:

1. Agung Syahputra [195150201111061]
2. Dimas Nauval Ar Razid [195150201111067]
3. Dyah Ayu Wulandari [195150201111062]
4. Yolanda Saputri [195150201111064]

Program Studi Teknik Informatika

Jurusan Teknik Informatika

Fakultas Ilmu Komputer

Universitas Brawijaya

2021/2022

DAFTAR ISI

BAB I PEMBAGIAN TUGAS

Source Code Program : Agung, Dimas, Dyah, Yola (Bersama melalui google meet)

Laporan : Agung, Dimas, Dyah, Yola

BAB II PENDAHULUAN

A. Deskripsi Masalah

Penyakit diabetes merupakan salah satu penyakit paling banyak diderita oleh manusia seluruh dunia. Menurut WHO (World Health Organization) melaporkan bahwa penderita penyakit diabetes di dunia mendekati jumlah 350 juta orang. Pada tahun 2012 dilaporkan sekitar 1,5 juta kematian disebabkan oleh penyakit diabetes, lebih dari 80% dari jumlah kematian tersebut terjadi di negara-negara berkembang. WHO memprediksikan bahwa tahun 2030 penyakit diabetes menjadi salah satu dari 7 faktor penyebab utama terjadinya kematian di dunia.

Selain itu bahaya yang ditimbulkan penyakit diabetes adalah kebutaan, amputasi, dan gagal ginjal diakibatkan kurangnya kesadaran masyarakat dunia tentang bahaya penyakit diabetes. Penyakit diabetes disebabkan oleh pankreas tidak bisa memproduksi insulin yang cukup sehingga menyebabkan peningkatan produksi glukosa dalam darah (hiperglikemia). Insulin merupakan sebuah hormon berfungsi mengatur gula darah dan bertanggung jawab untuk menghasilkan energi yang dibutuhkan oleh manusia. Dalam kurun waktu, terdapat banyak pasien yang melakukan pemeriksaan kesehatan sehingga berdampak pada penumpukkan jumlah data untuk hasil pemeriksaan dalam mendiagnosa pasien yang berpotensi menderita penyakit diabetes dan menyulitkan dalam mengklasifikasikan data. Akan hal itu, sangat diperlukan teknik yang dapat mendukung dalam menyelesaikan masalah di atas dengan menerapkan data mining untuk klasifikasi dengan memanfaatkan algoritma yaitu Algoritma Naive Bayes dan Support-Vector-Machine (SVM).

Sebelumnya, terdapat sejumlah penelitian terdahulu yang dilakukan oleh beberapa peneliti seperti perbandingan algoritma Naive Bayes dan Support Vector Machine diantaranya adalah penelitian yang dilakukan (Riyanto, 2018) untuk mengklasifikasikan jumlah pembaca online dengan perbandingan metode Naive Bayes dan SVM, yang menghasilkan tingkat akurasi yang tepat yaitu algoritma SVM dengan nilai 63,39%. Selanjutnya penelitian yang dilakukan oleh (Arifin & Sasongko, 2018) membandingkan

algoritma Naive Bayes dan SVM untuk klasifikasi jalur minat SMA yang hasil akurasinya, metode SVM lebih akurat dibanding algoritma Naive Bayes dengan nilai akurasi 97.01%. Ada lagi penelitian yang dilakukan (Widyawati & Sutanto, 2019) yaitu membandingkan algoritma Naive Bayes dan SVM dalam klasifikasi SMS spam berbahasa Indonesia yang hasil akurasinya, algoritma Naive Bayes lah yang lebih unggul dalam hal recall 94% dan Presisi 95%. Sehingga akan muncul pertanyaan pada penelitian ini "Algoritma mana yang tingkat akurasinya lebih tepat dalam mengklasifikasian penyakit diabetes antara Naive Bayes dan Support-Vector-Machine?".

Pada penelitian ini, data berasal dari National Institute of Diabetes and Digestive and Kidney Diseases sebanyak 768 record dan semua pasien yang didiagnosa menderita diabetes atau tidak adalah wanita yang berusia minimal 21 tahun dari Pima Indian heritage. Dataset dari data tersebut terdiri dari beberapa variabel prediktor medis dan satu variabel target yaitu Outcome. Variabel prediktor meliputi jumlah kehamilan (pregnancies) yang dialami pasien, BMI, tingkat Insulin, Age, Glucose, Blood Pressure, Skin Thickness, dan Diabetes Pedigree Function. Tujuan dari penelitian ini untuk mengetahui algoritma mana yang tingkat akurasi yang paling akurat untuk klasifikasi penyakit diabetes antara algoritma-Naïve-Bayes dan Support Vector Machine serta manfaat dari penelitian ini yaitu peneliti dapat memahami konsep data mining.

B. Rancangan Solusi

Pada penelitian ini menggunakan metode algoritma Naïve Bayes dan Support Vector Machine untuk pengklasifikasian penyakit diabetes pada Pima Indians. Kedua metode tersebut akan dibandingkan sehingga dapat mengetahui metode manakah yang memiliki akurasi lebih baik dalam klasifikasi penyakit diabetes pada Pima Indians. Data sampel yang digunakan pada kedua metode akan dibagi menjadi dua kategori, yaitu 80% untuk dijadikan sebagai data training dan 20% dijadikan sebagai data testing.

Pada metode Support Vector Machine akan dilakukan proses pembagian data menjadi dua kelas menggunakan garis vektor. Setelah itu, dilanjutkan proses training yang diikuti proses testing sehingga pada akhirnya metode ini akan memberikan hasil akhir berupa banyaknya data uji yang berhasil diprediksi dengan benar dari seluruh data yang diujikan. Sementara itu, metode Naïve Bayes menggunakan teori probabilitas dalam

melakukan klasifikasi. Pada metode ini juga dilakukan tahapan training dan testing. Dalam tahapan training dilakukan proses perhitungan prior atau peluang awal munculnya kelas dan perhitungan probabilitas kondisional. Kemudian, dalam tahapan testing dilakukan proses perhitungan posterior untuk masing-masing kelas dan penentuan kelas sebagai data yang akan diuji berdasarkan nilai posterior terbesar. Proses yang dilakukan pada bagian akhir sama seperti pada metode Support Vector Machine, yaitu memberikan hasil akhir berupa banyaknya data uji yang berhasil diprediksi dengan benar dari seluruh data yang diujikan. Masing-masing metode akan menghasilkan nilai prediksi atau akurasi yang berbeda-beda. Dari hasil prediksi tersebut nantinya dapat disimpulkan metode manakah yang lebih akurat untuk klasifikasi penyakit diabetes pada Pima Indians.

C. Dataset

Dataset : https://www.kaggle.com/uciml/pima-indians-diabetes-database

Dalam kasus ini, variabel yang kami gunakan adalah Pregnancies (jumlah kehamilan), Glucose (konsentrasi glukosa dalam plasma darah), Blood Pressure (tekanan darah), Skin Thickness (ketebalan lipatan kulit trisep), Insulin (banyaknya insulin dalam darah), BMI (Berat Badan), Diabetes (peluang mengalami diabetes berdasarkan gen), Age (Usia), Outcome (Hasil). Kami menggunakan variabel-variabel di atas karena pada dasarnya variabel-variabel tersebut merupakan faktor-faktor penyebab diabetes. Konsep pengambilan sampel pada data ini adalah dari seluruh populasi yaitu Seluruh masyarakat Pima Indians diambil masyarakat yang berjenis kelamin perempuan sebagai sampel penelitian. Kemudian dari sampel tersebut, diambil 80% untuk dijadikan data *training* dan sisanya yaitu 20% untuk dijadikan *testing*.

D. Implementasi

❖ EDA & Data Preprocessing

1. Menyiapkan library yang dibutuhkan yaitu numpy, pandas, matplotlib.pyplot dan seaborrn

```
[ ]  #Menyiapkan Pustaka
     import numpy as np
     import pandas as pd

     #Visualizations
     import matplotlib.pyplot as plt
     import seaborn as sns
     sns.set()
```

2. Import dataset dari https://raw.githubusercontent.com/agungsya/tes/master/diabetes.csv

```
[ ] ! wget https://raw.githubusercontent.com/agungsya/tes/master/diabetes.csv
```

```
--2021-05-27 17:38:18--  https://raw.githubusercontent.com/agungsya/tes/master/diabetes.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... con
HTTP request sent, awaiting response... 200 OK
Length: 23873 (23K) [text/plain]
Saving to: 'diabetes.csv.10'

diabetes.csv.10     100%[===================>]  23.31K  --.-KB/s    in 0.002s

2021-05-27 17:38:18 (13.7 MB/s) - 'diabetes.csv.10' saved [23873/23873]
```

3. Membaca data

```
[ ] # Baca data
    df = pd.read_csv("diabetes.csv")
    df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

4. Mencetak ukuran data beserta info dataset

```
[ ] print(f'Ukuran Data : {df.shape}')
```

```
Ukuran Data : (768, 9)
```

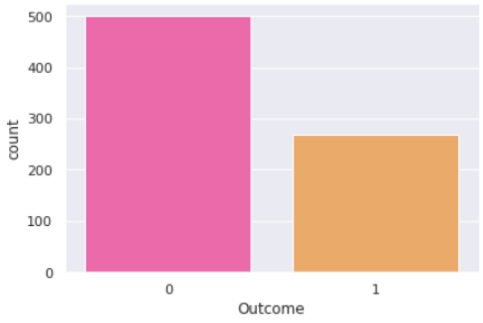
```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

5. Menampilkan EDA

```
[ ] sns.countplot(df['Outcome'],palette='spring');
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the follow
  FutureWarning
```



6. Melakukan pemrosesan data untuk menampilkan beberapa data yang akan dipilih untuk dilakukan clustering

```
[ ]  sns.pairplot(df,hue='Outcome');
```

Dari gambar diatas, kami dapat melihat clusters dengan pengelompokkan yang baik, yaitu:

1. Glucose dan BloodPressure ;
2. SkinThickness dan BloodPressure ;

7. Impute Missing Value

```
[ ]  df['Glucose']=df['Glucose'].replace(0,np.nan)
     df['BloodPressure']=df['BloodPressure'].replace(0,np.nan)
     df['SkinThickness']=df['SkinThickness'].replace(0,np.nan)
     df['Insulin']=df['Insulin'].replace(0,np.nan)
     df['BMI']=df['BMI'].replace(0,np.nan)
```

```
[ ]  #cek missing value
     total=df.isnull().sum().sort_values(ascending = False)
     print(total)
```

```
Insulin                     374
SkinThickness               227
BloodPressure                35
BMI                          11
Glucose                       5
Outcome                       0
Age                           0
DiabetesPedigreeFunction      0
Pregnancies                   0
dtype: int64
```

8. Sebelum melakukan imputasi pada nilai missing, dicek terlebih dahulu skewness dari data, untuk menentukan imputasi yang tepat

```
[ ]  datatr=['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
     df[datatr].skew(axis=0, skipna=True)
```

```
Glucose          0.530989
BloodPressure    0.134153
SkinThickness    0.690619
Insulin          2.166464
BMI              0.593970
dtype: float64
```

```
[ ]  df1=df.loc[df['Outcome']==0]
     df2=df.loc[df['Outcome']==1]
```

```
[ ]  df1[datatr].skew(axis=0, skipna=True)
```

```
Glucose          0.654938
BloodPressure    0.161298
SkinThickness    0.358207
Insulin          2.512829
BMI              0.469779
dtype: float64
```

```
[ ]  df2[datatr].skew(axis=0, skipna=True)
```

```
Glucose          0.090633
BloodPressure    0.077426
SkinThickness    1.427105
Insulin          1.886631
BMI              1.020176
dtype: float64
```

```
[ ] df1[datatr].describe()
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI
count	497.000000	481.000000	361.000000	264.000000	491.000000
mean	110.643863	70.877339	27.235457	130.287879	30.859674
std	24.776906	12.161223	10.026491	102.482237	6.560737
min	44.000000	24.000000	7.000000	15.000000	18.200000
25%	93.000000	62.000000	19.000000	66.000000	25.600000
50%	107.000000	70.000000	27.000000	102.500000	30.100000
75%	125.000000	78.000000	33.000000	161.250000	35.300000
max	197.000000	122.000000	60.000000	744.000000	57.300000

```
[ ] df2[datatr].describe()
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI
count	266.000000	252.000000	180.000000	130.000000	266.000000
mean	142.319549	75.321429	33.000000	206.846154	35.406767
std	29.599199	12.299866	10.327595	132.699898	6.614982
min	78.000000	30.000000	7.000000	14.000000	22.900000
25%	119.000000	68.000000	27.000000	127.500000	30.900000
50%	140.000000	74.500000	32.000000	169.500000	34.300000
75%	167.000000	84.000000	39.000000	239.250000	38.925000
max	199.000000	114.000000	99.000000	846.000000	67.100000

```
[ ] #imputasi class mean pada attribute Glucose dan Insulin
    df1['Glucose'].fillna(df1['Glucose'].mean(),inplace=True)
    df2['Glucose'].fillna(df2['Glucose'].mean(),inplace=True)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:4536: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  downcast=downcast,
```

```
[ ] df1['Insulin'].fillna(df1['Insulin'].median(),inplace=True)
    df2['Insulin'].fillna(df2['Insulin'].median(),inplace=True)
    data_imputasi=df1.append(df2)
    data_imputasi
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:4536: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  downcast=downcast,
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	1	85.0	66.0	29.0	102.5	26.6	0.351	31	0
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
5	5	116.0	74.0	NaN	102.5	25.6	0.201	30	0
7	10	115.0	NaN	NaN	102.5	35.3	0.134	29	0
10	4	110.0	92.0	NaN	102.5	37.6	0.191	30	0
...	...	...	...	...	...	...	...	...	...
755	1	128.0	88.0	39.0	110.0	36.5	1.057	37	1
757	0	123.0	72.0	NaN	169.5	36.3	0.258	52	1
759	6	190.0	92.0	NaN	169.5	35.5	0.278	66	1
761	9	170.0	74.0	31.0	169.5	44.0	0.403	43	1
766	1	126.0	60.0	NaN	169.5	30.1	0.349	47	1

768 rows × 9 columns

```
[ ] #Imputasi Mean pada bloodpressure, skinthickness, dan BMI
    mean1 = data_imputasi['BloodPressure'].mean()
    data_imputasi['BloodPressure'].fillna(mean1,inplace=True)
    mean2 = data_imputasi['SkinThickness'].mean()
    data_imputasi['SkinThickness'].fillna(mean2,inplace=True)
    mean3 = data_imputasi['BMI'].mean()
    data_imputasi['BMI'].fillna(mean3,inplace=True)
```

```
[ ] data_imputasi
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	1	85.0	66.000000	29.00000	102.5	26.6	0.351	31	0
3	1	89.0	66.000000	23.00000	94.0	28.1	0.167	21	0
5	5	116.0	74.000000	29.15342	102.5	25.6	0.201	30	0
7	10	115.0	72.405184	29.15342	102.5	35.3	0.134	29	0
10	4	110.0	92.000000	29.15342	102.5	37.6	0.191	30	0
...	...	...	...	...	...	...	...	...	...
755	1	128.0	88.000000	39.00000	110.0	36.5	1.057	37	1
757	0	123.0	72.000000	29.15342	169.5	36.3	0.258	52	1
759	6	190.0	92.000000	29.15342	169.5	35.5	0.278	66	1
761	9	170.0	74.000000	31.00000	169.5	44.0	0.403	43	1
766	1	126.0	60.000000	29.15342	169.5	30.1	0.349	47	1

768 rows × 9 columns

```
[ ] data_imputasi.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.000000	6.00000	17.00
Glucose	768.0	121.697358	30.462008	44.000	99.75000	117.000000	141.00000	199.00
BloodPressure	768.0	72.405184	12.096346	24.000	64.00000	72.202592	80.00000	122.00
SkinThickness	768.0	29.153420	8.790942	7.000	25.00000	29.153420	32.00000	99.00
Insulin	768.0	141.753906	89.100847	14.000	102.50000	102.500000	169.50000	846.00
BMI	768.0	32.457464	6.875151	18.200	27.50000	32.400000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.372500	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.000000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.000000	1.00000	1.00

```
#Check Missing Value
total=data_imputasi.isnull().sum().sort_values(ascending = False)
print(total)
```

```
Outcome                  0
Age                      0
DiabetesPedigreeFunction 0
BMI                      0
Insulin                  0
SkinThickness            0
BloodPressure            0
Glucose                  0
Pregnancies              0
dtype: int64
```

9. Menghitung Normalization Min Max

```
[ ]  def cetak_rentang(df_input):
       list_fitur = df_input.columns[:-1] #mengambil nama kolom, kecuali yang terakhir (kelas)
       for fitur in list_fitur:
         max = df_input[fitur].max()
         min = df_input[fitur].min()
         print("Rentang fitur ",fitur," adalah ",max-min)
```

```
[ ]  cetak_rentang(data_imputasi)
```

```
Rentang fitur  Pregnancies  adalah  17
Rentang fitur  Glucose  adalah  155.0
Rentang fitur  BloodPressure  adalah  98.0
Rentang fitur  SkinThickness  adalah  92.0
Rentang fitur  Insulin  adalah  832.0
Rentang fitur  BMI  adalah  48.89999999999999
Rentang fitur  DiabetesPedigreeFunction  adalah  2.342
Rentang fitur  Age  adalah  60
```

```
[ ]  def minmax(df_input):
       list_fitur = df_input.columns[:-1]
       for fitur in list_fitur:
         max = df_input[fitur].max()
         min = df_input[fitur].min()
         df_input[fitur] = (df_input[fitur]-min)/(max-min)
       return df_input
```

```
[ ]  data_normal = minmax(data_imputasi)
```

```
[ ] data_normal.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	0.226180	0.501273	0.493930	0.240798	0.153550	0.291564	0.168179	0.204015	0.348958
std	0.198210	0.196529	0.123432	0.095554	0.107092	0.140596	0.141473	0.196004	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.058824	0.359677	0.408163	0.195652	0.106370	0.190184	0.070773	0.050000	0.000000
50%	0.176471	0.470968	0.491863	0.240798	0.106370	0.290389	0.125747	0.133333	0.000000
75%	0.352941	0.625806	0.571429	0.271739	0.186899	0.376278	0.234095	0.333333	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
[ ] data_normal
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	0.058824	0.264516	0.428571	0.239130	0.106370	0.171779	0.116567	0.166667	0
3	0.058824	0.290323	0.428571	0.173913	0.096154	0.202454	0.038002	0.000000	0
5	0.294118	0.464516	0.510204	0.240798	0.106370	0.151329	0.052519	0.150000	0
7	0.588235	0.458065	0.493930	0.240798	0.106370	0.349693	0.023911	0.133333	0
10	0.235294	0.425806	0.693878	0.240798	0.106370	0.396728	0.048249	0.150000	0
...	...	...	...	...	...	...	...	...	...
755	0.058824	0.541935	0.653061	0.347826	0.115385	0.374233	0.418019	0.266667	1
757	0.000000	0.509677	0.489796	0.240798	0.186899	0.370143	0.076857	0.516667	1
759	0.352941	0.941935	0.693878	0.240798	0.186899	0.353783	0.085397	0.750000	1
761	0.529412	0.812903	0.510204	0.260870	0.186899	0.527607	0.138770	0.366667	1
766	0.058824	0.529032	0.367347	0.240798	0.186899	0.243354	0.115713	0.433333	1

768 rows × 9 columns

```
[ ] cetak_rentang(data_normal)
```

```
Rentang fitur  Pregnancies  adalah  1.0
Rentang fitur  Glucose  adalah  1.0
Rentang fitur  BloodPressure  adalah  1.0
Rentang fitur  SkinThickness  adalah  1.0
Rentang fitur  Insulin  adalah  1.0
Rentang fitur  BMI  adalah  1.0
Rentang fitur  DiabetesPedigreeFunction  adalah  1.0
Rentang fitur  Age  adalah  1.0
```

❖ **Metode K-Means Clustering**

1. Mengimport waktu yaitu timeit, library yaitu Kmeans dan confusion_matrix, classification_report, accuracy_score

```
[ ]  #time counter
     import timeit
```

```
[ ]  #import library
     from sklearn.cluster import KMeans
     from sklearn.metrics import confusion_matrix,classification_report,accuracy_score

     #Melakukan clustering
     startkmc = timeit.default_timer()
     kmeans = KMeans(3,init='k-means++')
     stopkmc = timeit.default_timer()
     kmeans.fit(data_normal.drop('Outcome',axis=1))
     print(confusion_matrix(data_normal.Outcome,kmeans.labels_))
```

```
[[ 45 337 118]
 [104  52 112]
 [  0   0   0]]
```

2. Menampilkan kmeans labels

```
[ ]  print(kmeans.labels_)
```

```
[1 1 1 2 1 2 1 0 2 1 2 2 2 1 1 2 1 2 0 2 2 2 1 1 1 1 1 1 2 1 0 0 1 1 1 1 1
 2 1 0 1 0 1 1 2 1 1 1 1 2 1 1 2 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 0 1 1 1 1
 1 1 1 1 2 1 1 2 1 1 1 1 1 1 2 2 1 0 1 2 1 2 1 0 1 0 1 1 1 0 2 0 1 1 1 1 1
 1 1 1 2 2 1 1 1 1 2 1 2 2 1 1 0 1 1 2 1 1 1 0 2 2 2 2 1 1 1 0 1 1 1 1 1 1
 1 0 2 0 2 1 2 1 1 1 1 1 0 0 1 2 2 0 1 1 1 1 2 1 1 2 1 2 2 2 0 1 1 1 2 0 1
 2 1 1 1 1 2 1 1 1 1 1 1 1 1 2 2 2 1 2 1 0 1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 2
 2 2 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 0 1 1 1 0
 1 1 1 1 0 1 1 1 1 1 2 0 1 2 1 1 1 1 1 1 1 2 1 2 1 2 2 1 2 1 2 1 1 1 0 0 1
 1 2 1 2 2 2 2 1 1 1 0 0 1 2 1 1 1 1 2 1 1 2 1 1 2 1 2 1 1 2 1 2 1 1 2 2 2
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 0 1 1 1 0 0 1 1 2 1 1 2 1 2 2 2 1 1 1 1 1
 2 0 1 1 1 1 0 1 1 2 1 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 0 1 1 1 1 2 1 1 1 0
 1 1 1 1 1 2 1 1 1 1 2 2 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 2 2 1 2 2 2 1 2 0 2
 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 0 1 1 1 0 2 1 1 2 1 1 1 2 1 2 1 1 0 1 1 2
 1 1 2 1 1 1 2 2 1 1 1 2 1 1 2 2 1 1 1 2 0 0 1 0 2 2 0 2 2 0 2 1 0 2 2 2 2
 0 2 1 2 2 0 2 2 0 2 2 1 1 2 0 0 2 2 0 0 1 0 2 2 2 2 0 1 1 1 2 0 2 0 2 2 0
 0 2 0 2 2 0 0 0 0 0 2 0 2 0 2 2 0 1 1 0 2 2 0 0 2 2 1 1 2 0 2 0 0 2 0 2 0
 2 1 1 2 2 1 2 0 1 0 1 2 1 0 2 2 0 1 0 1 0 2 0 1 0 2 1 1 0 2 0 2 1 1 2 1 1
 0 2 2 0 0 2 1 2 0 0 2 1 0 0 2 0 2 2 2 0 0 1 0 1 2 0 2 0 0 1 0 0 1 2 0 0 1
 0 0 2 1 0 1 1 2 2 2 1 0 0 0 2 2 2 0 2 0 2 2 1 0 2 1 2 0 0 2 0 1 1 0 0 2 2
 0 2 2 0 0 2 0 0 0 0 2 2 1 2 2 2 2 0 0 2 0 1 0 2 2 2 2 2 0 2 1 0 1 0 2 2 2
 0 2 0 2 2 1 2 0 0 2 0 1 1 0 1 2 2 0 0 2 0 0 2 0 0 2 2 1]
```

3. Menampilkan klasifikasi data

```
[ ] labels = kmeans.labels_.copy()
    X=data_normal.copy()
    labels_true=df['Outcome'].copy()
```

```
[ ] print(classification_report(data_normal.Outcome,kmeans.labels_))
```

```
              precision    recall  f1-score   support

           0       0.30      0.09      0.14       500
           1       0.13      0.19      0.16       268
           2       0.00      0.00      0.00         0

    accuracy                           0.13       768
   macro avg       0.15      0.09      0.10       768
weighted avg       0.24      0.13      0.15       768

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetric
  _warn_prf(average, modifier, msg_start, len(result))
```

```
[ ] f"Accuracy : {np.round(100*accuracy_score(data_normal.Outcome,kmeans.labels_),2)}"
```

```
'Accuracy : 12.63'
```

❖ **Evaluasi Clustering (K-Means Clustering)**

1. Davies-Bouldin Index

```
[ ] from sklearn.metrics import davies_bouldin_score
    davies_bouldin_score(X, labels)
```

```
2.114777786380618
```

2. Silhouette Coefficient

```
[ ] from sklearn import metrics
    from sklearn.metrics import pairwise_distances
    metrics.silhouette_score(X, labels, metric='euclidean')
```

```
0.1959644881449931
```

3. Rand Index

```
[ ] from sklearn import metrics
    metrics.adjusted_rand_score(labels_true, labels)
```

```
-0.0033869116460147368
```

4. Mutual Information Based Scores

```
[ ]  from sklearn import metrics
     metrics.adjusted_mutual_info_score(labels_true, labels)
```

```
-0.00037520472959636804
```

5. Homogeneity

```
[ ]  from sklearn import metrics
     metrics.homogeneity_score(labels_true, labels)
```

```
0.0015364237427036296
```

❖ **Hierarchical Clustering Single Linkage**

1. Mengimport waktu yaitu timeit, library yaitu AgglomerativeClustering, confusion_matrix,classification_report,accuracy_score

```
[ ]  #time counter
     import timeit
```

```
[ ]
     from sklearn.cluster import AgglomerativeClustering
     from sklearn.metrics import confusion_matrix,classification_report,accuracy_score

     #Melakukan clustering
     startagc = timeit.default_timer()
     agc = AgglomerativeClustering(n_clusters=3,linkage='single').fit(data_normal.drop('Outcome',axis=1))
     stopagc = timeit.default_timer()
     print(confusion_matrix(data_normal.Outcome,agc.labels_))
```

```
[[500   0   0]
 [266   1   1]
 [  0   0   0]]
```

2. Menampilkan agc labels

```
[ ]  print(agc.labels_)
```

```
[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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 2 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 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
 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]
```

3. Menampilkan klasifikasi data

```
[ ]  print(classification_report(data_normal.Outcome,agc.labels_))
```

```
              precision    recall  f1-score   support

           0       0.65      1.00      0.79       500
           1       1.00      0.00      0.01       268
           2       0.00      0.00      0.00         0

    accuracy                           0.65       768
   macro avg       0.55      0.33      0.27       768
weighted avg       0.77      0.65      0.52       768

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefine
  _warn_prf(average, modifier, msg_start, len(result))
```

```
[ ]  f"Accuracy : {np.round(100*accuracy_score(data_normal.Outcome,agc.labels_),2)}"
```

```
'Accuracy : 65.23'
```

❖ Evaluasi Clustering (Hierarchical Clustering Single Linkage)

1. Davies-Bouldin Index

```
[ ]  from sklearn.metrics import davies_bouldin_score
     davies_bouldin_score(X, labels)
```

```
0.4958042433529049
```

2. Silhouette Coefficient

```
[ ]  from sklearn import metrics
     from sklearn.metrics import pairwise_distances
     metrics.silhouette_score(X, labels, metric='euclidean')
```

```
0.3306126838587031
```

3. Rand Index

```
[ ]  from sklearn import metrics
     metrics.adjusted_rand_score(labels_true, labels)
```

```
0.004510848913906455
```

4. Mutual Information Based Scores

```
[ ]  from sklearn import metrics
     metrics.adjusted_mutual_info_score(labels_true, labels)
```

```
0.003201877159732065
```

5. Homogeneity

```
[ ]  from sklearn import metrics
     metrics.homogeneity_score(labels_true, labels)
```

```
0.00424864534514032
```

❖ **Metode Klasifikasi Naïve Bayes**

1. Mengimport waktu yaitu timeit, mengimport train test split

```
[ ]  #time counter
     import timeit
```

```
[ ]  # Import train_test_split function
     from sklearn.model_selection import train_test_split
```

2. Membagi data menjadi data training dan data testing

```
[ ]  array = data_normal.values
     x = array[:,0:8]
     y = array[:,8]
     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 123)
```

3. Mengimport library, memanggil fungsi klasifikasi Naive Bayes dan memasukkan data training

```
[ ]  # Import Gaussian Naive Bayes model
     from sklearn.naive_bayes import GaussianNB

     # Mengaktifkan/memanggil/membuat fungsi klasifikasi Naive bayes
     modelnb = GaussianNB()

     # Memasukkan data training pada fungsi klasifikasi naive bayes
     nbtrain = modelnb.fit(x_train, y_train)
     nbtrain.class_count_
```

```
array([402., 212.])
```

4. Menentukan hasil prediksi

```
[ ]  # Menentukan hasil prediksi dari x_test
     y_pred = nbtrain.predict(x_test)
     y_pred
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1.,
       1., 0., 0., 0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 1., 0., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0.,
       0., 0., 1., 0., 1., 1., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0.,
       0., 1., 1., 0., 0., 0., 0., 1., 0., 1., 1., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 1., 0.,
       1., 0., 0., 0., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0.,
       0., 1., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0.])
```

5. Menentukan probabilitas hasil prediksi

```
# Menentukan probabilitas hasil prediksi
nbtrain.predict_proba(x_test)
```

```
array([[9.96230982e-01, 3.76901845e-03],
       [9.44042634e-01, 5.59573660e-02],
       [9.69620224e-01, 3.03797762e-02],
       [8.04081704e-01, 1.95918296e-01],
       [9.41234550e-01, 5.87654500e-02],
       [6.40679132e-01, 3.59320868e-01],
       [9.84491104e-01, 1.55088963e-02],
       [8.79300587e-01, 1.20699413e-01],
       [9.88425912e-01, 1.15740884e-02],
       [9.92854206e-01, 7.14579386e-03],
       [9.30330984e-01, 6.96690160e-02],
       [9.90517670e-01, 9.48232986e-03],
       [9.50166739e-01, 4.98332606e-02],
       [3.48688354e-01, 6.51311646e-01],
       [7.13877986e-01, 2.86122014e-01],
       [9.70301044e-01, 2.96989562e-02],
       [6.41131822e-02, 9.35886818e-01],
       [2.55749733e-02, 9.74425027e-01],
       [9.82053351e-01, 1.79466494e-02],
       [6.37962136e-01, 3.62037864e-01],
       [6.00220552e-01, 3.99779448e-01],
       [3.09821588e-01, 6.90178412e-01],
       [3.01677321e-01, 6.98322679e-01],
       [7.25657450e-01, 2.74342550e-01],
       [8.76896503e-01, 1.23103497e-01],
       [3.22791899e-01, 6.77208101e-01],
       [9.92265239e-01, 7.73476083e-03],
       [9.92612624e-01, 7.38737583e-03],
       [1.99401612e-01, 8.00598388e-01],
       [8.43079958e-01, 1.56920042e-01],
       [3.29255846e-02, 9.67074415e-01],
       [5.96211354e-01, 4.03788646e-01],
       [9.85006395e-01, 1.49936049e-02],
       [8.68107130e-01, 1.31892870e-01],
       [9.69206329e-01, 3.07936714e-02],
       [7.28159381e-01, 2.71840619e-01],
       [8.47333594e-01, 1.52666406e-01],
       [9.17837169e-01, 8.21628309e-02],
       [9.74129664e-01, 2.58703358e-02],
       [8.96335236e-02, 9.10366476e-01],
```

6. Mengimport confusion matrix model dan hasilnya

```
[ ]  # import confusion_matrix model
     from sklearn.metrics import confusion_matrix
     confusion_matrix(y_test, y_pred)
```

```
array([[87, 11],
       [25, 31]])
```

7. Merapikan hasil confusion matrix

```
[ ]  # Merapikan hasil confusion matrix
     y_actual1 = pd.Series([1, 0,1,0,1,0,1,0,1,0,0,1,1,0,1,1,0,0], name = "actual")
     y_pred1 = pd.Series([1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1], name = "prediction")
     df_confusion = pd.crosstab(y_actual1, y_pred1)
     df_confusion
```

prediction	0	1
actual		
0	7	2
1	1	8

❖ **Evaluasi klasifikasi Naive Bayes**

```
[ ]  # Menghitung nilai akurasi dari klasifikasi naive bayes
     from sklearn.metrics import classification_report
     print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0.0	0.78	0.89	0.83	98
1.0	0.74	0.55	0.63	56
accuracy			0.77	154
macro avg	0.76	0.72	0.73	154
weighted avg	0.76	0.77	0.76	154

❖ **Metode Klasifikasi SVM**

1. Mengimport waktu yaitu timeit, library train test split, SVC, dan membagi data menjadi data training dan testing

```
[ ]  #time counter
     import timeit
```

```
[ ]  #import library
     from sklearn.model_selection import train_test_split
     from sklearn.svm import SVC

     #membagi data menjadi data train dan test
     array = data_normal.values
     x = array[:,0:8]
     y = array[:,8]
     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 123)
```

2. Menghitung SVC pada klasifikasi SVM

```
[ ]  svc_model = SVC()
     svc_model.fit(x_train, y_train)
     startsvm = timeit.default_timer()
     preds_svm = svc_model.predict(x_test)
     stopsvm = timeit.default_timer()
     print(confusion_matrix(y_test,preds_svm))
```

```
[[89  9]
 [28 28]]
```

❖ **Evaluasi klasifikasi SVM**

```
[ ]  # Menghitung nilai akurasi dari klasifikasi
     print(classification_report(y_test,preds_svm))
```

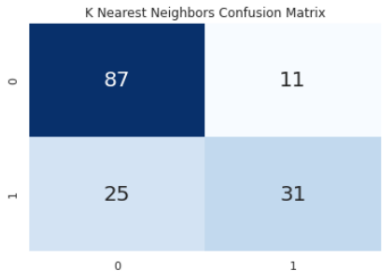
```
              precision    recall  f1-score   support

         0.0       0.76      0.91      0.83        98
         1.0       0.76      0.50      0.60        56

    accuracy                           0.76       154
   macro avg       0.76      0.70      0.72       154
weighted avg       0.76      0.76      0.75       154
```

E. Manualisasi Evaluasi Klasifikasi

❖ Evaluasi Klasifikasi Perbandingan Akurasi

- Naive Bayes

```
[ ]  plt.title("K Nearest Neighbors Confusion Matrix")
     sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,cmap="Blues",fmt="d",cbar=False, annot_kws={"size": 20});
```



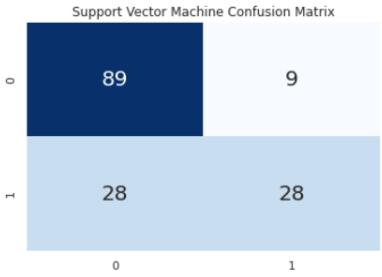
1. True Positives (TP) : 87
2. False Negatives (FN) : 11
3. False Positive (FP) : 25
4. True Negatives (TN) : 31

Akurasi

```
[ ]  akurasi = (87 + 31) / (87+11+25+31)
     print ("Accuracy : ",akurasi)

     Accuracy :  0.7662337662337663
```

- SVM

```
[ ]  plt.title("Support Vector Machine Confusion Matrix")
     sns.heatmap(confusion_matrix(y_test,preds_svm),annot=True,cmap="Blues",fmt="d",cbar=False, annot_kws={"size": 20});
```



1. True Positives (TP) : 89
2. False Negatives (FN) : 9
3. False Positive (FP) : 28
4. True Negatives (TN) : 28

Akurasi

```
[ ]  akurasi = (89 + 28) / (89+28+9+28)
     print ("Accuracy : ",akurasi)

     Accuracy :  0.7597402597402597
```

❖ **Evaluasi Klasifikasi Perbandingan Recall**

- Naive Bayes

▾ Recall

```
[ ]  recall = 87 / (87 + 11)
     print ("Recall : ",recall)
```

```
Recall :  0.8877551020408163
```

- SVM

▾ Recall

```
[ ]  recall = 89 / (89 + 9)
     print ("Recall : ",recall)
```

```
Recall :  0.9081632653061225
```

❖ **Evaluasi Klasifikasi Perbandingan Precision**

- Naive Bayes

▾ Precision

```
[ ]  precision = 87 / (87 + 25)
     print ("Precision : ",precision)
```

```
Precision :  0.7767857142857143
```

- SVM

Precision

```
[ ]  precision = 89 / (89 + 28)
     print ("Precision : ",precision)

     Precision :  0.7606837606837606
```

❖ **Evaluasi Klasifikasi Perbandingan F1_Measure**

- Naive Bayes

```
[ ]  f1 = (2*precision*recall) / (precision + recall)
     print ("F1-Measure : ",f1)

     F1-Measure :  0.8285714285714286
```

- SVM

```
[ ]  f1 = (2*precision*recall) / (precision + recall)
     print ("F1-Measure : ",f1)

     F1-Measure :  0.8279069767441861
```

F. Hasil dan Analisis

Ketika data dilakukan clustering terlebih dahulu yang bertujuan untuk mengelompokkan data yang memiliki kesamaan ciri dan memisahkan data ke dalam klaster yang berbeda untuk objek-objek yang memiliki ciri yang berbeda. Berbeda dengan klasifikasi, yang memiliki kelas yang telah didefinisikan sebelumnya. Dalam klasterisasi, klaster akan terbentuk sendiri berdasarkan ciri objek yang dimiliki dan kriteria pengelompokan yang telah ditentukan. Untuk menunjukkan klasterisasi dari sekumpulan data, suatu kriteria pengelompokan haruslah ditentukan sebelumnya.

Perbedaan kriteria pengelompokan akan memberikan dampak perbedaan hasil akhir dari klaster yang terbentuk. Metode Clustering yang kami gunakan yaitu metode K-Means Clustering dan Hierarchical Clustering Single Linkage. Metode K-Means Clustering merupakan salah satu metode pengelompokan data non hirarki yang digunakan untuk mempartisi N objek data ke dalam K kelompok. Setiap kelompok data memiliki jarak terdekat dengan centroidnya masing-masing. Sedangkan Metode Hierarchical Clustering adalah metode analisis kelompok yang berusaha untuk membangun sebuah hirarki kelompok data. • Strategi pengelompokannya umumnya ada 2 jenis yaitu Agglomerative (Bottom-Up) dan Divisive (Top-Down). (Pada bagian ini akan dibatasi hanya menggunakan konsep Agglomerative).

Hasil dari clustering tersebut selanjutnya dilakukan evaluasi clustering menggunakan Davies-Bouldin Index, Silhouette Coefficient, Rand Index, Mutual Information Based Scores dan Homogeneity. Langkah selanjutnya adalah melakukan klasifikasi seperti pada tugas kelompok sebelumnya, yaitu kami menggunakan metode klasifikasi Naive Bayes dan metode klasifikasi Support Vector Machine (SVM). Melalui hasil klasifikasi setiap metode tersebut, maka dilakukan evaluasi klasifikasi pada masing-masing metode klasifikasi. Pada evaluasi klasifikasi, pertama yang dilakukan adalah melakukan perhitungan akurasi pada metode Naive Bayes dan SVM yang dapat dilihat hasilnya bahwa perbandingan nilai akurasi dari kedua metode tersebut adalah lebih baik menggunakan metode klasifikasi Naive Bayes yaitu dengan hasil 0.7662337662337663, sedangkan jika menggunakan metode SVM hasil akurasinya hanya sebesar 0.7597402597402597. Metode evaluasi klasifikasi selanjutnya adalah Recall, dapat dilihat hasilnya bahwa perbandingan nilai recall dari kedua metode tersebut adalah lebih baik menggunakan metode klasifikasi SVM yaitu dengan hasil 0.9081632653061225, sedangkan jika menggunakan metode Naive Bayes hasil recallnya hanya sebesar 0.8877551020408163. Metode evaluasi klasifikasi ketiga adalah Precision, dapat dilihat hasilnya bahwa perbandingan nilai precision dari kedua metode tersebut adalah lebih baik menggunakan metode klasifikasi Naive Bayes yaitu dengan hasil 0.7767857142857143, sedangkan jika menggunakan metode SVM hasil precisionnya hanya sebesar 0.7606837606837606. Metode evaluasi klasifikasi keempat adalah F1_Measure, dapat dilihat hasilnya bahwa perbandingan nilai F1_Measure dari kedua

metode tersebut adalah hampir sama, hanya memiliki sedikit selisih namun tetap lebih baik menggunakan metode klasifikasi Naive Bayes yaitu dengan hasil 0.8285714285714286, sedangkan jika menggunakan metode SVM hasil precisionnya hanya sebesar 0.8279069767441861. Dari empat perbandingan evaluasi klasifikasi pada metode klasifikasi Naive Bayes dan SVM adalah hasilnya akan lebih baik menggunakan metode klasifikasi Naive Bayes karena dari keempat perbandingan, Naive Bayes mendapatkan tiga kali hasil lebih baik yaitu pada perhitungan akurasi, Precision dan F1_Measure.

BAB III KESIMPULAN

Jadi menurut percobaan yang telah kelompok kami lakukan maka dapat disimpulkan bahwa metode klasifikasi Naive Bayes lebih baik dari metode klasifikasi SVM dikarenakan dari empat perbandingan evaluasi klasifikasi pada metode klasifikasi Naive Bayes dan SVM adalah hasilnya akan lebih baik menggunakan metode klasifikasi Naive Bayes karena dari keempat perbandingan, Naive Bayes mendapatkan tiga kali hasil lebih baik yaitu pada perhitungan akurasi, Precision dan F1_Measure. Pada tugas kali ini dilakukan clustering terlebih dahulu sehingga hasil dari evaluasi klasifikasi adalah lebih baik menggunakan metode klasifikasi Naive Bayes dibandingkan dengan metode klasifikasi SVM. Jika dibandingkan dengan tugas kelompok yang sebelumnya hasilnya adalah tetap sama yaitu metode klasifikasi Naive Bayes lebih baik dari metode klasifikasi SVM.

BAB IV TAUTAN GOOGLE COLAB DAN VIDEO

1. Google Colab
   https://colab.research.google.com/drive/1spFJhPOvT_4ReGxo_7X5ifs13aR-Bs_z?usp=sharing

2. Video Menjalankan Program
   https://drive.google.com/file/d/1YZYNuKjOfELl814rY-vy6YQ81D4KzcS_/view?usp=sharing