LAPORAN TUGAS BESAR

KLUSTERISASI DATA

DISUSUN UNTUK MEMENUHI TUGAS MATA KULIAH
PEMBELAJARAN MESIN

DOSEN PENGAMPU :

DEDE ROHIDIN AMIN

CORNELIUS STEPAHANUS ALFREDO (1301180287)

MEYZO NAUFAL ROMZI (1301184299)

IF – 42 – 04

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

1. FORMULASI MASALAH

Dalam tugas besar kali ini diminta untuk membuat program yang berfungsi untuk memprediksi mengenai apakah pelanggan tertarik untuk membeli sebauh kendaraan baru berdasarkan data pelanggan yang telah diberikan dengan menerapkan metode classification. Di dalam dataset yang telah diberikan terdapat beberapa permasalahan yaitu adanya *missing value* dan pencilan di atribut yang ada pada dataset tersebut yang harus melalui proses *cleaning data* terlebih dahulu sebelum melakukan classification dan menentukan metode classification yang tepat dengan dataset yang telah diberikan.

2. Explorasi Persiapan data

Data yang dipakai dalam melakukan implementasi classification yaitu data train dan data test dari dalam dataset yang telah diberikan memiliki ukuran data sebagai berikut:

Data train:

```
Data Test shape
Rows:  285831
Columns:  11
```

Data test:

```
Data Test shape
Rows:  47639
Columns:  11
```

Tipe data pada masing-masing dataset:

```
#cek apakah masih ada data kategorikal atau tidak
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285831 entries, 0 to 285830
Data columns (total 11 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Jenis_Kelamin      271391 non-null  float64
 1   Umur               271617 non-null  float64
 2   SIM                271427 non-null  float64
 3   Kode_Daerah        271525 non-null  float64
 4   Sudah_Asuransi     271602 non-null  float64
 5   Umur_Kendaraan     271556 non-null  float64
 6   Kendaraan_Rusak    271643 non-null  float64
 7   Premi              271262 non-null  float64
 8   Kanal_Penjualan    271532 non-null  float64
 9   Lama_Berlangganan  271839 non-null  float64
 10  Tertarik           285831 non-null  int64
dtypes: float64(10), int64(1)
memory usage: 24.0 MB
```

```
print("Data Types in Data Test: ")
print(df_test.dtypes)
```

```
Data Types in Data Test:
Jenis_Kelamin        object
Umur                  int64
SIM                   int64
Kode_Daerah           int64
Sudah_Asuransi        int64
Umur_Kendaraan       object
Kendaraan_Rusak      object
Premi                 int64
Kanal_Penjualan       int64
Lama_Berlangganan     int64
Tertarik              int64
dtype: object
```

Penjelasan Kolom Fitur:

- SIM --> 0 : Tidak punya SIM 1 : Punya SIM
- Kode_Daerah --> Kode area tempat tinggal pelanggan
- Sudah_Asuransi --> 1 : Pelanggan sudah memiliki asuransi kendaraan, 0 : Pelanggan belum memiliki asuransi kendaraan
- Umur_Kendaraan --> Umur kendaraan
- Kendaraan_Rusak --> 1 : Kendaraan pernah rusak sebelumnya. 0 : Kendaraan belum pernah rusak.
- Premi --> Jumlah premi yang harus dibayarkan per tahun.
- Kanal_Penjualan --> Kode kanal untuk menghubungi pelanggan (email, telpon, dll)
- Lama_Berlangganan --> Sudah berapa lama pelanggan menjadi klien perusahaan
- Tertarik --> 1 : Pelanggan tertarik, 0 : Pelanggan tidak tertarik

Pengecekan missing value pada masing: dataset:

Data train:

```
Jenis_Kelamin        14440
Umur                 14214
SIM                  14404
Kode_Daerah          14306
Sudah_Asuransi       14229
Umur_Kendaraan       14275
Kendaraan_Rusak      14188
Premi                14569
Kanal_Penjualan      14299
Lama_Berlangganan    13992
Tertarik                 0
dtype: int64
```

Data test:

```
Checking Null Value in Data Test:
Jenis_Kelamin        0
Umur                 0
SIM                  0
Kode_Daerah          0
Sudah_Asuransi       0
Umur_Kendaraan       0
Kendaraan_Rusak      0
Premi                0
Kanal_Penjualan      0
Lama_Berlangganan    0
Tertarik             0
dtype: int64
```

Deskripsi data:

```
display(df_train)
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertar
0	0.0	30.0	1.0	33.0	1.0	0.0	0.0	28029.0	152.0	97.0	
1	1.0	48.0	1.0	39.0	0.0	1.0	1.0	25800.0	29.0	158.0	
2	NaN	21.0	1.0	46.0	1.0	0.0	0.0	32733.0	160.0	119.0	
3	0.0	58.0	1.0	48.0	0.0	0.5	0.0	2630.0	124.0	63.0	
4	1.0	50.0	1.0	35.0	0.0	1.0	NaN	34857.0	88.0	194.0	
...	...	...	...	...	...	...	...	...	...	...	
285826	0.0	23.0	1.0	4.0	1.0	0.0	0.0	25988.0	152.0	217.0	
285827	0.0	21.0	1.0	46.0	1.0	0.0	0.0	44686.0	152.0	50.0	
285828	0.0	23.0	1.0	50.0	1.0	0.0	0.0	49751.0	152.0	226.0	
285829	1.0	68.0	1.0	7.0	1.0	0.5	0.0	30503.0	124.0	270.0	
285830	1.0	45.0	1.0	28.0	0.0	0.5	1.0	36480.0	26.0	44.0	

285831 rows × 11 columns

```
display(df_test)
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	Wanita	49	1	8	0	1-2 Tahun	Pernah	46963	26	145	0
1	Pria	22	1	47	1	< 1 Tahun	Tidak	39624	152	241	0
2	Pria	24	1	28	1	< 1 Tahun	Tidak	110479	152	62	0
3	Pria	46	1	8	1	1-2 Tahun	Tidak	36266	124	34	0
4	Pria	35	1	23	0	1-2 Tahun	Pernah	26963	152	229	0
...	...	...	...	...	...	...	...	...	...	...	...
47634	Pria	61	1	46	0	> 2 Tahun	Pernah	31039	124	67	0
47635	Pria	41	1	15	0	1-2 Tahun	Pernah	2630	157	232	0
47636	Pria	24	1	29	1	< 1 Tahun	Tidak	33101	152	211	0
47637	Pria	59	1	30	0	1-2 Tahun	Pernah	37788	26	239	1
47638	Pria	52	1	31	0	1-2 Tahun	Tidak	2630	124	170	0

47639 rows × 11 columns

Transform data:

Data train:

```
2]: df_train.drop('id',axis=1,inplace=True)
```

```
3]: #untuk mengganti type data dari kategorikal menjadi numerical
df_train['Kendaraan_Rusak'] = df_train['Kendaraan_Rusak'].replace(['Pernah','Tidak'],[1,0])
df_train['Umur_Kendaraan'] = df_train['Umur_Kendaraan'].replace(['< 1 Tahun','1-2 Tahun','> 2 Tahun'],[0,0.5,1])
df_train['Jenis_Kelamin'] = df_train['Jenis_Kelamin'].replace(['Wanita','Pria'],[0,1])
```

Data test:

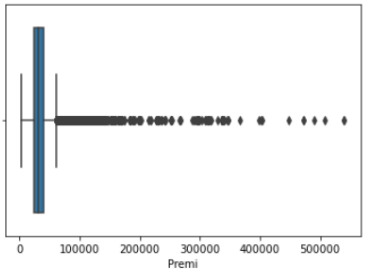
```
df_test['Kendaraan_Rusak'] = df_test['Kendaraan_Rusak'].replace(['Pernah','Tidak'],[1,0])
df_test['Umur_Kendaraan'] = df_test['Umur_Kendaraan'].replace(['< 1 Tahun','1-2 Tahun','> 2 Tahun'],[0,0.5,1])
df_test['Jenis_Kelamin'] = df_test['Jenis_Kelamin'].replace(['Wanita','Pria'],[0,1])
```

Sebelum melakukan preprosesing data ada baik nya kita mencari data outlier:

```
[7]: #cek ada pencilan atau outliers pada data atau tidak
sns.boxplot('Premi', data=df_train)
```

```
c:\python38\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
ersion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywo
sult in an error or misinterpretation.
  warnings.warn(
```

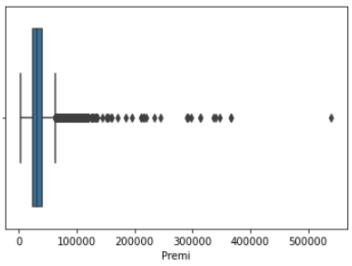
```
t[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1e378803490>
```



```
[8]: sns.boxplot('Premi', data=df_test)

c:\python38\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
ersion 0.12, the only valid positional argument will be `data`, and pas
sult in an error or misinterpretation.
  warnings.warn(

[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1e31a049550>
```



Ketika sudah melakukan pengecekan apakah ada outlier dan missing value pada masing-masing data jika ada maka kita harus mencari missing value tersebut dan menghilangkan outlier:

Data train:

```
#untuk mengassign value yang hilang dengan nilai mean atau modus disesuaikan data nya lebih cocok menggunakan mean atau modus
df_train['Jenis_Kelamin'].fillna(value=df_train['Jenis_Kelamin'].mode()[0],inplace=True)
df_train['Umur'].fillna(value=math.ceil(df_train['Umur'].mean()),inplace=True)
df_train['SIM'].fillna(value=df_train['SIM'].mode()[0],inplace=True)
df_train['Kode_Daerah'].fillna(value=df_train['Kode_Daerah'].mode()[0],inplace=True)
df_train['Sudah_Asuransi'].fillna(value=df_train['Sudah_Asuransi'].mode()[0],inplace=True)
df_train['Umur_Kendaraan'].fillna(value=df_train['Umur_Kendaraan'].mode()[0],inplace=True)
df_train['Kendaraan_Rusak'].fillna(value=df_train['Kendaraan_Rusak'].mode()[0],inplace=True)
df_train['Premi'].fillna(value=math.ceil(df_train['Premi'].mean()),inplace=True)
df_train['Kanal_Penjualan'].fillna(value=df_train['Kanal_Penjualan'].mode()[0],inplace=True)
df_train['Lama_Berlangganan'].fillna(value=math.ceil(df_train['Lama_Berlangganan'].mean()),inplace=True)

#setelah di assign dengan nilai baru cek lagi apakah masih ada handling missing value atau tidak
df_train.isnull().sum()
```

```
Jenis_Kelamin        0
Umur                 0
SIM                  0
Kode_Daerah          0
Sudah_Asuransi       0
Umur_Kendaraan       0
Kendaraan_Rusak      0
Premi                0
Kanal_Penjualan      0
Lama_Berlangganan    0
Tertarik             0
dtype: int64
```

```
: #fungsi untuk mencari pencilan nya berada dimana
  def finding_outlier(df):
      Q1= df.quantile(0.25)
      Q3= df.quantile(0.75)
      IQR= Q3-Q1
      df_final= df[(df<(Q1-(1.5*IQR))) | (df<(Q3+(1.5*IQR)))]
      return df_final

  print(finding_outlier(df_train['Premi']))
```

```
0         28029.0
1         25800.0
2         32733.0
3          2630.0
4         34857.0
           ...
285826    25988.0
285827    44686.0
285828    49751.0
285829    30503.0
285830    36480.0
Name: Premi, Length: 276960, dtype: float64
```

```
#fungsi untuk menghilangkan outliers
def remove_outlier(df):
    Q1= df.quantile(0.25)
    Q3= df.quantile(0.75)
    IQR= Q3-Q1
    df_final= df[((df<(Q1-(1.5*IQR))) | (df<(Q3+(1.5*IQR))))]
    return df_final
```

```
df = remove_outlier(df_train[['Premi']])
df.dropna(axis= 0)
df
```

	Premi
0	28029.0
1	25800.0
2	32733.0
3	2630.0
4	34857.0
...	...
285826	25988.0
285827	44686.0
285828	49751.0
285829	30503.0
285830	36480.0

Data test:

```
df_test['Umur'].fillna(value=math.ceil(df_test['Umur'].mean()),inplace=True)
df_test['Premi'].fillna(value=math.ceil(df_test['Premi'].mean()),inplace=True)
df_test['Lama_Berlangganan'].fillna(value=math.ceil(df_test['Lama_Berlangganan'].mean()),inplace=True)
df_test['Umur_Kendaraan'].fillna(value=df_test['Umur_Kendaraan'].mode()[0],inplace=True)
df_test['Jenis_Kelamin'].fillna(value=df_test['Jenis_Kelamin'].mode()[0],inplace=True)
df_test['SIM'].fillna(value=df_test['SIM'].mode()[0],inplace=True)
df_test['Kode_Daerah'].fillna(value=df_test['Kode_Daerah'].mode()[0],inplace=True)
df_test['Sudah_Asuransi'].fillna(value=df_test['Sudah_Asuransi'].mode()[0],inplace=True)
df_test['Kendaraan_Rusak'].fillna(value=df_test['Kendaraan_Rusak'].mode()[0],inplace=True)
df_test['Kanal_Penjualan'].fillna(value=df_test['Kanal_Penjualan'].mode()[0],inplace=True)
```

```
def finding_outlier(df):
    Q1= df.quantile(0.25)
    Q3= df.quantile(0.75)
    IQR= Q3-Q1
    df_final= df[(df<(Q1-(1.5*IQR))) | (df<(Q3+(1.5*IQR)))]
    return df_final
print(finding_outlier(df_test['Premi']))
```

```
0        46963
1        39624
3        36266
4        26963
5        42721
         ...
47634    31039
47635     2630
47636    33101
47637    37788
47638     2630
Name: Premi, Length: 46368, dtype: int64
```

19]:
```
def remove_outlier(df):
    Q1= df.quantile(0.25)
    Q3= df.quantile(0.75)
    IQR= Q3-Q1
    df_final= df[((df<(Q1-(1.5*IQR))) | (df<(Q3+(1.5*IQR))))]
    return df_final
```

20]:
```
df = remove_outlier(df_train[['Premi']])
df.dropna(axis= 0)
df
```

20]:

	Premi
0	28029.0
1	25800.0
2	32733.0
3	2630.0
4	34857.0
...	...
285826	25988.0
285827	44686.0
285828	49751.0
285829	30503.0
285830	36480.0

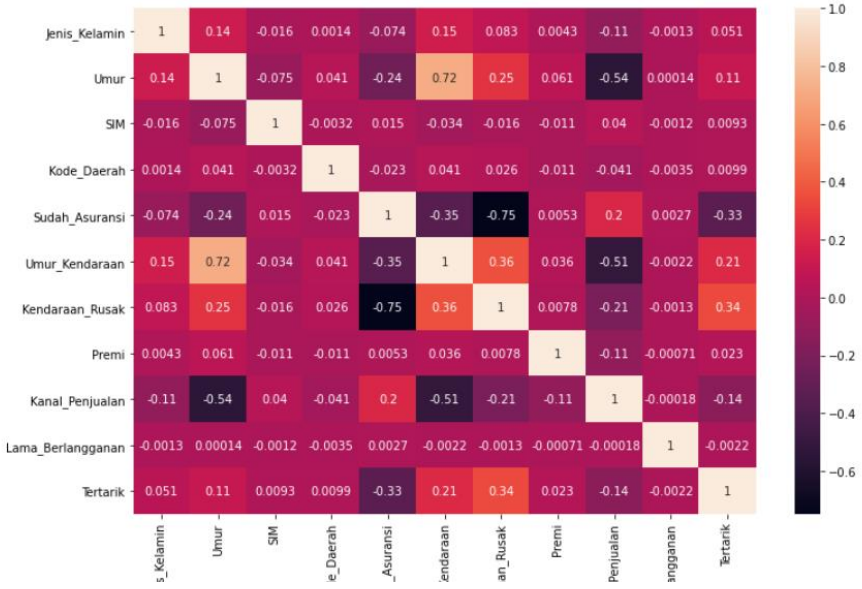
285831 rows × 1 columns

Berikut adalah hasil perhitungan corelasi masing-masing dataset:

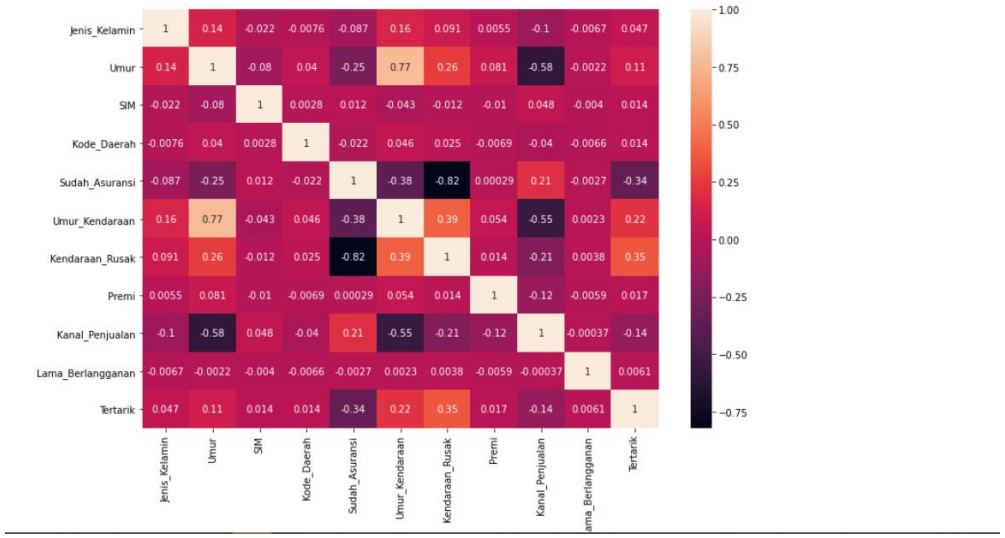
:
```
# untuk mengcek korelasi yang dimiliki antar data
# ket: semakin terang atau mendekati nilai satu maka data tersebut memiliki korelasi yang tinggi

plt.figure(figsize=(12,8))
sns.heatmap(df_train.corr(),annot= True)
```

:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e31a08fbe0>
```



```
]: plt.figure(figsize=(12,8))
   sns.heatmap(df_test.corr(),annot= True)
]: <matplotlib.axes._subplots.AxesSubplot at 0x1e3787d3ee0>
```



3. PEMODELAN

Pemodelan yang digunakan untuk melakukan classification pada tugas ini adalah Gradient Boosting Classifier. Alasan mengapa menggunakan model Gradient Boosting Classifier karena keakuratan dari algoritma ini dan dalam data science, algoritma ini jarang digunakan.

Disini digunakan 2 pemodelan untuk melakukan proses klasifikasi yaitu Gradient Boosting Classifier sebagai pemodelan utama dan Gaussian Naïve bayes sebagai pebanding dari pemodelan sebelum nya.dan nanti akan dievaluasi berdasarkan tingkat akurasi dari masing-masing model.

Naive Bayes merupakan sebuah pengklasifikasian probabilistik sederhana yang menghitung sekumpulan probabilitas dengan menjumlahkan frekuensi dan kombinasi nilai dari dataset yang diberikan. Dalam metode ini, akan diasumsikan semua atribut independen atau tidak saling ketergantungan yang diberikan oleh nilai pada variabel masing-masing kelas. Lalu muncul Gaussian Naive Bayes yang merupakan perkembangan dari metode Naive Bayes. Dalam Gaussian Naive Bayes, distribusi data

yang dipakai merupakan distribusi normal dan data yang diolah adalah data kontinyu. Berikut adalah rumus dari Gaussian Naive Bayes :

$$P = (X_i = x_i \mid Y_i = y_i) = \frac{1}{\sqrt{2\pi\sigma ij}} e^{-\frac{(x_i - u_{ij})2}{2\sigma^2 ij}}$$

Keterangan :

P : Peluang

Xi : Atribut ke i

xi : Nilai atribut ke i

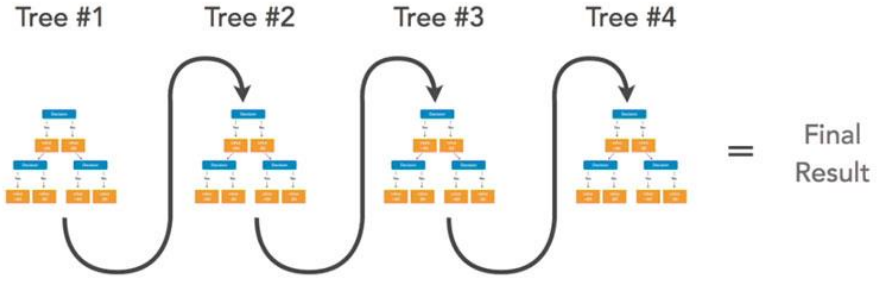
Y : Kelas yang dicari

yj : Sub kelas Y yang dicari

u : Mean, menyatakan rata rata dari seluruh atribut

o : Deviasi standar, menyatakan varian dari seluruh atribut

Gradient boosting adalah algoritma machine learning yang menggunakan ensamble dari decision tree untuk memprediksi nilai.adapun Ensamble learning algorithm adalah algortima yang menggunakan banyak simple machine learning model yang bekerja bersama untuk menghasilkan prediksi yang tepat. Dan struktur data dari gradient boosting adalah decision tree.adapun metode ini menggunakan simple decision tree yang berkesinambungan, setiap tree baru yang dibuat adalah perbaikan dari error tree sebelumnya.dengan tujuan membuat model dari hubungan luas dan harga rumah tersebut dengan menggunakan gradient boosting. Contoh :



4. EVALUASI

Evaluasi metric yang dipakai pada laporan ini yaitu, accuracy, running time, f1-score, recall, precision, dan confusion matrix. Metric tersebut dipilih untuk menentukan performa dari suatu metode. Penjelasan dari confusion matrix sebagai berikut :

		Nilai sebenarnya	
		TRUE	FALSE
Nilai predisksi	TRUE	TP (True Positive) *Corect result*	FP (False Positive) *Unexpected result*
	FALSE	FN (False Negative) *Missing result*	TN (True Negative) *Corect absence of result*

Jadi dari confusion matrix tersebut dapat diperoleh rumus-rumus lainnya seperti: Accuracy merupakan rasio prediksi Benar (positif dan negatif) dengan keseluruhan data. Berikut rumusnya:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Berikut rumusnya:

$$recall = \frac{TP}{TP + FN}$$

Precision merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positf. Berikut rumusnya:

$$precision = \frac{TP}{TP + FP}$$

F-1 score merupakan perbandingan rata-rata presisi dan recall yang dibobotkan. Berikut rumusnya:

F1 Score = 2 * (Recall*Precission) / (Recall + Precission)

5.EKSPERIMEN

Berikut hasil eksperimen dari masing-masing model:

Gradient Boosting Classifier:

```
]: start_time = time.time()

   model = GradientBoostingClassifier()
   model.fit(X_train, y_train)
   y_pred = model.predict(X_test)
   acc = round(model.score(X_test, y_test) * 100, 2)


   gbt_time = (time.time() - start_time)
   print(acc)
   print("Running Time: %s" % datetime.timedelta(seconds=gbt_time))
```

```
87.69
Running Time: 0:00:47.438239
```

```
]: print('Confusion Matrix\n', confusion_matrix(y_test, y_pred))
   print('\n')
   print(classification_report(y_test, y_pred))
```

```
Confusion Matrix
 [[41775     3]
 [ 5861     0]]


              precision    recall  f1-score   support

           0       0.88      1.00      0.93     41778
           1       0.00      0.00      0.00      5861

    accuracy                           0.88     47639
   macro avg       0.44      0.50      0.47     47639
weighted avg       0.77      0.88      0.82     47639
```

Gaussian Naïve Bayes

```
: naiveBayes = GaussianNB()
  naiveBayes.fit(X_latih, y_latih)
```

```
: GaussianNB()
```

```
: hasilPred = naiveBayes.predict(X_tes)
```

```
: print('Confusion Matris\n', confusion_matrix(y_tes, hasilPred))
  print('\n')
  print(classification_report(y_tes, hasilPred))
```

```
Confusion Matris
 [[24855 16923]
 [  153  5708]]


              precision    recall  f1-score   support

           0       0.99      0.59      0.74     41778
           1       0.25      0.97      0.40      5861

    accuracy                           0.64     47639
   macro avg       0.62      0.78      0.57     47639
weighted avg       0.90      0.64      0.70     47639
```

6. KESIMPULAN

Dapat dilihat dari Hasil eksperimen tersebut Gradient Boosting Clasiffier lebih unggul dalam hal accuracy dibandingkan dengan Gaussian Naïve Bayes dengan akurasi 0.88 sedangkan Gaussian Naïve Bayes akurasi nya hanya sebesar 0.64.

Dapat dismpulkan bahwa pada dataset ini lebih baik menggunakan pemodelan Gradient Boosting Classifier dibandingkan Gaussian Naïve Bayes