LAPORAN TUGAS BESAR MATA KULIAH
PEMBELAJARAN MESIN
CLASSIFICATION

Oleh:
Mega Vebika Shyahrin (1301194456)
Muhammad Zalfa Thoriq (130119)

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

2021

1. Formulasi Masalah

Pada tugas besar tahap 2 ini kami diminta untuk memprediksi apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak berdasarkan data pelanggan di dealer dan memprediksi ketepatan akurasi tersebut menggunakan supervised learning menggunakan teknik classification dengan Metode pemodelan yang kami gunakan adalah Algoritma K-Nearest Neighbors (KNN).

2. Eksplorasi dan Persiapan Data

2.1 Upload Dataset

Dataset yang kami upload adalah sebanyak 2 yaitu kendaraan_train dan kendaraan_test yang kemudian disimpan dalam variabel data_1 dan data_2.

```
#read data kendaraan test
data_2 = pd.read_excel('kendaraan_test.xlsx')
data_2.head()
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	Wanita	22	1	52	0	1-2 Tahun	Pernah	32895	124	71	1
1	Pria	54	1	52	0	1-2 Tahun	Pernah	43388	124	198	0
2	Wanita	24	1	52	0	1-2 Tahun	Pernah	45032	124	171	0
3	Wanita	78	1	52	0	> 2 Tahun	Pernah	42825	26	208	1
4	Wanita	45	1	52	0	1-2 Tahun	Pernah	2630	26	228	0

2.2 Cek Data Null

Untuk mengecek missing value pada dataset yang telah diupload.

```
# read data kendaraan train
data_1 = pd.read_excel('kendaraan_train.xlsx')
data_1.head()
```

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	1.0	Wanita	30.0	1.0	33.0	1	< 1 Tahun	Tidak	28029.0	152.0	97.0	0.0
1	2.0	Pria	48.0	1.0	39.0	0	> 2 Tahun	Pernah	25800.0	29.0	158.0	0.0
2	3.0	NaN	21.0	1.0	46.0	1	< 1 Tahun	Tidak	32733.0	160.0	119.0	0.0
3	4.0	Wanita	58.0	1.0	48.0	0	1-2 Tahun	Tidak	2630.0	124.0	63.0	0.0
4	5.0	Pria	50.0	1.0	35.0	0	> 2 Tahun	NaN	34857.0	88.0	194.0	0.0

```
# melihat type data kendaraan train
data_1.isnull().sum()
```

```
id                      5
Jenis_Kelamin       14445
Umur                14219
SIM                 14409
Kode_Daerah         14311
Sudah_Asuransi      14233
Umur_Kendaraan      14280
Kendaraan_Rusak     14193
Premi               14574
Kanal_Penjualan     14304
Lama_Berlangganan   13997
Tertarik                5
dtype: int64
```

```
# melihat type data kendaraan test
data_2.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
```

Dapat dilihat kendaraan train memiliki banyak missing value sedangkan kendaraan test tidak ada missing value.

2.3 Memperlihatkan Info Setiap Variabel

Untuk menampilkan detail dari dataset seperti jumlah kolom, nama kolom, jumlah data, tipe data dan lain-lain.

```
# melihat type data kendaraan train
data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285836 entries, 0 to 285835
Data columns (total 12 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   id                 285831 non-null  float64
 1   Jenis_Kelamin      271391 non-null  object 
 2   Umur               271617 non-null  float64
 3   SIM                271427 non-null  float64
 4   Kode_Daerah        271525 non-null  float64
 5   Sudah_Asuransi     271603 non-null  object 
 6   Umur_Kendaraan     271556 non-null  object 
 7   Kendaraan_Rusak    271643 non-null  object 
 8   Premi              271262 non-null  float64
 9   Kanal_Penjualan    271532 non-null  float64
 10  Lama_Berlangganan  271839 non-null  float64
 11  Tertarik           285831 non-null  float64
dtypes: float64(8), object(4)
memory usage: 26.2+ MB
```

```
# melihat type data kendaraan test
data_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47639 entries, 0 to 47638
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Jenis_Kelamin      47639 non-null  object
 1   Umur               47639 non-null  int64 
 2   SIM                47639 non-null  int64 
 3   Kode_Daerah        47639 non-null  int64 
 4   Sudah_Asuransi     47639 non-null  int64 
 5   Umur_Kendaraan     47639 non-null  object
 6   Kendaraan_Rusak    47639 non-null  object
 7   Premi              47639 non-null  int64 
 8   Kanal_Penjualan    47639 non-null  int64 
 9   Lama_Berlangganan  47639 non-null  int64 
 10  Tertarik           47639 non-null  int64 
dtypes: int64(8), object(3)
memory usage: 4.0+ MB
```

2.4 Menghapus Data yang Tidak Digunakan

Data yang tidak lagi digunakan akan dihapus. Dalam tugas besar ini data yang tidak lagi digunakan adalah data id.

```
#drop kolom yang tidak digunakan (hanya id yg didrop karena agar manyamakan data test)
data_1 = data_1.drop(columns=['id'])
data_1.head()
```

	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	Wanita	30.0	1.0	33.0	1	< 1 Tahun	Tidak	28029.0	152.0	97.0	0.0
1	Pria	48.0	1.0	39.0	0	> 2 Tahun	Pernah	25800.0	29.0	158.0	0.0
2	NaN	21.0	1.0	46.0	1	< 1 Tahun	Tidak	32733.0	160.0	119.0	0.0
3	Wanita	58.0	1.0	48.0	0	1-2 Tahun	Tidak	2630.0	124.0	63.0	0.0
4	Pria	50.0	1.0	35.0	0	> 2 Tahun	NaN	34857.0	88.0	194.0	0.0

2.5 Mengubah Data Kategori Menjadi Numeric

Pada proses ini data kategori yang diubah menjadi numeric adalah data jenis kelamin, data kendaraan rusak, dan data umur kendaraan.

```
# mengubah data jenis kelamin menjadi angka
dummies = pd.get_dummies(data_1['Jenis_Kelamin'], prefix='kelamin')
data_1 = pd.concat([data_1, dummies], axis=1)
data_1 = data_1.drop(['Jenis_Kelamin'], axis = 1)

dummies = pd.get_dummies(data_2['Jenis_Kelamin'], prefix='kelamin')
data_2 = pd.concat([data_2, dummies], axis=1)
data_2 = data_2.drop(['Jenis_Kelamin'], axis = 1)

# mengubah data kendaraan rusak menjadi angka
dummies = pd.get_dummies(data_1['Kendaraan_Rusak'], prefix='kendaraan_rusak')
data_1 = pd.concat([data_1, dummies], axis=1)
data_1 = data_1.drop(['Kendaraan_Rusak'], axis = 1)

dummies = pd.get_dummies(data_2['Kendaraan_Rusak'], prefix='kendaraan_rusak')
data_2 = pd.concat([data_2, dummies], axis=1)
data_2 = data_2.drop(['Kendaraan_Rusak'], axis = 1)

# mengubah data umur kendaraan menjadi angka
dummies = pd.get_dummies(data_1['Umur_Kendaraan'], prefix='Umur_kendaraan')
data_1 = pd.concat([data_1, dummies], axis=1)
data_1 = data_1.drop(['Umur_Kendaraan'], axis = 1)

dummies = pd.get_dummies(data_2['Umur_Kendaraan'], prefix='Umur_kendaraan')
data_2 = pd.concat([data_2, dummies], axis=1)
data_2 = data_2.drop(['Umur_Kendaraan'], axis = 1)
```

```
[49] data_1.head()
```

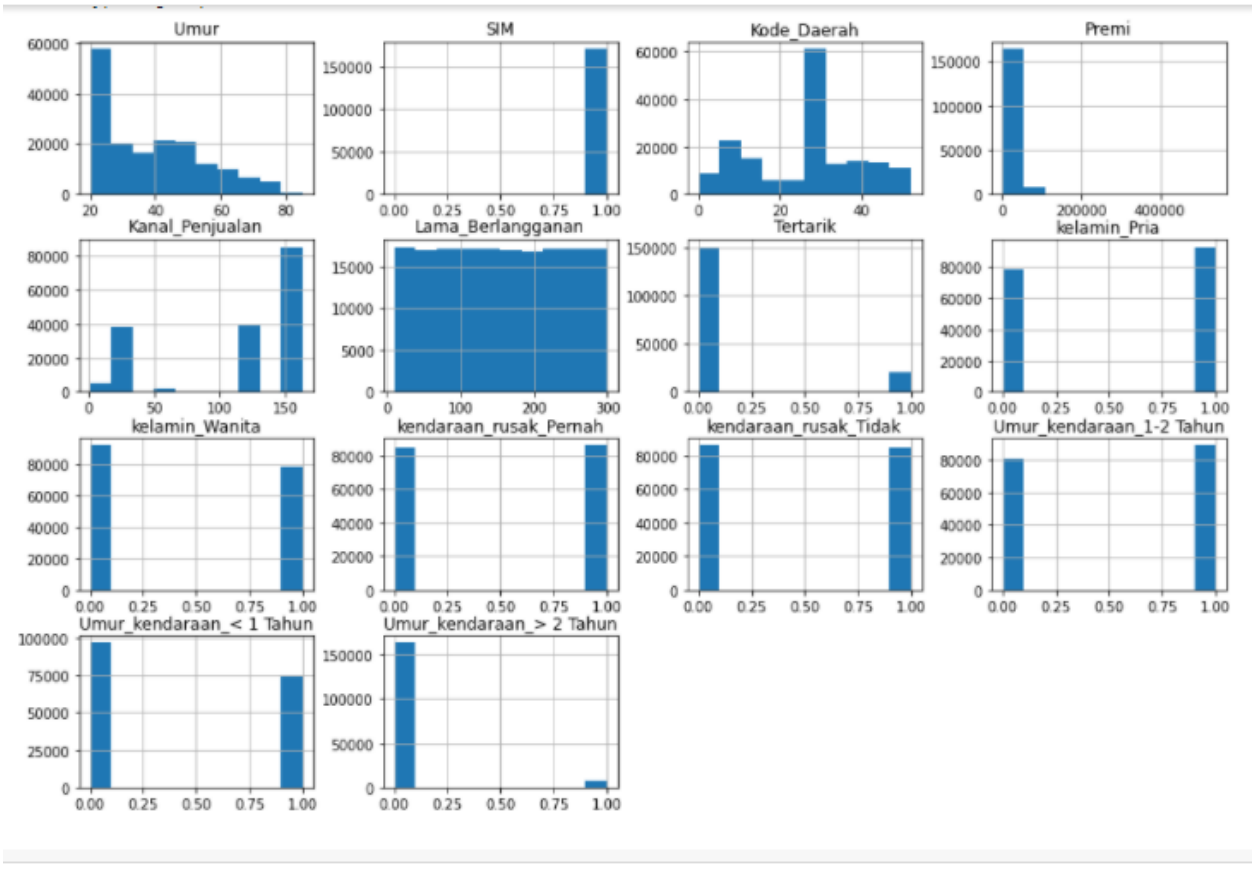
	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik	kelamin_Pria	kelamin_Wanita	kendaraan_rusak_Pernah	kendaraan_rusak_Tidak	Umur_kendaraan_1-2 Tahun	Umur_kendaraan_< 1 Tahun	Umur_kendaraan_> 2 Tahun
0	30.0	1.0	33.0	1	28029.0	152.0	97.0	0.0	0	1	0	1	0	1	0
1	48.0	1.0	39.0	0	25800.0	29.0	158.0	0.0	1	0	1	0	0	0	1
3	58.0	1.0	48.0	0	2630.0	124.0	63.0	0.0	0	1	0	1	1	0	0
5	21.0	1.0	35.0	1	22735.0	152.0	171.0	0.0	1	0	0	1	0	1	0
8	20.0	1.0	8.0	1	30786.0	160.0	31.0	0.0	0	1	0	1	0	1	0

```
[50] data_2.head()
```

	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik	kelamin_Pria	kelamin_Wanita	kendaraan_rusak_Pernah	kendaraan_rusak_Tidak	Umur_kendaraan_1-2 Tahun	Umur_kendaraan_< 1 Tahun	Umur_kendaraan_> 2 Tahun
0	22	1	52	0	32895	124	71	1	0	1	1	0	1	0	0
1	54	1	52	0	43388	124	198	0	1	0	1	0	1	0	0
2	24	1	52	0	45032	124	171	0	0	1	1	0	1	0	0
3	78	1	52	0	42825	26	208	1	0	1	1	0	0	0	1
4	45	1	52	0	2630	26	228	0	0	1	1	0	1	0	0

2.6 Histogram

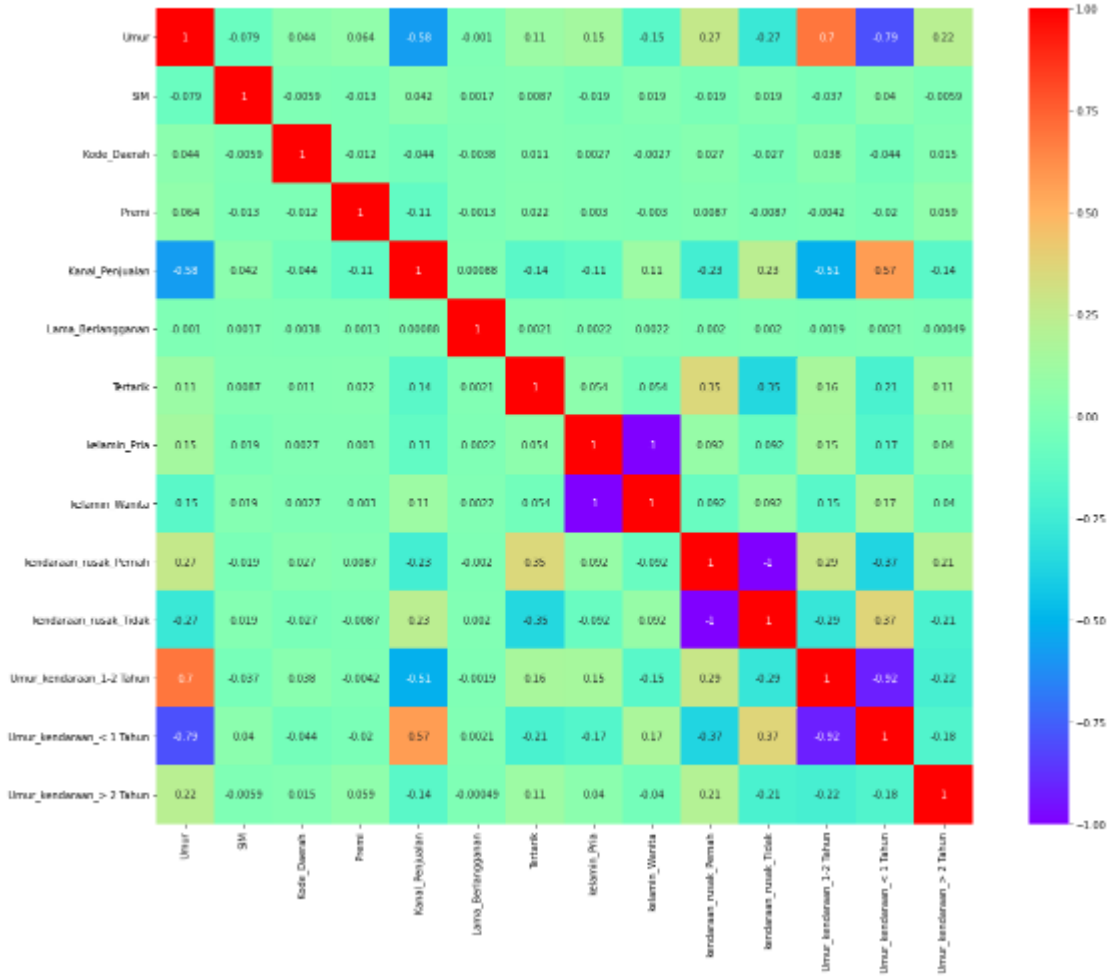
Dengan menggunakan diagram histogram dapat dilihat jumlah sesuatu data misalnya pada kolom tertarik berjumlah dibawah 50.000 sedangkan yang tidak tertarik berjumlah 150.000.



2.7 Korelasi

Dengan menggunakan tabel korelasi kami dapat menentukan korelasi suatu data dengan contohnya pada kolom umur memiliki rentang 0,001 sampai 1 dan semakin angka mendekati 1 maka korelasi tersebut semakin bagus.

```
sns.heatmap(data_1.corr(), annot=True, ax=ax, square=True, cmap='rainbow', vmin=-1,vmax=1, annot_kws={'size': 10})
plt.show()
```



2.8 Seleksi Data

Pada tahap ini terjadi proses pemisahan antara yang 'tertarik' dengan lainnya. Variabel 't1' digunakan untuk menyimpan selain 'tertarik' sedangkan variabel 'tertarik_1' digunakan untuk menyimpan data yang 'tertarik'.

```
# memisahkan Data tertarik pada kendaraan train
t1 = data_1.drop(["Tertarik"], axis = 1)
t1.head()
```

	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Premi	Kanal_Penjualan	Lama_Berlangganan	kelamin_Pria	kelamin_Wanita	kendaraan_rusak_Pernah	kendaraan_rusak_Tidak	Umur_kendaraan_1-2 Tahun	Umur_kendaraan_< 1 Tahun	Umur_k
0	30.0	1.0	33.0	1	28029.0	152.0	97.0	0	1	0	1	0	1	
1	48.0	1.0	39.0	0	25800.0	29.0	158.0	1	0	1	0	0	0	
3	58.0	1.0	48.0	0	2630.0	124.0	63.0	0	1	0	1	1	0	
5	21.0	1.0	35.0	1	22735.0	152.0	171.0	1	0	0	1	0	1	
8	20.0	1.0	8.0	1	30786.0	160.0	31.0	0	1	0	1	0	1	

```
# memisahkan Data tertarik pada kendaraan test
t2 = data_2.drop(["Tertarik"], axis = 1)
t2.head()
```

	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Premi	Kanal_Penjualan	Lama_Berlangganan	kelamin_Pria	kelamin_Wanita	kendaraan_rusak_Pernah	kendaraan_rusak_Tidak	Umur_kendaraan_1-2 Tahun	Umur_kendaraan_<1 Tahun	Umur_ker
0	22	1	52	0	32895	124	71	0	1	1	0	1	0	
1	54	1	52	0	43388	124	198	1	0	1	0	1	0	
2	24	1	52	0	45032	124	171	0	1	1	0	1	0	
3	78	1	52	0	42825	26	208	0	1	1	0	0	0	
4	45	1	52	0	2630	26	228	0	1	1	0	1	0	

```
# menampilkan hanya data tertarik pada kendaraan train
tertarik_1 = data_1["Tertarik"]
tertarik_1.head()
```

```
0    0.0
1    0.0
3    0.0
5    0.0
8    0.0
Name: Tertarik, dtype: float64
```

```
# menampilkan hanya data tertarik pada kendaraan train
tertarik_2 = data_2["Tertarik"]
tertarik_2.head()
```

```
0    1
1    0
2    0
3    1
4    0
Name: Tertarik, dtype: int64
```

3. Pemodelan Data

Metode pemodelan yang kami gunakan adalah Algoritma K-Nearest Neighbors (KNN). Algoritma K-Nearest Neighbor (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek yang berdasarkan dari data pembelajaran yang jaraknya paling dekat dengan objek tersebut. KNN merupakan algoritma supervised learning dimana hasil dari query instance yang baru diklasifikan berdasarkan mayoritas dari kategori pada algoritma KNN. Dimana kelas yang paling banyak muncul yang nantinya akan menjadi kelas hasil dari klasifikasi.

Berikut ini adalah cara kerja algoritma KNN:

1. Tentukan jumlah tetangga (K) yang akan digunakan untuk pertimbangan penentuan kelas.
2. Hitung jarak dari data baru ke masing-masing data point di dataset.
3. Ambil sejumlah K data dengan jarak terdekat, kemudian tentukan kelas dari data baru tersebut.

Pengaktifan fungsi klasifikasi menggunakan KNN :

```
[ ]  # KNN
     knn = KNeighborsClassifier (n_neighbors = 4)
```

```
[ ]  # memasukkan data train pada fungsi classification untuk KNN
     knn.fit(t1, tertarik_1)

     KNeighborsClassifier(n_neighbors=4)
```

Penggunaan algoritma KNN sendiri memiliki kelebihan dan kekurangan masing-masing, berikut ini adalah kelebihan dan kekurangannya :

Kelebihan :
1. Algoritma K-NN kuat dalam mentraining data yang noisy,
2. Algoritma K-NN sangat efektif jika datanya besar,
3. Mudah diimplementasikan.

Kekurangan :
1. Algoritma K-NN perlu menentukan nilai parameter K,
2. Sensitif pada data pencilan,
3. Rentan pada variabel yang non-informatif.

4. Evaluasi

Pada tahap evaluasi kami melakukan percobaan sebanyak 4 kali dengan range nilai K dari 1 sampai 5.

```
knn = []
for k in range(1,5):
    knn_Class = KNeighborsClassifier(n_neighbors = k)
    knn_Class.fit(t1, tertarik_1)
    knn.append(knn_Class.score(t2, tertarik_2))
```

```
plt.plot([k for k in range(1, 5)], knn, color = 'blue')
for i in range(1,5):
    plt.text(i, knn[i-1], (i, knn[i-1]))
plt.xticks([i for i in range(1, 5)])
plt.xlabel('Jumlah K')
plt.ylabel('Score')
plt.title('Score K Classifier untuk nilai K yang berbeda')
```

```
Text(0.5, 1.0, 'Score K Classifier untuk nilai K yang berbeda')
```

Bisa dilihat dari gambar tersebut bahwa k yang paling optimal adalah 4 karena memiliki score 0,868 dimana jumlah tersebut adalah yang tertinggi.

5. Eksperimen

Pada tahap eksperimen ini kami melakukannya dengan metode KNN(K-Nearest Neighbors. Data yang digunakan dalam eksperimen sama dengan data yang digunakan pada pemodelan. Hasil dari eksperimen menunjukkan keakuratan sebesar 0,87.

PREDIKSI

```
# prediksi
prediksi = knn.predict (t2)
prediksi
```

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

Menentukan hasil Prediksi dari data train.

PROBABILITAS

```
# probabilitas
knn.predict_proba(t2)
```

```
array([[1.  , 0.  ],
       [0.5 , 0.5 ],
       [1.  , 0.  ],
       ...,
       [1.  , 0.  ],
       [1.  , 0.  ],
       [0.75, 0.25]])
```

Menentukan hasil Probabilitas dari Prediksi.

MATRIX

```
[90] # menampilkan matrix hasil prediksi (atas benar, bawah salah, kiri true, kanan false)
     print(confusion_matrix(tertarik_2, prediksi))
```

```
[[41177   601]
 [ 5691   170]]
```

Nilai dari Matrix :

True Positive (TP) : 41177
True Negative (TN): 5691
False Positive (FP): 601
False Negative (FN): 170

```
[91] # finalisasi hasil prediksi
     print(classification_report(tertarik_2, prediksi))
```

```
                  precision    recall  f1-score   support

               0       0.88      0.99      0.93     41778
               1       0.22      0.03      0.05      5861

        accuracy                           0.87     47639
       macro avg       0.55      0.51      0.49     47639
    weighted avg       0.80      0.87      0.82     47639
```

Akurasi hasil prediksi adalah 0.87 atau 87%.

6. Kesimpulan

Setelah melakukan pencarian hasil prediksi terhadap dataset yang telah diupload menggunakan metode algoritma K-Nearest Neighbor (KNN) menghasilkan K = 4 adalah paling optimal. Atau sesuai dengan hasil dari matrix yang kami cari yaitu : True positive sebesar 41177, true negative sebesar 5691, false positive sebesar 601, dan false negative sebesar 170. Kemudian kami juga mendapatkan hasil dari prediksi pelanggan untuk membeli kendaraan baru sebesar 87% dari keseluruhan dataset.