NAMA ANGGOTA KELOMPOK :

1. Miftah Shidqi Rabbani (1301184371)
2. Raafli Mulki Fatiha Sihite (1301184384)

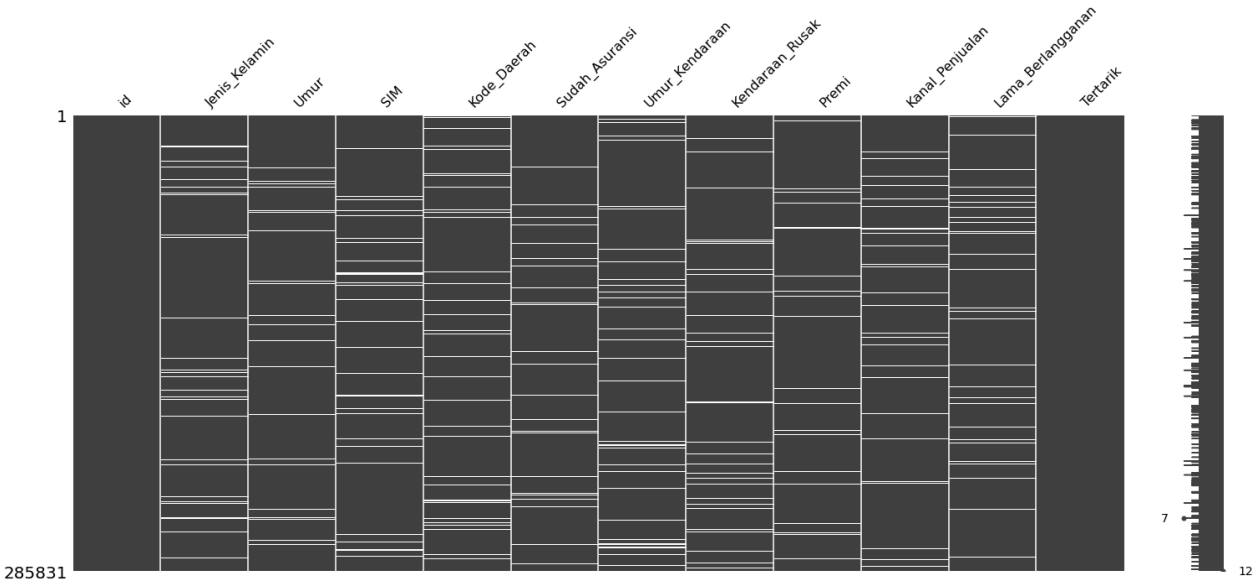
Link Video Youtube :https://youtu.be/GWtXkGek6H8

Formulasi Masalah

Memprediksi ketertarikan pelanggan dalam membeli kendaraan baru atau tidak berdasarkan data dari dealer

Eksplorasi dan Persiapan Data

Karena terdapat missing value seperti gambar dibawah ini.



Sehingga dilakukan input missing value.

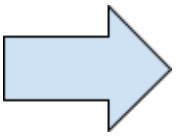
1. Input Missing Value

```
#input missing value
data["Jenis_Kelamin"] = data["Jenis_Kelamin"].replace(np.NaN, np.random.choice(a=["Pria", "Wanita"],p=[0.54, 0.46]))
data["Umur"] = data["Umur"].replace(np.NaN, data["Umur"].mean())
data["SIM"] = data["SIM"].replace(np.NaN, np.random.choice(a=[1, 0],p=[0.9956, 0.0044]))
data["Kode_Daerah"] = data["Kode_Daerah"].replace(np.NaN, statistics.mode(data["Kode_Daerah"]))
data["Sudah_Asuransi"] = data["Sudah_Asuransi"].replace(np.NaN, np.random.choice(a=[0, 1],p=[0.5417, 0.4583]))
data["Umur_Kendaraan"] = data["Umur_Kendaraan"].replace(np.NaN, np.random.choice(a=["< 1 Tahun", "1-2 Tahun", "> 2 Tahun"],p=[0.431, 0.527, 0.042]))
data["Kendaraan_Rusak"] = data["Kendaraan_Rusak"].replace(np.NaN, np.random.choice(a=["Pernah", "Tidak"],p=[0.5355, 0.4645]))
data["Premi"] = data["Premi"].replace(np.NaN, data["Premi"].mean())
data["Kanal_Penjualan"] = data["Kanal_Penjualan"].replace(np.NaN, 152)
data["Lama_Berlangganan"] = data["Lama_Berlangganan"].replace(np.NaN, data["Lama_Berlangganan"].mean())
```

Untuk tiap kolom dilakukan input value dengan cara yang berbeda. Saya lakukan hal tersebut berdasarkan percobaan yang saya lakukan sebelumnya terhadap dataset tersebut. Misal untuk kolom “Jenis_Kelamin” dilakukan pmeilihan acak antara “Pria” dan “Wanita” dengan probabilitas masing-masing 54% dan 46%. Probabilitas tersebut saya dapatkan dari banyaknya kemunculan Pria maupun Wanita pada dataset ”Kendaraan_Train”. Untuk kolom selanjutnya saya lakukan hal serupa.

2. Konversi tipe data

```
id                    int64
Jenis_Kelamin        object
Umur                float64
SIM                 float64
Kode_Daerah         float64
Sudah_Asuransi      float64
Umur_Kendaraan       object
Kendaraan_Rusak      object
Premi               float64
Kanal_Penjualan     float64
Lama_Berlangganan   float64
Tertarik              int64
dtype: object
```



```
id                    int64
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
```

Untuk tiap kolom dengan tipe data Float diubah menjadi Integer, Untuk memudahkan dilakukannya klasifikasi nantinya.

3. Scaling

```
#Scale Kendaraan Rusak
scale_Kendaraan_Rusak = {"Tidak":0,"Pernah":1}
data["Kendaraan_Rusak"] = data["Kendaraan_Rusak"].replace(scale_Kendaraan_Rusak)
data_test["Kendaraan_Rusak"] = data_test["Kendaraan_Rusak"].replace(scale_Kendaraan_Rusak)

#Scale Jenis Kelamin
scale_Jenis_Kelamin = {"Wanita":0,"Pria":1}
data["Jenis_Kelamin"] = data["Jenis_Kelamin"].replace(scale_Jenis_Kelamin)
data_test["Jenis_Kelamin"] = data_test["Jenis_Kelamin"].replace(scale_Jenis_Kelamin)

#Scale Umur Kendaraan
umurKD = pd.get_dummies(data["Umur_Kendaraan"], drop_first=True)
data = pd.concat([data,umurKD], axis = 1)
data.drop(["Umur_Kendaraan","id","Kode_Daerah","Kanal_Penjualan"], axis=1,inplace=True)

umurKD = pd.get_dummies(data_test["Umur_Kendaraan"], drop_first=True)
data_test = pd.concat([data_test,umurKD], axis = 1)
data_test.drop(["Umur_Kendaraan","Kode_Daerah","Kanal_Penjualan"], axis=1,inplace=True)
```

Untuk kolom Kendaraan_Rusak dan Jenis_Kelamin dilakukan scaling, Sehingga yang awal valuenya berupa object diubah menjadi integer. Dan untuk kolom Umur_Kendaraan, diambil valuenya lalu untuk kategori “< 1 Tahun” dan “> 2 Tahun” dijadikan kolom yang baru lalu kolom “Umur_Kendaraan” di drop sehingga terjadi perubahan seperti ini:

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
0	1	Wanita	30	1	33	1	< 1 Tahun	Tidak	28029	152	97	0
1	2	Pria	48	1	39	0	> 2 Tahun	Pernah	25800	29	158	0
2	3	Wanita	21	1	46	1	< 1 Tahun	Tidak	32733	160	119	0
3	4	Wanita	58	1	48	0	1-2 Tahun	Tidak	2630	124	63	0
4	5	Pria	50	1	35	0	> 2 Tahun	Pernah	34857	88	194	0

	Jenis_Kelamin	Umur	SIM	Sudah_Asuransi	Kendaraan_Rusak	Premi	Lama_Berlangganan	Tertarik	< 1 Tahun	> 2 Tahun
0	0	30	1	1	0	28029	97	0	1	0
1	1	48	1	0	1	25800	158	0	0	1
2	0	21	1	1	0	32733	119	0	1	0
3	0	58	1	0	0	2630	63	0	0	0
4	1	50	1	0	1	34857	194	0	0	1

4. Feature Selection

Dilakukan Feature Selection yang berbeda antar 2 klasifikasi:

A. Logistic Regression

```
#Feature Selection
X1 = data.iloc[:,[0,1,2,3,4,5,6,8,9]].values #DATASET Kendaraan_train
y1 = data.iloc[:,7].values

X2 = data_test.iloc[:,[0,1,2,3,4,5,6,8,9]].values #DATASET Kendaraan_test
y2 = data_test.iloc[:,7].values
```

Semua kolom kecuali kolom “Tertarik” dijadikan fitur untuk dilakukan pemodelan menggunakan metode Logistic Regression. Dan untuk kolom “Tertarik” dengan urutan kolom ke-7 dijadikan class untuk dilakukan prediksi dari fitur yang dipilih. Lakukan hal tersebut untuk data train dan data test

B. KNN

```
#Feature Selection
X1 = data.iloc[:,[1,5,6]].values  #DATASET Kendaraan_train
y1 = data.iloc[:,7].values

X2 = data_test.iloc[:,[1,5,6]].values #DATASET Kendaraan_test
y2 = data_test.iloc[:,7].values
```

Pada metode KNN, hanya 3 kolom yang dijadikan fitur, yaitu “Umur”, “Premi”, dan “Lama_Berlangganan” karena untuk mempermudah dilakukan pengukuran jarak nantinya. Dan kolom “Tertarik” dijadikan class untuk dilakukan prediksi dari fitur yang dipilih. Lakukan hal tersebut untuk data train dan data test.

5. Data Scaling

```
#Data Scaling
from sklearn import preprocessing

min_max_scaler = preprocessing.MinMaxScaler()
X1 = min_max_scaler.fit_transform(X1)
X2 = min_max_scaler.fit_transform(X2)
```

Antara metode Logistic Regression dan KNN dilakukan scaling yang sama, yaitu untuk merubah semua value menjadi di range 0 - 1.

6. Data Splitting

```
#Data Splitting
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3)
```

Antara metode Logistic Regression dan KNN dilakukan Data Splitting yang sama berdasarkan dari dataset Kendaraan_Train. Dari dataset tersebut dipecah menjadi 2, yaitu train dan test. Untuk train sebesar 70% dan test sebesar 30%.

Pemodelan Dan Evaluasi

Dalam program ini, kami menggunakan dua jenis Classification, yaitu Logistic Regression dan K-Nearest Neighbors. Kami menggunakan Classification ini karena mudah untuk di implementasikan.

Kami kemudian melakukan training terhadap logmodel menggunakan dataset Kendaraan train yang telah didapatkan dari data splitting.

1. Pemodelan pada Classification Logistic Regression

Pemodelan Logistic menggunakan dataset Kendaraan_train yang telah di splitting.

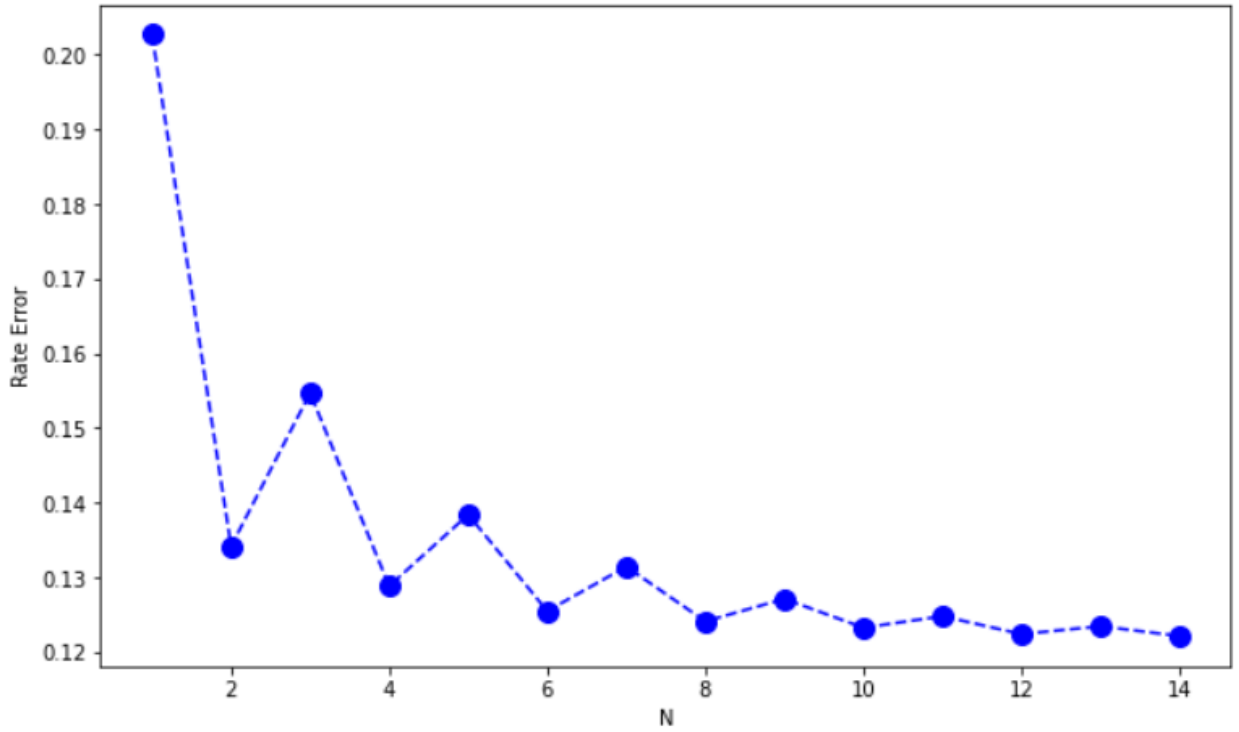
```
#Data Classification Logistic Regression Kendaraan_train
from sklearn.linear_model import LogisticRegression

logmodel = LogisticRegression(class_weight = 'auto')
logmodel.fit(X_train,y_train)
```

2. Pemodelan pada Classification K-Nearest Neighbors

Untuk melakukan pemodelan yang tepat, pertama kami mencari nilai K terbaik dari range 1 - 15, dilakukan looping berulang-ulang dengan data error rate tiap K disimpan dalam list.

```
#Mencari N terbaik
error_rate = []
for i in range(1,15):
 knn = KNeighborsClassifier(n_neighbors=i)
 knn.fit(X_train,y_train)
 pred_i = knn.predict(X_test)
 error_rate.append(np.mean(pred_i != y_test))
```

Berikut adalah Visualisasi data dari error rate tiap K:



Didapatkan bahwa K terbaik adalah 14

```
print("Nilai N terbaik : ",error_rate.index(min(error_rate))+1)
```

```
Nilai N terbaik :  14
```

Eksperimen

1. Logistic Regression

Setelah Pemodelan selesai dibuat, selanjutnya kami melakukan Eksperimen terhadap model menggunakan dataset Kendaraan train yang telah di splitting, dan juga melakukan uji coba menggunakan dataset Kendaraan test

```
predictions = logmodel.predict(X_test)
predictions2 = logmodel.predict(X2)
```

Berikut adalah hasil uji coba yang didapatkan

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94     75285
           1       0.40      0.00      0.00     10465

    accuracy                           0.88     85750
   macro avg       0.64      0.50      0.47     85750
weighted avg       0.82      0.88      0.82     85750

Akurasi Menggunakan dataset Kendaraan_train   : 0.8779475218658892

Akurasi Menggunakan dataset Kendaraan_test    : 0.8769495581351414
Confusion Matrix :
[[41775     3]
 [ 5859     2]]
```

2. K - Nearest Neighbors

Berdasarkan data nilai K terbaik yang kami dapatkan di tahap sebelumnya, kami membuat model KNN berdasarkan data tersebut menggunakan dataset Kendaraan train yang telah di splitting, lalu dilakukan uji coba terhadap data Kendaraan train yang telah di splitting, dan data Kendaraan test.

```
#Define the model
logmodel = KNeighborsClassifier(n_neighbors=14, p=2, metric='euclidean')

#FIT MODEL
logmodel.fit(X_train,y_train)

#PREDICTION
predictions = logmodel.predict(X_test)
predictions2 = logmodel.predict(X2)
```

Berikut adalah hasil uji coba yang didapatkan

```
Akurasi Menggunakan dataset Kendaraan_train : 0.8778425655976676

Akurasi Menggunakan dataset Kendaraan_test  : 0.8762778395852138
Confusion Matrix :
[[41717    61]
 [ 5833    28]]
```

Kesimpulan

Pada tugas kali ini adalah melakukan Classification pada dataset yang diberikan untuk memberi gambaran apakah orang tersebut memiliki minat untuk membeli kendaraan yang baru atau tidak.

Kedua jenis Classification menghasilkan akurasi yang sama terhadap dataset yang sama, sehingga kinerja dari kedua Classification sama.