LAPORAN
TUGAS BESAR PEMBELAJARAN MESIN

Disusun oleh :

Rakhmat Rifaldy (1301180407)
IF-42-04

Jurusan S1 Informatika
Fakultas Informatika
Telkom University

A. Formulasi Masalah

Dari dataset yang diberikan akan dicoba mencari analisisnya untuk menentukan apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak berdasarkan data pelanggan di dealer.

B. Eksplorasi dan Persiapan Data

Import Library

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import random as rd
import copy
import math
```

Read Data

```
df_train = pd.read_csv('kendaraan_train.csv')
df_train
```

	id	Jenis_Kelamin	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Umur_Kendaraan	Kendaraan_Rusak	Premi	Kanal_Penjualan	Lama_Berlanggana
0	1	Wanita	30.0	1.0	33.0	1.0	< 1 Tahun	Tidak	28029.0	152.0	97.
1	2	Pria	48.0	1.0	39.0	0.0	> 2 Tahun	Pernah	25800.0	29.0	158.
2	3	NaN	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	32733.0	160.0	119.
3	4	Wanita	58.0	1.0	48.0	0.0	1-2 Tahun	Tidak	2630.0	124.0	63.
4	5	Pria	50.0	1.0	35.0	0.0	> 2 Tahun	NaN	34857.0	88.0	194.
...	...	...	...	...	...	...	...	...	...	...	...
285826	285827	Wanita	23.0	1.0	4.0	1.0	< 1 Tahun	Tidak	25988.0	152.0	217.
285827	285828	Wanita	21.0	1.0	46.0	1.0	< 1 Tahun	Tidak	44686.0	152.0	50.
285828	285829	Wanita	23.0	1.0	50.0	1.0	< 1 Tahun	Tidak	49751.0	152.0	226.
285829	285830	Pria	68.0	1.0	7.0	1.0	1-2 Tahun	Tidak	30503.0	124.0	270.
285830	285831	Pria	45.0	1.0	28.0	0.0	1-2 Tahun	Pernah	36480.0	26.0	44.

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

Describe Data

```
df_train.describe()
```

	id	Umur	SIM	Kode_Daerah	Sudah_Asuransi	Premi	Kanal_Penjualan	Lama_Berlangganan	Tertarik
count	285831.000000	271617.000000	271427.000000	271525.000000	271602.000000	271262.000000	271532.000000	271839.000000	285831.000000
mean	142916.000000	38.844336	0.997848	26.405410	0.458778	30536.683472	112.021567	154.286302	0.122471
std	82512.446734	15.522487	0.046335	13.252714	0.498299	17155.000770	54.202457	83.694910	0.327830
min	1.000000	20.000000	0.000000	0.000000	0.000000	2630.000000	1.000000	10.000000	0.000000
25%	71458.500000	25.000000	1.000000	15.000000	0.000000	24398.000000	29.000000	82.000000	0.000000
50%	142916.000000	36.000000	1.000000	28.000000	0.000000	31646.000000	132.000000	154.000000	0.000000
75%	214373.500000	49.000000	1.000000	35.000000	1.000000	39377.750000	152.000000	227.000000	0.000000
max	285831.000000	85.000000	1.000000	52.000000	1.000000	540165.000000	163.000000	299.000000	1.000000

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285831 entries, 0 to 285830
Data columns (total 12 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   id                 285831 non-null  int64
 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     271602 non-null  float64
 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  int64
dtypes: float64(7), int64(2), object(3)
memory usage: 26.2+ MB
```

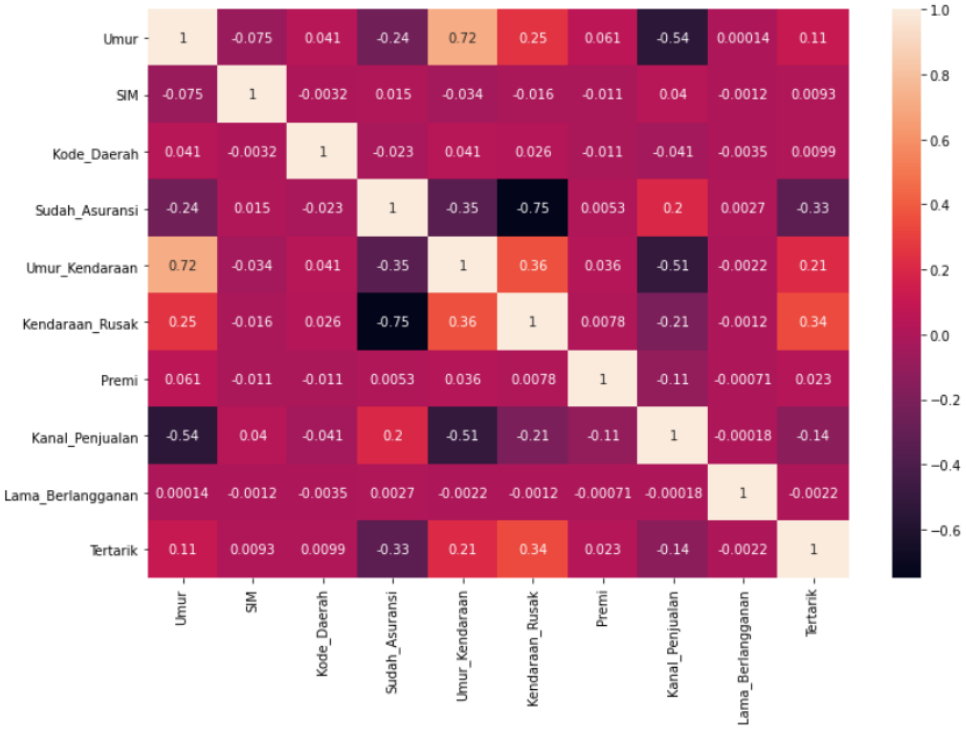
Dilihat dari info() data-data tersebut mempunyai data yang null atau tidak ada dan masih ada yang typenya object

```
df_train[['Jenis_Kelamin','SIM','Kode_Daerah','Sudah_Asuransi','Umur_Kendaraan',
          'Kendaraan_Rusak','Kanal_Penjualan','Tertarik']] = df_train[['Jenis_Kelamin','SIM','Kode_Daerah',
        'Sudah_Asuransi','Umur_Kendaraan','Kendaraan_Rusak','Kanal_Penjualan','Tertarik']].fillna(df_train.mode().iloc[0])
df_train[['Umur','Premi','Lama_Berlangganan']] = df_train[['Umur','Premi',
                                'Lama_Berlangganan']].fillna(df_train[['Umur','Premi','Lama_Berlangganan']].mean())
df_train=df_train.set_index('id')
```

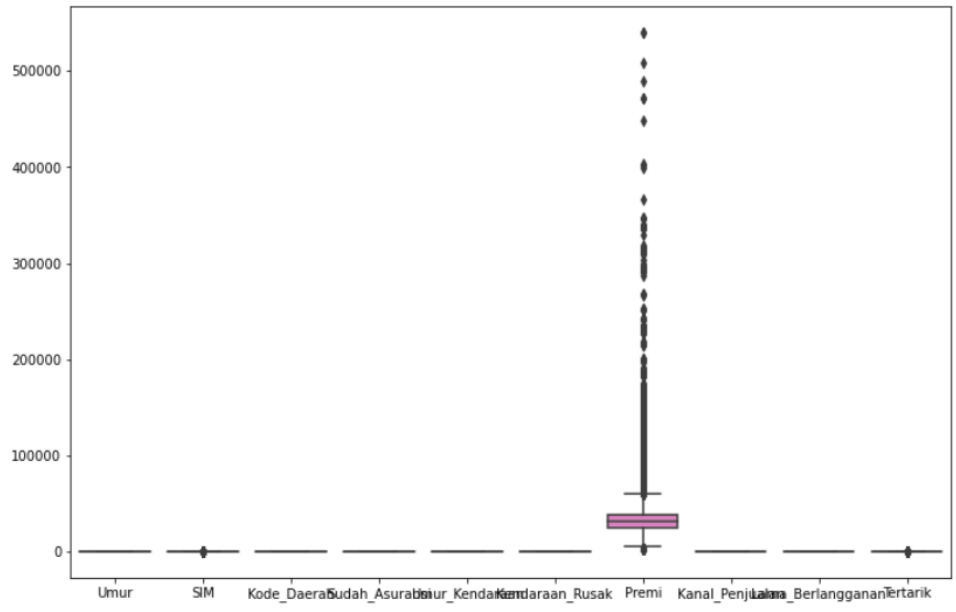
Dari data column yang null tersebut akan diisikan dengan data modus atau mean tergantung dari tipe setiap column tersebut.

```
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])
```

Lalu dilihat dari column Kendaraan_Rusak dan Umur_Kendaraan, data tersebut bisa diubah menjadi numeric yang diganti seperti diatas



Dari dataframe yang ada, dibuat matriks corelasinya untuk melihat korelasi antar column



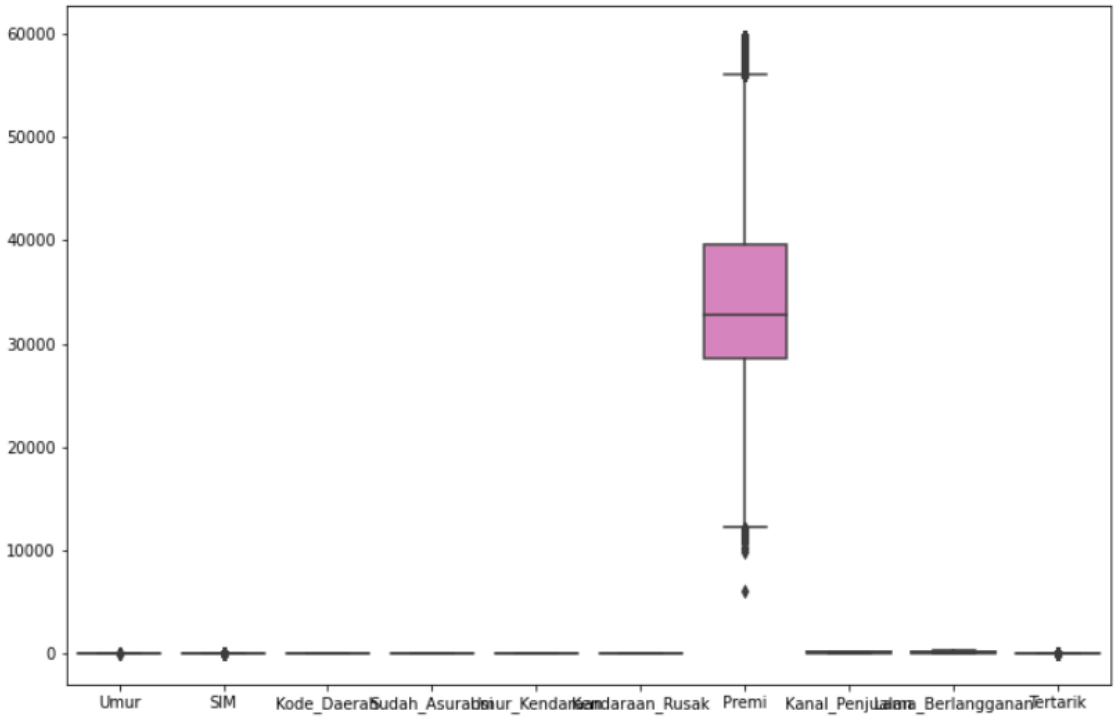
Lalu di cek menggunakan boxplot untuk melihat data outlier. Ternyata ada data outlier/pencilan dari column ‘Premi’. Oleh karena itu kita atasi data pencilan tersebut.

```python
###Mengatasi Outlier
def handling_outlier(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    IQR = q3-q1
    LB=q1 - (IQR * 1.5)
    UB=q3 + (IQR *1.5)
    dff = df[((df>LB) & (df<UB))]
    return dff
```

Menggunakan Lower Bound dan Upper Bound, dicari data apa saja yang berada diantara batas atas(Upper Bound) dan batas bawah(Lower Bound)

```python
df_train['Premi'] = handling_outlier(df_train[['Premi']])
df_train.dropna(axis=0,inplace=True)
```

Lalu function tersebut diimplementasikan ke main program. Setelah data-data yang mempunyai pencilan dijadikan data NaN atau kosong, data tersebut di drop.



Berikut tampilan data setelah data pencilannya didrop.

```
df_train = df_train.sample(1000)
scaler = StandardScaler()
df_scaler = scaler.fit_transform(df_train[['Premi','Lama_Berlangganan']])
df_scaler
```

Karena dataset yang dipunya terlalu banyak, dibuatlah sample data yang berisikan 1000 baris data. Setelah itu data di scalling menggunakan library StandarScaler dari scikitlearn.

C. Pemodelan

```
class Kmeans:
    def __init__(self,X,K):
        ##menginilisialisasikan variabel untuk class
        self.X=X
        self.Output={}
        self.Centroids=np.array([]).reshape(self.X.shape[1],0)
        self.K=K
        self.m=self.X.shape[0]

    def kmeansss(self,X,K):
        ##proses kmeans

        #menentukan posisi awal centroid
        i=rd.randint(0,X.shape[0])
        Centroid_temp=np.array([X[i]])

        #mencari jarak setiap data dengan centroid
        for k in range(1,K):
            D=np.array([])
            for x in X:
                D=np.append(D,np.min(np.sum((x-Centroid_temp)**2)))
            #menghitung probabilitas dari setiap centroid...
            #...agar tidak berdekatan dengan centroid lain
            prob=D/np.sum(D)
            cummulative_prob=np.cumsum(prob)
            r=rd.random()
            i=0
            for j,p in enumerate(cummulative_prob):
                if r<p:
                    i=j
                    break
            Centroid_temp=np.append(Centroid_temp,[X[i]],axis=0)
        return Centroid_temp.T
```

```
34
35      def fit(self,n_iter):
36          #inisialisasi centroid secara acak
37          self.Centroids=self.kmeansss(self.X,self.K)
38          #menghitung jarak dengan eculidian distances dan memasukan n cluster
39          for n in range(n_iter):
40              EuclidianDistance=np.array([]).reshape(self.m,0)
41              for k in range(self.K):
42                  tempDist=np.sum((self.X-self.Centroids[:,k])**2,axis=1)
43                  EuclidianDistance=np.c_[EuclidianDistance,tempDist]
44              C=np.argmin(EuclidianDistance,axis=1)+1
45
46              #update posisi centroid
47              Y={}
48              for k in range(self.K):
49                  Y[k+1]=np.array([]).reshape(2,0)
50              for i in range(self.m):
51                  Y[C[i]]=np.c_[Y[C[i]],self.X[i]]
52
53              for k in range(self.K):
54                  Y[k+1]=Y[k+1].T
55              for k in range(self.K):
56                  self.Centroids[:,k]=np.mean(Y[k+1],axis=0)
57
58              self.Output=Y
59
60      def predict(self):
61          return self.Output,self.Centroids.T
```

Dari codingan merupakan class dari kmeans yang berisi def kmeansss untuk menentukan centroid yang dimana centroid dibuat supaya tidak berdekatan dengan centroid lainnya. Memisahkan centroidnya dengan cara mencari probabilitas yang cocok untuk proses ini.

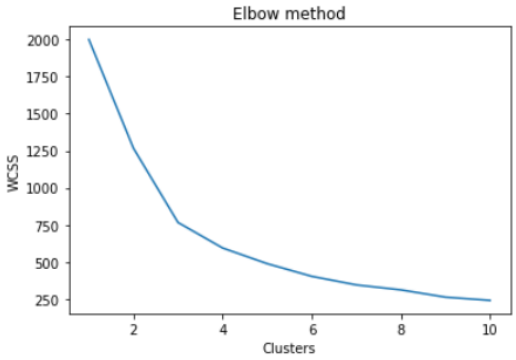
Setelah itu ada def fit yang berfungsi untuk pengulangan cluster sehingga mendapatkan cluster dengan posisi centroid terbaik.

D. Evaluasi

```
1  ##Menghitung best cluster menggunakan elbow method dengan WCSS
2  WCSS_array=np.array([])
3  n_iter=100
4  for K in range(1,11):
5      kmeans=Kmeans(df_scaler,K)
6      kmeans.fit(n_iter)
7      Output,Centroids=kmeans.predict()
8      wcss=0
9      for k in range(K):
10         wcss+=np.sum((Output[k+1]-Centroids[k,:])**2)
11     WCSS_array=np.append(WCSS_array,wcss)
12
```

Menggunakan metode elbow dengan WCSS untuk mencari the K terbaik untuk clustering. K diulang sampai 10 kali lalu didapatkan hasil WCSS nya. WCSS adalah Within-Cluster Sum of Squares yang menghitung jarak centroid dengan data-data clusternya.

```
#within-cluster sums of squares (WCSS)
K_array=np.arange(1,11,1)
plt.plot(K_array,WCSS_array)
plt.xlabel('Clusters')
plt.ylabel('WCSS')
plt.title('Elbow method')
plt.show()
```

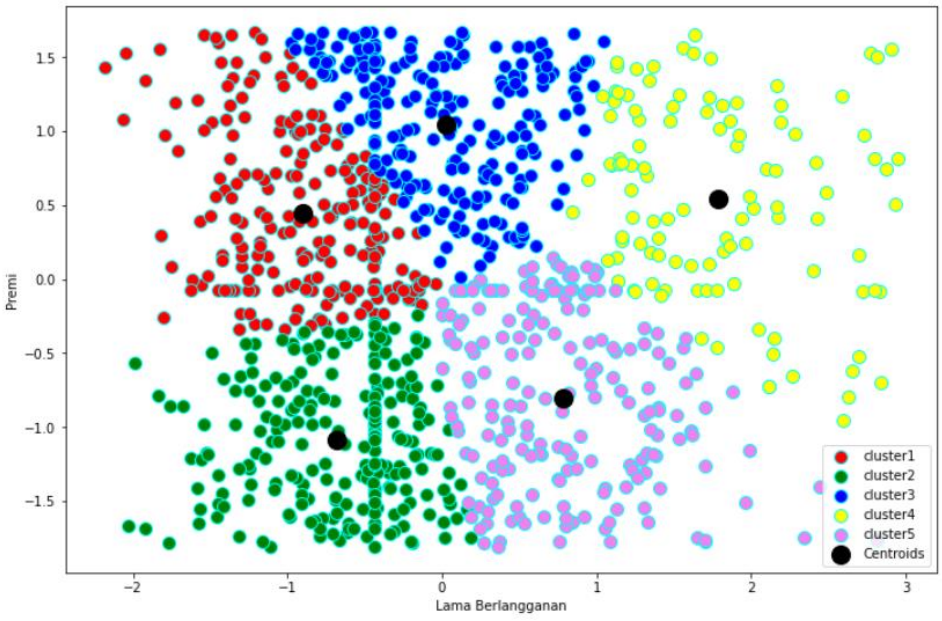


Dari table diatas diambil K = 5 karena di K = 5, grafik data sudah mulai landai yang mendandakan clustering tersebut mempunyai centroid yang stabil posisinya

E. Eksperimen

```
K = 5
kmeans=Kmeans(df_scaler,K)
kmeans.fit(n_iter)
Output,Centroids=kmeans.predict()
```

Dengan menggunakan K = 5 dan pengolahan kmeans menggunakan class yang sudah dibuat sebelumnya maka akan menghasilkan plot data clustering seperti berikut.



F. Kesimpulan

Kesimpulan dari eksperimen yang dilakukan adalah dengan menggunakan metode elbow WCSS bisa didapatkan clustering terbaik yaitu K = 5. Lalu dapat disimpulkan juga dengan menggunakan data ‘Lama_Berlangganan’ dan data ‘Premi’ dikombinasikan dengan K = 5 dapat dihasilkan clustering yang baik dan jelas. Hal ini terbukti bahwa model yang dibuat dapat bekerja secara optimal