Laporan Tugas Besar
Pembelajaran Mesin

Oleh : Aldiyan Farhan N – 1301180344

Link Video Presentasi :

https://youtu.be/elJHlqG0c0s

1. Permasalahan

Diberikannya data pembelian kendaraan untuk mengelompokkan pelanggan berdasarkan data pelanggan di dealer tanpa memperhatikan label kelas apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak. Permasalahan untuk tugas besar 1 ini adalah bagaimana data yang diberikan dapat di clusterisasi.

2. Explorasi dan Persiapan Data

a. Memasukan File

Memasukan data dengan melakukan pembanggilan URL yang memuat data set yang akan di explorasi :

```
[3] #kendaraan_train.csv
    datatraining_url ="https://cdn.discordapp.com/attachments/707989365015707758/907169095689072680/kendaraan_train.csv"
    training=requests.get(datatraining_url).content
    datatrain = pd.read_csv(io.StringIO(training.decode('utf-8')))

    #kendaraan_test.csv
    datatest_url="https://cdn.discordapp.com/attachments/707989365015707758/907169077833891931/kendaraan_test.csv"
    testing=requests.get(datatest_url).content
    datatest = pd.read_csv(io.StringIO(testing.decode('utf-8')))
```

b. Pengecekan Missing Value

Melakukan pengecekan untuk adanya value yang bernilai kosong/null. Data yang bernilai kosong/null akan mempengaruki pemrosesan data yang menyebabkan pemrosesan tidak berjalan optimal

```
for column in missing_datatrain.columns.values.tolist():
  print(column)
  print(missing_datatrain[column].value_counts())
  print("")
```

```
id
False    285831
Name: id, dtype: int64

Jenis_Kelamin
False    271391
True      14440
Name: Jenis_Kelamin, dtype: int64

Umur
False    271617
True      14214
Name: Umur, dtype: int64

SIM
False    271427
True      14404
Name: SIM, dtype: int64

Kode_Daerah
False    271525
True      14306
Name: Kode_Daerah, dtype: int64

Sudah_Asuransi
False    271602
True      14229
Name: Sudah_Asuransi, dtype: int64

Umur_Kendaraan
False    271556
True      14275
Name: Umur_Kendaraan, dtype: int64

Kendaraan_Rusak
False    271643
True      14188
Name: Kendaraan_Rusak, dtype: int64

Premi
False    271262
True      14569
Name: Premi, dtype: int64

Kanal_Penjualan
False    271532
True      14299
Name: Kanal_Penjualan, dtype: int64

Lama_Berlangganan
False    271839
True      13992
Name: Lama_Berlangganan, dtype: int64

Tertarik
False    285831
Name: Tertarik, dtype: int64
```

c. Drop Coloum

Dilakukan untuk menghapus kolom yang tidak digunakan untuk menghilangkan redundan

```
[21] datatrain.drop('id', axis=1, inplace=True)
     datatrain.drop('Tertarik', axis=1, inplace=True)
     datatrain
```

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

285831 rows × 10 columns

```
[22] datatest.drop('Tertarik', axis=1, inplace=True)
     datatest
```

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

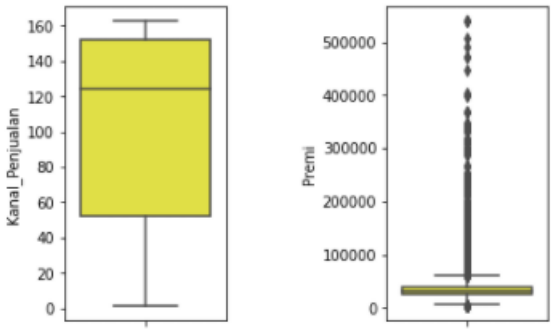
47639 rows × 10 columns

d. Pengecekan Outlier

Setelah selesai dengan missing value, tahap selanjutnya adalah mencari kolom dimana terdapat data outlier yang kemudian di modelkan dengan menggunakan boxplot supaya lebih mudah melihat dimana outlier tersebut berada

```
[44] def outlier(data):
       plt.figure(figsize=(60, 60))
       f, axes = plt.subplots(1, 2)
       sns.boxplot(y= data['Kanal_Penjualan'], ax= axes[0], color='yellow')
       sns.boxplot(y= data['Premi'], ax=axes[1], color='yellow')
       plt.subplots_adjust(wspace=1)

     outlier(dataCluster)
```

```
<Figure size 4320x4320 with 0 Axes>
```



e. Data Cleansing

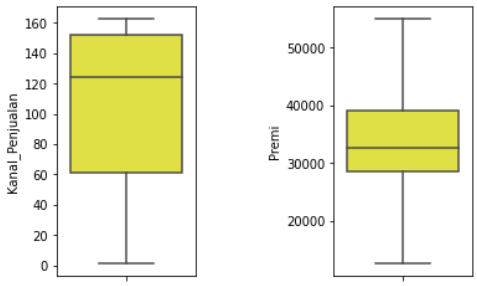
Melakukan data cleansing kemudian cek kembali adakah outlier di premi

```
[ ] #handle outlier Premi
    while True:
      qlo1, qlo3 = np.percentile(dataCluster['Premi'],[25,75])
      iqrlo = qlo3 - qlo1
      lowerlo = qlo1 - (1.5 * iqrlo)
      upperlo = qlo3 + (1.5 * iqrlo)
      outlierlo = dataCluster[(dataCluster['Premi'] < (lowerlo)) | (dataCluster['Premi'] > (upperlo))]
      print('amount of outlier data',outlierlo.shape[0]) #jumlah outlier data
      idxlo = outlierlo.index
      dataCluster.drop(idxlo, inplace=True) #drop outlier data
      if (outlierlo.shape[0] <= 0):
        break

    dataCluster['Premi'].describe()
```

```
amount of outlier data 64480
amount of outlier data 4132
amount of outlier data 1247
amount of outlier data 405
amount of outlier data 125
amount of outlier data 40
amount of outlier data 12
amount of outlier data 1
amount of outlier data 0
count    263028.000000
mean      34017.495005
std        7986.237800
min       12470.000000
25%       28458.000000
50%       32642.000000
75%       39123.000000
max       55120.000000
Name: Premi, dtype: float64
```

```
[ ] def check_outlier_encode(data):
      plt.figure(figsize=(60, 60))
      f, axes = plt.subplots(1, 2)
      sns.boxplot(y=data['Kanal_Penjualan'], ax= axes[0], color='yellow')
      sns.boxplot(y= data['Premi'], ax=axes[1], color='yellow')
      plt.subplots_adjust(wspace=1)

    check_outlier_encode(dataCluster)

    dataCluster.to_csv(r'/content/data_ready.csv', index=False, header=True)
```

```
<Figure size 4320x4320 with 0 Axes>
```



f. Normalisasi

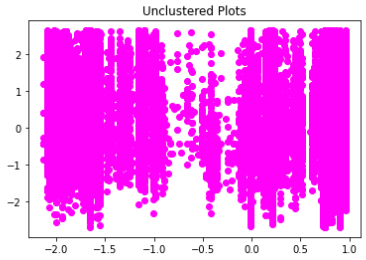
```
[ ]  from sklearn.preprocessing import StandardScaler

     normalize = StandardScaler()
     dc = normalize.fit_transform(dataCluster)

     dc
```

```
array([[ 0.74905745, -0.74985325],
       [-1.59773366, -1.02895892],
       [ 0.90169427, -0.16083887],
       ...,
       [ 0.21482858, -0.37295417],
       [ 0.74905745, -0.11475951],
       [-1.65497247,  0.47212621]])
```

3. Pemodelan

a. Unclustered Plot

```
[ ]  plt.scatter(dc[:,0],dc[:,1],c='magenta',label='unclustered data')

     plt.title('Unclustered Plots')

     plt.show()
```

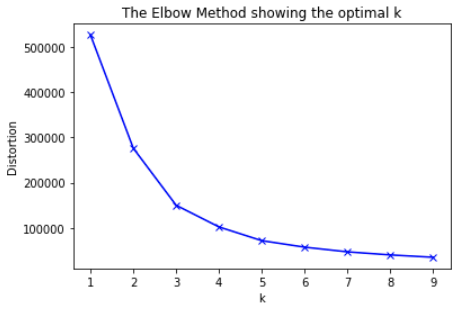


b. K-Means Functions

```
[ ]  def k_means(x,k, no_iterations):
         idx = np.random.choice(len(x), k)
         EuclidianDistance = np.array([]).reshape(x.shape[0],0)
         #PILIH CENTROID SECARA RANDOM
         centroids = x[idx, :]

         #MENCARI JARAK ANTARA TIAP CENTROID DAN SEMUA DATA POINT
         for i in range(k):
           tempDist = np.sum((x-centroids[i])**2,axis = 1)
           EuclidianDistance = np.c_[EuclidianDistance, tempDist]

         points = np.array([np.argmin(i) for i in EuclidianDistance])

         #MENGULANGI CARA DIATAS SEBANYAK ITERASI YANG DIINPUT
         for _ in range(no_iterations):
             centroids = []
             EuclidianDistance = np.array([]).reshape(x.shape[0],0)
             for idx in range(k):
                 #MELAKUKAN UPDATE CENTROID DENGAN MENGAMBIL RATA RATA CLUSTER
                 temp_cent = x[points==idx].mean(axis=0)
                 centroids.append(temp_cent)

             centroids = np.vstack(centroids) #UPDATE CENTROID

             for i in range(k):
               tempDist = np.sum((x-centroids[i])**2,axis = 1)
               EuclidianDistance = np.c_[EuclidianDistance, tempDist]

             points = np.array([np.argmin(i) for i in EuclidianDistance])

         return points, centroids
```

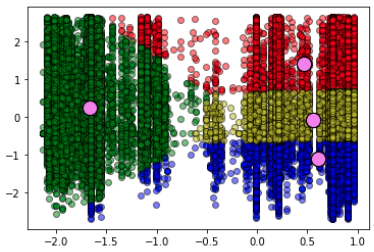
c. Elbow Method

```
from sklearn.cluster import KMeans
distortions = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k, max_iter=300)
    kmeanModel.fit(dc)
    distortions.append(kmeanModel.inertia_)

plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



d. Plotting Hasil K-Means

```
label, centroid = k_means(dc,4,300)
label_centre = ['centroid 1', 'centroid 2', 'centroid 3', 'centroid 4']
colors=['r', 'g', 'b', 'y', 'teal']
j = 0

u_labels = np.unique(label)

for i in u_labels:
    plt.scatter(dc[label == i , 0] , dc[label == i , 1] , c = colors[i], linewidths=1 ,alpha=0.5, edgecolors= 'k', label = i )

for centre in centroid:
    plt.scatter(centroid[: , 0] , centroid[: , 1] , s=200, c = 'violet',linewidths=1, edgecolors= 'k', label = label_centre[j])
    j += 1

plt.show()
```




```
from sklearn.metrics import silhouette_score
sil_score = []

for n_cluster in range(3, 7):
    kmeans = KMeans(n_clusters=n_cluster).fit(dc)
    label = kmeans.labels_
    sil_coeff = silhouette_score(dc, label, metric='euclidean')
    sil_score.append(sil_coeff)
    print('Nilai Silhoutte Method untuk n_clusters = {} adalah {}'.format(n_cluster, sil_coeff))
```

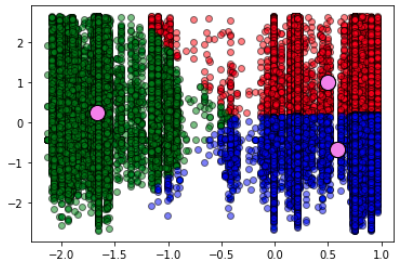
```
Nilai Silhoutte Method untuk n_clusters = 3 adalah 0.5163455216679663
Nilai Silhoutte Method untuk n_clusters = 4 adalah 0.5279529008500726
Nilai Silhoutte Method untuk n_clusters = 5 adalah 0.4608899212455691
Nilai Silhoutte Method untuk n_clusters = 6 adalah 0.44770756948382057
```

4. Observasi

a. Eksperimen Dengan Nilai K Yang Didapat

K = 3

```
[ ] label, centroid = k_means(dataWithOutlier,3,300)
    label_centre = ['centroid 1', 'centroid 2', 'centroid 3', 'centroid 4']
    colors=['r', 'g', 'b', 'y', 'teal']
    j = 0

    u_labels = np.unique(label)

    for i in u_labels:
        plt.scatter(dc[label == i , 0] , dc[label == i , 1] , c = colors[i], linewidths=1 ,alpha=0.5, edgecolors= 'k', label = i )

    for centre in centroid:
        plt.scatter(centroid[: , 0] , centroid[: , 1] , s=200, c = 'violet',linewidths=1, edgecolors= 'k', label = label_centre[j])
        j += 1

    plt.show()
```



K = 4

```
[ ] label, centroid = k_means(dataWithOutlier,4,300)
    label_centre = ['centroid 1', 'centroid 2', 'centroid 3', 'centroid 4']
    colors=['r', 'g', 'b', 'y', 'teal']
    j = 0

    u_labels = np.unique(label)

    for i in u_labels:
        plt.scatter(dc[label == i , 0] , dc[label == i , 1] , c = colors[i], linewidths=1 ,alpha=0.5, edgecolors= 'k', label = i )

    for centre in centroid:
        plt.scatter(centroid[: , 0] , centroid[: , 1] , s=200, c = 'violet',linewidths=1, edgecolors= 'k', label = label_centre[j])
        j += 1

    plt.show()
```