**LAPORAN TUGAS BESAR**

**MK PEMBELAJARAN MESIN 2021/2022**

Nama : Daniel Septyadi

NIM : 1301180009

Kelas : IF42GAB

## A. Permasalahan

Data yang diberikan merupakan hasil pendataan di sebuah dealer tentang ketertarikan pelanggan untuk membeli kendaraan baru. Di dalam data tersebut terdapat SIM, kode daerah, punya/tidaknya asuransi, umur kendaraan, rusak/tidaknya kendaraan yang dimiliki, premi yang harus dibayar, kanal penjualan, lama berlangganan, dan terakhir apakah pelanggan tersebut tertarik atau tidak membeli kendaraan baru.

Masalah yang ingin diselesaikan disini adalah algoritma cluster mampu melakukan clusterisasi terhadap data kendaraan untuk kemudian akan diprediksi apakah pelanggan tertarik untuk membeli kendaraan baru atau tidak berdasarkan data pelanggan di dealer.

## B. Eksplorasi dan Persiapan Data

Tidak dilakukan spliting karena melakukan clustering dan hanya perlu dilakukan eksplorasi data dan preprocessing/pembersihan data. bisa kita lihat data yang sudah di tampilkan, untuk data : Jenis_Kelamin, SIM, Kode Daerah, Sudah_Asuransi, Umur_Kendaraan, Kendaraan_Rusak tidak bisa dijadikan clustering karena data tersebut tidak akan variatif. Yang hanya bisa dipakai untuk dijadikan clustering yaitu : Umur, Premi, Kanal_Penjualan dan Lama Berlangganan.

**#choose data**

```
[ ] dataset1 = pd.read_csv('kendaraan_train.csv')
    dataset2 = pd.read_csv('kendaraan_test.csv')
    dataset = pd.concat([dataset1,dataset2])
    dataset.head()
```

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

```
#Choose Dataset
clean_data = dataset.drop(['Tertarik','Kode_Daerah','Kanal_Penjualan','id','Jenis_Kelamin','Sudah_Asuransi','Umur_Kendaraan','Kendaraan_Rusak','SIM','Premi'], axis = 1)
clean_data.head()
```

| | Umur | Lama_Berlangganan |
|---|---|---|
| 0 | 30.0 | 97.0 |
| 1 | 48.0 | 158.0 |
| 2 | 21.0 | 119.0 |
| 3 | 58.0 | 63.0 |
| 4 | 50.0 | 194.0 |

Disini saya memilih umur dan lama berlangganan untuk melakukan pengelompokan pelanggan berdasarkan umur berapa saja yang berlangganan lebih lama dan lebih sedikit. Sehingga bisa kita tentukan target pasar yang mendapatkan promo berlangganan berdasarkan clusternya.

**#check empty data**

```
#Check Empty Data

print("Data yang Kosong : ")
print(clean_data['Umur'].isnull().sum())

print("Jumlah Keseluruhan Data : ")
print(len(clean_data))


print("Persentase Umur yang Kosong : ")
print((clean_data['Umur'].isnull().sum()/len(clean_data['Umur']))*100)
print("Persentase Lama_Berlangganan yang Kosong : ")
print((clean_data['Lama_Berlangganan'].isnull().sum()/len(clean_data['Lama_Berlangganan']))*100)
```

```
Data yang Kosong :
14214
Jumlah Keseluruhan Data :
333470
Persentase Umur yang Kosong :
4.262452394518248
Persentase Lama_Berlangganan yang Kosong :
4.195879689327376
```

Disini kita akan tampilkan data yang kosong, Jumlah keseluruhan Data, persentase umur yang kosong dan persentase lama berlangganan yang kosong.

**#handle Missing Value with sum mean()**,

```
#Handle Missing Value with sum mean()
average_age = clean_data['Umur'].mean()
clean_data['Umur'] = clean_data['Umur'].fillna(average_age)

average_age = clean_data['Lama_Berlangganan'].mean()
clean_data['Lama_Berlangganan'] = clean_data['Lama_Berlangganan'].fillna(average_age)

print("Jumlah Data Kosong : ")
print(clean_data.isnull().sum())
```

Ada beberapa cara untuk menangani missing value , tapi disini saya memilih untuk tidak membuang datanya yaitu dengan menggunakan rata-rata. Karena untuk missing value itu jika kita menghilangkan datanya/dihapus akan berdampak pada clustering/modeling nantinya. Untuk menaruh nilai rata rata ke baris yg kosong/missing value disini kita akan cari rata-rata rumus: (jumlah seluruh angka dalam tabel/panjang seluruh data ) * 100.

**#function scaler**

```
#Scaler function
def scalerFunction(data):
    scaler_data = []
    x_min = data.min()
    x_max = data.max()
    for x in data:
        x = x/x_max
        scaler_data.append(x)
    return scaler_data
```

Menyamakan nilai umur dan lama berlangganan parameter nilainya diubah jadi rentang 0 ke 1, (Contoh: Umur : 1-100, sementara untuk lama berlangganan : 11 - 110 hari , jadi untuk menyamakan nilai umur dan lama berlangganan tersebut kita menggunakan scaler function. Rumus : x = x/x_max (setiap data dibagi dengan nilai data maximal).

**#call scaller**

```
#Call Scaller
umur_scaler = scalerFunction(clean_data['Umur'])
lamaberlangganan_scaler = scalerFunction(clean_data['Lama_Berlangganan'])
```

**#handle by Pandas**

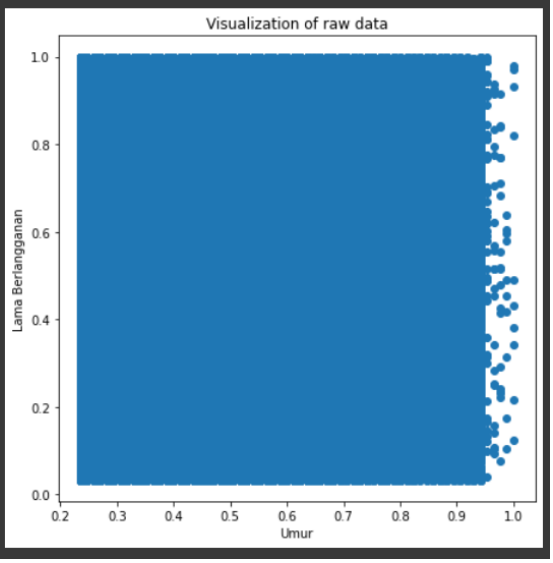
```
#Handle By Pandas
fixed_data = pd.DataFrame({'Umur':umur_scaler,'Lama_Berlangganan':lamaberlangganan_scaler})
fixed_data
```

| | Umur | Lama_Berlangganan |
|---|---|---|
| 0 | 0.352941 | 0.324415 |
| 1 | 0.564706 | 0.528428 |
| 2 | 0.247059 | 0.397993 |
| 3 | 0.682353 | 0.210702 |
| 4 | 0.588235 | 0.648829 |
| ... | ... | ... |
| 333465 | 0.717647 | 0.224080 |
| 333466 | 0.482353 | 0.775920 |
| 333467 | 0.282353 | 0.705686 |
| 333468 | 0.694118 | 0.799331 |
| 333469 | 0.611765 | 0.568562 |

333470 rows × 2 columns

Panggil fungsi scaller yg tadi sudah dimasukkan yaitu umur dan lama berlangganan yang selanjutnya dihandle oleh pandas dipindahkan ke tabel baru.

```
plt.figure(figsize=(7, 7))
plt.scatter(fixed_data['Umur'],fixed_data['Lama_Berlangganan'])
plt.xlabel('Umur')
plt.ylabel('Lama Berlangganan')
plt.title('Visualization of raw data');
```




Lalu diakhir tampilkan visualisasikan persebaran data keseluruhannya.

## C. Pemodelan

K-means , menentapkan clustering pertama secara acak, lalu mencari jarak terdekat ke cluster menggunakan rumus euclidean distance.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

$p, q$ = two points in Euclidean n-space

$q_i, p_i$ = Euclidean vectors, starting from the origin of the space (initial point)

$n$ = n-space

**#define start cluster**

```
[10] #Defenite start cluster
     def randomCluster(number_of_cluster):
         cluster_rows = np.random.randint(0,len(fixed_data),size=number_of_cluster)
         return cluster_rows
```

**#change native cluster_array to numpy cluster_array**

```
[11] #Change native cluster_array to numpy cluster_array
     def cluster_to_np(get_centroid):
         pcluster = []
         for get in get_centroid:
             pcluster_temp = []
             pcluster_temp.append(fixed_data['Umur'][get])
             pcluster_temp.append(fixed_data['Lama_Berlangganan'][get])
             pcluster.append(pcluster_temp)

         pcluster = np.array(pcluster)
         return pcluster
```

**#count Disctance fucntion**

```
[12] #Count Distance function
     def distance(x_position,centoroid_x,y_position,centoroid_y):
         dist = np.sqrt(((centoroid_x-x_position)**2)+((centoroid_y-y_position)**2))
         return dist
```

## #K-Means Function

```
[13] #K-Means Function
     # def kMeans(fixed datasets,cluster):
     def intialCentroid(fixed_data,clusters):
         labels = []
         umur = fixed_data['Umur']
         lama = fixed_data['Lama_Berlangganan']
         for cls in range(len(clusters)):
             labels_temp = []
             centroid_x = clusters[cls][0]
             centroid_y = clusters[cls][1]
             for i in range(len(fixed_data)):
                 labels_temp.append(distance(umur[i],centroid_x,lama[i],centroid_y))
             labels.append(labels_temp)
         return labels
```

## #setMinimumDistance

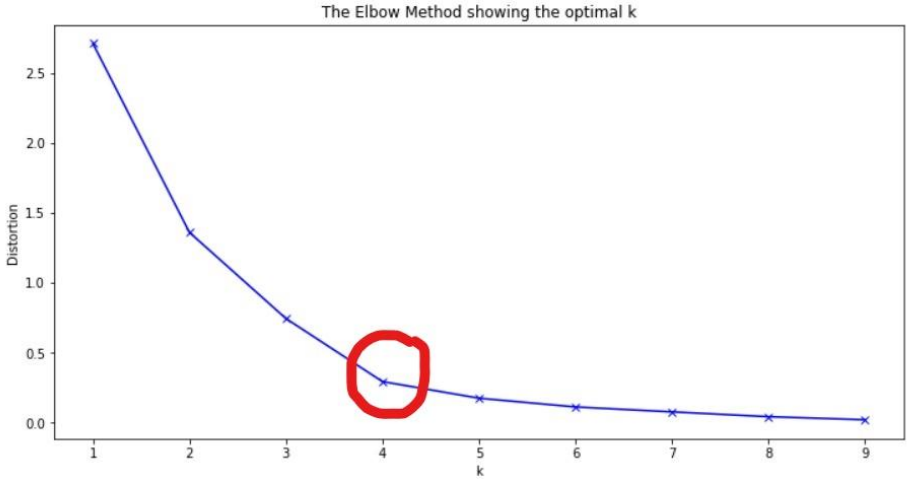
```
[14] def setMinimimumDistance(datas,labels):
         cluster_labels = []
         for n in range(len(labels[1])):
             min_position = np.where(datas[n]==datas[n].min())
             cluster_labels.append(min_position[0][0])
         return cluster_labels
```

## #findNewCluster

```
[ ] def findNewCluster(number_of_cluster,fixed_data):
        new_cluster = []
        for i in range(number_of_cluster):
            get_avg = []
            #Average every cluster for new cluster
            get_cluster = fixed_data.loc[fixed_data['Cluster']==i]
            get_avg.append(get_cluster['Umur'].mean())
            get_avg.append(get_cluster['Lama_Berlangganan'].mean())
            new_cluster.append(get_avg)

        new_cluster = np.array(new_cluster)
        return new_cluster
```

## D. Evaluasi




Dalam metode elbow untuk melihat k terbaik adalah melihat cekungan siku itulah k terbaik. Disini tampak terlihat menurut metode elbow(siku) k terbaik adalah bentuk siku yg terbentuk adalah = 4

## E. Eksperimen.

```
def kMeansAlgorithm(total_cluster,max_iteration,fixed_data):
    min_val = []
    for iteration in range(max_iteration):
        if (iteration==0):
            #Initial starting centroid with random
            get_centroid = randomCluster(total_cluster)
            pcluster = cluster_to_np(get_centroid)
        else:
            new_cluster = findNewCluster(total_cluster,fixed_data)
            # stop if cluster not changes
            if ((new_cluster==pcluster).all().any()):
                break
            pcluster = new_cluster
        #Count Distance and initial centoroid
        labels = intialCentroid(fixed_data, pcluster)

        #Find Min for variance
        min_val.append(min(labels[0]))


        #Change labels to pandas Dataframe
        datas = pd.DataFrame(labels)
        #Set minimum distance
        cluster_labels = setMinimimumDistance(datas,labels)
        #Merger labels with dataset
        fixed_data = pd.DataFrame({'Umur':umur_scaler,'Lama_Berlangganan':lamaberlangganan_scaler,'Cluster':cluster_labels})


    #Count Innertia/Variance for Elbow
    value_temp = 0
    values = 0
    innertia = 0
    avg_innertia = np.mean(min_val)
    for variance_iter in range(len(min_val)):
        value_temp = (min_val[variance_iter]-avg_innertia)**2
        values = values+value_temp
    innertia = value_temp/len(labels[0])
    return fixed_data, pcluster,innertia
```

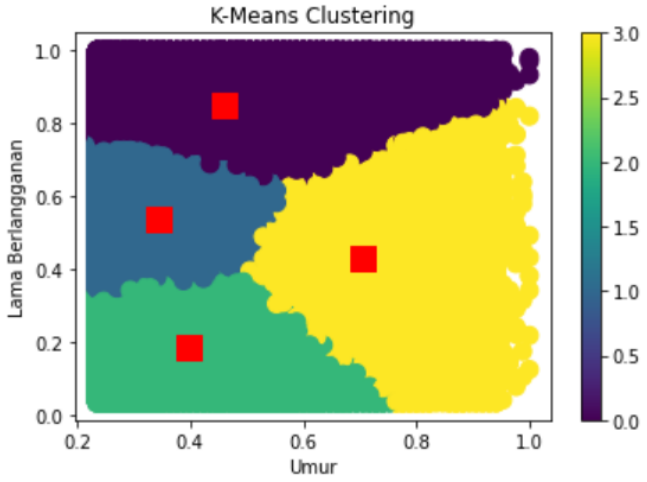
Kita melakukan eksperimen dengan menggunakan *coding from scratch* dibuat dari awal tanpa menggunakan library *sklearn* dengan rumus rumus yang ada yaitu dengan menggunakan rumus *euclidean distance*.

```
distortions = []
K = range(1,10)
for k in K:
    kmeans = kMeansAlgorithm(k,123,fixed_data)
    innertia_ = kmeans[2]
    distortions.append(innertia_)
```

Lalu dibagian ini kita melalukan eksperimen dengan melakukan perulangan k sebanyak 10 kali menentukan mana k yang paling baik (1-10)

## F. Kesimpulan

```
#visual hasil kluster
output = plt.scatter(final_data['Umur'], final_data['Lama_Berlangganan'], s = 100, c = final_data['Cluster'], marker = "o", alpha = 1, )
plt.scatter(pcluster[:,0], pcluster[:,1], c='red', s=200, alpha=1 , marker="s");
plt.title("K-Means Clustering")
plt.xlabel('Umur')
plt.ylabel('Lama Berlangganan')
plt.colorbar (output)
plt.show()
```




Dari kesimpulan yang didapat kalau k yang terbaik itu didapatkan berdasarkan hasil evaluasi , Jadi k yang terbaik itu adalah 4 , kita mencari nilai yang memiliki variansi terbaik berdasarkan metode elbow(bentuk siku) muncul bentuk siku itulah yang kita ambil sebagai nilai terbaik. Ada 4 jenis konsumen berdasarkan lama berlangganan dan umurnya. Dengan 4 jenis clustering tersebut kita bisa menentukan respon kita terhadap pelanggan mana saja pelanggan yang bisa mendapatkan reward/promo supaya menjadi pelanggan tetap.

## G. Lampiran

Link Video : https://youtu.be/0G0zOBQYcYE

Link Code : Clustering_Program_1301180009

## H. Referensi


https://g.co/kgs/iUN5wq

https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb