LAPORAN MACHINE LEARNING

CLUSTERING DAN CLASSIFICATION

Made Adi Widyananda - 1301174158

1. Formulasi Masalah

Pada tugas machine learning ini saya mendapatkan dataset yaitu Used Car dimana dataset tersebut merupakan data yang menyimpan sekumpulan data-data mobil bekas yang akan dijual, dengan memanfaatkan machine learning saya membuat suatu label baru pada dataset yang nantinya akan berguna bagi orang yang akan membeli mobil bekas. Dengan memanfaatkan data tahun mobil dan odometernya saya akan membuat suatu label yaitu keterangan dengan isi didalam label tersebut adalah mobil itu sangat direkomendasikan, direkomendasikan, kurang direkomendasikan, dan tidak direkomendasikan.

Dalam pembuatannya saya juga akan menggunakan 2 eksperimen, yaitu di dalam penyiapan datanya. Untuk eksperimennya saya akan membandingkan antara data dengan menghilangkan outliers dan data yang tidak menghilangkan outliers, dimana outliers merupakan data yang memiliki nilai yang sangat jauh dibandingkan data lainnya, data yang tidak menghilangkan outliers akan memiliki data yang lebih banyak namun dengan rentang jarak data yang lebih jauh, sedangkan data yang menghilangkan outliers akan memiliki data yang lebih sedikit namun rentang jaraknya tidak jauh. Sehingga dari eksperimen yang dilakukan akan membandingkan hasil data mana yang lebih baik diantara 2 model tersebut.

Dalam melaksanakan eksperimen tersebut, saya akan mengolah data Used_Car agar menghasilkan data baru yang dapat digunakan. Pertama kami menyiapkan datanya sebelum di proses, yaitu dengan cara menghilangkan missing values(0 dan NaN) agar data yang tersisa merupakan data yang berisi nilai, melakukan scaling normalization digunakan agar rentang nilai yang digunakan menjadi lebih kecil(0 - 1) , dan pada salah satu program saya akan menghilangkan outliers sebagai pembeda pada eksperimen.

Selanjutnya setelah melakukan persiapan data, saya melakukan clustering dengan metode Kmeans, metode Kmeans bekerja dengan cara mengambil beberapa data random dalam suatu dataset, setelah itu dijadikan kelas yang nantinya akan digunakan sebagai kelas baru dan dirata-ratakan tiap kelas dan nilainya digunakan sebagai pembaruan kelas tersebut, sehingga mendapatkan kelas baru lagi, dan seterusnya seperti itu sehingga tidak ada perubahan kelas lagi, dengan cara seperti itu data akan terkelompok secara efektif. dan setelah didapatkan kelas yang baru maka kita mendapatkan data_set yang baru. Setelah itu data tersebut akan digunakan pada classification, pada program melakukan 2 classification yaitu Linear SVM, dimana Linear SVM bekerja dengan cara membuat hyperline dan memaksimalkan jarak antar kelas dan KNN bekerja dengan cara melihat kecocokan antara data yang baru dengan data di sekitarnya, dan melihat akurasi dari classifitio yang telah dilakukan menggunakan metrics.

2. Penjelasan Program

Untuk program dengan menghilangkan outliers saya namakan WOoutliers, sedangkan program yang tanpa menghilangkan outliers saya namakan Woutliers.

Library

Library yang digunakan program WOoutliers

```
#Library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from sklearn.preprocessing import MinMaxScaler
import random
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

Penjelasan:

1. Import Pandas : digunakan untuk mengambil dataset dari file yang berbeda dan persiapan data.
2. Import Numpy : digunakan untuk pengolahan objek seperti array.
3. Import Matplotlib : digunakan untuk menggambarkan data seperti plot grafik.
4. From spicy import stats : digunakan untuk menghilangkan outliers.
5. From sklearn.preprocessing import MinMaxScaler : digunakan untuk scaling normalization.
6. Import random : digunakan untuk mendapatkan angka random.
7. From sklearn.model_selection import train_test_split: digunakan untuk membagi data.
8. From sklearn import svm : digunakan untuk melakukan classification SVM.
9. From sklearn.neighbors import KNeighborsClassifier : digunakan untuk melakukan classification KNN.
10. From sklearn import metrics : digunakan untuk mengevaluasi classification.

Library yang digunakan program Woutliers

```
#Library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
import random
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

Library yang digunakan pada program WOoutliers hampir sama dengan program Woutliers, hanya saja di dalam WOoutliers tidak terdapat library From spicy import stats yang digunakan untuk menghilanglan outliers.

Pengambilan Dataset

```
#Pengambilan dataset
uc_data = pd.read_csv("used_cars.csv")
uc_data.shape
```

```
(20001, 26)
```

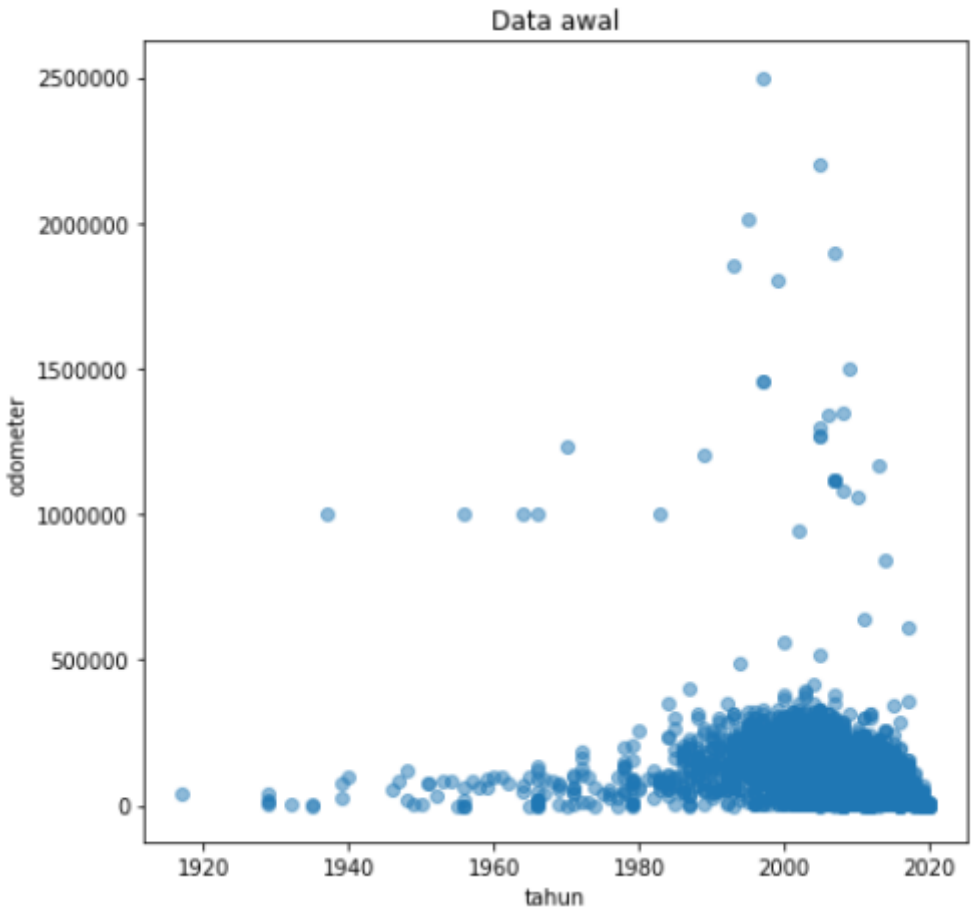
Pada program WOoutliers dan Woutliers menggunakan cara yang sama dalam pengambilan dataset, yaitu menggunakan library pandas, dan dengan fungsi shape dapat melihat banyak baris dan banyak kolom dari dataset.

Plot Grafik Antara Tahun dan Odometer

Pada program WOoutliers dan Woutliers menggunakan metode yang sama dalam pembuatan grap.

```
#Plot grap antara tahun dan odometer
fig,grap = plt.subplots(figsize=(7,7))
grap.scatter(uc_data["year"], uc_data["odometer"], alpha=0.5)
plt.title('Data awal')
plt.xlabel('tahun')
plt.ylabel('odometer')
```

Dengan menggunakan library Matplotlib kita membuat grap dengan ukuran 7x7 dengan xlabel adalah tahun(year) dan ylabel adalah odometer dengan alpha 0,5 yang menunjukan trasnparansi objek dalam grap dan juga judul grap yaitu Data awal, dan menghasilkan grap seperti ini.



Menghilangkan Missing Values (0 dan NaN)

Pada program Woutliers dan WOoutliers menggunakan metode yang sama dalam pembuatan menghilangkan missing values.

```
#Menghilangkan missing values (0 dan NaN)
use_data = uc_data[['year','odometer']].replace(0, np.nan)
use_data.dropna(inplace=True)
use_data.shape
```

```
(17570, 2)
```

Use_data merupakan data yang nantinya digunakan dalam clustering, use_data mengambil data berupa kolom tahun(year) dan odometer dan menghilangkan 0 dan juga nan dengan mendrop table yang berisi value 0 dan NaN, sehingga dapat dilahat baris yang awalnya berjumlah 20001 menjadi 17570 dan kolom yang awalnya 26 menjadi 2(year dan odometer).

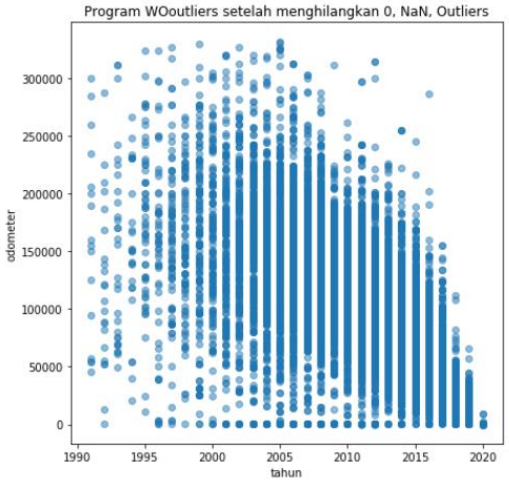
Menghilangkan Outliers Pada Program WOoutliers

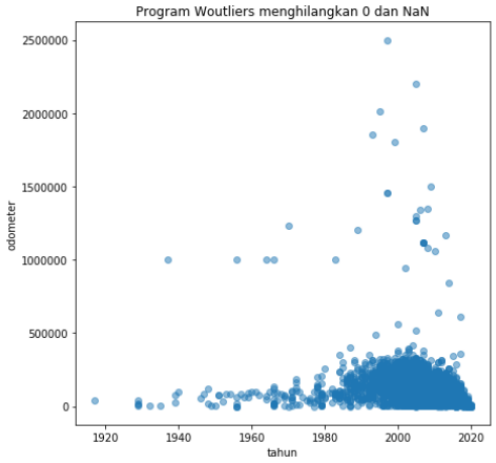
```
#Menghilangkan outliers
z = np.abs(stats.zscore(use_data))
threshold = 3
use_data = use_data[(z < threshold).all(axis=1)]
use_data.shape
```

```
(17302, 2)
```

Menghilangkan Outliers pada program WOoutliers dengan nilai threshold 3, dan dapat dilihat jumlah baris yang awalnya 17570 menjadi 17302.

Perbandingan Grap antara Program WOoutliers (setelah menghilangkan outliers) dan Program Woutliers






```
#Plot grap antara tahun dan odometer setelah menghilangkan 0, NaN, Outliers
fig, grap = plt.subplots(figsize=(7,7))
grap.scatter(use_data["year"], use_data["odometer"], alpha=0.5)
plt.title('Program WOoutliers setelah menghilangkan 0, NaN, Outliers')
plt.xlabel('tahun')
plt.ylabel('odometer')
```

```
Text(0, 0.5, 'odometer')
```

```
#Plot grap antara tahun dan odometer setelah menghilangkan 0 dan NaN
fig,grap = plt.subplots(figsize=(7,7))
grap.scatter(use_data["year"], use_data["odometer"], alpha=0.5)
plt.title('Program Woutliers menghilangkan 0 dan NaN')
plt.xlabel('tahun')
plt.ylabel('odometer')
```

```
Text(0, 0.5, 'odometer')
```

Dapat dilihat pada grap Program WOoutliers tidak terdapat objek data yang memilki jarak yang jauh dengan data yang lain, sedangkan pada program Woutliers tetap terdapat data yang memiliki jarak yang jauh dengan data yang lain, begitu juga dengan rentang tahun dan odometernya, pada program WOoutliers rentang tahun dan odometernya lebih kecil dibandingkan program Woutliers.

Melakukan Scaling Normalization Pada Data

```
#melakukan scaling normalization pada Program WOoutliers
new_data = use_data
scaler = MinMaxScaler()
use_data = scaler.fit_transform(use_data)
use_data = pd.DataFrame(use_data)
use_data.head()
```

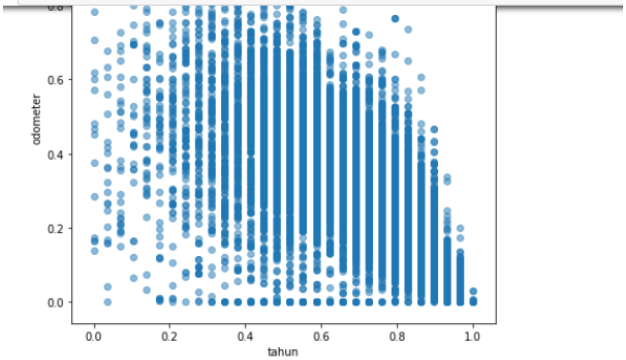
	0	1
0	0.724138	0.191263
1	0.862069	0.000027
2	0.827586	0.022750
3	0.862069	0.000027
4	0.931034	0.211293

```
#melakukan scaling normalization pada Program Woutliers
new_data = use_data
scaler = MinMaxScaler()
use_data = scaler.fit_transform(use_data)
use_data = pd.DataFrame(use_data)
use_data.head()
```

	0	1
0	0.922330	0.025400
1	0.961165	0.000004
2	0.951456	0.003021
3	0.961165	0.000004
4	0.980583	0.028060

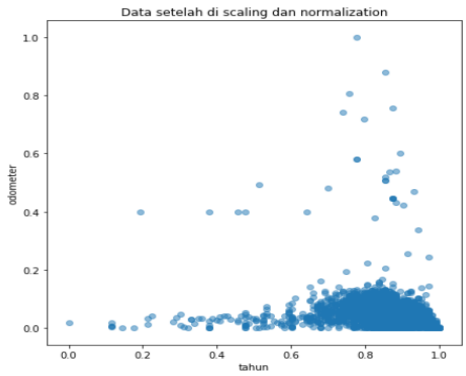
Pada program Woutliers dan WOoutliers menggunakan metode yang sama yaitu melakukan scaling normalization, metode ini digunakan agar rentang jarak data tahun dan juga odometer menjadi 0 – 1 sehingga memiliki rentang nilai yang sama dan tidak ada variable yang mendominasi dan juga kolom tahun(year) berubah menjadi 0 dan kolom odometer berubah menjadi 1. Disini juga kita membuat new_data yang nantinya digunakan untuk hasil akhir yang tujuannya untuk menyimpan data tahun dan odometer sebelum rentangnya diubah menjadi 0 - 1.

```
fig, grap = plt.subplots(figsize=(7,7))
grap.scatter(use_data[0], use_data[1], alpha=0.5)
plt.title('Program WOoutliers setelah di scaling dan normalization')
plt.xlabel('tahun')
plt.ylabel('odometer')
```




```
fig, grap = plt.subplots(figsize=(7,7))
grap.scatter(use_data[0], use_data[1], alpha=0.5)
plt.title('Program Woutliers setelah di scaling dan normalization')
plt.xlabel('tahun')
plt.ylabel('odometer')
```

```
Text(0, 0.5, 'odometer')
```



Clustering (Kmeans)

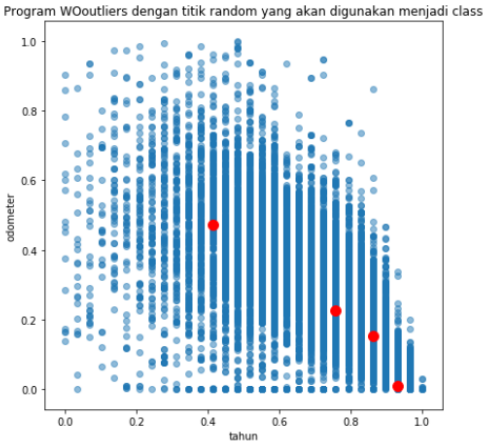
1. Menyiapkan 4 titik random sebagai kelas

```
#clustering membuat 4 class
#mengambil random nilai dari use_data digunakan sebagai titik class
k = 4;
datax = use_data[0]
datay = use_data[1]
centerx = []
centery = []
for i in range(k):
    r = random.randint(0,17302)
    centerx.append(datax[r])
    centery.append(datay[r])
```

Pada program Woutliers dan WOoutliers menggunakan metode yang sama yaitu melakukan Clustering dengan metode Kmeans namun saat mencari nilai random disesuaikan dengan banyak data Woutliers(0 - 17570) WOoutliers(0 – 17302. Dapat dilihat pada kode diatas, variable k menunjukkan jumlah titik random yang nantinya akan digunakan di perulangan, variable datax merupakan salinan use_data kolom 0 (tahun) dan datay merupakan salinan use_data 1 (odometer), array centerx digunakan untuk menyimpan nilai random dari datax, dan array centery digunakan untuk menyimpan nilai random dari datay. Pada bagian perulangan for sebanyak k terdapat variable r yang digunakan untuk menyimpan nilai random, setelah itu array centerx dan centery akan diisi dengan nilai dari datax dan datay sesuai dengan index r. Setelah perulangan selesai kita akan mendapatkan 4 titik random.

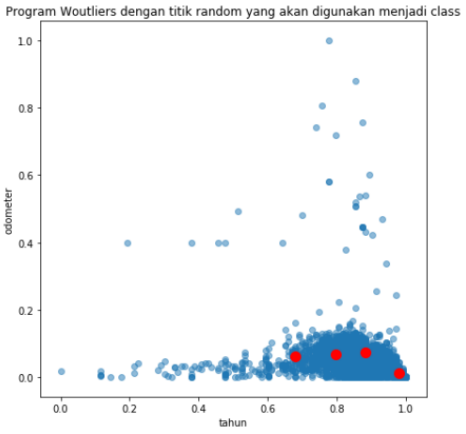
```
fig, grap = plt.subplots(figsize=(7,7))
grap.scatter(use_data[0], use_data[1], alpha=0.5)
grap.scatter(centerx,centery,s=100, c="r")
plt.title('Program WOoutliers dengan titik random yang akan digunakan menjadi class')
plt.xlabel('tahun')
plt.ylabel('odometer')
```

```
Text(0, 0.5, 'odometer')
```




```
fig, grap = plt.subplots(figsize=(7,7))
grap.scatter(use_data[0], use_data[1], alpha=0.5)
grap.scatter(centerx,centery,s=100, c="r")
plt.title('Program Woutliers dengan titik random yang akan digunakan menjadi class')
plt.xlabel('tahun')
plt.ylabel('odometer')
```

```
Text(0, 0.5, 'odometer')
```



Dapat dilihat pada grap diatas titik random yang dibuat adalah titik yang berwarna merah dengan ukuran 100.

Pembuatan Fungsi Yang Digunakan Untuk Clustering

```
#Mendefinisikan fungsi yang akan digunakan pada clustering
def Distance(centerx,centery,datax,datay):
    distance1 = np.sqrt(((centerx[0] - datax[i])**2) + ((centery[0] - datay[i])**2))
    distance2 = np.sqrt(((centerx[1] - datax[i])**2) + ((centery[1] - datay[i])**2))
    distance3 = np.sqrt(((centerx[2] - datax[i])**2) + ((centery[2] - datay[i])**2))
    distance4 = np.sqrt(((centerx[3] - datax[i])**2) + ((centery[3] - datay[i])**2))
    if ((distance1 < distance2) and (distance1 < distance3) and (distance1 < distance4)):
            return(0)
    elif ((distance2 < distance1) and (distance2 < distance3) and (distance2 < distance4)):
            return(1)
    elif ((distance3 < distance1) and (distance3 < distance2) and (distance3 < distance4)):
            return(2)
    elif ((distance4 < distance1) and (distance4 < distance2) and (distance4 < distance3)):
            return(3)

def Average(data,index,var):
    i = 0
    jum = 0
    pem = 0
    for i in range(len(data)):
        if (data['keterangan'][i]== index):
            jum = jum + data[var][i]
            pem = pem + 1
    return(jum/pem)
```

Pada program Woutliers dan WOoutliers menggunakan fungsi yang sama. Fungsi Distance digunakan untuk menghitung jarak suatu objek(titik) pada dataset terhadap 4 titik yang sudah ditentukan sebelumnya sebagai kelas, dan setelah mendapatkan jarak antar titik, program akan membandingkan titik mana yang paling terdekat dari objek tersebut, setelah itu fungsi akan mengeluarkan nilai yang menentukan objek tersebut masuk kedalam suatu kelas diantara 4 kelas tersebut (0,1,2,3).

Fungsi Average digunakan untuk menghitung rata-rata pada suatu kelas, yang nantinya rata-rata tersebut akan digunakan untuk menentukan kelas yang baru. Bisa

dilihat pada gambar diatas terdapat variable i yang diisi dengan nilai 0 yang nantinya digunakan pada perulangan, dan terdapat variable jum yang menyimpan jumlah dari data selama perulangan, dan variable pem digunakan untuk menyimpan nilai pembagi. Perulangan diulang sebanyak panjang data yang dimasukkan, dan inputan variable index digunakan untuk menentukan kelas mana yang dihitung rata-ratanya dan var adalah variable yang digunakan untuk menentukan kolom mana yang akan dihitung jumlahnya.

Menjalankan Clustering(Kmeans)

```
#Menjalankan CLustering
precenterx = [0,0,0,0]
precentery = [0,0,0,0]
while True:
    precenterx = centerx
    precentery = centery
    clustering_data = []
    for i in range(len(use_data)):
        clustering_data.append(Distance(centerx,centery,datax,datay))
    use_data['keterangan'] = clustering_data
    centerx[0] = Average(use_data,0,0)
    centerx[1] = Average(use_data,1,0)
    centerx[2] = Average(use_data,2,0)
    centerx[3] = Average(use_data,3,0)
    centery[0] = Average(use_data,0,1)
    centery[1] = Average(use_data,1,1)
    centery[2] = Average(use_data,2,1)
    centery[3] = Average(use_data,3,1)

    if((precenterx[0] == centerx[0]) and (precenterx[1] == centerx[1]) and (precenterx[2] == centerx[2]) and (precenterx[3] == c
            break
```

Pada program Woutliers dan WOoutliers menggunakan Clustering(Kmeans) dengan kode yang sama. Dapat dilihat pada gambar terdapat array precenterx dan precentery yang nantinya digunakan untuk membandingkan center yang baru dengan center setelahnya. Setelah itu kita melakukan perulangan while dengan kondisi true, perulangan akan tetap dijalankan, di dalam perulangan tersebut precenterx diisi nilainya dengan centerx dan precentery diisi nilainya dengan centerx untuk nantinya akan dibandingkan, setelah itu terdapat array clustering data yang menyimpan hasil clustering. Di dalam perulangan while tersebut, terdapat perulangan for sebanyak jumlah data yang ada pada use_data, dan di dalam perulangan tersebut array clustering data ditambah(append) dengan fungsi Distance, dimana seluruh data yang ada pada use_data akan di cek dan dimasukkan ke salah satu dari 4 kelas yang telah ditentukan sebelumnya. Setelah semua data dimasukkan ke kelas, maka use_data akan menambah kolom baru yaitu kolom keterangan yang menyimpan data hasil clustering. Setelah itu 4 kelas/titik tersebut nilainya akan dirubah sesuai dengan hasil rata-rata tiap kelas, setelah mendapatkan nilai kelas yang baru, kelas tersebut akan dibandingkan dengan kelas sebelumnya, jika kelas tersebut memiliki nilai yang sama dengan nilai kelas sebelumnya menandakan sudah tidak ada perubahan pada kelas lagi dan perulangan dihentikan.

Data Setelah Clustering

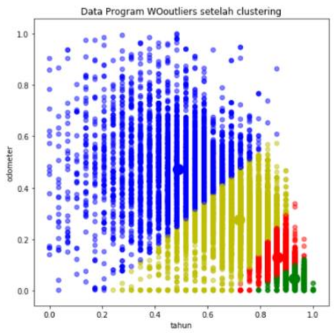
```
fig, grap = plt.subplots(figsize=(7,7))
use_data['keterangan'] = use_data['keterangan'].replace([0,1,2,3],["r","y","b","g"])
grap.scatter(use_data[0], use_data[1], color=use_data['keterangan'], alpha=0.5)
grap.scatter(centerx[0],centery[0],s=200, c="r")
grap.scatter(centerx[1],centery[1],s=200, c="y")
grap.scatter(centerx[2],centery[2],s=200, c="b")
grap.scatter(centerx[3],centery[3],s=200, c="g")
plt.title('Data Program WOoutliers setelah clustering')
plt.xlabel('tahun')
plt.ylabel('odometer')
use_data.head(20)
```

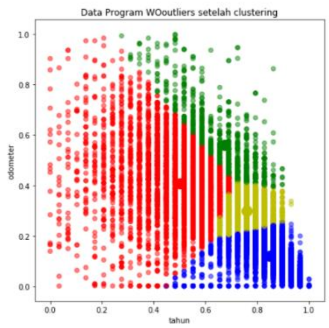
```
fig, grap = plt.subplots(figsize=(7,7))
use_data['keterangan'] = use_data['keterangan'].replace([0,1,2,3],["r","y","b","g"])
grap.scatter(use_data[0], use_data[1], color=use_data['keterangan'], alpha=0.5)
grap.scatter(centerx[0],centery[0],s=400, c="r")
grap.scatter(centerx[1],centery[1],s=400, c="y")
grap.scatter(centerx[2],centery[2],s=400, c="b")
grap.scatter(centerx[3],centery[3],s=400, c="g")
plt.title('Data Program Woutliers setelah clustering')
plt.xlabel('tahun')
plt.ylabel('odometer')
use_data.head(20)
```

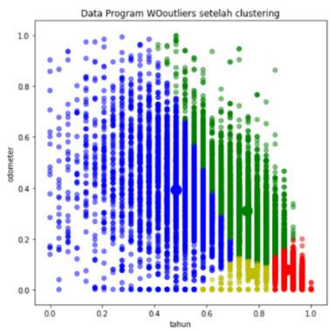
	0	1	keterangan
0	0.724138	0.191263	b
1	0.862069	0.000027	y
2	0.827586	0.022750	y
3	0.862069	0.000027	y
4	0.931034	0.211293	y
5	0.620690	0.361616	r
6	0.896552	0.287600	y
7	0.551724	0.269094	r
8	0.758621	0.680815	g
9	0.724138	0.386001	g
10	0.103448	0.508547	g
11	0.758621	0.471038	g
12	0.551724	0.407489	g
13	0.827586	0.221721	b
14	0.448276	0.710210	g
15	0.793103	0.476372	g
16	0.827586	0.384079	g
17	0.379310	0.539450	g
18	0.413793	0.557890	g
19	0.965517	0.063039	y

	0	1	keterangan
0	0.922330	0.025400	y
1	0.961165	0.000004	g
2	0.951456	0.003021	g
3	0.961165	0.000004	g
4	0.980583	0.028060	g
5	0.893204	0.048022	y
6	0.970874	0.038193	g
7	0.873786	0.035736	y
8	0.932039	0.090412	y
9	0.922330	0.051261	y
10	0.747573	0.067535	b
11	0.932039	0.062553	g
12	0.873786	0.054114	y
13	0.951456	0.029444	g
14	0.844660	0.094315	r
15	0.941748	0.063262	g
16	0.951456	0.051006	g
17	0.825243	0.071639	r
18	0.834951	0.074087	r
19	0.990291	0.008372	g

Sample Data Grap







Dari sample data diatas kita dapat menarik kesimpulan bahwa nilai label x dari suatu kelas pada grap dapat menentukan kelas mana yang terbaik, terbaik dalam artian bahwa kondisi mobil tersebut memiliki tahun yang paling baru dan odometer yang rendah sehingga memiliki kondisi yang bagus.

```
#Mengisi keterangan berdasarkan data yang dihasilkan
copy_centerx = []
for i in range(len(centerx)):
    copy_centerx.append(centerx[i])


centerx.sort()
centerx.reverse()
for i in range(len(centerx)):
    for j in range(len(centerx)):
        if (centerx[i] == copy_centerx[j]):
            copy_centerx[j] = i

use_data['keterangan'] = use_data['keterangan'].replace(["r","y","b","g"],copy_centerx)
for_classification = use_data['keterangan']
use_data['keterangan'] = use_data['keterangan'].replace([0,1,2,3],['Sangat Direkomendasikan','Direkomendasikan','Kurang Direkome
new_data['keterangan'] = use_data['keterangan']
new_data.head(20)
```

Maka dari itu saya membuat fungsi dimana akan menentukan keterangan yang diisi di setiap objek berdasarkan nilai label x dari setiap kelas. Dapat dilihat pada gambar diatas, terdapat array copy_centerx yang digunakan untuk menyimpan data dari centerx dengan menggunakan perulangan. Setelah itu data pada centerx akan diurutkan dari nilai yang terbesar ke terkecil. Setelah itu program akan melakukan perulangan, dan di dalam perulangan tersebut terdapat perulangan juga yang didalamnya terdapat kondisi bila nilai centerx dengan index i sama dengan nilai copy_centerx dengan index j, maka copy_centerx dengan index j akan diisi nilainya dengan index i dan seterusnya sampai perulangan selesai, sehingga akan menghasilkan urutan dari nilai label x pada array copy_centerx. Setelah itu kolom keterangan pada use data yang sebelumnya nilainya diubah dengan kode warna untuk menampilkan grafik, di ubah dengan data dari copy_centerx, terdapat juga array for_classification yang menyimpan data keterangan pada use_data yang digunakan nanti saat classification. Setelah itu kolom keterangan diubah lagi nilainya, bila nilai 0 akan diganti dengan “Sangat Direkomendasikan”, bila nilainya 1 akan diganti dengan “Direkomendasikan”, bila nilainya 2 akan diganti dengan “Kurang Direkomendasikan”, bilai nilainya 3 akan diganti dengan “Tidak Direkomendasikan” Setelah itu new_data yang sebelumnya dibuat sebelum scaling normalization ditambah kolom dengan nama keterangan dan diisi dengan kolom keterangan pada use_data, sehingga mendapatkan dataset baru.

|:

	year	odometer	keterangan
0	2012.0	63500.0	Kurang Direkomendasikan
1	2016.0	10.0	Sangat Direkomendasikan
2	2015.0	7554.0	Sangat Direkomendasikan
3	2016.0	10.0	Sangat Direkomendasikan
4	2018.0	70150.0	Direkomendasikan
5	2009.0	120057.0	Kurang Direkomendasikan
6	2017.0	95484.0	Direkomendasikan
7	2007.0	89340.0	Kurang Direkomendasikan
8	2013.0	226031.0	Tidak Direkomendasikan
9	2012.0	128153.0	Kurang Direkomendasikan
10	1994.0	168838.0	Tidak Direkomendasikan
11	2013.0	156385.0	Kurang Direkomendasikan
12	2007.0	135287.0	Tidak Direkomendasikan
13	2015.0	73612.0	Kurang Direkomendasikan
14	2004.0	235790.0	Tidak Direkomendasikan
15	2014.0	158156.0	Kurang Direkomendasikan
16	2015.0	127515.0	Kurang Direkomendasikan
17	2002.0	179098.0	Tidak Direkomendasikan
18	2003.0	185220.0	Tidak Direkomendasikan
19	2019.0	20930.0	Sangat Direkomendasikan

Program WOoutliers

	year	odometer	keterangan
0	2012.0	63500.0	Sangat Direkomendasikan
1	2016.0	10.0	Sangat Direkomendasikan
2	2015.0	7554.0	Sangat Direkomendasikan
3	2016.0	10.0	Sangat Direkomendasikan
4	2018.0	70150.0	Sangat Direkomendasikan
5	2009.0	120057.0	Kurang Direkomendasikan
6	2017.0	95484.0	Direkomendasikan
7	2007.0	89340.0	Kurang Direkomendasikan
8	2013.0	226031.0	Direkomendasikan
9	2012.0	128153.0	Direkomendasikan
10	1994.0	168838.0	Tidak Direkomendasikan
11	2013.0	156385.0	Direkomendasikan
12	2007.0	135287.0	Kurang Direkomendasikan
13	2015.0	73612.0	Sangat Direkomendasikan
14	2004.0	235790.0	Tidak Direkomendasikan
15	2014.0	158156.0	Direkomendasikan
16	2015.0	127515.0	Direkomendasikan
17	2002.0	179098.0	Tidak Direkomendasikan
18	2003.0	185220.0	Kurang Direkomendasikan
19	2019.0	20930.0	Sangat Direkomendasikan

Program Woutliers

Membuat File Dataset Baru

```
#Menyimpan dataset baru
new_data.to_csv('WOoutliers_Used_car.csv')
```

```
#Menyimpan dataset baru
new_data.to_csv('Woutliers_Used_car.csv')
```

Membuat file format csv sebagai dataset baru dari hasil clustering.

Classification

Pada program Woutliers dan WOoutliers melakukan classification dengan metode yang sama yaitu Linear SVM dan KNN

```
#Classification
data_classification = use_data
data_classification['keterangan'] = for_classification
data_classification = data_classification[[0,1,'keterangan']].replace(np.nan)
use_data.dropna(inplace=True)
```

```
#Membuat data train dan test
variable = np.array(data_classification[[0,1]])
label = np.array(data_classification['keterangan'])
Variable_train, Variable_test, Label_train, Label_test = train_test_split(variable, label, test_size=0.2)
```

Sebelum melakukan classification saya menyiapkan datanya terlebih dahulu, data_classification diisi dengan use_data namun pada kolom keterangan diisi dengan for_classification dimana agar data pada kolom keterangan bernilai 0,1,2,3 dan setelah itu mendrop baris pada data_classification bila ada baris yang berisi NaN. Setelah itu kita menentukan variable dan label dan setelah itu melakukan split data.

Melakukan Classification

```
#Linear SVM
csvm = svm.SVC(kernel='linear')
csvm.fit(Variable_train, Label_train)
csvm_pred = csvm.predict(Variable_test)
```

```
#KNN
cknn = KNeighborsClassifier(n_neighbors=5)
cknn.fit(Variable_train, Label_train)
cknn_pred = cknn.predict(Variable_test)
```

Classification digunakan untuk menentukan kelas dari suatu data yang baru, Linear SVM bekerja dengan cara membuat hyperline dan memaksimalkan jarak antar kelas, sedangkan KNN bekerja dengan cara melihat kecocokan antara data yang baru dengan data di sekitarnya.

Akurasi Classification

```
#Cek akurasi
print("Akurasi Linear SVM :",metrics.accuracy_score(Label_test,csvm_pred))
print("Akurasi KNN        :",metrics.accuracy_score(Label_test,cknn_pred))
```

```
Akurasi Linear SVM : 0.9786188962727536
Akurasi KNN        : 0.9994221323316961
```

Program WOoutliers

```
#Cek akurasi
print("Akurasi Linear SVM :",metrics.accuracy_score(Label_test,csvm_pred))
print("Akurasi KNN        :",metrics.accuracy_score(Label_test,cknn_pred))
```

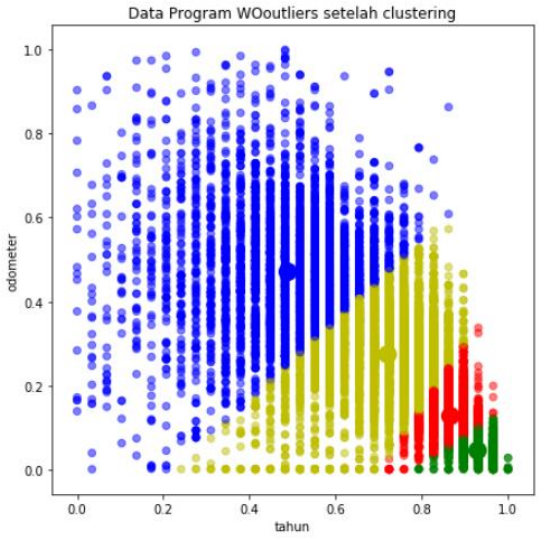
```
Akurasi Linear SVM : 0.9117814456459875
Akurasi KNN        : 0.99800796812749
```

Program Woutliers

Menggunakan metrics untuk mengecek akurasi dari Linear SVM dan KNN.

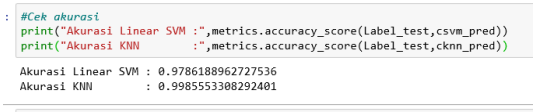
3. Eksperimen

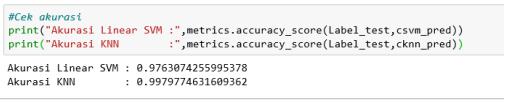
Eksperimen membandingkan hasil clustering dengan menghilangkan outliers dan tanpa menghilangkan outliers.

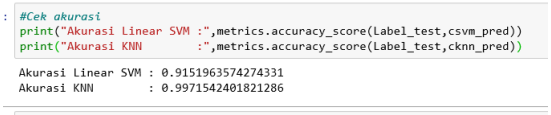


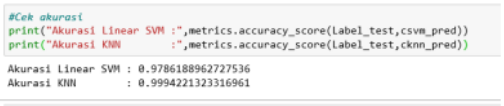
Dari gambar diatas kita dapat melihat pada data Program WOoutliers (menghilangkan outliers) pembagian kelas terlihat sangat jelas, merata dan bisa dilihat juga bahwa suatu kelas tidak mendominasi dan juga data yang dihasilkan tidak ambigu contohnya melihat pada label tahun, untuk data berwarna biar cakupannya perkiraan antara 0,0 – 0,7, untuk warna kuning perkiraan antara 0,2-0,8 untuk warna merah perkiraan antara 0,7 – 0,96 untuk warna hijau perkiraan antara 0,78 – 1,0. Sedangkan pada data Program Woutliners (tanpa menghilangkan outliners) kita dapat melihat kelas dengan warna merah pada label tahun sangat mendominasi dengan memiliki rentang nilai perkiraan antara 0,0 – 0,94 dan warna kuning, biru, dan hijau memiliki rentan nilai yang jauh lebih kecil, hal ini menyebabkan pembagian kelas yang tidak merata, dan terdapat data yang mendominasi dan ambigu seperti objek/titik merah yang berada pada x(0,2) dan y(0,4) merupakan kelas yang sama dengan objek/titik pada x(0,94) dan y (0,0) bila kita bayangkan ke data Used_Car dan melihat dari nilai pada label x(tahun), kelas warna merah memiliki nilai x yang terkecil sehingga termasuk pada keterangan “Tidak Direkomendasikan” pada new_data, sehingga mobil dengan odometer yang masih rendah

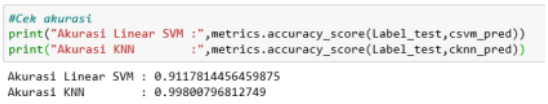
dan tahun yang baru memiliki keterangan “Tidak Direkomendasikan” yang mengakibatkan data yang dihasilkan tidak sesuai dengan tujuan.











Program WOoutliers

Program Woutliers

Pada gambar diatas kita bisa melihat sample hasil akurasi classification Linear SVM dan KNN, dari data diatas terlihat dari 2 program memiliki akurasi KNN yang hamper sama, sedangkan pada SVM memiliki hasil yang berbeda dimana akurasi pada Program Woutliers memiliki akurasi yang lebih kecil dibandingkan WOoutliers, karena pembeda dari 2 program ini adalah outliers maka berdasarkan data tersebut kita dapat mengetahui bahwa data tanpa menghilangkan outliers bila dijadikan data latih pada classification Linear SVM akan mengakibatkan akurasinya menurun, karena kita tahu bahwa Linear SVM bekerja dengan cara membuat hyperline dan memaksimalkan jarak antar kelas, sedangkan pada eksperimen clustering diketahui bahwa terdapat jarak antar kelas yang mendominasi dan ambigu, sehingga akan sangat berdampak pada classification Linear SVM.

4. KESIMPULAN

Dari eksperimen yang telah dilakukan, dapat disimpulkan bahwa suatu dataset yang akan digunakan pada clustering dan classification bila tidak melakukan penghapusan outliers pada data tersebut akan mengakibatkan data yang ambigu yang mengakibatkan hasil dari kelas yang dibuat pada clustering tidak sesuai dengan tujuan, dan bila dataset baru tersebut dijadikan data train pada classification Linear SVM akan menghasilkan akurasi yang tidak maksimal, sedangkan bila kita melakukan penghapusan outliers pada dataset yang akan digunakan untuk clustering dan classification, data kelas yang dihasilkan setelah clustering sesuai dengan tujuan dan bila dataset baru tersebut digunakan sebagai data train pada classification Linear SVM maka akan menghasilkan akurasi yang lebih baik dibandingkan dengan dataset yang sebelumnya tanpa penghapusan outliers.