```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder # used for encoding categori
from sklearn.model_selection import train_test_split # used for splitting training and tes
from sklearn.preprocessing import MinMaxScaler # used for feature scaling
from sklearn.decomposition import PCA #used for feature extraction
```

Data Preparation

Mounting to my Drive, dataset awal terdapat 20000 tuple dan 26 atribut.

```
df = pd.read_csv("drive/My Drive/ProjectMalin/used_cars.csv")
```

Data Preprocessing

1. Sebelum memulai semuanya, penting untuk memahami dataset.
2. Lakukan analisis pada atribut dalam dataset. Berhubung jumlah dimensi dataset awal tidak terlalu banyak maka analisa dapat dilakukan secara manual. Pengecekan dapat dilakukan ddengan memanggil isnull() dari dataframe atau melihat 5 tuple teratas.

```
df.isnull().sum()
```

```
Unnamed: 0         0
region             0
price              0
year              12
manufacturer     705
model            265
condition       9152
cylinders       7085
fuel              73
odometer        2389
title_status     110
transmission     190
drive           4642
type            3659
paint_color     5514
dtype: int64
```

3. [Data Cleaning] Setelah mengecek nilai null, dapat dilihat atribut dengan missing value lebih dari 50% total tuple. Mada dari itu kolom 'size' dan 'county' di-drop.
4. [Data Cleaning] Dataset ini tidak memiliki label, namun dari data dan link pada url dalam dataset dapat dilihat bahwa dataset ini untuk jual beli mobil. Asumsi yang dilakukan

adalah disini tidak akan dilakukan pemrosesan teks, link url, dan gambar. Maka dari itu kolom 'url', 'region_url', 'image_url', 'description' di-drop.

5. [Data Cleaning] Selanjutnya perlu diasumsikan dari atribut tersisa, mana saja yang tidak dibutuhkan dalam pembelian mobil. Maka dari itu kolom 'vin' di-drop.
6. [Data Cleaning] Ada lebih dari 1 atribut yang menyangkut tempat yaitu: 'region', 'state', 'lat', 'long'. Namun ada dikerucutkan menjadi satu atribut saja yaitu 'region' karena orang-orang cenderung membeli mobil di daerahnya dengan pemahaman lebih dekat dibanding kota lain dalam satu negara bagian (state). Long dan lat terasa tidak dibutuhkan karena kita tidak mengolah data spasial. Maka atribut 'state', 'long', 'lat' di-drop.
7. [Data Cleaning] Drop atribut unnamed karena sama dengan index. Atribut ini biasanya auto-generated sama mengubah dataframe menjadi csv. Selanjutnya, tahap ini akan sering dilakukan.
8. [Data Cleaning] Drop atribut 'id' karena menurut saya tidak memiliki dampak signifikan.

```
#3 Drop atribut size dan county
df.drop(df.columns[df.columns.str.contains('county',case = False)],axis = 1, inplace = Tru
df.drop(df.columns[df.columns.str.contains('size',case = False)],axis = 1, inplace = True)

#4 Drop url, region_url, image_url, dan description
df.drop(df.columns[df.columns.str.contains('url',case = False)],axis = 1, inplace = True)
df.drop(df.columns[df.columns.str.contains('region_url',case = False)],axis = 1, inplace =
df.drop(df.columns[df.columns.str.contains('image_url',case = False)],axis = 1, inplace =
df.drop(df.columns[df.columns.str.contains('description',case = False)],axis = 1, inplace

#5 Drop vin
df.drop(df.columns[df.columns.str.contains('vin',case = False)],axis = 1, inplace = True)

#6 Drop state, lat, long
df.drop(df.columns[df.columns.str.contains('state',case = False)],axis = 1, inplace = True
df.drop(df.columns[df.columns.str.contains('lat',case = False)],axis = 1, inplace = True)
df.drop(df.columns[df.columns.str.contains('long',case = False)],axis = 1, inplace = True)

#7 Drop unnamed
df.drop(df.filter(regex="Unnamed"),axis=1, inplace=True)

#8 Drop ID
df.drop(df.columns[df.columns.str.contains('id',case = False)],axis = 1, inplace = True)
```

9. [Data Cleaning] Menghapus rows yang memiliki missing value pada atribut year.

```
# Drop rows if all the selected columns contains NaN only i.e.
df = df.dropna(axis = 0, subset=['year'])
```

10. [Data Cleaning] Menghapus row yang memiliki missing value >= 6

```
df = df[df.isnull().sum(axis=1) < 6]
```

11. [Data Cleaning] Menghapus rows dengan price < 1500

```
df.drop(df[df.price < 1500 ].index, inplace=True)
```

12. [Data Cleaning - Missing Value] Tentukan tipe data: int = [price, odometer], nominal = [region, manufacturer, model, paint color, fuel], ordinal = [year, condition, cylinders, title_status, transmission, drive, type]. Dari tiap tipe data, tentukan yang masih memiliki missing value. Handle missing value year dengan di drop, odometer dengan rata-rata, dan sisanya dengan modus.

```
df.isnull().sum()
```

```
#Isi missing value odometer dengan mean
df['odometer'] = df['odometer'].fillna(df['odometer'].mean())
```

Condition suatu mobil penting untuk menentukan mobil itu diminati atau tidak. Maka dari itu kita lihat distribusi datanya terlebih dahulu sebelum mengisi dengan modus.

```
#Mengisi missing value condition
df['condition'].value_counts()
#Rank: 1. salvage, 2. fair, 3. good,
#4.excellent, 5. like new 6. new
```

Dengan asumsi atribut condition adalah atribut ordinal dengan susunan berikut : salvage < fair < good < excellent < like new < new. Maka saya memutuskan untuk tetap mengisi condition dengan modus, karena modusnya merupakan nilai tengah

```
a_kategori = ['manufacturer','model','condition','cylinders','fuel','title_status','transm
for j in a_kategori:
  df[j] = df[j].fillna(df[j].mode().values[0])
```

Cek jika masih ada missing value

```
df.isnull().sum()
```

13. [Data Cleaning - Outlier] Mengatasi outlier pada atribut 'price', 'year', dan 'odometer'. Atribut 'year' memiliki tipe data float64, maka masih bisa dilakukan deteksi outlier. Selain itu deteksi outlier juga perlu untuk mendeteksi non-sense data seperti mobil yang tahun pembuatannya di bawah tahun ditemukannya mobil (jika terdapat tuple seperti ini).

```
#import seaborn untuk melihat persebaran price, year, dan odometer dalam boxplot
import seaborn as sns
```

```
sns.boxplot(x=df['year'])
```

```
sns.boxplot(x=df['price'])
```

```
sns.boxplot(x=df['odometer'])
```

Boxplot yang menampilkan persebaran data price dan odometer tidak dapat memvisualisasikan dengan baik karena persebaran data yang terlalu abstrak, maka dari itu didefinisikan fungsi remove outlier untuk diaplikasikan pada ketiga atribut ini.

```
def rmvOutlier(df,column):
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    interq = q3-q1
    batasBawah = q1-1.5*interq
    batasAtas = q3+1.5*interq
    df_out =  df.loc[(df[column] > batasBawah) & (df[column] < batasAtas)]
    return df_out
```

```
#Remove outlier dilakukan dua kali, yang pertama untuk handling outlier dan yang kedua unt
```

```
#Remove outlier dilakukan dua kali, yang pertama untuk handling outlier dan yang kedua unt
df = rmvOutlier(df, 'year')
df = rmvOutlier(df, 'year')
```

Menampilkan boxplot year setelah outlier dihapus

```
sns.boxplot(x=df['year'])
```

```
df = rmvOutlier(df, 'price')
df = rmvOutlier(df, 'price')
```

```
sns.boxplot(x=df['price'])
```

```
df = rmvOutlier(df, 'odometer')
df = rmvOutlier(df, 'odometer')
```

```
sns.boxplot(x=df['odometer'])
```

```
#Mengubah data untuk nantinya ditempelkan atribut kelas
df.to_csv("drive/My Drive/ProjectMalin/Preprocessing1/clean_data.csv", index=False)
```

```
ds = df
```

14. [Data Cleaning - Mengubah Nominal dan Ordinal Menjadi Numerik] Data ordinal diubah dengan replacing dan data nominal diubah menggunakan encoder. Sebelumnya lakukan perhitungan value count dari tiap kolom. Misalnya title_status, sudah tidak ada mobil yang parts only.

```
#Mengganti value data ordinal dengan replacing dari tingkat rendah ke tinggi berdasar valu
df['condition'] = df['condition'].replace({'salvage': 1, 'fair': 2, 'good': 3, 'excellent'

df['cylinders'] = df['cylinders'].replace({'other': 1, '3 cylinders': 2, '4 cylinders': 3,
                                           '6 cylinders':5, '8 cylinders':6, '10 cylinders
#sudah tidak ada title_status == 'parts only'
df['title_status'] = df['title_status'].replace({'missing': 1, 'salvage':2, 'rebuilt': 3,

df['transmission'] = df['transmission'].replace({'other':1, 'manual':2, 'automatic':3})

df['drive'] = df['drive'].replace({'fwd':1, 'rwd':2, '4wd':3})

df['type'] = df['type'].replace({'other':1, 'hatchback':2, 'wagon':3, 'sedan':4, 'mini-var
                                 'SUV':8, 'offroad':9, 'truck':10, 'bus':11, 'coupe':12, '
```

```
kategori = ['region','manufacturer','model','fuel','paint_color']
for i in kategori:
  num = LabelEncoder()
  df[i] = num.fit_transform(df[i].astype('str'))
```

Cek tipe data tiap atribut dan ubah ke numerik

```
df.dtypes
```

cast odometer dan year menjadi int64

```
df['odometer'] = df['odometer'].astype(np.int64)
```

```
df['year'] = df['year'].astype('int64')
```

14. [Data Cleaning - Scaling] Scaling menggunakan minmax.

```
#Scaling menggunakan minmax
mms = MinMaxScaler()
df_new = mms.fit_transform(df)
df_new = pd.DataFrame(df_new,columns=df.columns)
```

```
df = df_new
```

```
df.to_csv("drive/My Drive/ProjectMalin/Preprocessing1/used_cars_after_scaling.csv", index
```

```
df_alrscal = df
```

Range data tiap kolom sudah sama.

Reduksi Dimensi

Saat ini saya menggunakan PCA dengan library dengan n components 0.3 agar menghasilkan 2 atribut.

```
pca = PCA(n_components=0.30, whiten=True)

#Mencari Hasil PCA, ditransform ke 0.3 supaya tersisa 2 atribut untuk clustering
df_pca = pca.fit_transform(df)
```

```
df_pca = pd.DataFrame(df_pca)
```

```
df_pca.count()
```

```
fig = plt.figure(figsize=(5,5))
plt.scatter(df_pca[0], df_pca[1], color = 'lightsalmon')
plt.xlim(-2.5,2.5) #limit disesuaikan dengan persebaran data agar terlihat jelas
plt.ylim(-2.5,2.5)
plt.show()
```

Clustering

Data dengan 2 atribut dari hasil PCA digunakan untuk membangun model K-Means clustering 3 kelas. Hal ini dikarenakan K-Means clustering requires 2 atribut.

```
a = df_pca.iloc[:, 0] #matrix baru, pake semua baris, tapi kecuali kolom terakhir
b = df_pca.iloc[:, 1] #matrix baru, pake semua baris, tapi kolom kelas saja

c = {'x':a, 'y':b}

df_pca = pd.DataFrame(c)

np.random.seed(200)
k = 3
```

```
centroids = {
    i+1: [np.random.randint(-1.5, 2), np.random.randint(-1.6,2.5)]
    for i in range(k)
}

fig = plt.figure(figsize=(5,5))
plt.scatter(df_pca['x'], df_pca['y'], color='lightsalmon')
colmap = {1: 'r', 2: 'g', 3: 'b'}
for i in centroids.keys():
    plt.scatter(*centroids[i], color = colmap[i])
plt.xlim(-1.5,2.2)
plt.ylim(-2,2.5)
plt.show()
```

```
def assignment(df, centroids):
    for i in centroids.keys():
        df['distance_from_{}'.format(i)] = (
            np.sqrt(
                (df['x'] - centroids[i][0]) ** 2
                + (df['y'] - centroids[i][1]) ** 2
            )
        )
    centroid_distance_cols = ['distance_from_{}'.format(i) for i in centroids.keys()]
    df['closet'] = df.loc[:, centroid_distance_cols].idxmin(axis = 1)
    df['closet'] = df['closet'].map(lambda x : int(x.lstrip('distance_from_')))
    df['color'] = df['closet'].map(lambda x : colmap[x])
    return df

df_pca = assignment(df_pca,centroids)
print(df_pca.head())

fig = plt.figure(figsize=(5,5))
plt.scatter(df_pca['x'], df_pca['y'], color = df_pca['color'], alpha = 0.5, edgecolor = 'k
for i in centroids.keys():
    plt.scatter(*centroids[i], color = colmap[i])
plt.xlim(-1.5,2.2)
plt.ylim(-2,2.5)
```

```
plt.show()
```

```
import copy

old_centroids = copy.deepcopy(centroids)
def update(k):
  for i in centroids.keys():
    centroids[i][0] = np.mean(df_pca[df_pca['closet'] == i]['x'])
    centroids[i][1] = np.mean(df_pca[df_pca['closet'] == i]['y'])
  return k

centroids = update(centroids)

fig = plt.figure(figsize=(5,5))
ax = plt.axes()
plt.scatter(df_pca['x'], df_pca['y'], color = df_pca['color'], alpha = 0.5, edgecolor = 'k
for i in centroids.keys():
    plt.scatter(*centroids[i], color = colmap[i])
plt.xlim(-1.5,2.2)
plt.ylim(-2,2.5)

for i in old_centroids.keys():
  old_x = old_centroids[i][0]
  old_y = old_centroids[i][1]
  dx = (centroids[i][0] - old_centroids[i][0]) * 0.75
  dy = (centroids[i][1] - old_centroids[i][1]) * 0.75
  ax.arrow(old_x, old_y, dx, dy, head_width = 2, head_length=3, fc = colmap[i], ec= colmap

plt.show()
```

```
#Ulang assignment
df_pca = assignment(df_pca,centroids)

#Plot results
fig = plt.figure(figsize=(5,5))
plt.scatter(df_pca['x'], df_pca['y'], color = df_pca['color'], alpha = 0.5, edgecolor = 'k
for i in centroids.keys():
    plt.scatter(*centroids[i], color = colmap[i])
plt.xlim(-1.5,2.2)
plt.ylim(-2,2.5)
plt.show()
```

```
while True:
  closest_centroids = df_pca['closet'].copy(deep=True)
  centroids = update(centroids)
  df_pca = assignment(df_pca, centroids)
  if closest_centroids.equals(df_pca['closet']):
    break

#Membuat plot hasil
```

```
#membuat plot hasil
fig = plt.figure(figsize=(5,5))
plt.scatter(df_pca['x'], df_pca['y'], color = df_pca['color'], alpha = 0.5, edgecolor = 'k
for i in centroids.keys():
    plt.scatter(*centroids[i], color = colmap[i])
plt.xlim(-1.5,2.2)
plt.ylim(-2,2.5)
plt.show()
```

```
df_pca.head()
```

Atribut closet adalah atribut kelas hasil Clustering. Masukkan atribut ini ke dataframe setelah scaling untuk melakukan klasifikasi.

```
df['class'] = df_pca['closet']
```

Classification

Dalam pengerjaan project ini dilakukan klasifikasi dengan menggunakan model Naive Bayes, KNN, dan Random Forest. Data masukan adalah data sesudah scaling karena kedua model ini membaca masukan numerikal.

```
#Libraries for Gaussian NB and KNN
import collections
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RepeatedKFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

Split dataset dilakukan satu kali sekaligus untuk model Naive Bayes dan KNN, maka data train dan data test pada kedua model adalah sama. Dengan ini diharapkan bisa secara real melihat perbandingan performansi kedua model ini.

```
#Split dataset untuk klasifikasi hanya dilakukan satu kali, maka data test NB sama dengan
X = df.iloc[:, :-1].values #matrix baru, pake semua baris, tapi kecuali kolom terakhir
y = df.iloc[:, 14].values #matrix baru, pake semua baris, tapi kolom kelas saja
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)


gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)


print(y_pred)
print('\n')
print("Akurasi Model Naive Bayes:",metrics.accuracy_score(y_test, y_pred))
print('\n')
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
print('\n')
print("Classification Report for Naive Bayes")
print(classification_report(y_test,y_pred))
```

```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)


print(y_pred)
print('\n')
print("Akurasi Model KNN:",metrics.accuracy_score(y_test, y_pred))
print('\n')
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
print('\n')
print("Classification Report for KNN")
print(classification_report(y_test,y_pred))
```

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)


print(y_pred)
print('\n')
print("Akurasi Model Random Forest:",metrics.accuracy_score(y_test, y_pred))
print('\n')
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
print('\n')
print("Classification Report for Random Forest")
```

```
print(classification_report(y_test,y_pred))
```

Merapihkan dataframe untuk dijadikan file csv

```
ds['class'] = df['class']
```

```
ds.to_csv("drive/My Drive/ProjectMalin/Preprocessing1/clean_data_with_label.csv", index =
```

```
df_alrscal.to_csv("drive/My Drive/ProjectMalin/Preprocessing1/after_scaling_with_label.csv
```

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder # used for encoding categori
from sklearn.model_selection import train_test_split # used for splitting training and tes
from sklearn.preprocessing import MinMaxScaler # used for feature scaling
from sklearn.decomposition import PCA #used for feature extraction
```

Data Preparation

Mounting to my Drive, dataset awal terdapat 20000 tuple dan 26 atribut.

```
df = pd.read_csv("drive/My Drive/ProjectMalin/used_cars.csv")
```

Data Preprocessing

1. Sebelum memulai semuanya, penting untuk memahami dataset.
2. Lakukan analisis pada atribut dalam dataset. Berhubung jumlah dimensi dataset awal tidak terlalu banyak maka analisa dapat dilakukan secara manual. Pengecekan dapat dilakukan ddengan memanggil isnull() dari dataframe atau melihat 5 tuple teratas.

```
df.isnull().sum()
```

```
Unnamed: 0         0
region             0
price              0
year              12
manufacturer     705
model            265
condition       9152
cylinders       7085
fuel              73
odometer        2389
title_status     110
transmission     190
drive           4642
type            3659
paint_color     5514
dtype: int64
```

3. [Data Cleaning] Setelah mengecek nilai null, dapat dilihat atribut dengan missing value lebih dari 50% total tuple. Mada dari itu kolom 'size' dan 'county' di-drop.
4. [Data Cleaning] Dataset ini tidak memiliki label, namun dari data dan link pada url dalam dataset dapat dilihat bahwa dataset ini untuk jual beli mobil. Asumsi yang dilakukan

adalah disini tidak akan dilakukan pemrosesan teks, link url, dan gambar. Maka dari itu kolom 'url', 'region_url', 'image_url', 'description' di-drop.

5. [Data Cleaning] Selanjutnya perlu diasumsikan dari atribut tersisa, mana saja yang tidak dibutuhkan dalam pembelian mobil. Maka dari itu kolom 'vin' di-drop.
6. [Data Cleaning] Ada lebih dari 1 atribut yang menyangkut tempat yaitu: 'region', 'state', 'lat', 'long'. Namun ada dikerucutkan menjadi satu atribut saja yaitu 'region' karena orang-orang cenderung membeli mobil di daerahnya dengan pemahaman lebih dekat dibanding kota lain dalam satu negara bagian (state). Long dan lat terasa tidak dibutuhkan karena kita tidak mengolah data spasial. Maka atribut 'state', 'long', 'lat' di-drop.
7. [Data Cleaning] Drop atribut unnamed karena sama dengan index. Atribut ini biasanya auto-generated sama mengubah dataframe menjadi csv. Selanjutnya, tahap ini akan sering dilakukan.

```
#3 Drop atribut size dan county
df.drop(df.columns[df.columns.str.contains('county',case = False)],axis = 1, inplace = Tru
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

df.drop(df.columns[df.columns.str.contains('size',case = False)],axis = 1, inplace = True)
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

#4 Drop url, region_url, image_url, dan description
df.drop(df.columns[df.columns.str.contains('url',case = False)],axis = 1, inplace = True)
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

df.drop(df.columns[df.columns.str.contains('region_url',case = False)],axis = 1, inplace =
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

df.drop(df.columns[df.columns.str.contains('image_url',case = False)],axis = 1, inplace =
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

df.drop(df.columns[df.columns.str.contains('description',case = False)],axis = 1, inplace
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

#5 Drop vin
df.drop(df.columns[df.columns.str.contains('vin',case = False)],axis = 1, inplace = True)
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

#6 Drop state, lat, long
df.drop(df.columns[df.columns.str.contains('state',case = False)],axis = 1, inplace = True
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

df.drop(df.columns[df.columns.str.contains('lat',case = False)],axis = 1, inplace = True)
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

df.drop(df.columns[df.columns.str.contains('long',case = False)],axis = 1, inplace = True)
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")

#7 Drop unnamed
df.drop(df.filter(regex="Unnamed"),axis=1, inplace=True)
df.to_csv('drive/My Drive/ProjectMalin/used_cars.csv', index=False, sep=',')
```

8. [Data Cleaning] Drop atribut 'id' karena menurut saya tidak memiliki dampak signifikan.

```
df.drop(df.columns[df.columns.str.contains('id',case = False)],axis = 1, inplace = True)
df.to_csv("drive/My Drive/ProjectMalin/used_cars.csv")
```

9. [Data Cleaning - Missing Value] Analisa missing value pada setiap atribut
10. [Data Cleaning - Missing Value] Tentukan tipe data: int = [price, odometer], nominal = [region, manufacturer, model, paint color, fuel], ordinal = [year, condition, cylinders, title_status, transmission, drive, type]
11. [Data Cleaning - Missing Value] Dari tiap tipe data, tentukan yang masih memiliki missing value. Handle missing value odometer dengan rata-rata, dan sisanya dengan modus.

```
#Isi missing value odometer dengan mean
df['odometer'] = df['odometer'].fillna(df['odometer'].mean())
```

```
a_kategori = ['year','manufacturer','model','condition','cylinders','fuel','title_status',
for j in a_kategori:
  df[j] = df[j].fillna(df[j].mode().values[0])
```

Cek jika masih ada missing value

```
df.isnull().sum()
```

12. [Data Cleaning - Outlier] Mengatasi outlier pada atribut 'price', 'year', dan 'odometer'. Atribut 'year' memiliki tipe data float64, maka masih bisa dilakukan deteksi outlier. Selain itu deteksi outlier juga perlu untuk mendeteksi non-sense data seperti mobil yang tahun pembuatannya di bawah tahun ditemukannya mobil (jika terdapat tuple seperti ini).

```
#import seaborn untuk melihat persebaran price, year, dan odometer dalam boxplot
import seaborn as sns
```

```
import seaborn as sns

sns.boxplot(x=df['year'])
```

```
sns.boxplot(x=df['price'])
```

```
sns.boxplot(x=df['odometer'])
```

Boxplot yang menampilkan persebaran data price dan odometer tidak dapat memvisualisasikan dengan baik karena persebaran data yang terlalu abstrak, maka dari itu didefinisikan fungsi remove outlier untuk diaplikasikan pada ketiga atribut ini.

```
def rmvOutlier(df,column):
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    interq = q3-q1
    batasBawah = q1-1.5*interq
    batasAtas = q3+1.5*interq
    df_out =  df.loc[(df[column] > batasBawah) & (df[column] < batasAtas)]
    return df_out


#Remove outlier dilakukan dua kali, yang pertama untuk handling outlier dan yang kedua unt
df = rmvOutlier(df, 'year')
df = rmvOutlier(df, 'year')
```

Menampilkan boxplot year setelah outlier dihapus

```
sns.boxplot(x=df['year'])
```

```
df = rmvOutlier(df, 'price')
df = rmvOutlier(df, 'price')
```

```
sns.boxplot(x=df['price'])
```

```
df = rmvOutlier(df, 'odometer')
df = rmvOutlier(df, 'odometer')


sns.boxplot(x=df['odometer'])
```

```
#Mengubah data untuk nantinya ditempelkan atribut kelas
df.to_csv("drive/My Drive/ProjectMalin/Preprocessing2/clean_data.csv")


ds = df
```

13. [Data Cleaning - Mengubah Nominal dan Ordinal Menjadi Numerik] Data ordinal diubah dengan replacing dan data nominal diubah menggunakan encoder.

```
#Mengganti value data ordinal dengan replacing dari tingkat rendah ke tinggi berdasar valu
df['condition'] = df['condition'].replace({'salvage': 1, 'fair': 2, 'good': 3, 'excellent'

df['cylinders'] = df['cylinders'].replace({'other': 1, '3 cylinders': 2, '4 cylinders': 3,
                                           '6 cylinders':5, '8 cylinders':6, '10 cylinders

df['title_status'] = df['title_status'].replace({'parts only': 1, 'missing' : 2, 'salvage'

df['transmission'] = df['transmission'].replace({'other':1, 'manual':2, 'automatic':3})

df['drive'] = df['drive'].replace({'fwd':1, 'rwd':2, '4wd':3})
```

```
df['type'] = df['type'].replace({'other':1, 'hatchback':2, 'wagon':3, 'sedan':4, 'mini-var
                                 'SUV':8, 'offroad':9, 'truck':10, 'bus':11, 'coupe':12, '
```

```
kategori = ['region','manufacturer','model','fuel','paint_color']
for i in kategori:
  num = LabelEncoder()
  df[i] = num.fit_transform(df[i].astype('str'))
```

Cek tipe data tiap atribut dan ubah ke numerik

```
df.dtypes
```

cast odometer dan year menjadi int64

```
df['odometer'] = df['odometer'].astype(np.int64)
```

```
df['year'] = df['year'].astype('int64')
```

14. [Data Cleaning - Scaling] Scaling menggunakan minmax.

```
#Scaling menggunakan minmax
mms = MinMaxScaler()
df_new = mms.fit_transform(df)
df_new = pd.DataFrame(df_new,columns=df.columns)
```

```
df = df_new
```

```
df.to_csv("drive/My Drive/ProjectMalin/Preprocessing2/used_cars_after_scaling.csv")
```

```
df_alrscal = df
```

Range data tiap kolom sudah sama.

Reduksi Dimensi

Saat ini saya menggunakan PCA dengan library dengan n components 0.3 agar menghasilkan 2 atribut.

```
pca = PCA(n_components=0.30, whiten=True)

#Mencari Hasil PCA, ditransform ke 0.3 supaya tersisa 2 atribut untuk clustering
df_pca = pca.fit_transform(df)



df_pca = pd.DataFrame(df_pca)


df_pca.count()
```

```
fig = plt.figure(figsize=(5,5))
plt.scatter(df_pca[0], df_pca[1], color = 'lightsalmon')
plt.xlim(-2.5,2.5) #limit disesuaikan dengan persebaran data agar terlihat jelas
plt.ylim(-2.5,2.5)
plt.show()
```

Clustering

Data dengan 2 atribut dari hasil PCA digunakan untuk membangun model K-Means clustering 3

```
a = df_pca.iloc[:, 0] #matrix baru, pake semua baris, tapi kecuali kolom terakhir
b = df_pca.iloc[:, 1] #matrix baru, pake semua baris, tapi kolom kelas saja

c = {'x':a, 'y':b}

df_pca = pd.DataFrame(c)

np.random.seed(200)
k = 3

centroids = {
    i+1: [np.random.randint(-1.5, 2), np.random.randint(-1.6,2.5)]
    for i in range(k)
}

fig = plt.figure(figsize=(5,5))
plt.scatter(df_pca['x'], df_pca['y'], color='lightsalmon')
colmap = {1: 'r', 2: 'g', 3: 'b'}
for i in centroids.keys():
    plt.scatter(*centroids[i], color = colmap[i])
plt.xlim(-1.5,2.2)
plt.ylim(-2,2.5)
plt.show()
```

```
def assignment(df, centroids):
    for i in centroids.keys():
        df['distance_from_{}'.format(i)] = (
            np.sqrt(
                (df['x'] - centroids[i][0]) ** 2
                + (df['y'] - centroids[i][1]) ** 2
            )
        )
```

```
        centroid_distance_cols = ['distance_from_{}'.format(i) for i in centroids.keys()]
        df['closet'] = df.loc[:, centroid_distance_cols].idxmin(axis = 1)
        df['closet'] = df['closet'].map(lambda x : int(x.lstrip('distance_from_')))
        df['color'] = df['closet'].map(lambda x : colmap[x])
        return df

    df_pca = assignment(df_pca,centroids)
    print(df_pca.head())

    fig = plt.figure(figsize=(5,5))
    plt.scatter(df_pca['x'], df_pca['y'], color = df_pca['color'], alpha = 0.5, edgecolor = 'k
    for i in centroids.keys():
        plt.scatter(*centroids[i], color = colmap[i])
    plt.xlim(-1.5,2.2)
    plt.ylim(-2,2.5)
    plt.show()
```

```
    import copy

    old_centroids = copy.deepcopy(centroids)
    def update(k):
      for i in centroids.keys():
        centroids[i][0] = np.mean(df_pca[df_pca['closet'] == i]['x'])
        centroids[i][1] = np.mean(df_pca[df_pca['closet'] == i]['y'])
      return k

    centroids = update(centroids)

    fig = plt.figure(figsize=(5,5))
    ax = plt.axes()
```

```
    plt.scatter(df_pca['x'], df_pca['y'], color = df_pca['color'], alpha = 0.5, edgecolor = 'k
    for i in centroids.keys():
        plt.scatter(*centroids[i], color = colmap[i])
    plt.xlim(-1.5,2.2)
    plt.ylim(-2,2.5)

    for i in old_centroids.keys():
      old_x = old_centroids[i][0]
      old_y = old_centroids[i][1]
      dx = (centroids[i][0] - old_centroids[i][0]) * 0.75
      dy = (centroids[i][1] - old_centroids[i][1]) * 0.75
      ax.arrow(old_x, old_y, dx, dy, head_width = 2, head_length=3, fc = colmap[i], ec= colmap

    plt.show()
```

```
    #Ulang assignment
    df_pca = assignment(df_pca,centroids)

    #Plot results
    fig = plt.figure(figsize=(5,5))
    plt.scatter(df_pca['x'], df_pca['y'], color = df_pca['color'], alpha = 0.5, edgecolor = 'k
    for i in centroids.keys():
        plt.scatter(*centroids[i], color = colmap[i])
    plt.xlim(-1.5,2.2)
    plt.ylim(-2,2.5)
    plt.show()
```

```
while True:
  closest_centroids = df_pca['closet'].copy(deep=True)
  centroids = update(centroids)
  df_pca = assignment(df_pca, centroids)
  if closest_centroids.equals(df_pca['closet']):
    break

#Membuat plot hasil
fig = plt.figure(figsize=(5,5))
plt.scatter(df_pca['x'], df_pca['y'], color = df_pca['color'], alpha = 0.5, edgecolor = 'k
for i in centroids.keys():
    plt.scatter(*centroids[i], color = colmap[i])
plt.xlim(-1.5,2.2)
plt.ylim(-2,2.5)
plt.show()
```

```
df_pca.head()
```

Atribut closet adalah atribut kelas hasil Clustering. Masukkan atribut ini ke dataframe setelah scaling untuk melakukan klasifikasi.

```
df['class'] = df_pca['closet']
```

```
df.head()
```

Classification

Dalam pengerjaan project ini dilakukan klasifikasi dengan menggunakan model Naive Bayes, KNN, dan Random Forest. Data masukan adalah data sesudah scaling karena kedua model ini membaca masukan numerikal.

```
#Libraries for Gaussian NB and KNN
import collections
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RepeatedKFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

Split dataset dilakukan satu kali sekaligus untuk model Naive Bayes dan KNN, maka data train dan data test pada kedua model adalah sama. Dengan ini diharapkan bisa secara real melihat perbandingan performansi kedua model ini.

```
#Split dataset untuk klasifikasi hanya dilakukan satu kali, maka data test NB sama dengan
X = df.iloc[:, :-1].values #matrix baru, pake semua baris, tapi kecuali kolom terakhir
y = df.iloc[:, 14].values #matrix baru, pake semua baris, tapi kolom kelas saja
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)


print(y_pred)
print('\n')
print("Akurasi Model Naive Bayes:",metrics.accuracy_score(y_test, y_pred))
print('\n')
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
print('\n')
print("Classification Report for Naive Bayes")
print(classification_report(y_test,y_pred))
```

```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)


print(y_pred)
print('\n')
print("Akurasi Model KNN:",metrics.accuracy_score(y_test, y_pred))
print('\n')
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
print('\n')
print("Classification Report for KNN")
print(classification_report(y_test,y_pred))
```

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)


print(y_pred)
print('\n')
print("Akurasi Model Random Forest:",metrics.accuracy_score(y_test, y_pred))
print('\n')
print('Confusion Matrix')
print(confusion_matrix(y_test, y_pred))
print('\n')
print("Classification Report for Random Forest")
print(classification_report(y_test,y_pred))
```

Merapihkan dataframe untuk dijadikan file csv

```
ds['class'] = df['class']
```

```
ds.to_csv("drive/My Drive/ProjectMalin/Preprocessing2/clean_data_with_label.csv", index =
```

```
df_alrscal.to_csv("drive/My Drive/ProjectMalin/Preprocessing2/after_scaling_with_label.csv
```

Implementasi *Unsupervised* dan *Supervised Learning* pada Iklan Penjualan Mobil Bekas

Anggitha Yohana Clara

Fakultas Informatika, Telkom University
Jalan Telekomunikasi no. 1, Kabupaten Bandung, Indonesia
anggithayohana@student.telkomuniversity.ac.id

I. PENDAHULUAN

Machine Learning merupakan cabang ilmu pengetahuan yang membuat komputer mampu belajar dari data, tanpa pemrograman secara eksplisit. *Machine Learning* menggunakan teknik statistika dalam proses pembelajarannya. *Machine Learning* menggunakan algoritma yang secara berulang-ulang belajar dari data. Berdasarkan teknik pembelajaran yang dilakukan, algoritma *Machine Learning* dapat dikelompokkan menjadi *unsupervised learning*, *supervised learning*, dan *reinforcement learning*. Penelitian ini akan menggunakan teknik *unsupervised learning* dan *supervised learning*.

Data awal yang dilakukan dalam proses *Machine Learning* berasal dari data aktual. Faktanya, dalam satu basis data biasanya terdapat banyak derau, pencilan, dan masalah lainnya. Masalah dalam basis data ini dapat diatasi dengan membersihkan data yang akan digunakan. Tahap pembersihan data ini disebut *preprocessing*. Ide dari tahap ini bukan hanya membersihkan tapi juga menyiapkan data agar dapat diolah.

Reduksi dimensi adalah salah satu teknik pengurangan dimensi. Reduksi dimensi dapat dilakukan dengan seleksi fitur ataupun ekstraksi fitur. Data yang telah dibersihkan seringkali masih berupa data dengan banyak atribut. Banyaknya atribut atau dimensi ini seringnya justru menyebabkan kompleksitas dan waktu proses yang dibutuhkan menjadi lebih lama. Oleh karena itu, dalam pengerjaan tugas besar ini penting untuk melakukan reduksi dimensi.

Clustering merupakan salah satu metode dalam teknik *unsupervised learning*. Metode ini mencari kemiripan antar data untuk kemudian dikelompokkan berdasar kemiripan tersebut. Kemiripan antar data ini dilihat dari jarak antar data. Keluaran dari metode ini berupa atribut baru dalam kelas sesuai *cluster.*

Klasifikasi adalah salah satu teknik *supervised learning*. Klasifikasi memodelkan data-data ke dalam kelas-kelas yang telah didefinisikan. Dalam memodelkan klasifikasi, dibutuhkan masukan berupa data latih yang memiliki kelas / label. Selanjutnya data latih akan menghasilkan model klasifikasi yang akan memodelkan data uji. Dari hasil pemodelan data uji dapat dilihat akurasi model yang dibangun.

II. PEMBAHASAN

A. Identifikasi Masalah

Dalam pengerjaan tugas besar *Machine* Learning, saya diberikan dataset untuk melakukan *unsupervised* dan *supervised learning.* Dataset yang akan diolah adalah used_car. Dataset yang diberikan harus dapat memodelkan *clustering* dan klasifikasi. Mahasiswa

dibebaskan untuk mengeksplorasi proses menyiapkan data dan mahasiswa harus melakukan evaluasi terhadap model yang dibangun.

Masalah krusial dari dataset yang diberikan adalah tidak adanya atribut kelas. Pada dataset mentah terdapat 20.000 baris data dan 26 atribut, namun dari ke-26 atribut tersebut tidak ada atribut kelas. Permasalahan ini mengharuskan proses pemberian label pada dataset supaya selanjutnya dapat dilakukan klasifikasi (*supervised learning* membutuhkan kelas data).

B. Dataset

Dataset yang digunakan adalah data dari situs iklan penjualan barang, craigslist. Dataset ini hanya berisi iklan mobil-mobil yang dijual dalam situs tersebut. Dataset ini memiliki 20.000 baris data dan 26 atribut. Sebelum melanjutkan ke data selanjutnya, penting untuk memahami dataset. Menurut pemahaman saya, dataset ini adalah dataset yang diambil dari mobil-mobil yang diiklankan untuk dijual.

C. Data Preprocessing

Preprocessing adalah tahap yang paling penting dalam pengerjaan tugas besar ini karena akan memberikan pengaruh besar pada tahap selanjutnya. Seluruh pemodelan akan dilakukan dengan 2 tahap *preprocessing* berbeda. Ada beberapa tahap yang dilakukan dalam *preprocessing*. Kode penuh *preprocessing* dan justifikasi dalam setiap tahap dapat dilihat dalam dokumentasi Google Colab. Berikut tahapan yang dilakukan dalam pengerjaan tugas besar ini :

- Preprocessing I / Teknik I
    1. Hal pertama yang dilakukan adalah menghapus semua atribut yang tidak diperlukan dan menyisakan atribut: region, price, year, manufacturer, model, condition, cylinders, fuel, odometer, title_status, transmission, drive, type, paint_color.
    2. Selanjutnya, saya menghapus baris yang memiliki *missing value* pada atribut year.
    3. Lalu, saya menghapus baris dengan *missing value* lebih dari sama dengan 6.
    4. Selanjutnya saya mengasumsikan tidak terdapat mobil dengan harga lebih rendah dari 1500 $.
    5. Kegiatan selanjutnya adalah menentukan tipe data dari setiap atribut.
    6. Selanjutnya adalah menentukan cara penanganan *missing value* pada data yang memiliki tipe data berbeda.
    7. Setelah itu menghapus outlier pada atribut price, odometer, dan year.
    8. Lalu menyimpan data ke dalam file csv untuk nantinya dimasukkan label.
    9. Kemudian mengganti nilai dari atribut yang memiliki nilai ordinal dengan *ranking* manual berdasar value (untuk atribut type, dilihat dari harga rata-rata mobil bekas di website cargurus.com). Dari seluruh data sudah tidak terdapat mobil dengan title_status = ‘missing only’ maka tidak perlu di-*replace*.
    10. Setelah itu mengganti nilai dari atribut yang memiliki nilai nominal dengan LabelEncoder dari *library* Python.

11. Selanjutnya, menjadikan seluruh atribut ke dalam tipe data int64 dengan *casting* atribut odometer dan year.
12. Setelah itu, menyamakan skala antar atribut dengan MinMax Scaling. Scaling menyebabkan seluruh data berada dalam rentang data yang sama yaitu dari 0 hingga 1. Hal ini disebabkan rumus dari MinMax Scaling yang menyebabkan nilai pada suatu kolom dalam tiap baris akan dihitung dengan membandingkan nilai max dan min dari keseluruhan kolom tersebut.

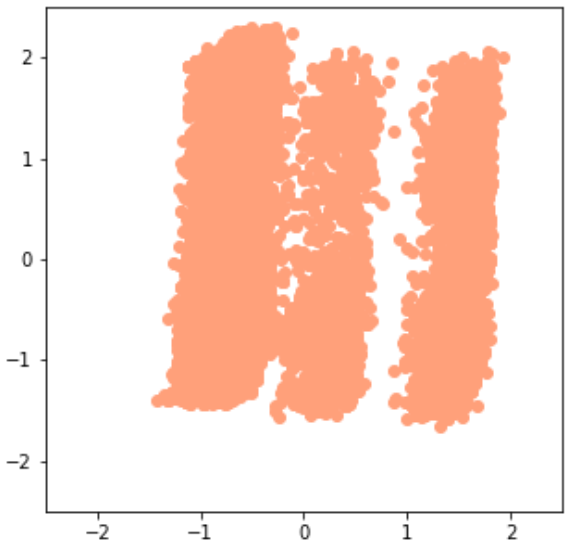
- Preprocessing II / Teknik II

  Perbedaan mendasar antara Preprocessing I dan II adalah:
  1. Saya tidak menghapus baris yang memiliki *missing value* pada atribut year.
  2. Saya tidak menghapus baris dengan *missing value* lebih dari sama dengan 6 atribut.
  3. Saya tidak mengasumsikan mobil selalu memiliki harga lebih dari 1500 $.
  4. Saat saya melakukan *replace* nilai atribut ordinal, seluruh nilai dari tiap atribut ordinal masih lengkap. Masih terdapat baris dengan nilai atribut title_status = ‘parts only’

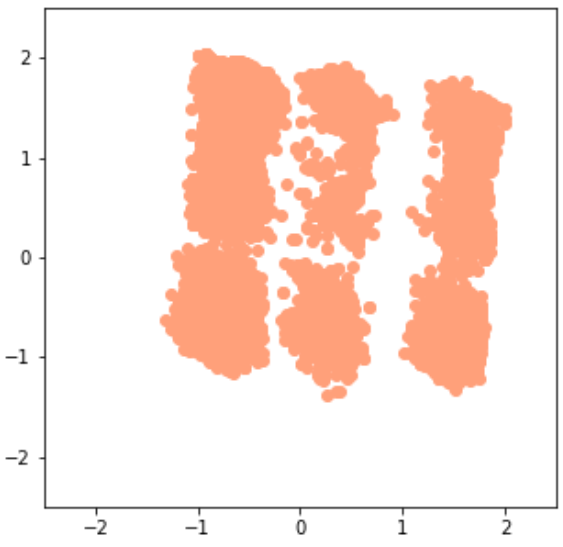
D. Reduksi Dimensi

Data ini akan melakukan *clustering*, dan metode yang saya pilih adalah K-Means Clustering. Metode ini menerima masukan data 2 kolom untuk kemudian diberi label. Maka dari itu saya harus melakukan transformasi pada data yang telah di-*scaling*. Saya memutuskan untuk mengurangi jumlah kolom dengan ekstraksi fitur menggunakan PCA. Proses ekstraksi fitur dengan PCA, saya lakukan dengan library karena tingkat kesulitan yang tinggi. Pada penentuan n komponen PCA, saya memilih nilai 0.3. Hal ini karena setelah dicoba dengan nilai yang berbeda akan menghasilkan atribut dengan jumlah lebih selain 2. Saya mengasumsikan nilai ini adalah ini optimal jika ingin mereduksi data menjadi 2 dimensi dari data sebelumnya. Hasil dari transformasi berupa array, maka hasil PCA perlu diubah ke dataframe. Berikut saya tampilkan hasil reduksi dimensi pada Teknik I dan Teknik II.

Hasil PCA Teknik I



Hasil PCA Teknik II



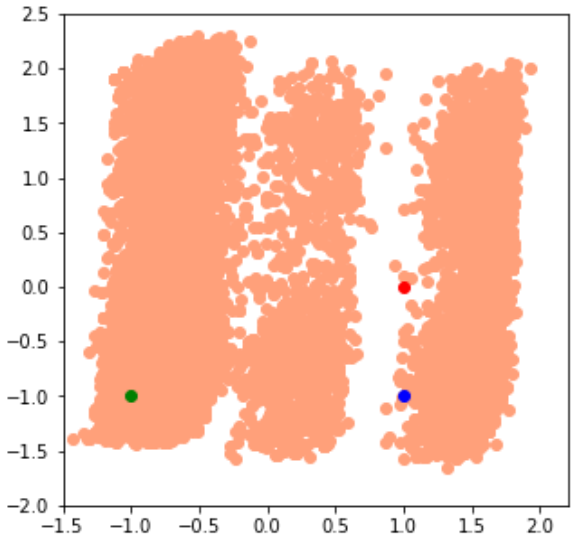
Perbedaan terlihat dari hasil reduksi dimensi kedua teknik tersebut. Meskipun sama-sama menggunakan PCA, namun sebaran datanya terlihat berbeda. Terdapat lebih banyak baris pada data hasil PCA Teknik II daripada Teknik I, namun data hasil PCA cenderung menumpuk dan tidak tersebar. Dari dua gambar sebelumnya, dapat kita lihat bahwa hasil PCA pada data Teknik I meng-*generate* posisi yang lebih tersebar.

E. Clustering

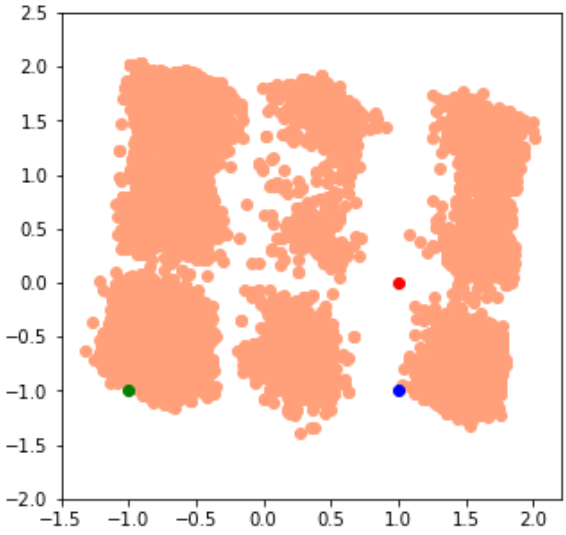
Dengan dataframe hasil PCA, maka selanjutnya dibangun model K-Means Clustering dengan k = 3. Kode dan alur *clustering* dapat dilihat dalam dokumentasi kode. Di bawah ini saya akan menampilkan hasil clustering 3 kelas. Warna merah menandakan kelas '1', warna hijau mendakan kelas '2', dan warna biru menandakan kelas '3'. Berikut adalah ilustrasi per-tahap K-Means Clustering dalam Teknik I dan Teknik II:

- Mencari 3 centroid untuk tiap cluster dan diterapkan Teknik 1 dan Teknik 2. Setelah dicari secara acak, secara kebetulan centroid dari kedua teknik sama.

Teknik I

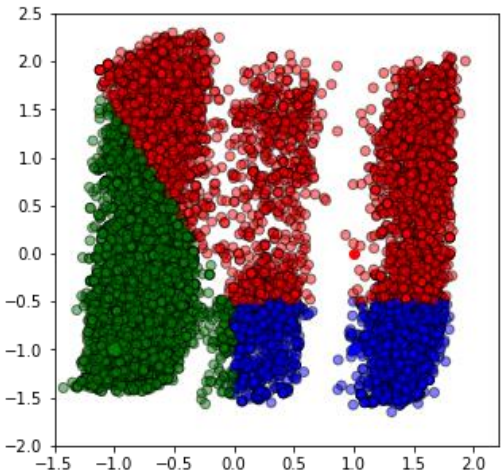


Teknik II

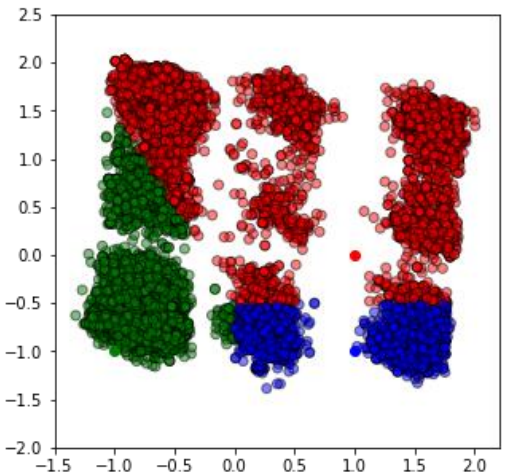


- Pengelompokan tiap *instance* ke dalam *cluster* dengan mencari centroid terdekat.

Teknik I

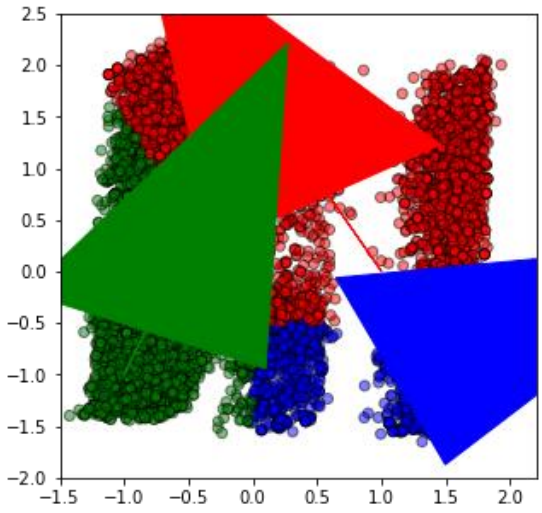


Teknik II

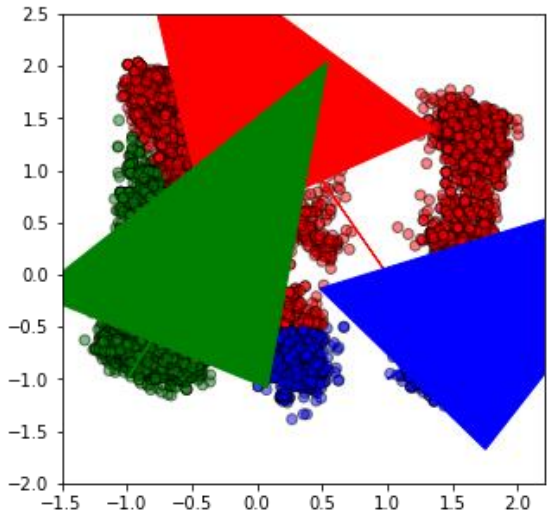


- Mencari arah dimensi *clustering* pada tiap *cluster*.

Teknik I

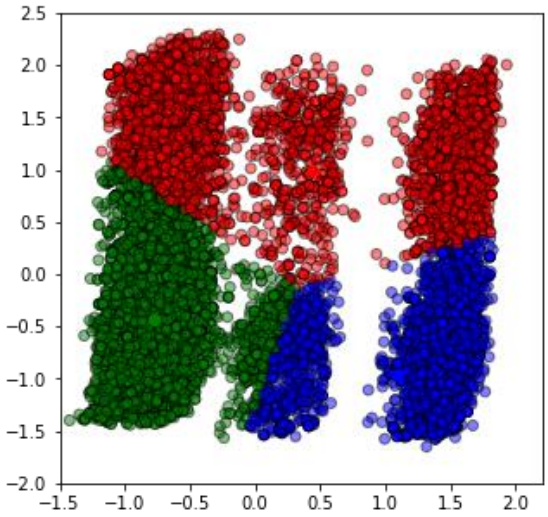


Teknik II

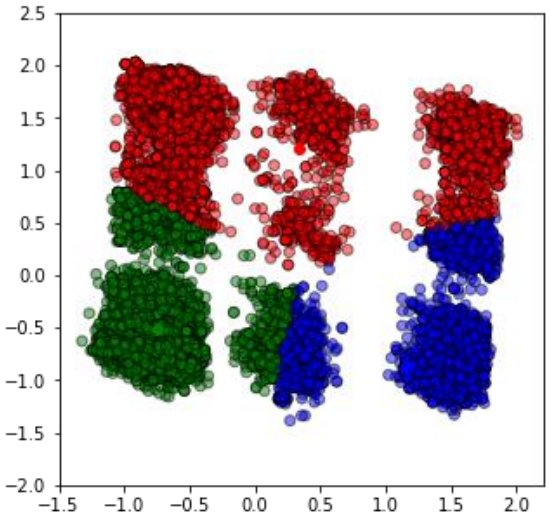


- Melakukan assignment sekali lagi pada tiap *cluster*, untuk mencari centroid baru.

Teknik I

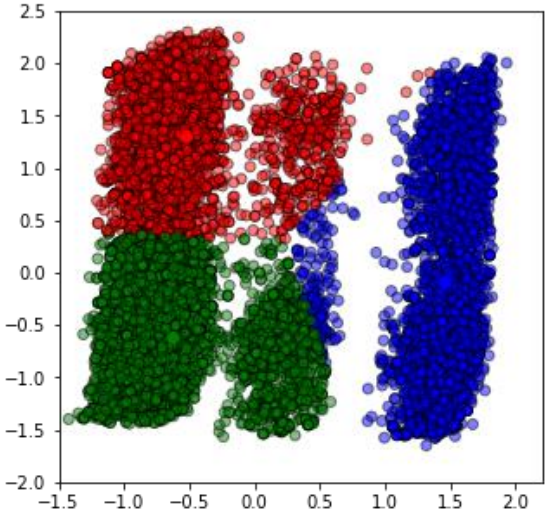


Teknik II

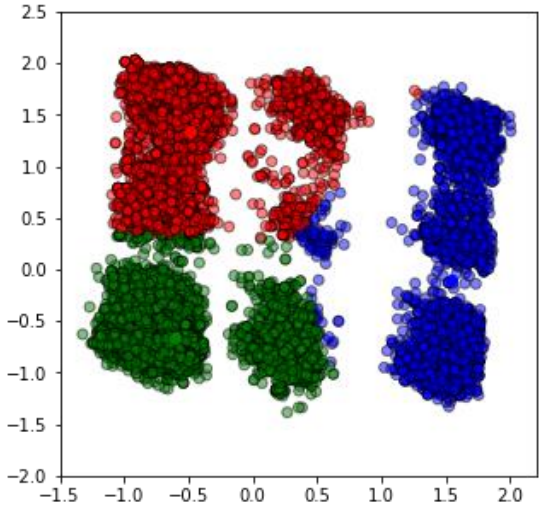


- Secara repetitif, tiap instance mencari cluster terdekat hingga stabil.

Teknik I



Teknik II



F. Klasifikasi

Pada tahap klasifikasi, akan dilakukan eksperimen dengan menggunakan algoritma Naive Bayes, KNN, dan Random Forest. Ketiga algoritma tersebut membutuhkan masukan berupa data numerikal, maka data yang digunakan adalah data setelah scaling yang telah ditempelkan label dari hasil clustering. Data dengan label ini akan di-split 80 % untuk data latih dan 20% untuk data uji. Menurut saya jika data latihnya hanya 75%, bisa jadi model yang dirancang kurang maksimal. Namun, jika data latihnya lebih besar lagi dari 80%, bisa jadi cakup data uji kurang luas. Data latih dan data uji untuk ketiga algoritma adalah sama agar dapat membandingkan secara real performansi keduanya. Pemodelan klasifikasi dilakukan dengan menggunakan library dari sklearn. Algoritma pada Teknik I dan Teknik II sama untuk setiap model.

- Algoritma Naive Bayes : Algoritma ini bekerja dengan perbandingan probabilitas.
- Algoritma K-Nearest Neighbor : Algoritma ini bekerja dengan pengelompokan *instance* berdasar kelas k jumlah tetangga terdekat.
- Algoritma Random Forest : Algoritma ini bekerja dengan menggunakan konsep *ensemble learning* yang umumnya memiliki performa lebih baik dan cepat.

Kode yang digunakan dalam klasifikasi:

- Split data

```
[ ]  X = df.iloc[:, :-1].values
     y = df.iloc[:, 14].values
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

- Naive Bayes

```
[ ]  gnb = GaussianNB()
     gnb.fit(X_train, y_train)
     y_pred = gnb.predict(X_test)
```

- KNN

```
[ ]  knn = KNeighborsClassifier()
     knn.fit(X_train, y_train)
     y_pred = knn.predict(X_test)
```

- Random Forest

```
[114] rf = RandomForestClassifier()
      rf.fit(X_train, y_train)
      y_pred = rf.predict(X_test)
```

G. Evaluasi

Evaluasi yang dilakukan adalah dengan menggunakan *Confusion Matrix* karena ini adalah salah satu cara evaluasi yang paling umum terhadap model klasifikasi yang saya bangun. Penggunaan *Confusion Matrix* memudahkan proses evaluasi karena dapat melihat akurasi, *precision, recall,* dan F1-score. Berikut adalah evaluasi terhadap model klasifikasi yang telah dibangun pada Teknik I dan Teknik II :

a. Evaluasi model klasifikasi Naive Bayes pada kedua teknik.

- Teknik I, akurasi 96,48%

```
Akurasi Model Naive Bayes: 0.9648005363727791


Confusion Matrix
[[ 706    4    0]
 [  71 1360    1]
 [  14   15  812]]


Classification Report for Naive Bayes
              precision    recall  f1-score   support

           1       0.89      0.99      0.94       710
           2       0.99      0.95      0.97      1432
           3       1.00      0.97      0.98       841

    accuracy                           0.96      2983
   macro avg       0.96      0.97      0.96      2983
weighted avg       0.97      0.96      0.97      2983
```

- Teknik II, akurasi 98,96%

```
Akurasi Model Naive Bayes: 0.9895690365083722


Confusion Matrix
[[ 935    0    0]
 [  22 1715    0]
 [  15    1  955]]


Classification Report for Naive Bayes
              precision    recall  f1-score   support

           1       0.96      1.00      0.98       935
           2       1.00      0.99      0.99      1737
           3       1.00      0.98      0.99       971

    accuracy                           0.99      3643
   macro avg       0.99      0.99      0.99      3643
weighted avg       0.99      0.99      0.99      3643
```

b. Evaluasi model klasifikasi KNN pada kedua teknik.

- Teknik I, akurasi 97,99%

```
Akurasi Model KNN: 0.9798860207844452


Confusion Matrix
[[ 688   21    1]
 [  20 1409    3]
 [   8    7  826]]


Classification Report for KNN
              precision    recall  f1-score   support

           1       0.96      0.97      0.96       710
           2       0.98      0.98      0.98      1432
           3       1.00      0.98      0.99       841

    accuracy                           0.98      2983
   macro avg       0.98      0.98      0.98      2983
weighted avg       0.98      0.98      0.98      2983
```

- Teknik II, akurasi 98,55%

```
Akurasi Model KNN: 0.9854515509195718


Confusion Matrix
[[ 909   22    4]
 [  15 1718    4]
 [   6    2  963]]


Classification Report for KNN
              precision    recall  f1-score   support

           1       0.98      0.97      0.97       935
           2       0.99      0.99      0.99      1737
           3       0.99      0.99      0.99       971

    accuracy                           0.99      3643
   macro avg       0.99      0.98      0.98      3643
weighted avg       0.99      0.99      0.99      3643
```

c. Evaluasi model klasifikasi Random Forest pada kedua teknik.

- Teknik I, akurasi 98,79%

```
Akurasi Model Random Forest: 0.9879316124706671


Confusion Matrix
[[ 706    4    0]
 [  12 1420    0]
 [  10   10  821]]


Classification Report for Random Forest
              precision    recall  f1-score   support

           1       0.97      0.99      0.98       710
           2       0.99      0.99      0.99      1432
           3       1.00      0.98      0.99       841

    accuracy                           0.99      2983
   macro avg       0.99      0.99      0.99      2983
weighted avg       0.99      0.99      0.99      2983
```

- Teknik II, akurasi 99,61%

```
Akurasi Model Random Forest: 0.9961570134504529


Confusion Matrix
[[ 931    4    0]
 [   3 1734    0]
 [   5    2  964]]


Classification Report for Random Forest
              precision    recall  f1-score   support

           1       0.99      1.00      0.99       935
           2       1.00      1.00      1.00      1737
           3       1.00      0.99      1.00       971

    accuracy                           1.00      3643
   macro avg       1.00      1.00      1.00      3643
weighted avg       1.00      1.00      1.00      3643
```

III. KESIMPULAN

Preprocessing adalah tahap yang membutuhkan waktu paling banyak dalam pengerjaan tugas besar ini. Tahap ini memiliki tantangan tersendiri karena harus menyiapkan data agar siap digunakan untuk tahap selanjutnya. Tahap ini menjadi semakin penting karena jika dilakukan dengan benar, model yang dibangun akan memiliki akurasi tinggi. Dalam pengerjaan tugas besar ini dilakukan 2 teknik berbeda dalam *preprocessing*. Perbedaan teknik yang dimaksud adalah pada eksekusi asumsi yang mempengaruh kurang lebih 3.300 baris perbedaan jumlah *clean data*.

Preprocessing jika dikombinasikan dengan teknik reduksi dimensi yang tepat dapat meningkatkan performansi model yang akan dibangun. Setelah melakukan PCA pada kedua data hasil Teknik I dan Teknik II didapat bahwa data pada Teknik I lebih memencar dibanding Teknik II. Sekilas akan terlihat bahwa data Teknik I lebih banyak dari data Teknik II, padahal yang terjadi adalah kebalikannya. Disini kita bisa melihat kepadatan data sangat berpengaruh.

Setelah melakukan *preprocessing* dan ekstraksi fitur, selanjutnya dilakukan pemberian label pada dataset dengan melakukan *clustering*. Label yang tercipta dari prosess *clustering* menurut saya adalah label yang menandakan rating untuk value of the product. Rating value of the product mengacu pada keseimbangan kualitas dari tiap aspek dengan harga yang diberikan. Label ini saya beri nama kolom ‘class’. Penambahan kolom ini saya lakukan pada data setelah *scaling*. Hal ini dilakukan karena seluruh algoritma yang akan digunakan

menerima masukan data berupa integer. Setelah dataset memiliki label, selanjutnya dilakukan klasifikasi dengan tiga model, yaitu Naive Bayes, KNN, dan Random Forest.

Pada sub-bab Evaluasi telah kita lihat bahwa secara keseluruhan, model dengan menggunakan algoritma Random Forest memberikan performansi paling baik. Hal ini dikarenakan algoritma Random Forest yang menerapkan *ensemble technique*, yaitu teknik untuk membentuk 'strong learners' dari kumpulan 'weak learners'. Namun bukan hanya pemilihan model, teknik *preprocessing* ternyata juga berpengaruh. Buktinya dapat kita lihat bahwa dalam setiap model, akurasi Teknik II selalu lebih tinggi. Hal ini membuktikan bahwa tidak selalu pengurangan baris dapat meningkatkan akurasi. Bisa jadi karena jumlah baris kurang, maka pemodelan data menjadi kurang akurat. Hal ini menjadi masuk akal karena jika kita kalikan seluruh nilai dari atribut ordinal saja, maka jumlah baris yang kita butuhkan masih kurang. Teknik 1 dengan 3.300 data lebih sedikit daripada Teknik II tentu saja menjadi kurang mampu memodelkan data dengan tepat.