Assignment – 4

GitHub link to access the source code and video which demonstrates my work.

https://github.com/ManojDhulipalla/ML-Assignments.git

Source code can be found under the source code file.

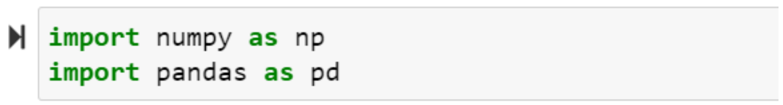
There is a README file which contains student Info, short description about assignment.

A Video is available in README file under Video section.

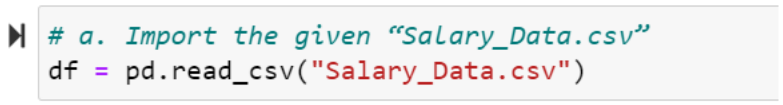
1. Apply Linear Regression to the provided dataset using underlying steps.

a. Import the given "Salary_Data.csv"

Using the pandal library, I imported Salary_Data csv file.



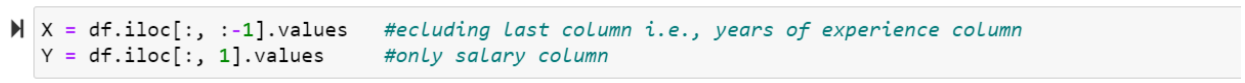
```python
import numpy as np
import pandas as pd
```



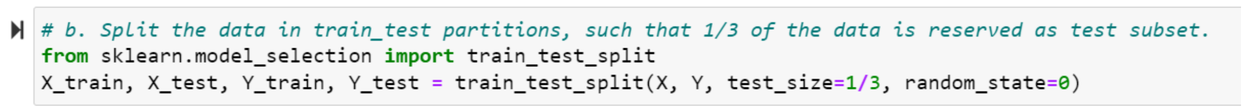
```python
# a. Import the given "Salary_Data.csv"
df = pd.read_csv("Salary_Data.csv")
```

b. Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.

Using train_test_split method from sklearn library, we divided the data into training and testing data. By setting test_size = 1/3 we can divide the data where the testing part contains 1/3rd of the data.



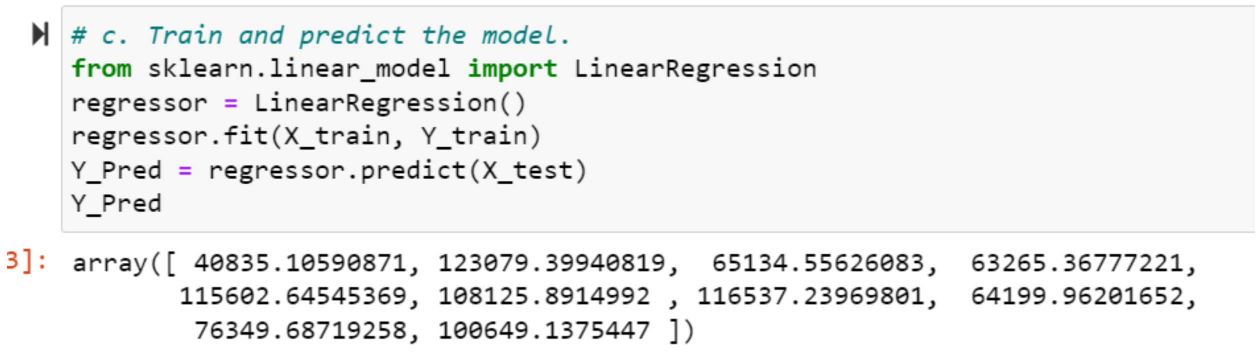
```python
X = df.iloc[:, :-1].values    #ecluding last column i.e., years of experience column
Y = df.iloc[:, 1].values      #only salary column
```



```python
# b. Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=1/3, random_state=0)
```

c. Train and predict the model.

The relationship between the data points is used to predict the outcomes and that relationship can be represented through a line in linear regression algorithm. We trained linear regression algorithm on our training data using LinearRegression method of sklearn library and predicted the values.

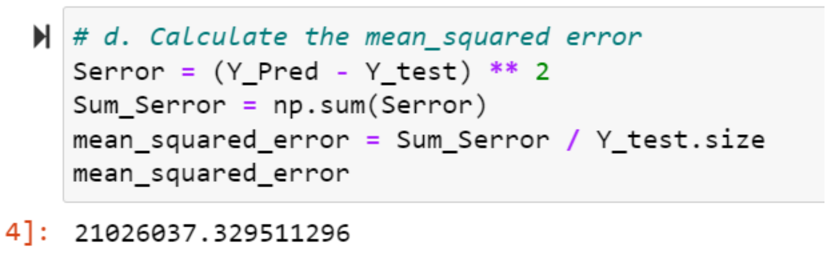


```python
# c. Train and predict the model.
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, Y_train)
Y_Pred = regressor.predict(X_test)
Y_Pred
```

```
3]: array([ 40835.10590871, 123079.39940819,  65134.55626083,  63265.36777221,
       115602.64545369, 108125.8914992 , 116537.23969801,  64199.96201652,
        76349.68719258, 100649.1375447 ])
```

d. Calculate the mean_squared error

Mean squared error is the average squared error i.e., the average squared difference between the predicted values and the testing values.

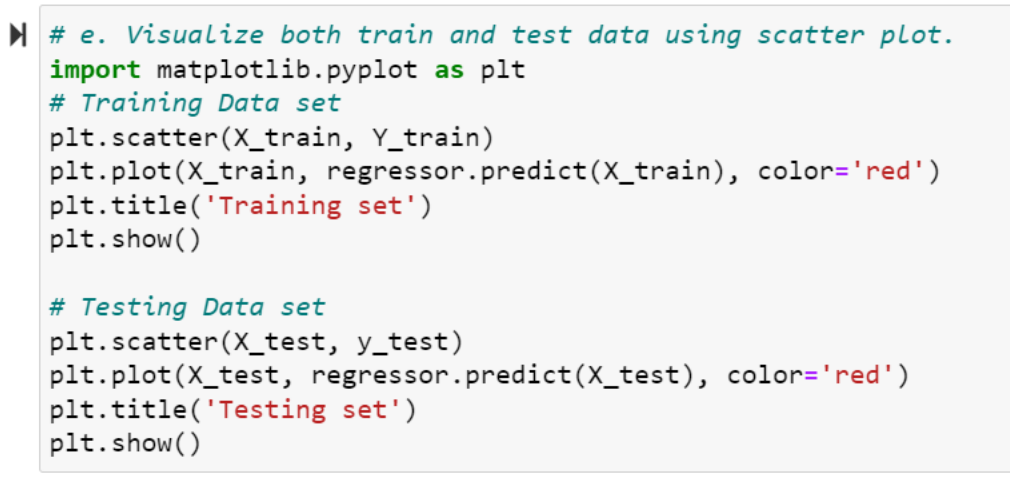


```
# d. Calculate the mean_squared error
Serror = (Y_Pred - Y_test) ** 2
Sum_Serror = np.sum(Serror)
mean_squared_error = Sum_Serror / Y_test.size
mean_squared_error
```

```
4]:  21026037.329511296
```

e. Visualize both train and test data using scatter plot.

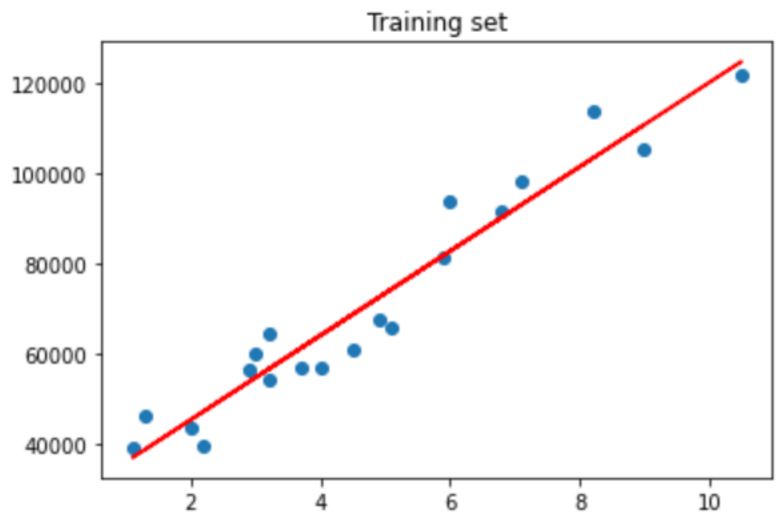
The scatter method of matplotlib library is used to represent the training and testing data visually through a scatter plot and plot method to draw a line between the data points.



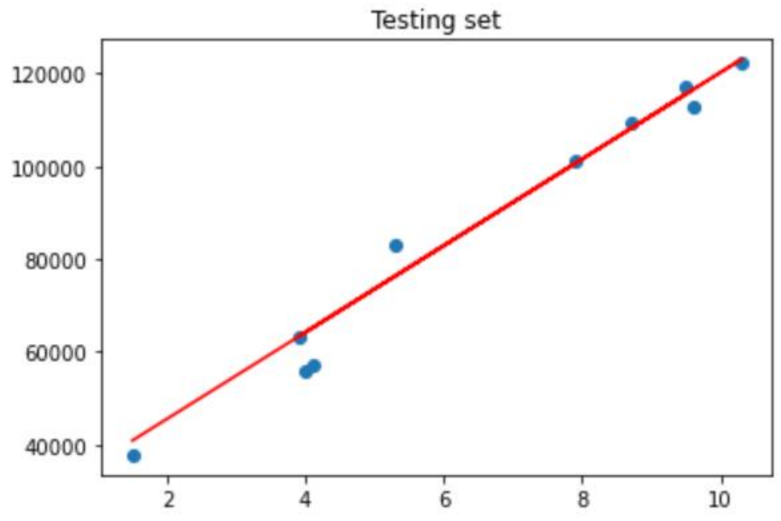
```
# e. Visualize both train and test data using scatter plot.
import matplotlib.pyplot as plt
# Training Data set
plt.scatter(X_train, Y_train)
plt.plot(X_train, regressor.predict(X_train), color='red')
plt.title('Training set')
plt.show()

# Testing Data set
plt.scatter(X_test, y_test)
plt.plot(X_test, regressor.predict(X_test), color='red')
plt.title('Testing set')
plt.show()
```

Testing data:

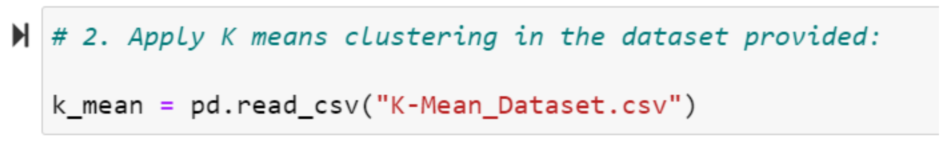


Training data:



2. Apply K means clustering in the dataset provided:

Using the pandal library, I imported K-Mean_Dataset csv file.

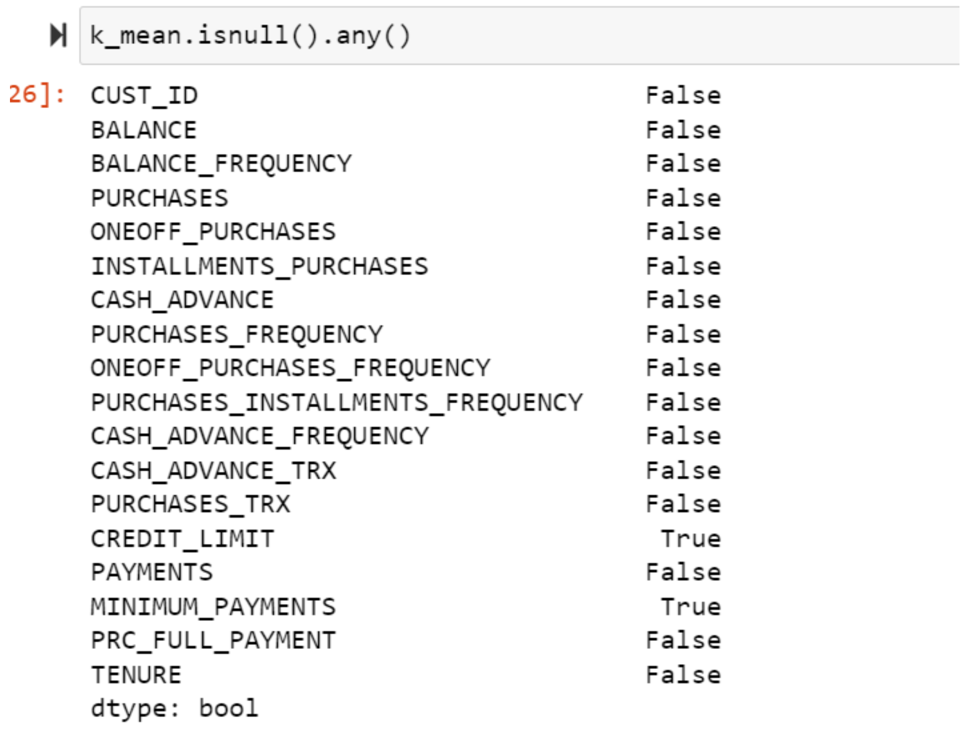


```
# 2. Apply K means clustering in the dataset provided:

k_mean = pd.read_csv("K-Mean_Dataset.csv")
```

• **Remove any null values by the means.**

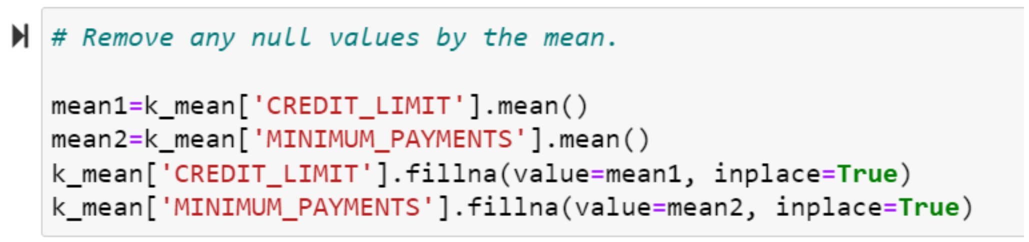
Using isnull method we can check that the dataset contains any null values.



```
k_mean.isnull().any()
```

```
26]:  CUST_ID                             False
      BALANCE                             False
      BALANCE_FREQUENCY                   False
      PURCHASES                           False
      ONEOFF_PURCHASES                    False
      INSTALLMENTS_PURCHASES              False
      CASH_ADVANCE                        False
      PURCHASES_FREQUENCY                 False
      ONEOFF_PURCHASES_FREQUENCY          False
      PURCHASES_INSTALLMENTS_FREQUENCY    False
      CASH_ADVANCE_FREQUENCY              False
      CASH_ADVANCE_TRX                    False
      PURCHASES_TRX                       False
      CREDIT_LIMIT                         True
      PAYMENTS                            False
      MINIMUM_PAYMENTS                     True
      PRC_FULL_PAYMENT                    False
      TENURE                              False
      dtype: bool
```

If any null values are present, we need to replace those values by means of individual columns using fillna method.



```
# Remove any null values by the mean.

mean1=k_mean['CREDIT_LIMIT'].mean()
mean2=k_mean['MINIMUM_PAYMENTS'].mean()
k_mean['CREDIT_LIMIT'].fillna(value=mean1, inplace=True)
k_mean['MINIMUM_PAYMENTS'].fillna(value=mean2, inplace=True)
```

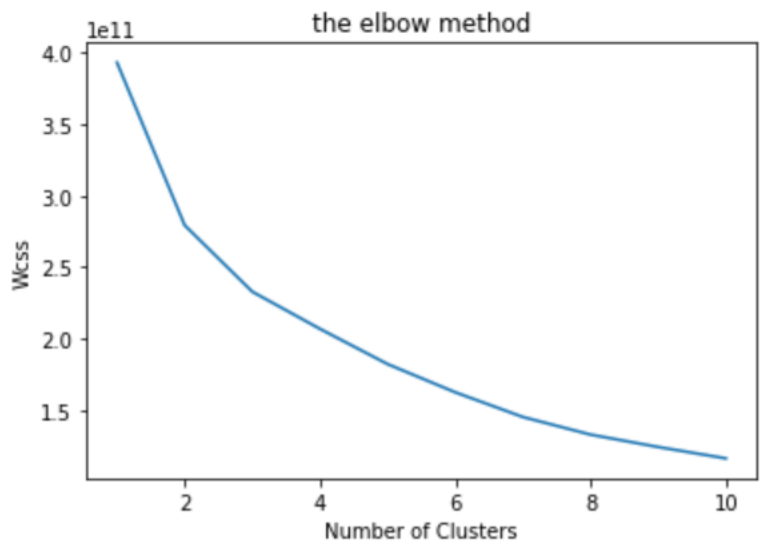
• Use the elbow method to find a good number of clusters with the K-Means algorithm

To find a good number of clusters required to fit our data together into clusters we have the elbow method. This elbow method results in optimal value of K. In K-Means algorithms, where K is the number of clusters. We can find the k value from the graph where the elbow point that the inertia values start decreases linearly.

```python
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,11),wcss)
plt.title('the elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```

From the graph below, we can define the number of clusters required to train KMeans algorithm on data set. From the graph, we can see at elbow point 3, wcss values start decreases linearly. So, we need 3 clusters to fit out data into clusters.



• **Calculate the silhouette score for the above clustering**

With k = 3 we can train KMeans algorithm on K-Mean_Dataset and predit the values using KMeans method of sklearn library. Using the predicted values, we can find the silhouette score. Silhouette score is used to calculate how good the clustering technique is. Silhouette score is the difference between the point and the nearest cluster that the point is not part of the cluster.

```python
#  Calculate the silhouette score for the above clustering

nclusters = 3  # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(x)

y_cluster_kmeans = km.predict(x)
from sklearn import metrics
score = metrics.silhouette_score(x, y_cluster_kmeans)
print(score)
```

```
0.46504469672047805
```

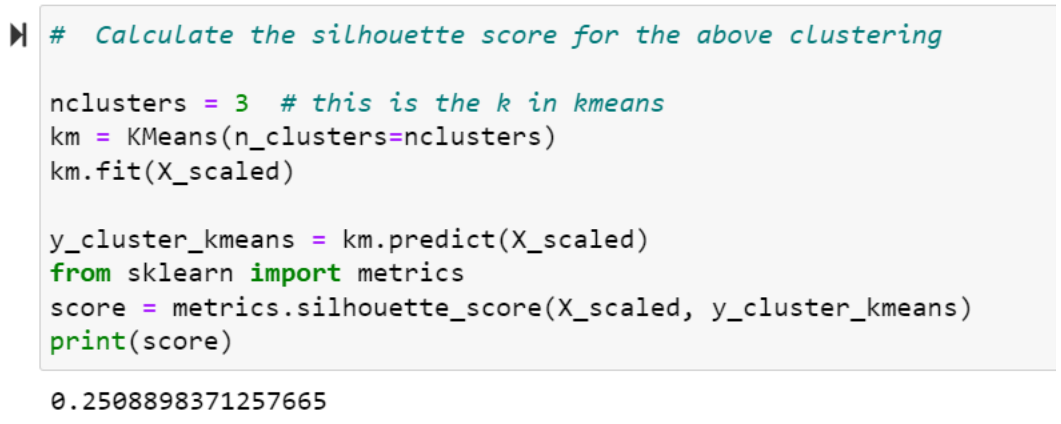
3. Try feature scaling and then apply K-Means on the scaled features. Did that improve the Silhouette score? If yes, can you justify why

We can perform feature scaling using many ways i.e., min-max scaler, standard scaler, and robust scaler. Here, we are using standard scaler to perform feature scaling. Sklearn library contains a preprocessing module to preprocess our raw data. Using preprocessing module, we will perform feature scaling. Feature scaling is used to normalize the range of all features. Initially some features contain a range in thousands or even hundreds. Among varies type of ranges, the features which have high range have some superiority over others.

```python
# feature scaling using standard scaler

from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(x)
X_scaled_array = scaler.transform(x)
X_scaled = pd.DataFrame(X_scaled_array, columns = x.columns)
```

Here we are training KMeans algorithm on the preprocessed data set and found the silhouette score for this model.



```
#  Calculate the silhouette score for the above clustering

nclusters = 3  # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X_scaled)

y_cluster_kmeans = km.predict(X_scaled)
from sklearn import metrics
score = metrics.silhouette_score(X_scaled, y_cluster_kmeans)
print(score)
```

```
0.2508898371257665
```

Even after preprocessing raw data, it does not improve the silhouette score. The silhouette score for this preprocessed data is less than the silhouette score for raw data.