Assignment 4

Name: Rohitha Sai Alla

No: 700734780

Video link: https://drive.google.com/file/d/1mMcR6zjmXVk7nwqvvD5TS_5jSYzRfKDJ/view?usp=sharing

Git link: https://github.com/RohithaSaiML/Assignment4

Linear regression

1.a importing the dataset

In [161]:
```python
# Simple Linear Regression
# Importing the libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# 1.a Importing the datasets

datasets = pd.read_csv('Salary_Data.csv')

X = datasets.iloc[:, :-1].values  #X is assigned to yearExperience
Y = datasets.iloc[:, 1].values #Y is assigned to salary
datasets.head(10)
```

Out[161]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0

1.b Splitting the data into 1/3rd test and other as train

```
In [162]: #1.b Splitting the dataset pinto train and test set

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)
```

1.c train the predict the model

```
In [163]: #Fitting the linear regression to training set
#1.c Train and predict the model

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, Y_train)

#predicting the model
y_predict = regressor.predict(X_test)
```

1.d to calculate the mean squared error

```
In [164]: #1.d to calculate mean squared error
from sklearn.metrics import mean_squared_error

print('Mean squared value: ', mean_squared_error(Y_test,y_predict))
```

Output:

```
Mean squared value:  21026037.329511296
```

1.e visualization of train data

```
In [165]: #1.e to visualize train and test data using scatter plot
# to visualize train results
plt.scatter(X_train, Y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience train set')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.show()
```

Output:

Visualization of test data

```
# to visulaize test results
plt.scatter(X_test, Y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs experience test set')
plt.xlabel('Experience')
plt.ylabel('Salary')
plt.show()
```

Output:

2.K means clustering

Remove null values and replacing with mean

```
In [170]: #2.1 to replace null values with mean
#CREDIT_LIMIT and MINIMUM_PAYMENTS have null
dataset[['CREDIT_LIMIT','MINIMUM_PAYMENTS']] = dataset[['CREDIT_LIMIT','MINIMUM_PAYMENTS']].fillna(value=dataset[['CREDIT_LIMIT',
dataset.isnull().any()
```

Output:

```
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                        False
PAYMENTS                            False
MINIMUM_PAYMENTS                    False
PRC_FULL_PAYMENT                    False
TENURE                              False
dtype: bool
```

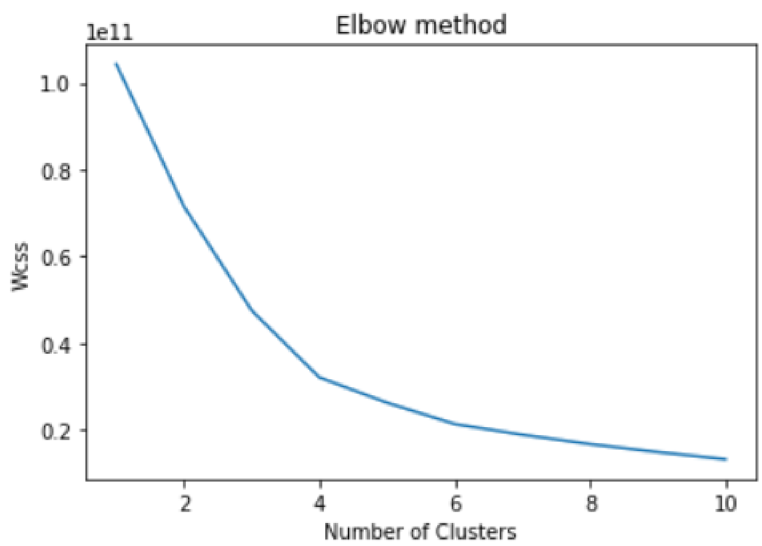
Applying elbow method to find clusters with k-means algorithm.

```
#2.2 using elbow method to find good number of clusters with K-Means algorithm

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('Elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('Wcss')
plt.show()
```

Output:



Calculating silhouette score

```
#2.3 to calculate silhouette score for the cluster

y_cluster_kmeans = km.predict(x)
score = metrics.silhouette_score(x, y_cluster_kmeans)
print('Silhouette score: ', score)
```

Output:

```
Silhouette score:  0.6176798152094864
```

```
#for 4 clusters we get least error
nclusters = 4
km = KMeans(n_clusters=nclusters)
km.fit(x)
```

```
KMeans(n_clusters=4)
```

Feature scaling the dataset

```
In [175]: #feature scaling the data
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder, StandardScaler
scaler = preprocessing.StandardScaler()
scaler.fit(x)
X_scaled_array = scaler.transform(x)
X_scaled = pd.DataFrame(X_scaled_array, columns = x.columns)
```

```
In [176]: #applying kmeans on scaled features
from sklearn.cluster import KMeans
nclusters = 4
km = KMeans(n_clusters=nclusters)
km.fit(X_scaled)
```

Output:

```
KMeans(n_clusters=4)
```

Calculating silhouette score for the scaled data

```
In [177]: #getting sillhouette score after scaling features
y_scaled_cluster_kmeans = km.pre`dict(X_scaled)
from sklearn import metrics
scaled_score = metrics.silhouette_score(X_scaled, y_scaled_cluster_kmeans)
print('Silhouette score: ', score)
print('Silhouette score on scaled features:', scaled_score)
```

Output:

```
Silhouette score:  0.6176798152094864
Silhouette score on scaled features: 0.5112148779873158
```

The Silhouette score has decreased after feature scaling. Due to normalization the data points are brought to same range due to which intra-cluster distance has decreased. So Silhouette score has decreased after scaling features.