Name : Garimella Suresh Babu
Student Id: 700741367
User Id :SXG13670@ucmo.edu

GitHub Link:-
https://github.com/Suresh-Garimella/Linear-Regression-Clustering

Video Link:-

https://github.com/Suresh-Garimella/Linear-Regression-Clustering/blob/main/ScreenRecording_Regression_Clustering.mkv

1. Apply Linear Regression to the provided dataset using underlying steps.
a. Import the given “Salary_Data.csv”

Imports : pandas , numpy and matplotlib libraries
Using the pandas library we read the Salary_Data.csv file.

```
In [128]:
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
In [129]:
# Importing the datasets

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

```
In [130]: datasets.head()
```

Out[130]:

	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

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

Imports : sklearn and train_test_split libraries.
Train_test_split is used to divide the dataset in to train part and testing part based on the given parameters.

Here 0.33 means the test data should be 1/3rd of the whole dataset.
Random_state = 0 represents the split should be same for every execution of the code.

In [165]:
```
from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X,Y,test_size=0.33,random_state = 0)
```

In [166]:
```
print(X_Train.shape,X_Test.shape)
print(Y_Train.shape,Y_Test.shape)
```

```
(20, 1) (10, 1)
(20,) (10,)
```

c. Train and predict the model.
Imports : LinearRegression
We used the LinearRegression method and passed the training_set to the fit() method.
Then using the returned object, we used the predict() method to return the model using the test dataset.

In [167]:
```
# Fitting Simple Linear Regression to the training set

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_Train, Y_Train)
```

Out[167]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [168]:
```
print(regressor.intercept_)
```

```
26816.19224403119
```

In [169]:
```
print(regressor.coef_)
```

```
[9345.94244312]
```

In [170]:
```
# print(regressor.y_pred)
Y_Pred = regressor.predict(X_Test)
Y_Pred
```

Out[170]:
```
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

Imports: metrics from sklearn library
Metrics contains a predefined method to return the mean squared value namely,mean_squared_error.
The mean square error is the average of the square of the difference between the observed and predicted values of a variable.

```
In [175]: from sklearn import metrics
```

```
In [176]: print("Mean Squared Error:", metrics.mean_squared_error(Y_Test,Y_Pred))

          Mean Squared Error: 21026037.329511296
```

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

Imports: plt from matplotlib library
We used the scatter() method from matplotlib library to Visualize the train and test data points in the graph.
plot() is used to represent the linear regression line.
title() - represents the Title of the graph
Xlabel and ylabel used to label the X-axis and Y-axis.

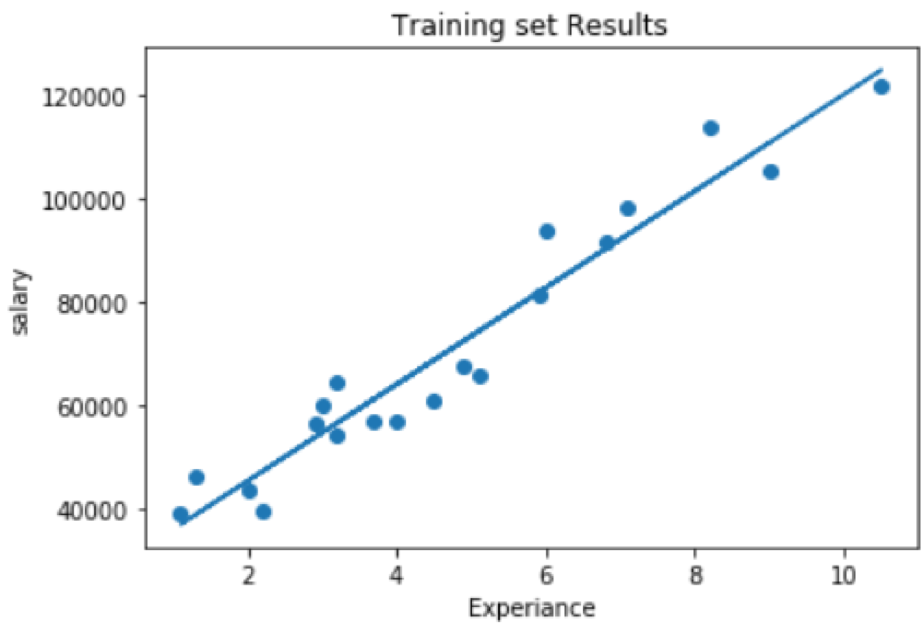
In [178]:

```
import matplotlib.pyplot as plt
```

In [179]:

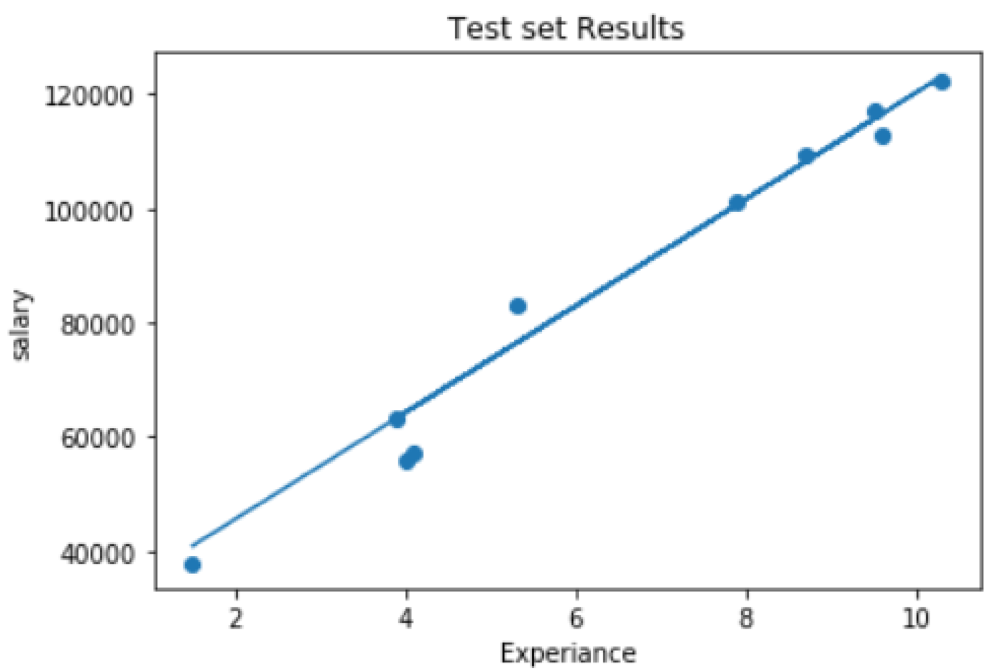
```

plt.scatter(X_Train, Y_Train)
plt.plot(X_Train,regressor.predict(X_Train))
plt.xlabel('Experiance')
plt.ylabel('salary')
plt.title('Training set Results')
plt.show()
```

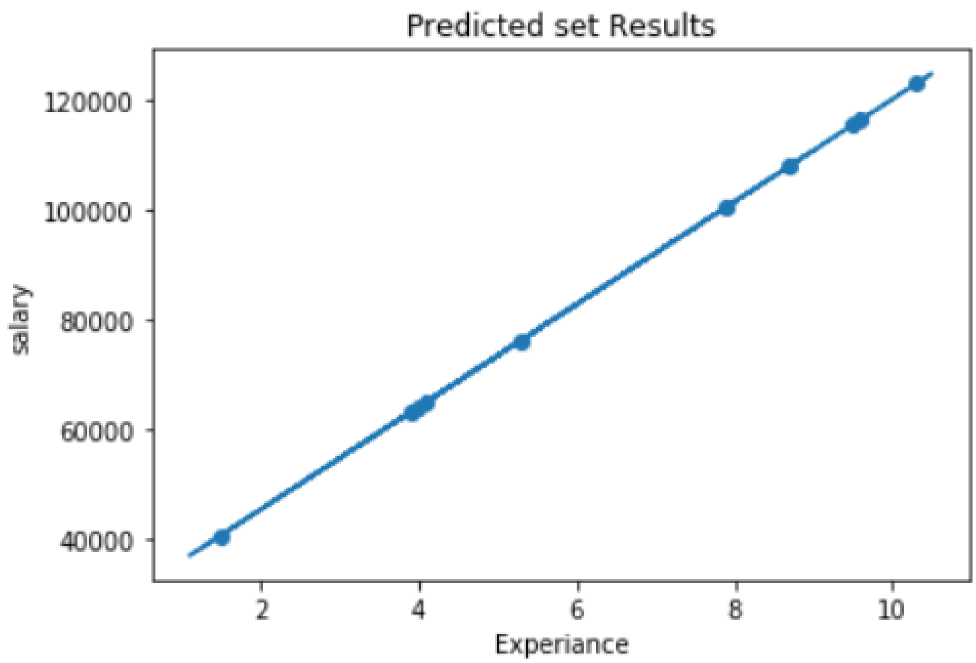


In [180]:

```
plt.scatter(X_Test, Y_Test)
plt.plot(X_Test,regressor.predict(X_Test))
plt.xlabel('Experiance')
plt.ylabel('salary')
plt.title('Test set Results')
plt.show()
```



In [181]:
```
plt.scatter(X_Test, Y_Pred)
plt.plot(X_Train,regressor.predict(X_Train))
plt.xlabel('Experiance')
plt.ylabel('salary')
plt.title('Predicted set Results')
plt.show()
```



2. Apply K means clustering in the dataset provided:
Using Pandas Library reading the K-Mean_Dataset.csv file into the dataset.

In [114]:
```
import pandas as pd
```

In [115]:
```
dataset = pd.read_csv("K-Mean_Dataset.csv")
```

- **Remove any null values by the mean.**

Finding columns containing the null values using the isnull() method.

```
In [116]: dataset.isnull().any()
```

```
Out[116]: 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
```

Here we can see “PAYMENTS” and “CREDIT_LIMIT” columns has Null values .
So, Replacing the null values with their respective means.

```
In [117]: mean1=dataset['CREDIT_LIMIT'].mean()
          mean2=dataset['MINIMUM_PAYMENTS'].mean()
          dataset['CREDIT_LIMIT'].fillna(value=mean1, inplace=True)
          dataset['MINIMUM_PAYMENTS'].fillna(value=mean2, inplace=True)
```

After replacing the null values with the means.

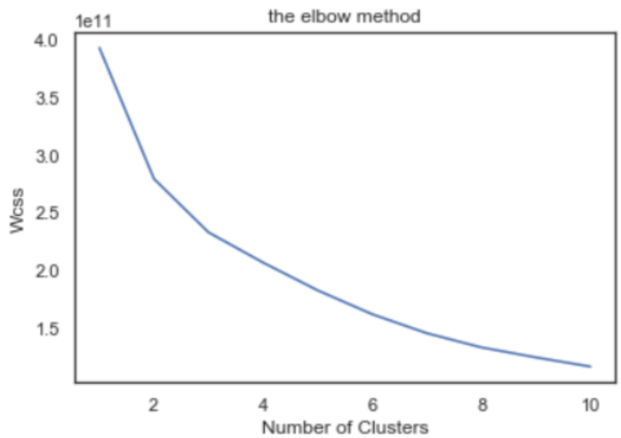
```
In [118]: dataset.isnull().any()
```

```
Out[118]: 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
```

• **Use the elbow method to find a good number of clusters with the K-Means algorithm**
The elbow method runs k-means clustering on the dataset for a range of values for k (say from 1-11) and then for each value of k computes an average score for all clusters.

In [119]:
```
# ##elbow method to know the number of clusters
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(table)
    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 we can see that there is a distortion at the x-axis value 3 .
which represents that the model will be good fit if we took 3 clusters from the given data set.

WCSS is the sum of squared distance between each point and the centroid in a cluster. When we plot the WCSS with the K value, the plot looks like an Elbow. As the number of clusters increases, the WCSS value will start to decrease. WCSS value is largest when K = 1.

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

Silhouette Score is a cluster validating coefficient.
Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.

```
In [122]: #  Calculate the silhouette score for the above clustering
          #since the elbow point is at 3
          nclusters = 3  # this is the k in kmeans
          km = KMeans(n_clusters=nclusters)
          km.fit(table)

          y_cluster_kmeans = km.predict(table)
          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

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing.
It basically helps to normalize the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm.
Here we are using standard scalar feature scaling technique to normalize the data.

```
In [123]: # feature scaling using standard scaler
          from sklearn import preprocessing
          scaler = preprocessing.StandardScaler()
          scaler.fit(table)
          X_scaled_array = scaler.transform(table)
          X_scaled = pd.DataFrame(X_scaled_array, columns = x.columns)
```

After scaling we used the K-means clustering algorithm to get the model.
Then we find the Silhoutte_score which gives us 0.24996085627555273

```
In [124]: #  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)

          0.24996085627555273
```

Silhouette score is the difference between the point and the nearest cluster that the point is not part of the cluster.
1: Means clusters are well apart from each other and clearly distinguished.
0: Means clusters are indifferent, or we can say that the distance between clusters is not significant.
-1: Means clusters are assigned in the wrong way.

No , here the feature scaling the silhouette score value didn’t moved near 1 compared to before. That means the clusters formed are near to each other unlike before feature scaling.