MACHINE LEARNING ASSIGNMENT README FILE.

1)
Firstly we have imported all the necessary libraries and imported our salaries dataset to the variable called salaries and print it using salaries.head function.

	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

Then we have splitted the dataset such that 1/3rd of data is to be reserved for the test dataset.

```
#Splitting the dataset into training set and test set
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split, cross_validate

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 1/3, random_state = 0)
```

Then we trained the model with the dataset and predicted the model.

```
#Splitting the dataset into training set and test set
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split, cross_validate

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 1/3, random_state = 0)
```

```
#Fitting multiple linear regression to the training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression()
```

```
#Predicitng the test set results
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[ 40835.10590871 123079.39940819  65134.55626083  63265.36777221
 115602.64545369 108125.8914992  116537.23969801  64199.96201652
  76349.68719258 100649.1375447 ]
```

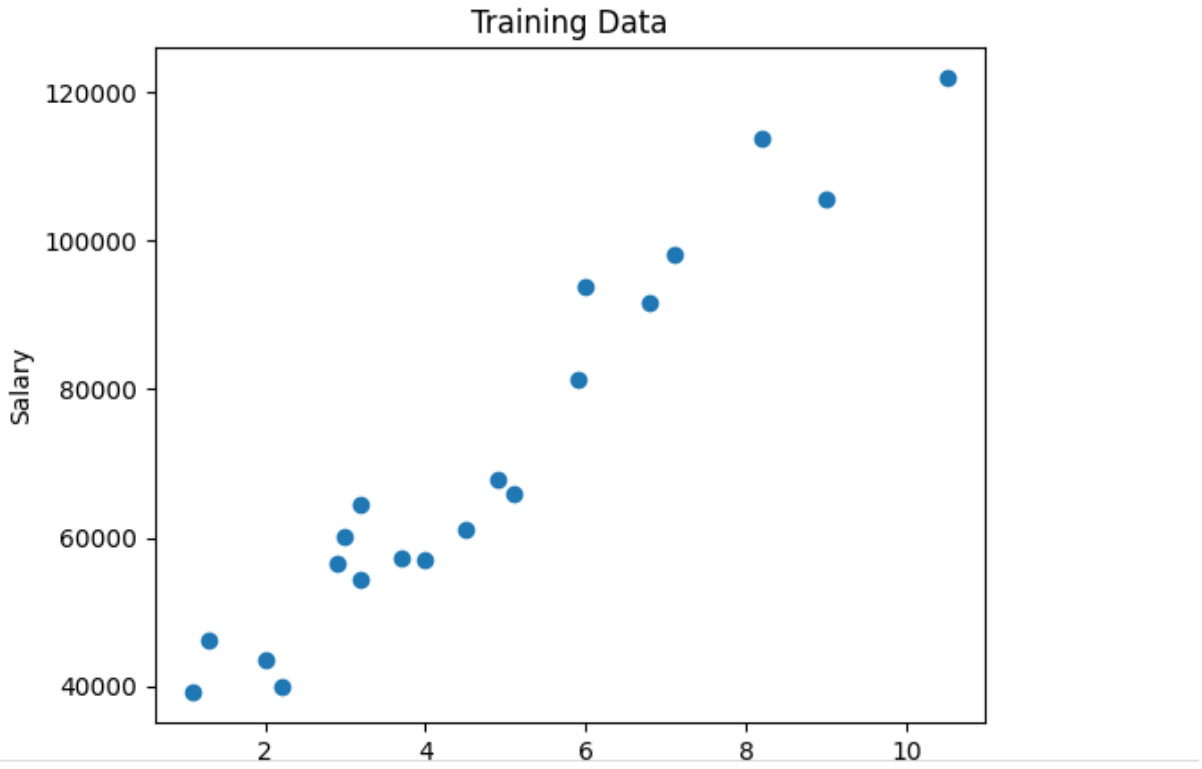
Then we have calculated the mean squared error value by importing sklearn metrics.

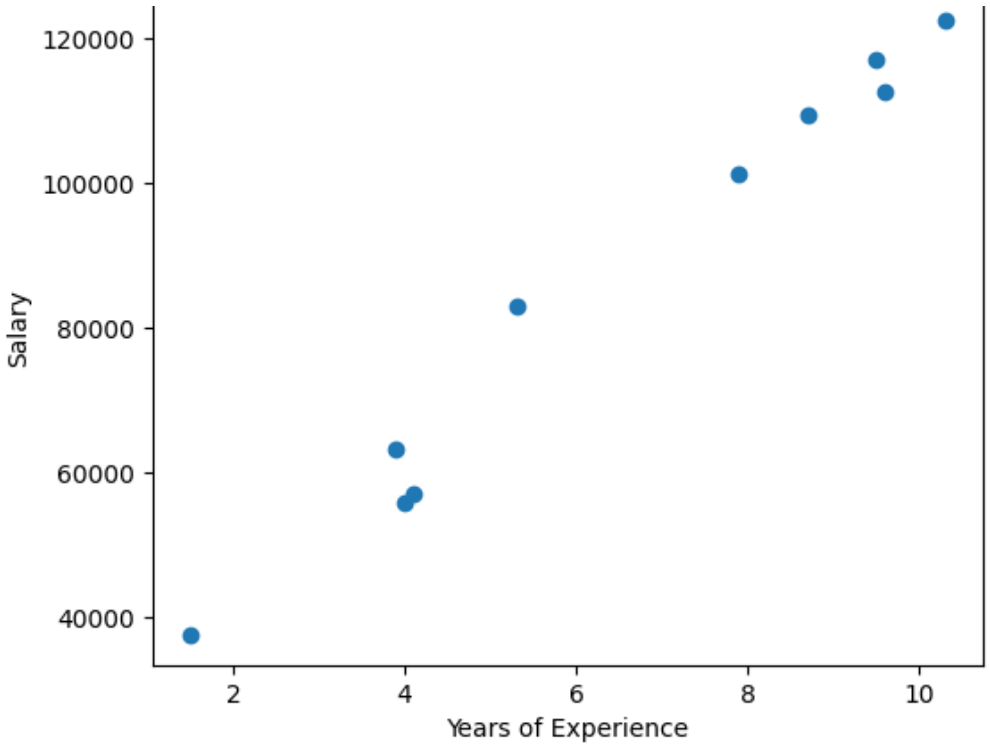
```
#Evaluating the model Calculating the R squared value
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)

#Calculate the mean_squared error
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

```
21026037.329511296
```

Then by importing the scatter function and plot function we have drawn the visualization of both the test and train dataset.



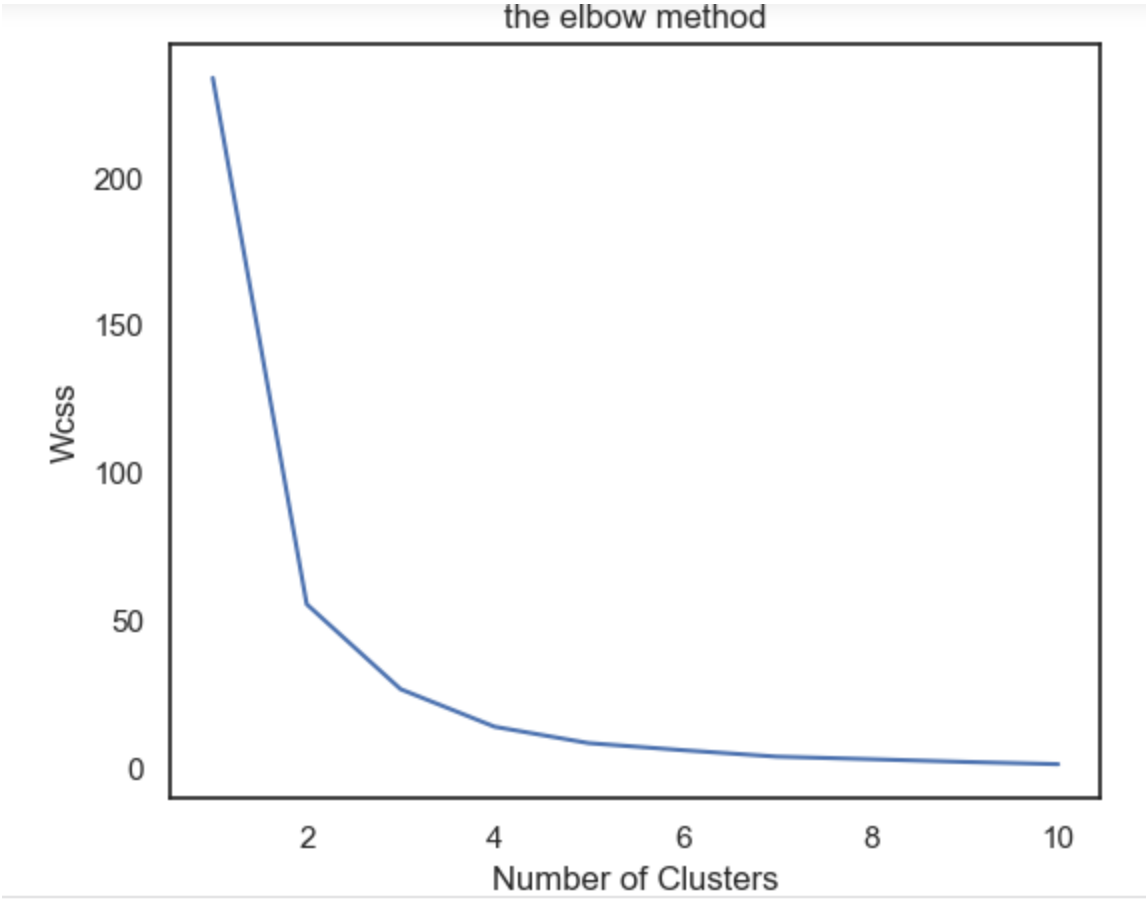


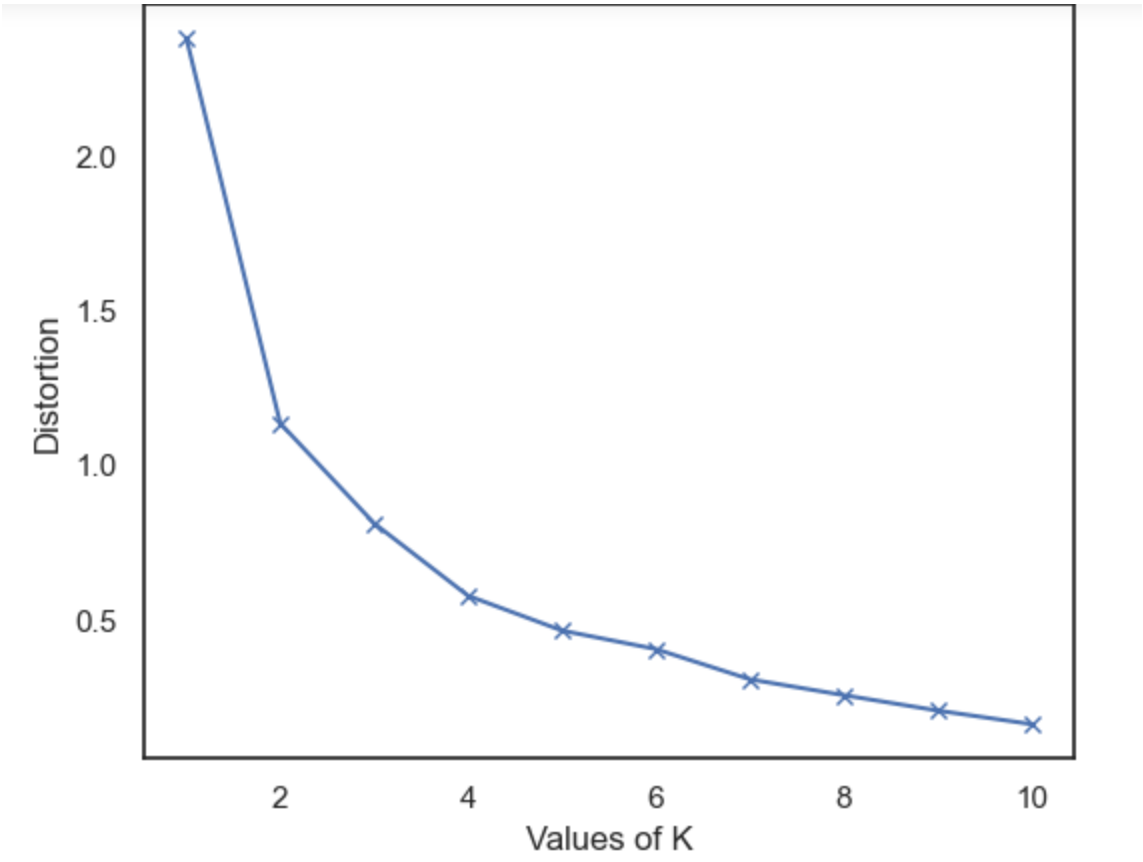
2)

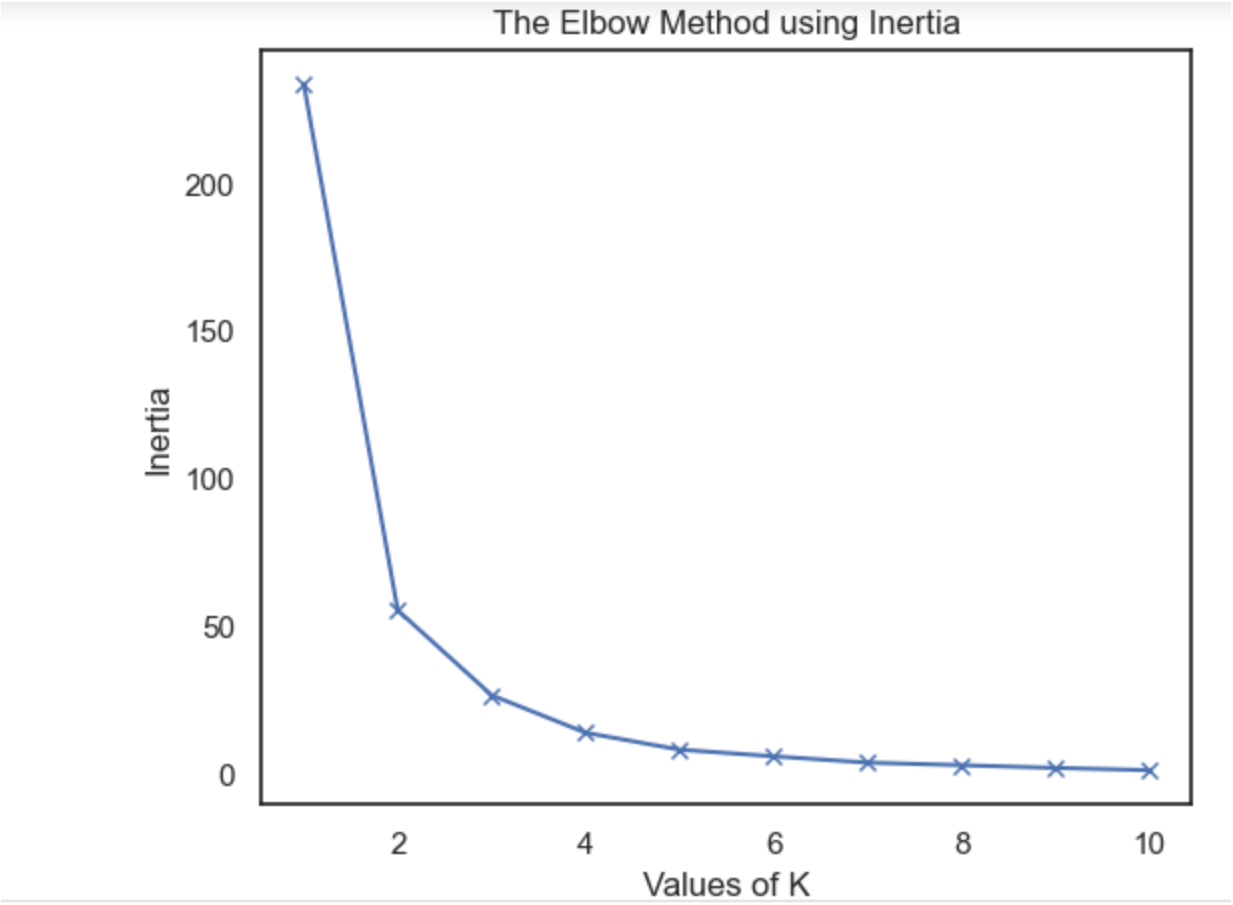
To remove all the null values, firstly we have displayed all the null values and then calculated the mean and then remove all the null values by the mean.

```
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

Then we used the elbow method to find the good number of clusters that are needed for the above k-means algorithm and plotted the graph.


the elbow method
Wcss
200
150
100
50
0
2
4
6
8
10
Number of Clusters


Distortion
2.0
1.5
1.0
0.5
2
4
6
8
10
Values of K

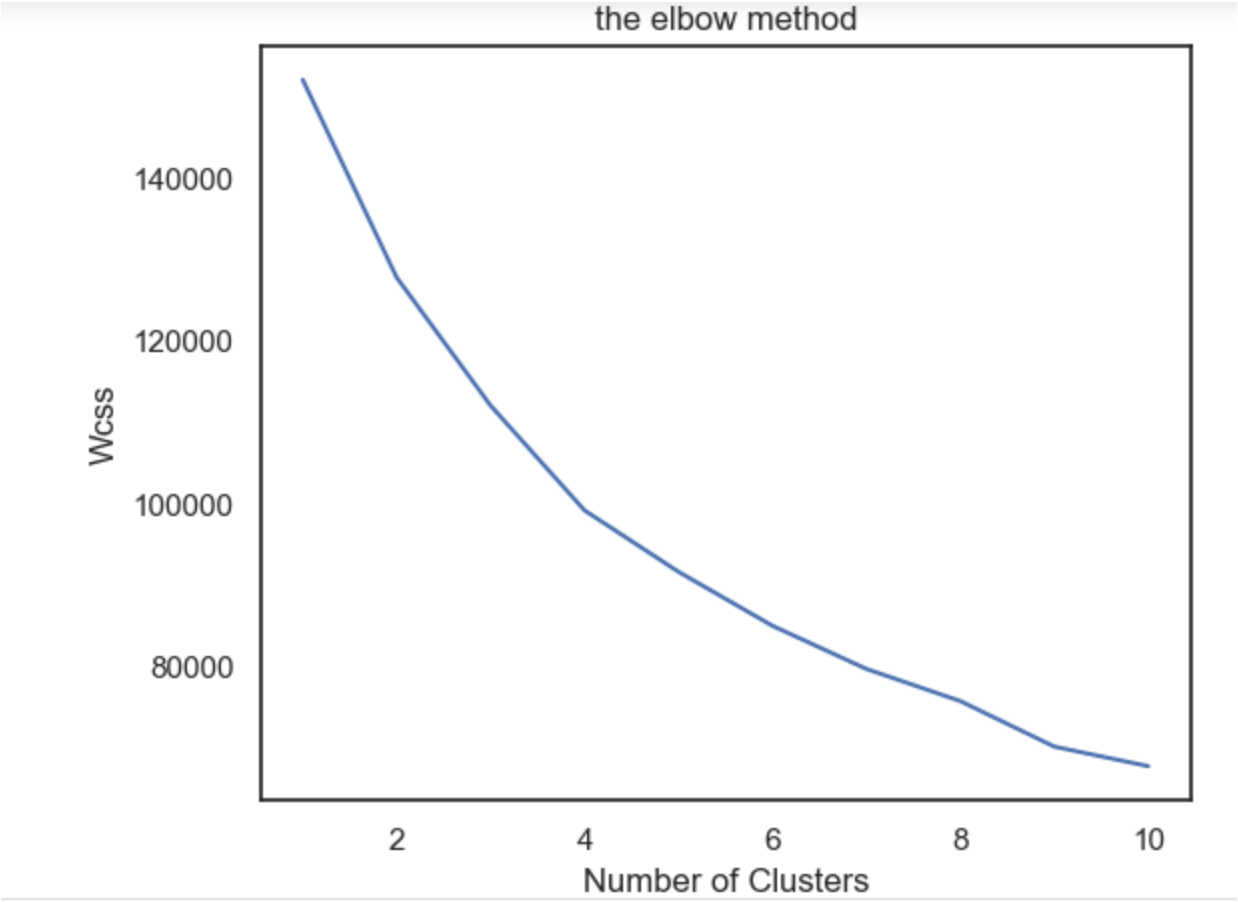


Then we have calculated the silhouette score which is a metric used to calculate the goodness of a clustering technique.

```
the Silhouette Score 0.5837513335884701
```

3)

Here for the above data we have done feature scaling and then applied k-means on the scaled featured data and drawn the graphs.



Then calculated the Silhouette score
And the value got improved because
silhouette score is a metric used to calculate the goodness of a clustering technique and here score is improved a lot after scaling.
As k-means clustering depends on euclidean distance to form the clusters, if one of the features would have much larger values than another it would dominate the distance results.
By scaling the features to the same range, the algorithm would be sensitive to all of them and not biased to the features with the greater magnitude.