MachineLearningAssignment4

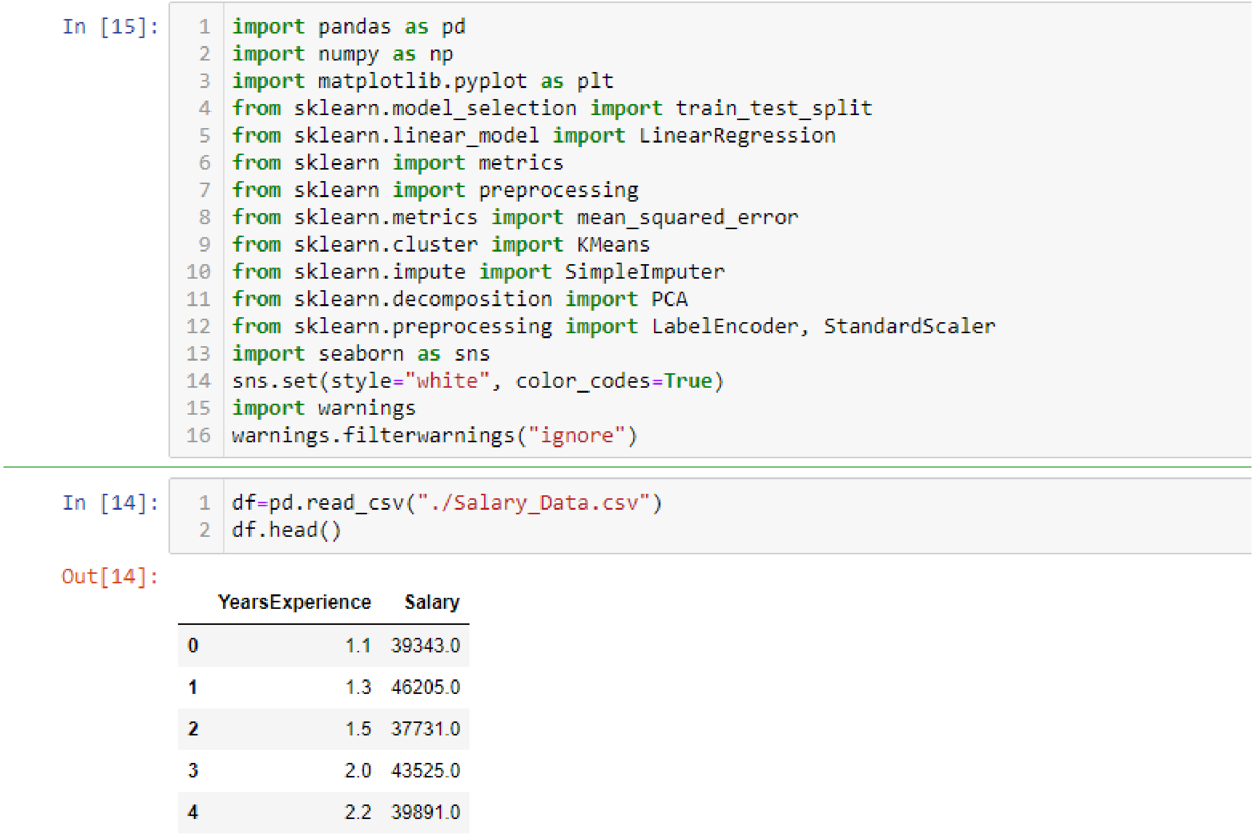
Name: Sanjana Arkatala

Student ID: 700687517

#Question1:

Read_csv() is used to read the dataset and train_test_split() is used to split the data into training and testing sets.

Mean_squared_error is used to find the mean square error.



In [15]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import mean_squared_error
from sklearn.cluster import KMeans
from sklearn.impute import SimpleImputer
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder, StandardScaler
import seaborn as sns
sns.set(style="white", color_codes=True)
import warnings
warnings.filterwarnings("ignore")
```

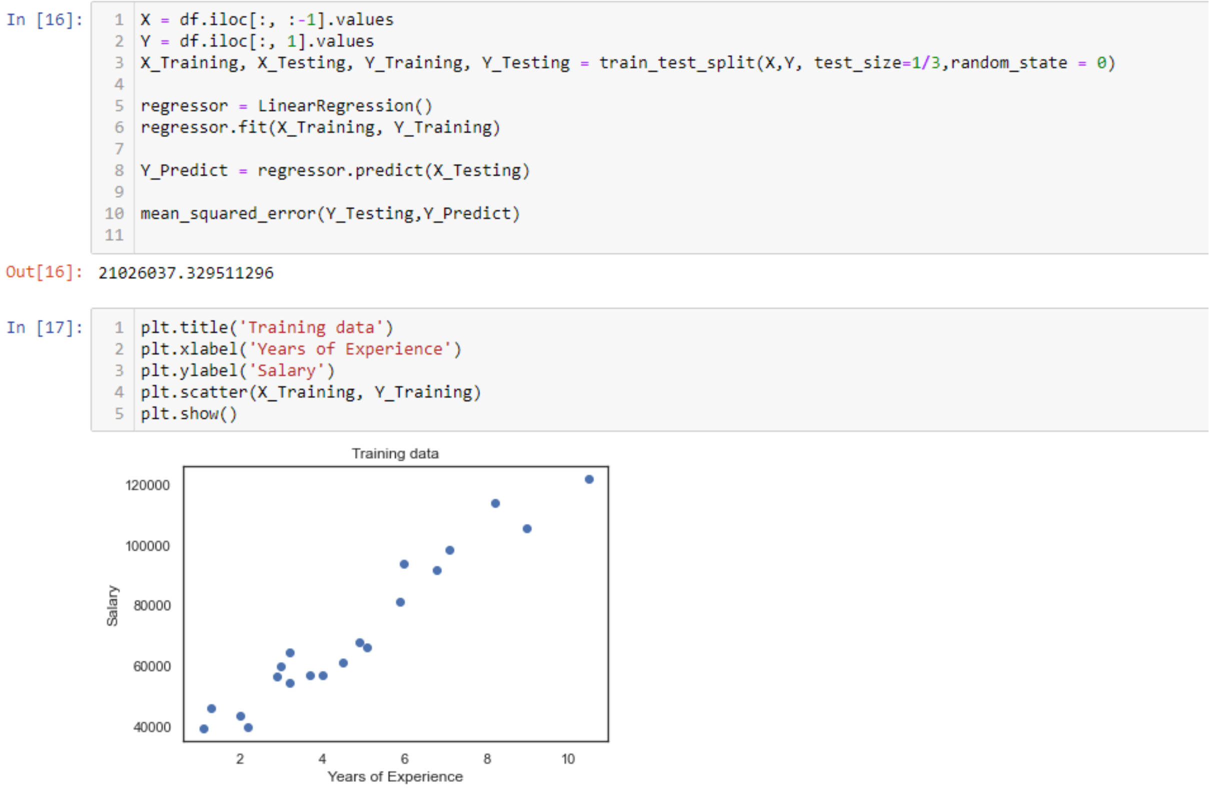
In [14]:

```python
df=pd.read_csv("./Salary_Data.csv")
df.head()
```

Out[14]:

```
   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
```

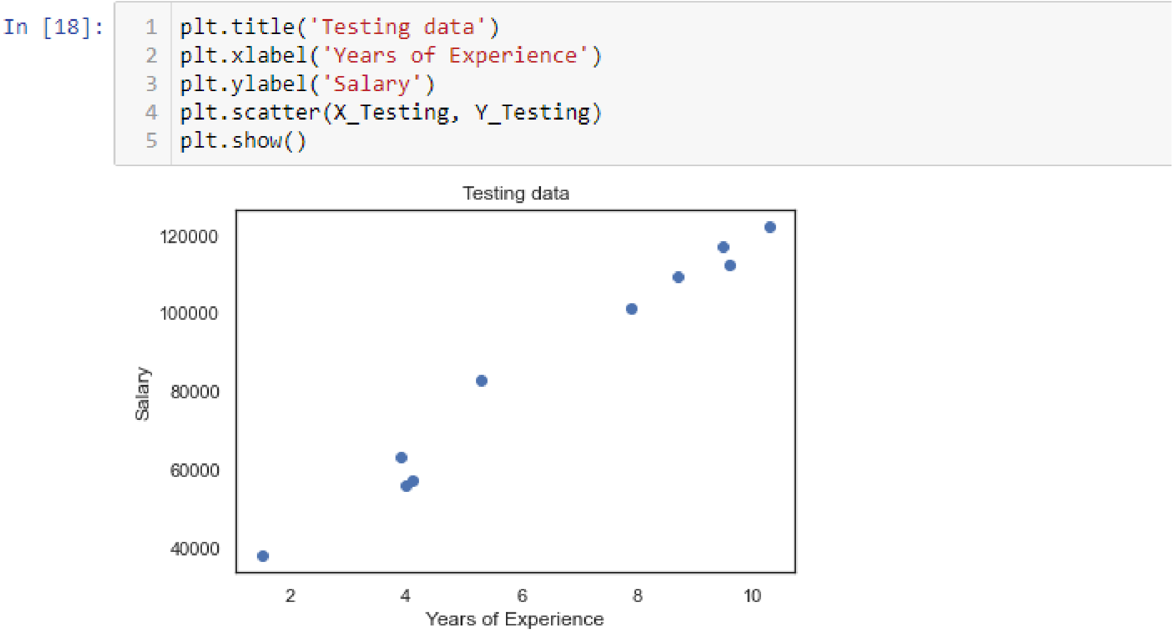
MachineLearningAssignment4



In [16]:
```
X = df.iloc[:, :-1].values
Y = df.iloc[:, 1].values
X_Training, X_Testing, Y_Training, Y_Testing = train_test_split(X,Y, test_size=1/3,random_state = 0)

regressor = LinearRegression()
regressor.fit(X_Training, Y_Training)

Y_Predict = regressor.predict(X_Testing)

mean_squared_error(Y_Testing,Y_Predict)

```

Out[16]:
```
21026037.329511296
```

In [17]:
```
plt.title('Training data')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.scatter(X_Training, Y_Training)
plt.show()
```



In [18]:
```
plt.title('Testing data')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.scatter(X_Testing, Y_Testing)
plt.show()
```

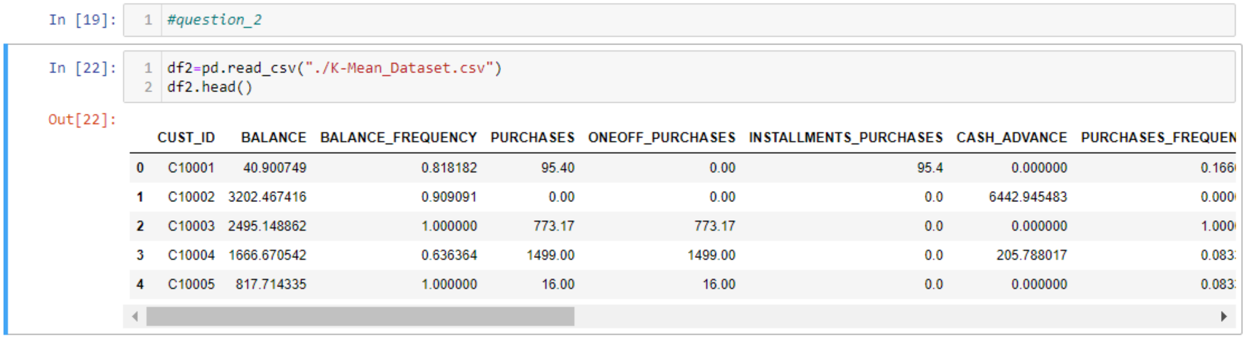
MachineLearningAssignment4

#Question-2

All the missing values are replaced with mean by using a simple imputer and the model is fitted with the resulting data.

The elbow graph bends at 2 so we consider this as the number of clusters.

The silhouette score is reduced after scaling



In [19]:

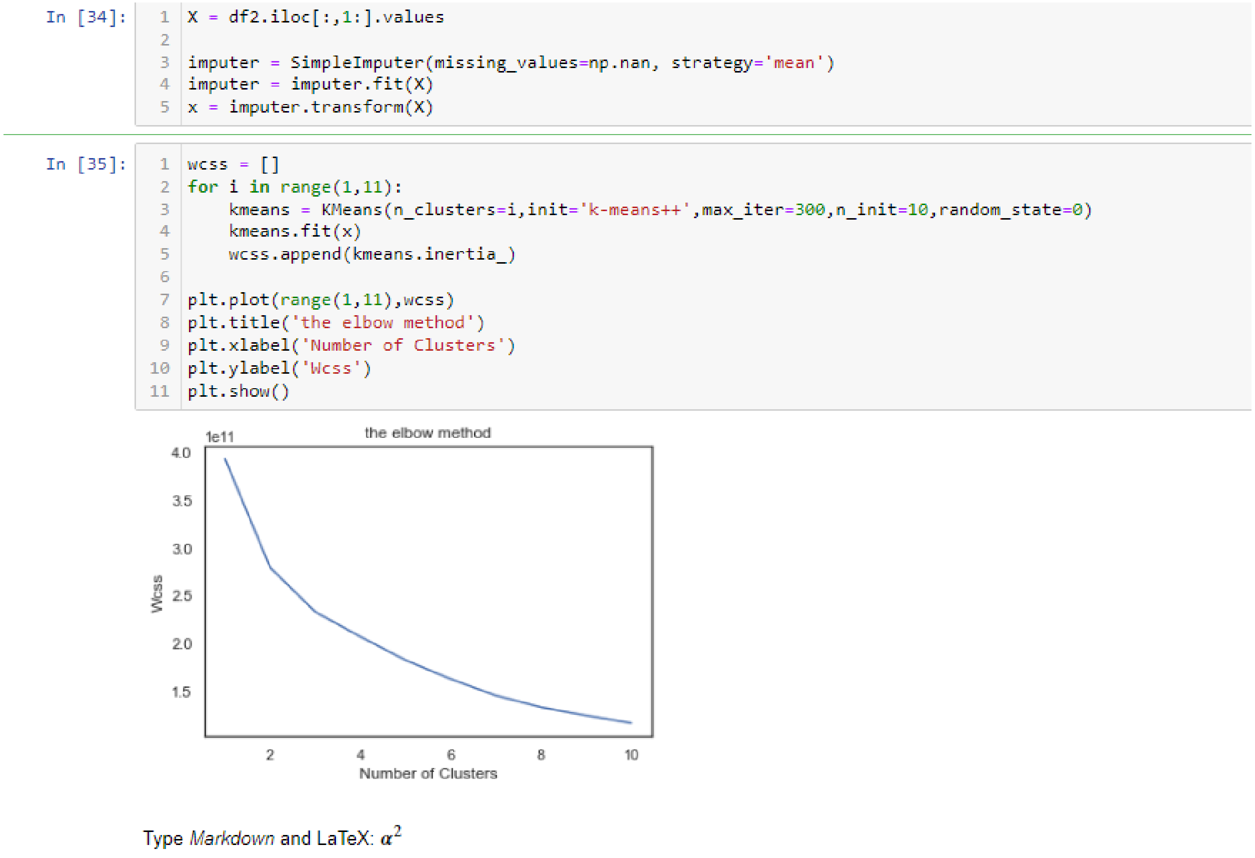
```
#question_2
```

In [22]:

```
df2=pd.read_csv("./K-Mean_Dataset.csv")
df2.head()
```

Out[22]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUEN
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083



In [34]:

```
X = df2.iloc[:,1:].values

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(X)
x = imputer.transform(X)
```

In [35]:

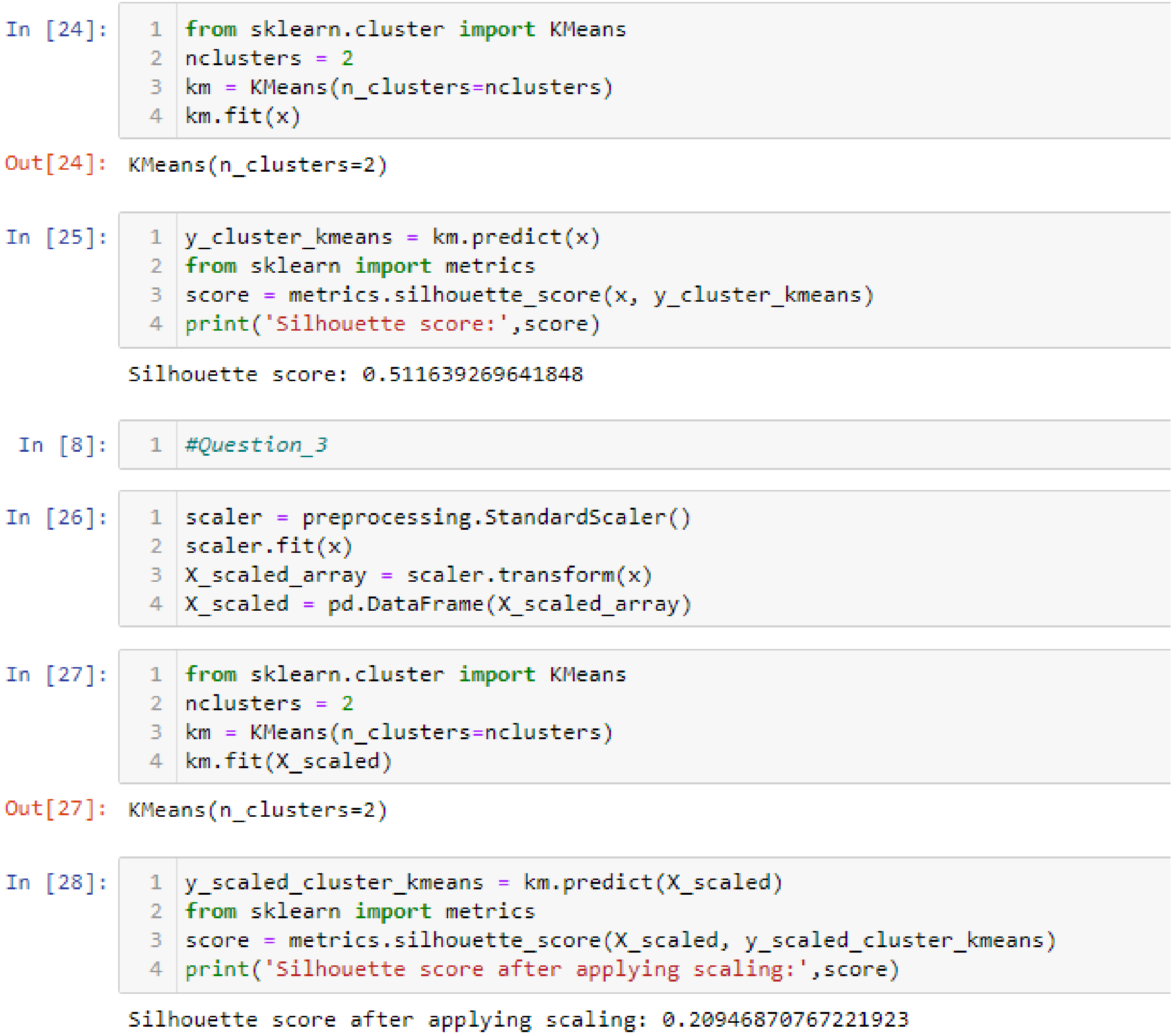
```
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)

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

Type *Markdown* and LaTeX: α^2

MachineLearningAssignment4

#Question3

Silhouette score is reduced after scaling



```
In [24]:  from sklearn.cluster import KMeans
          nclusters = 2
          km = KMeans(n_clusters=nclusters)
          km.fit(x)

Out[24]:  KMeans(n_clusters=2)

In [25]:  y_cluster_kmeans = km.predict(x)
          from sklearn import metrics
          score = metrics.silhouette_score(x, y_cluster_kmeans)
          print('Silhouette score:',score)

          Silhouette score: 0.511639269641848

In [8]:   #Question_3

In [26]:  scaler = preprocessing.StandardScaler()
          scaler.fit(x)
          X_scaled_array = scaler.transform(x)
          X_scaled = pd.DataFrame(X_scaled_array)

In [27]:  from sklearn.cluster import KMeans
          nclusters = 2
          km = KMeans(n_clusters=nclusters)
          km.fit(X_scaled)

Out[27]:  KMeans(n_clusters=2)

In [28]:  y_scaled_cluster_kmeans = km.predict(X_scaled)
          from sklearn import metrics
          score = metrics.silhouette_score(X_scaled, y_scaled_cluster_kmeans)
          print('Silhouette score after applying scaling:',score)

          Silhouette score after applying scaling: 0.20946870767221923
```

Github link: https://github.com/Sanjana9791/MachineLearningAssignment4.git