ML_Assignment4_Farhana_700735341

Programming elements:

Linear Regression, K-Means Clustering and Data Analysis

1. Apply Linear Regression to the provided dataset using underlying steps.
   a. Import the given "Salary_Data.csv"
   b. Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.
   c. Train and predict the model.
   d. Calculate the mean_squared error
   e. Visualize both train and test data using scatter plot.

```
In [1]:
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")
```

Here, we are import pandas,numpy,matplotlib.pyplot , train_test_split,LinearRegression,metrics,preprocessing,mean_squared_error,KMeans,SimpleImputer, PCA, LabelEncoder, StandardScaler, seaborn as sn and warnings.

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 [3]:

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

Out[3]:

	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

Here, we are reading Salary_Data csv file , the above is filepath given in command and the output shown for df.head().

df=pd.read_csv("datasets/datasets/Salary_Data.csv")

df.head()

	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

```
In [4]:
X = df.iloc[:, :-1].values
Y = df.iloc[:, 1].values
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X,Y, test_size=1/3,random_state = 0)
```

Here,
.iloc[] is primarily integer position based (from 0 to length-1 of the axis), but may also be used with a boolean array. We used Train_test_split

X = df.iloc[:, :-1].values

Y = df.iloc[:, 1].values

X_Train, X_Test, Y_Train, Y_Test = train_test_split(X,Y, test_size=1/3,random_state = 0)

```
In [5]:
regressor = LinearRegression()
regressor.fit(X_Train, Y_Train)

Y_Pred = regressor.predict(X_Test)
```

regressor = LinearRegression()

regressor.fit(X_Train, Y_Train)

Y_Pred = regressor.predict(X_Test)

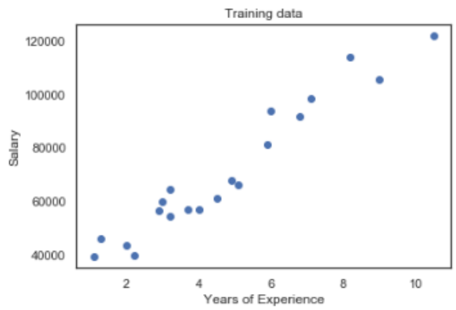
```
In [6]:
mean_squared_error(Y_Test,Y_Pred)
Out[6]: 21026037.329511296
```

mean_squared_error(Y_Test,Y_Pred)

`output:` 21026037.329511296

In [7]:

```
plt.title('Training data')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.scatter(X_Train, Y_Train)
plt.show()
```



plt.title('Training data')

plt.xlabel('Years of Experience')

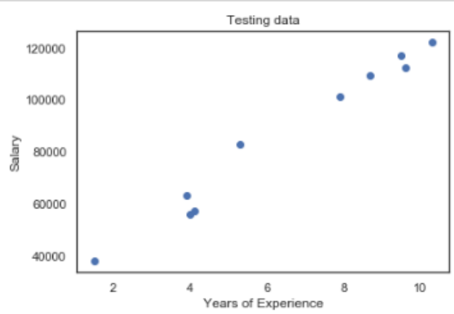
plt.ylabel('Salary')

plt.scatter(X_Train, Y_Train)

plt.show()

In [8]:

```
plt.title('Testing data')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.scatter(X_Test, Y_Test)
plt.show()
```



plt.title('Testing data')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.scatter(X_Test, Y_Test)

plt.show()

In [9]:
```
df2=pd.read_csv("datasets/datasets/K-Mean_Dataset.csv")
df2.head()
```

Out[9]:

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

Out[9]:

PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMENTS_FREQUENCY	CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_
0.166667	0.000000	0.083333	0.000000	
0.000000	0.000000	0.000000	0.250000	
1.000000	1.000000	0.000000	0.000000	
0.083333	0.083333	0.000000	0.083333	
0.083333	0.083333	0.000000	0.000000	

Out[9]:

CASH_ADVANCE_FREQUENCY	CASH_ADVANCE_TRX	PURCHASES_TRX	CREDIT_LIMIT	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
0.000000	0	2	1000.0	201.802084	139.509787	0.000000	12
0.250000	4	0	7000.0	4103.032597	1072.340217	0.222222	12
0.000000	0	12	7500.0	622.066742	627.284787	0.000000	12
0.083333	1	1	7500.0	0.000000	NaN	0.000000	12
0.000000	0	1	1200.0	678.334763	244.791237	0.000000	12

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

df2.head()

Once the data is loaded in, let's take a quick peek at the first 5 values using the `head()` method

---

In [30]:
```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(X)
X = imputer.transform(X)
```

imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

imputer = imputer.fit(X)

X = imputer.transform(X)

```
In [31]: df2.columns

Out[31]: Index(['CUST_ID', 'BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES',
                'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE',
                'PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY',
                'PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY',
                'CASH_ADVANCE_TRX', 'PURCHASES_TRX', 'CREDIT_LIMIT', 'PAYMENTS',
                'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT', 'TENURE'],
               dtype='object')
```

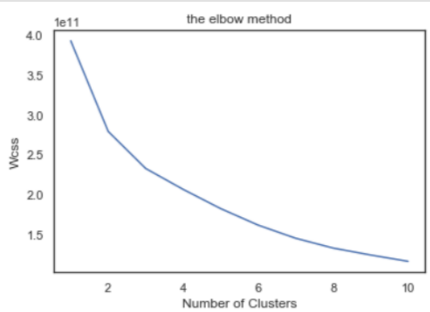
df2.columns

output:

Index(['CUST_ID', 'BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES',
 'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE',
 'PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY',
 'PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY',
 'CASH_ADVANCE_TRX', 'PURCHASES_TRX', 'CREDIT_LIMIT', 'PAYMENTS',
 'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT', 'TENURE'],
 dtype='object')

```
In [32]: 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()
```



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()

```
In [33]:  from sklearn.cluster import KMeans
          nclusters = 4 # this is the k in kmeans
          km = KMeans(n_clusters=nclusters)
          km.fit(X)

Out[33]:  KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
                 n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
                 random_state=None, tol=0.0001, verbose=0)
```

from sklearn.cluster import KMeans

nclusters = 4 # this is the k in kmeans

km = KMeans(n_clusters=nclusters)

km.fit(X)

output:

KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)

```
In [34]:  y_cluster_kmeans = km.predict(X)
          from sklearn import metrics
          score = metrics.silhouette_score(X, y_cluster_kmeans)
          print('Silhouette score:',score)

          Silhouette score: 0.464760862519683
```

y_cluster_kmeans = km.predict(X)

from sklearn import metrics

score = metrics.silhouette_score(X, y_cluster_kmeans)

print('Silhouette score:',score)

output:

Silhouette score: 0.464760862519683

```
In [35]:  columns=['']
```

```
In [36]:  scaler = preprocessing.StandardScaler()
          scaler.fit(X)
          X_scaled_array = scaler.transform(X)
          X_scaled = pd.DataFrame(X_scaled_array, columns = df2.columns[1:])
```

scaler = preprocessing.StandardScaler()

scaler.fit(X)

X_scaled_array = scaler.transform(X)

X_scaled = pd.DataFrame(X_scaled_array, columns = df2.columns[1:])

```
In [37]:  from sklearn.cluster import KMeans
          nclusters = 4
          km = KMeans(n_clusters=nclusters)
          km.fit(X_scaled)

Out[37]:  KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
                 n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
                 random_state=None, tol=0.0001, verbose=0)
```

from sklearn.cluster import KMeans

nclusters = 4

km = KMeans(n_clusters=nclusters)

km.fit(X_scaled)

output:

KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
n_clusters=4, n_init=10, n_jobs=None, precompute_distances='auto',
random_state=None, tol=0.0001, verbose=0)

```
In [38]:  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: 0.1976074492720698
```

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)

output:
Silhouette score after applying scaling: 0.1976074492720698

GitHub link: https://github.com/AishaFar/ML_Assignment4_Farhana_700735341
VideoLink: https://youtu.be/hB9vzteclKI