**Assignment - 4**

Programming elements: Linear Regression, K-Means Clustering and Data Analysis

In class programming:

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.

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")

data=pd.read\_csv("Salary\_Data.csv")

data.head()

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

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

X\_Train, X\_Test, Y\_Train, Y\_Test = train\_test\_split(X,Y, test\_size=1/3,random\_state = 0)

regression = LinearRegression()

regression.fit(X\_Train, Y\_Train)

Y\_Pred = regression.predict(X\_Test)

mean\_squared\_error(Y\_Test,Y\_Pred)

plt.title('Training data')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.scatter(X\_Train, Y\_Train)

plt.show()

plt.title('Testing data')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.scatter(X\_Test, Y\_Test)

plt.show()

2. Apply K means clustering in the dataset provided:

• Remove any null values by the mean.

• Use the elbow method to find a good number of clusters with the K-Means algorithm

• Calculate the silhouette score for the above clustering

data2=pd.read\_csv("K-Mean\_Dataset.csv")

data2.head()

X = data2.iloc[:,1:].values

imputer = SimpleImputer(missing\_values=np.nan, strategy='mean')

imputer = imputer.fit(X)

X = imputer.transform(X)

Data2.columns

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()

from sklearn.cluster import KMeans

nclusters = 4 # this is the k in kmeans

km = KMeans(n\_clusters=nclusters)

km.fit(X)

y\_cluster\_kmeans = km.predict(X)

from sklearn import metrics

score = metrics.silhouette\_score(X, y\_cluster\_kmeans)

print('Silhouette score:',score)

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

columns=['']

scaler = preprocessing.StandardScaler()

scaler.fit(X)

X\_scaled\_array = scaler.transform(X)

X\_scaled = pd.DataFrame(X\_scaled\_array, columns = data2.columns[1:])

from sklearn.cluster import KMeans

nclusters = 4

km = KMeans(n\_clusters=nclusters)

km.fit(X\_scaled)

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 is the measure for the performance or accuracy of the clustering model ranging from -1 to 1, where 1 is said to be best and -1 is worst. Here in this model, after feature scaling the Silhouette score of the model reduced from 0.46 to 0.19, this means it’s quite reduced. Feature scaling and other methods of normalizing the data will improve the stability of the model accuracy. Indeed this is a large dataset with many feature variables, so this might require much tuning of the normalization parameters to improve the score and all the data are much spread with various ranges.

**https://github.com/NXI57230/Assignment1\_700725723**