Machine Learning Assginment -4

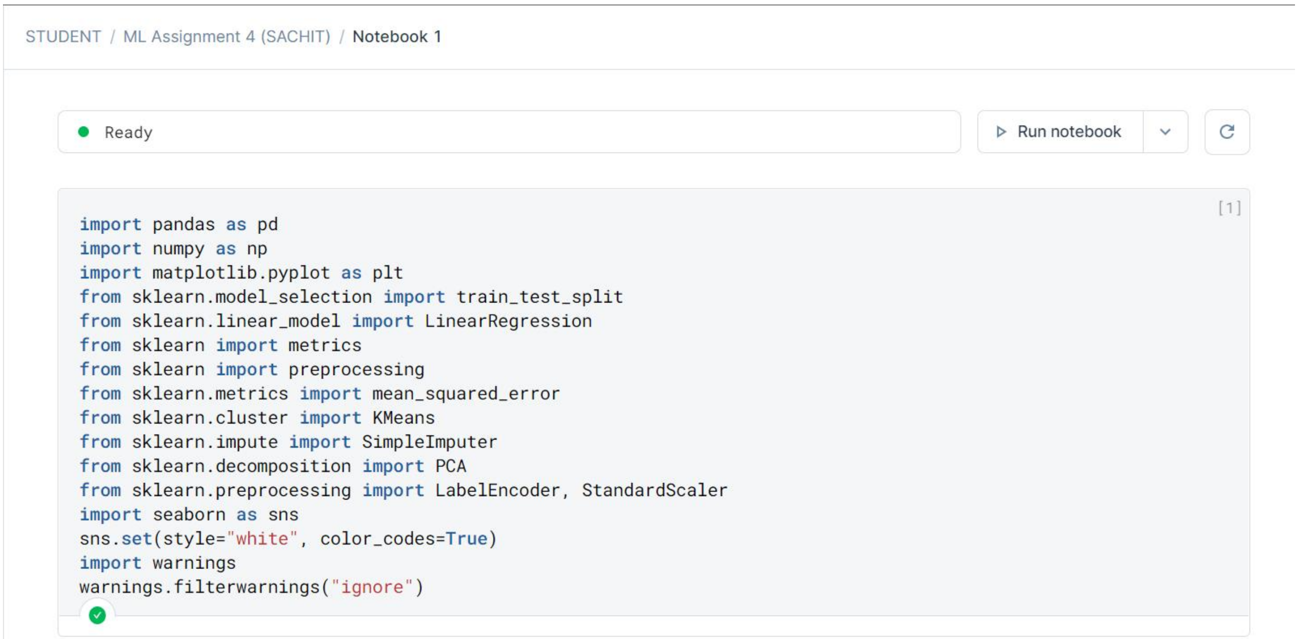
Name	**:**	**Sachit Kumar Tadishetty**
700#	**:**	**700734682**
UCM ID	**:**	**SXT46820**

VideoLink:
https://drive.google.com/drive/folders/1xq28VUjgRu-2ng9uIL5182B9izNjOkDe?usp=share_link

Github Link : https://github.com/sachit46820/ML-Assignment

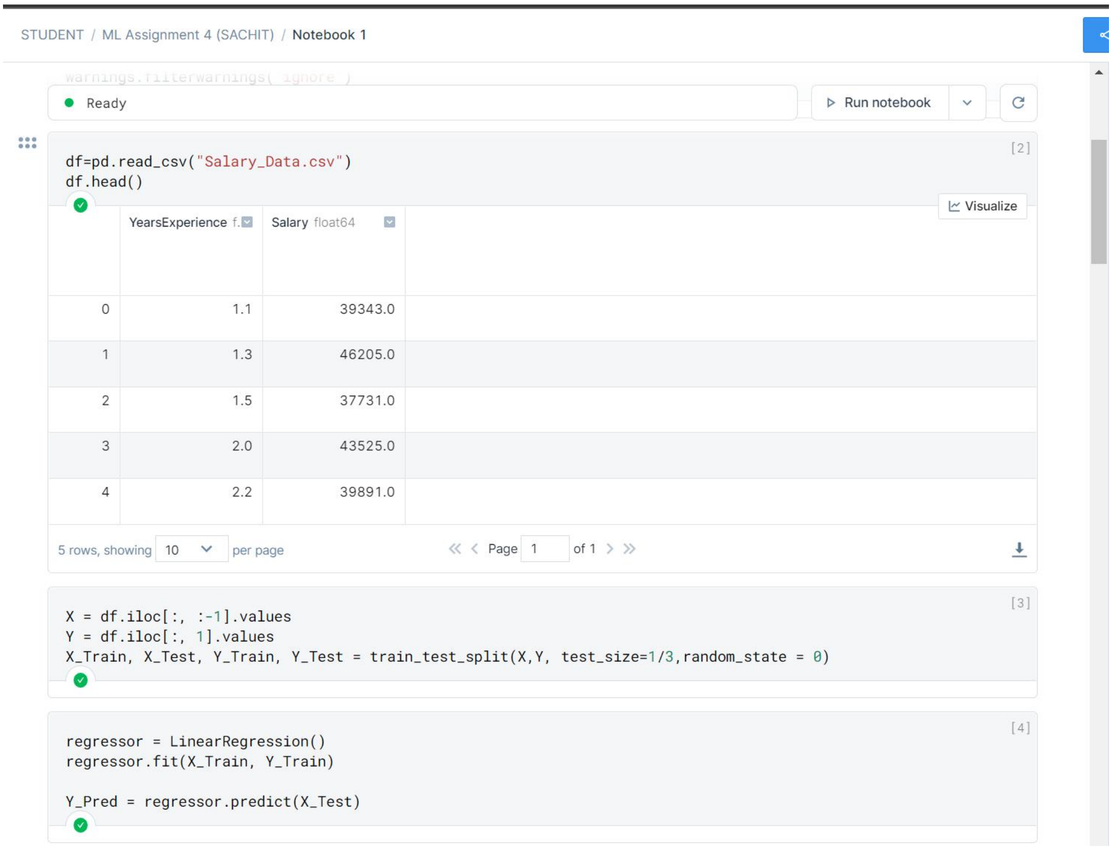
1) Apply Linear Regression to the provided dataset using underlying steps.

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")
```

- Used Linear regression to find the relationship between variables.
- Imported simpleimputer which helps in handling the missing data in the predictive dataset.
- Imported Kmeans for clustering the data points into K-clusters by minimising the variance.
- Imported metrics for evaluating the algorithms.



```
warnings.filterwarnings('ignore')
```

[2]

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

	YearsExperience f...	Salary float64
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

5 rows, showing 10 per page Page 1 of 1

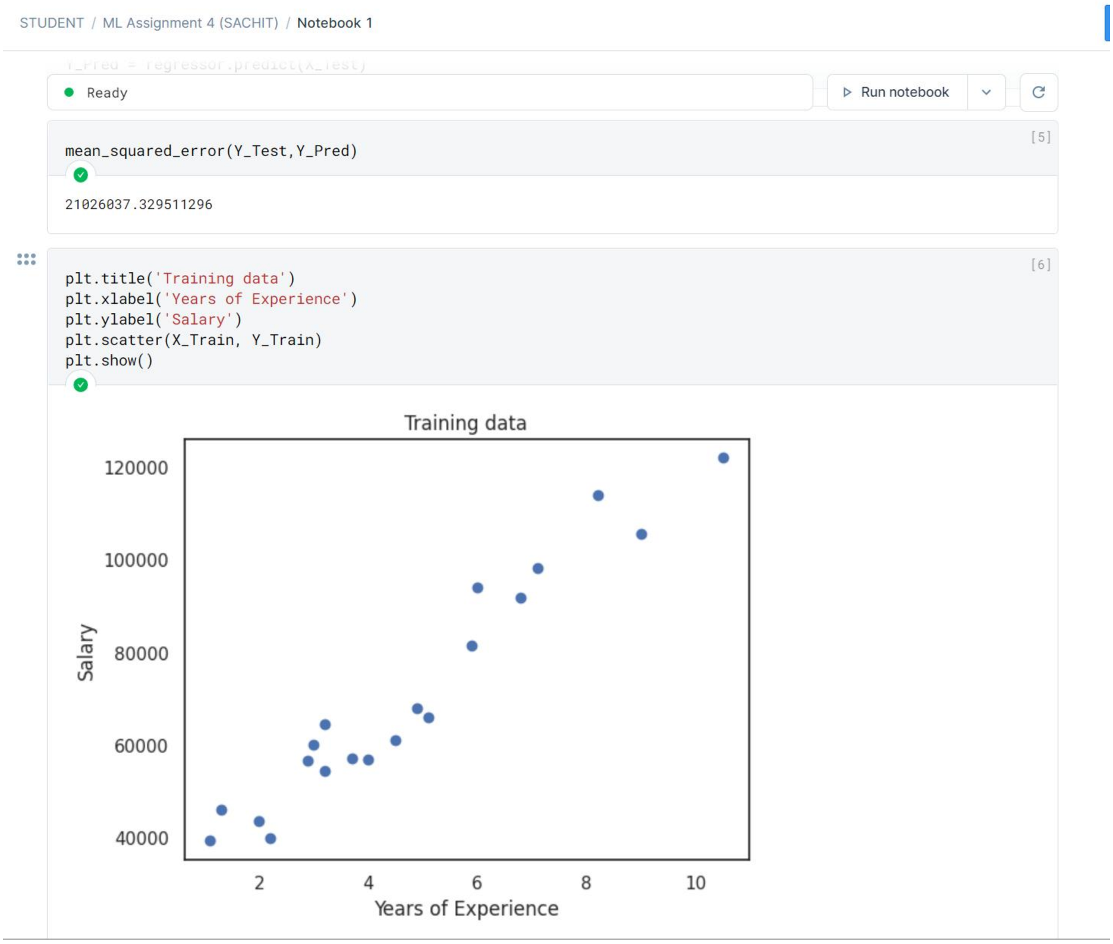
[3]

```
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)
```

[4]

```
regressor = LinearRegression()
regressor.fit(X_Train, Y_Train)

Y_Pred = regressor.predict(X_Test)
```



```
Y_Pred = regressor.predict(X_Test)
```

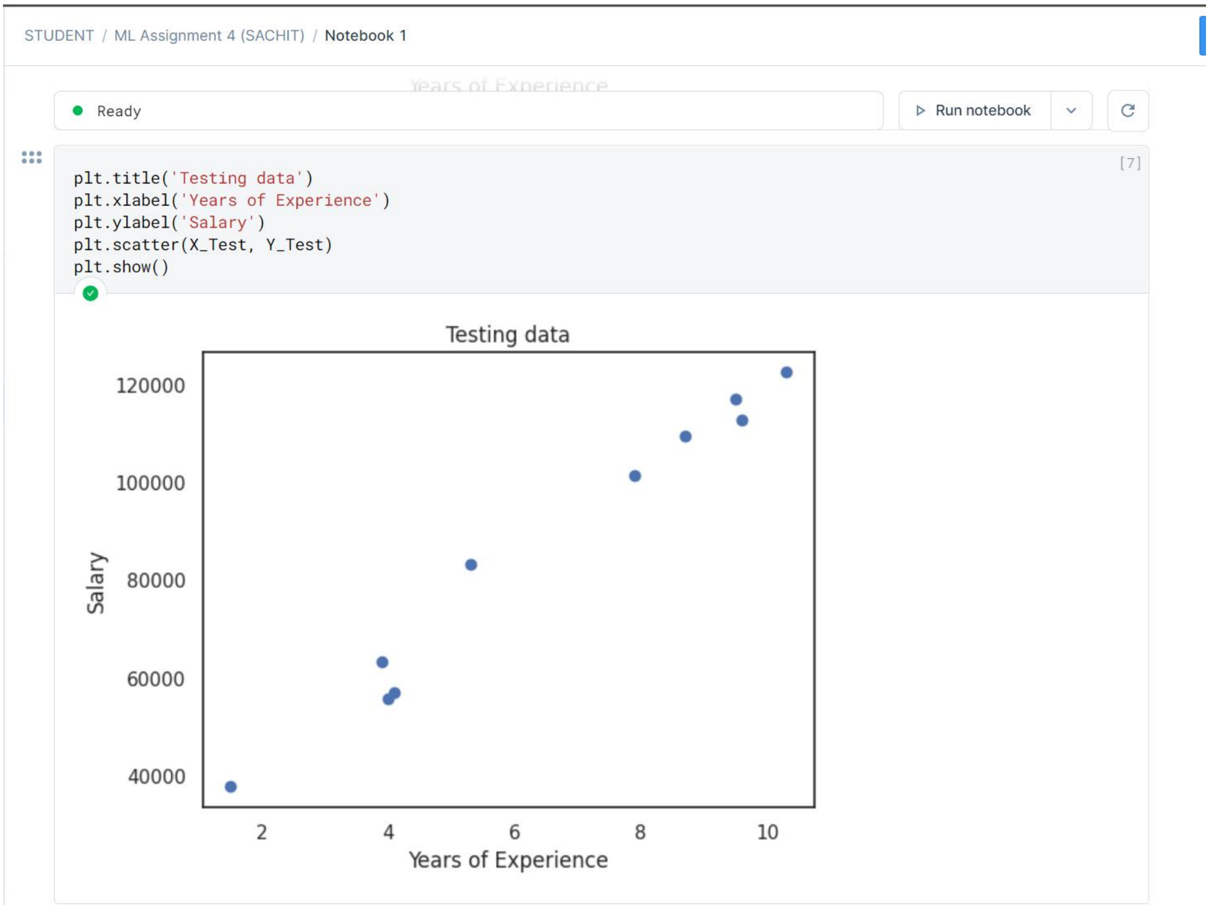
[5]

```
mean_squared_error(Y_Test,Y_Pred)
```

```
21026037.329511296
```

[6]

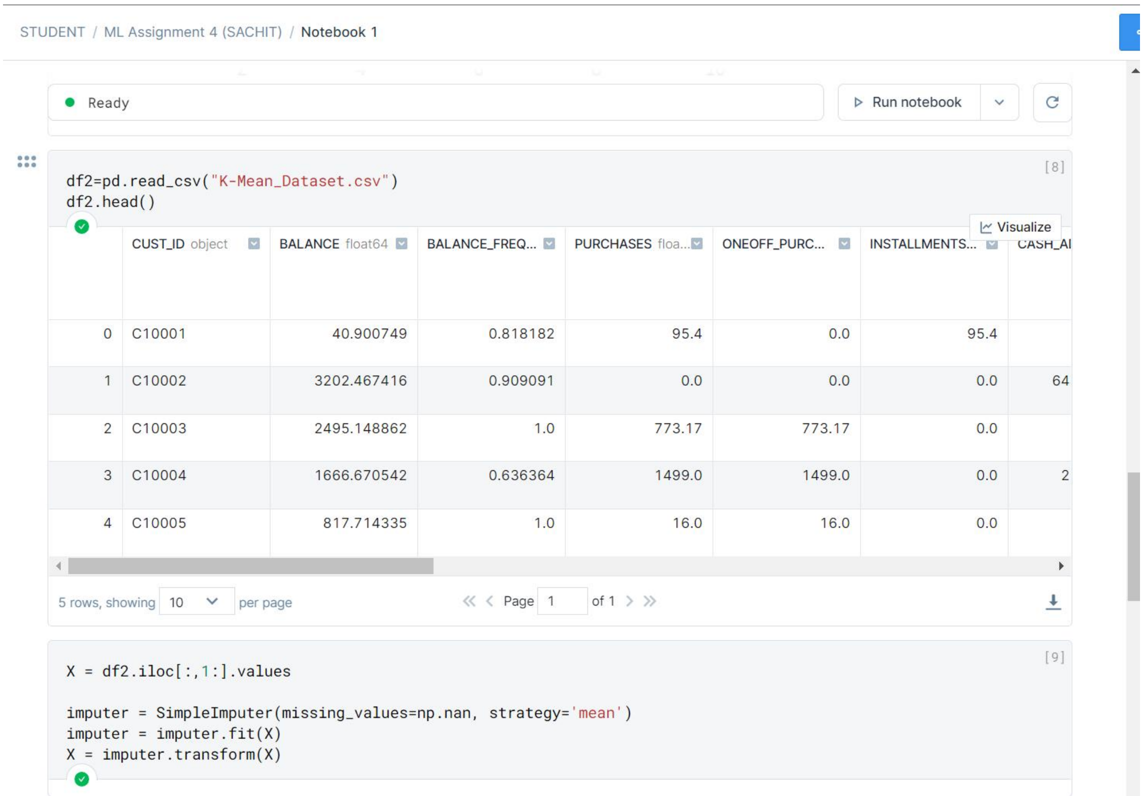
```
plt.title('Training data')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.scatter(X_Train, Y_Train)
plt.show()
```



```
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.



STUDENT / ML Assignment 4 (SACHIT) / Notebook 1

● Ready

[8]

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

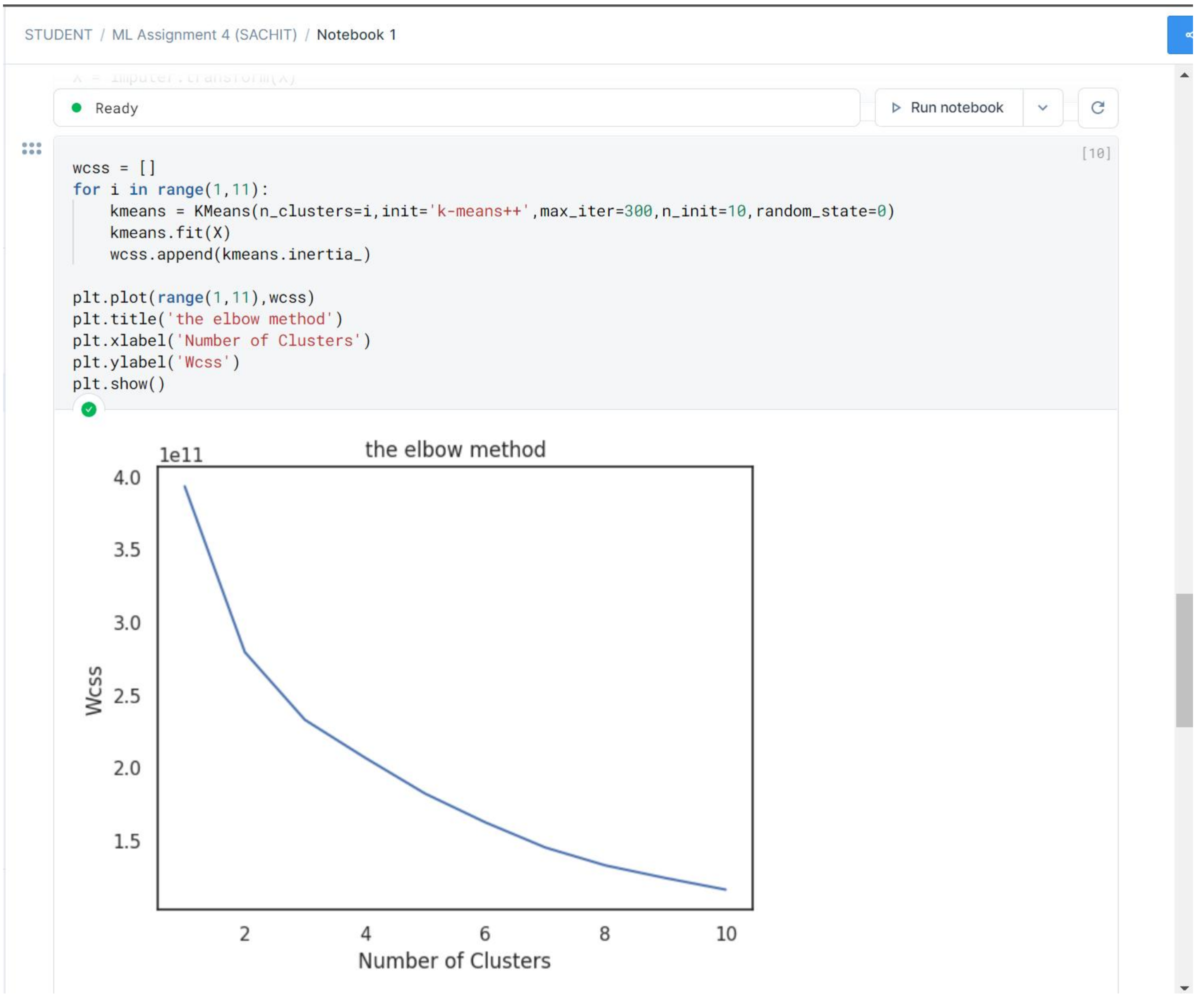
	CUST_ID object	BALANCE float64	BALANCE_FREQ...	PURCHASES floa...	ONEOFF_PURC...	INSTALLMENTS...	CASH_AI
0	C10001	40.900749	0.818182	95.4	0.0	95.4	
1	C10002	3202.467416	0.909091	0.0	0.0	0.0	64
2	C10003	2495.148862	1.0	773.17	773.17	0.0	
3	C10004	1666.670542	0.636364	1499.0	1499.0	0.0	2
4	C10005	817.714335	1.0	16.0	16.0	0.0	

5 rows, showing 10 per page   Page 1 of 1

[9]

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

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



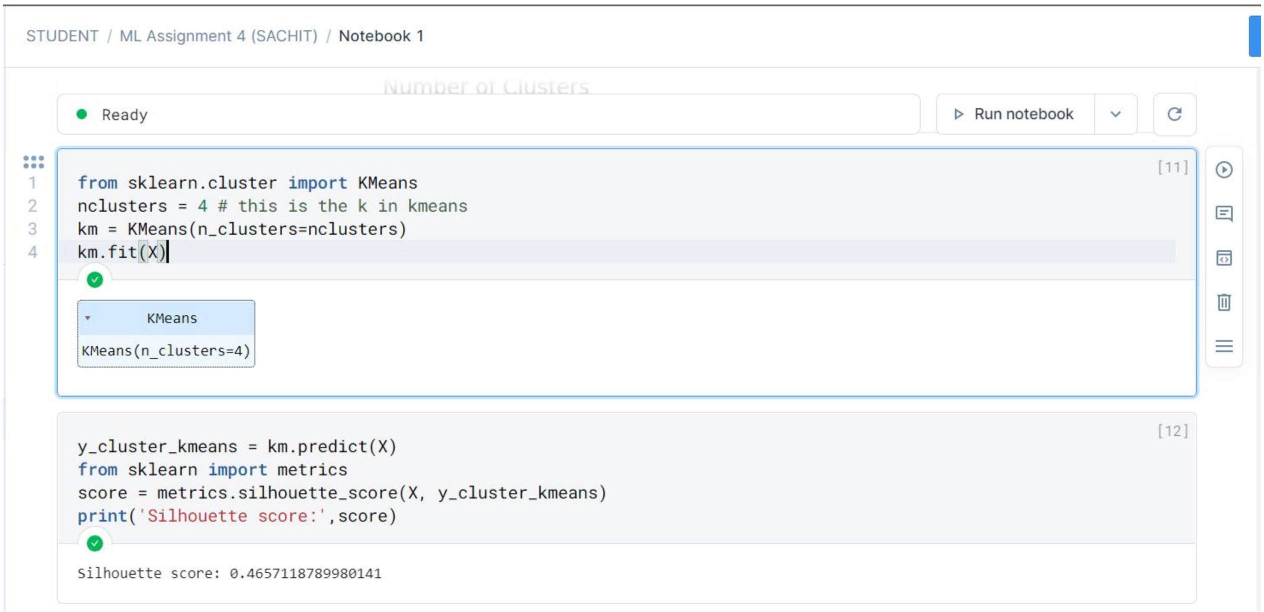
STUDENT / ML Assignment 4 (SACHIT) / Notebook 1

● Ready

[10]

```
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()
```



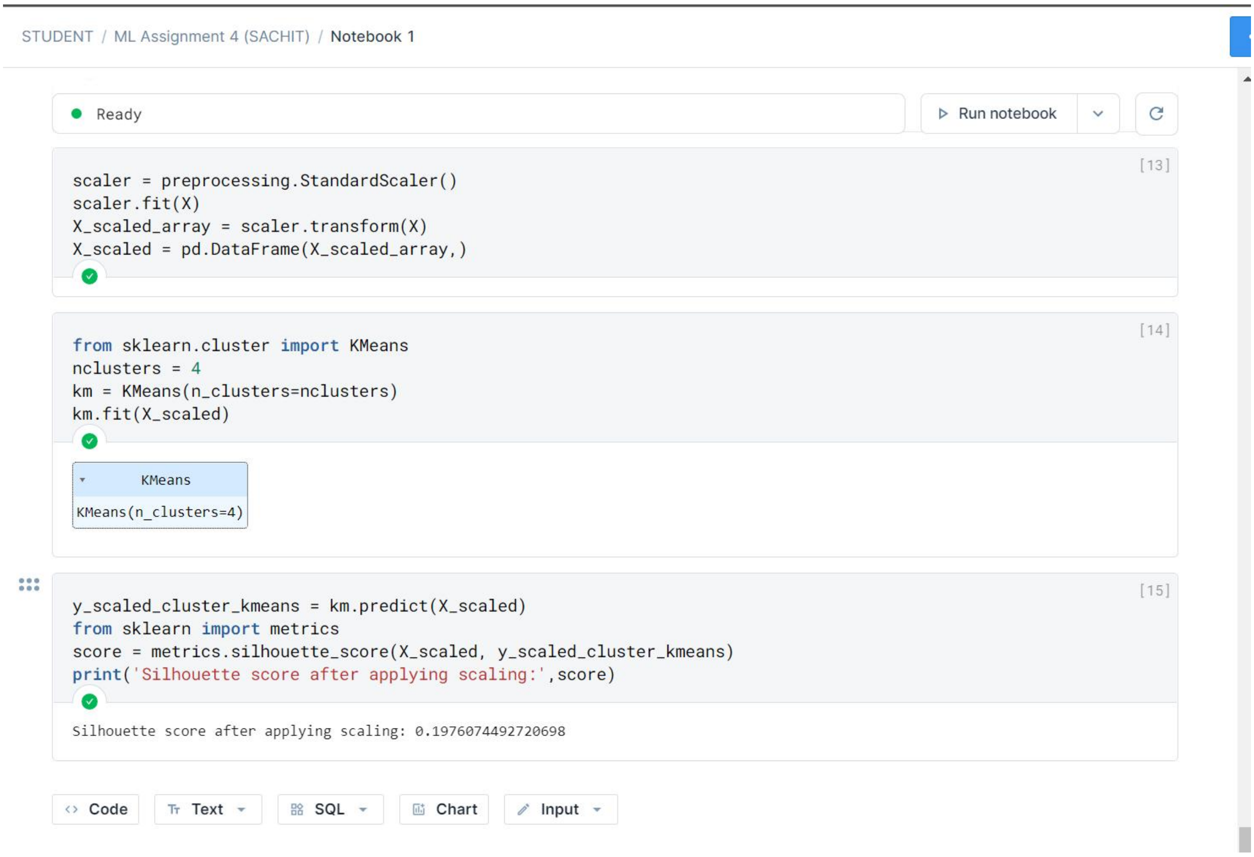
```
from sklearn.cluster import KMeans
nclusters = 4 # this is the k in kmeans
km = KMeans(n_clusters=nclusters)
km.fit(X)
```

```
KMeans
KMeans(n_clusters=4)
```

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

```
Silhouette score: 0.4657118789980141
```

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



```
scaler = preprocessing.StandardScaler()
scaler.fit(X)
X_scaled_array = scaler.transform(X)
X_scaled = pd.DataFrame(X_scaled_array,)
```

```
from sklearn.cluster import KMeans
nclusters = 4
km = KMeans(n_clusters=nclusters)
km.fit(X_scaled)
```

```
KMeans
KMeans(n_clusters=4)
```

```
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
```

No , the Silhouette score did not improve after feature scaling . because scaling will basically result in models that are disproportionally influenced by the subset of features on a large scale hence the performance decreases.