Experiment 11

Implementation of learning algorithms for an application

Name : Rahul Goel

Reg No: RA1911030010094

Aim:

A) Implementation of a Linear Regression algorithm to predict student's scores using the given dataset.
B) Implementation of Support Vector Classification algorithm to classify the cases of breast cancer

using the given dataset.
C) Implementation of K-means clustering algorithm to group the customers based on their
demographic detail using the given dataset.

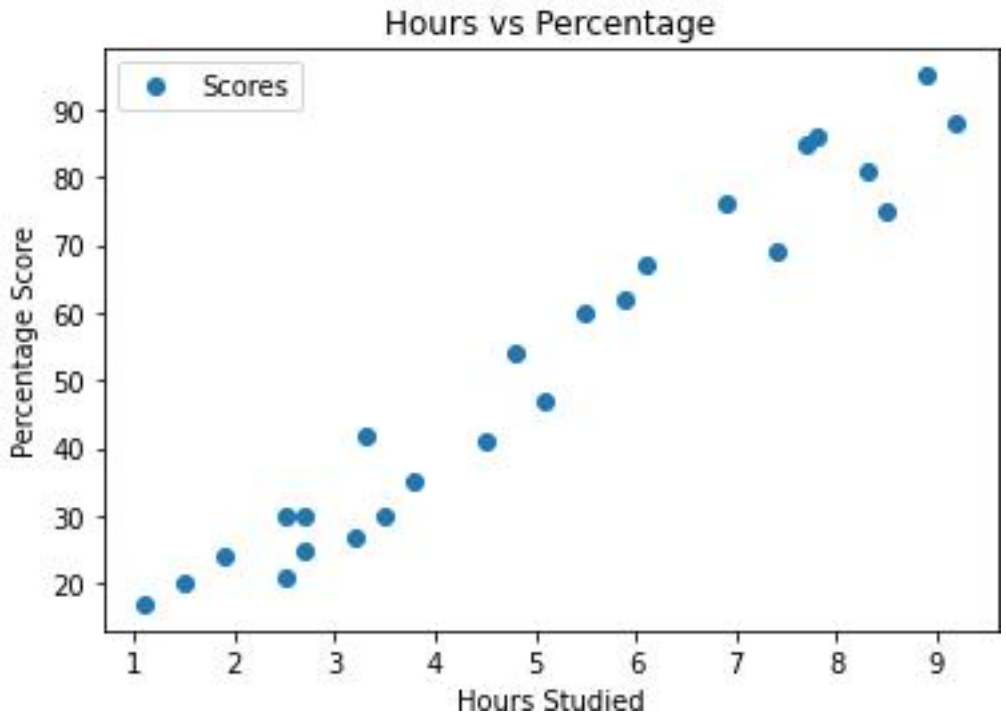
A: Linear Regression on Student's Score Code:

```
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
%matplotlib inline

dataset = pd.read_csv('student_scores.csv') dataset.head()
```

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

```
dataset.describe()
dataset.plot(x='Hours', y='Scores', style='o') plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied') plt.ylabel('Percentage Score')

plt.show()
```

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000




```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, y,test_size=0.2, random_state=0)
print('X train shape: ', X_train.shape)
print('Y train shape: ', Y_train.shape)
print('X test shape: ', X_test.shape)
print('Y test shape: ', Y_test.shape)

regressor = LinearRegression()
regressor.fit(X_train, y_train)
print(regressor.intercept_)
print(regressor.coef_)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}) print(df)
```

	Actual	Predicted
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

```
print('Mean Absolute Error:',metrics.mean_absolute_error (y_test, y_pred)) print('Mean Squared Error:',metrics.mean_squared_error (y_test, y_pred)) print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error (y_test, y_pred)))
```

B: Support Vector Classification algorithm to classify the cases of breast cancer

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVC
%matplotlib inline
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']], columns =
np.append(cancer['feature_names'], ['target']))
df_cancer.head()
```

```
Mean Absolute Error: 4.183859899002982
Mean Squared Error: 21.598769307217456
Root Mean Squared Error: 4.647447612100373
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	w compact
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.1622	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.1238	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.1444	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.2098	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.1374	0

5 rows × 30 columns

X = df_cancer.drop(['target'], axis = 1) # We drop our "target" feature and use all the remaining features in our dataframe to train the model. X.head()

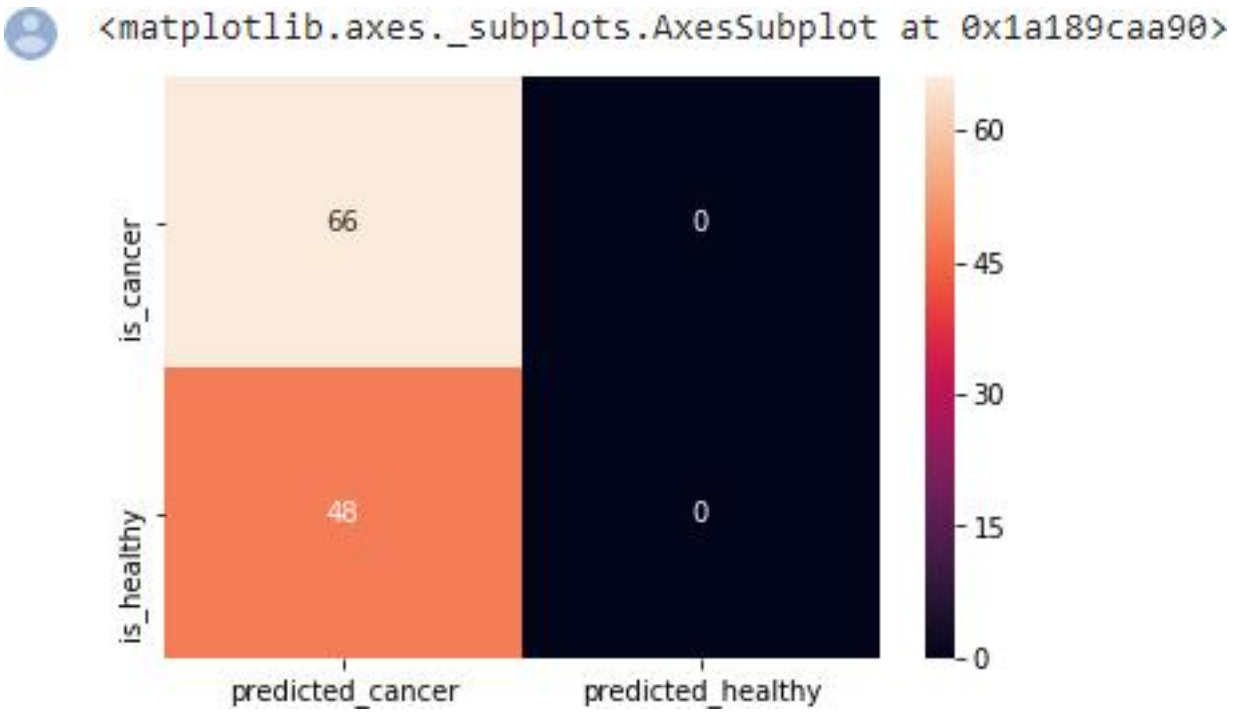
y = df_cancer['target']
y.head()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 20)

svc_model = SVC() svc_model.fit(X_train, y_train) y_predict = svc_model.predict(X_test)

cm = np.array(confusion_matrix(y_test, y_predict, labels=[1,0])) confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],

columns=['predicted_cancer','predicted_healthy']) sns.heatmap(confusion, annot=True)

	predicted_cancer	predicted_healthy
is_cancer	66	0
is_healthy	48	0



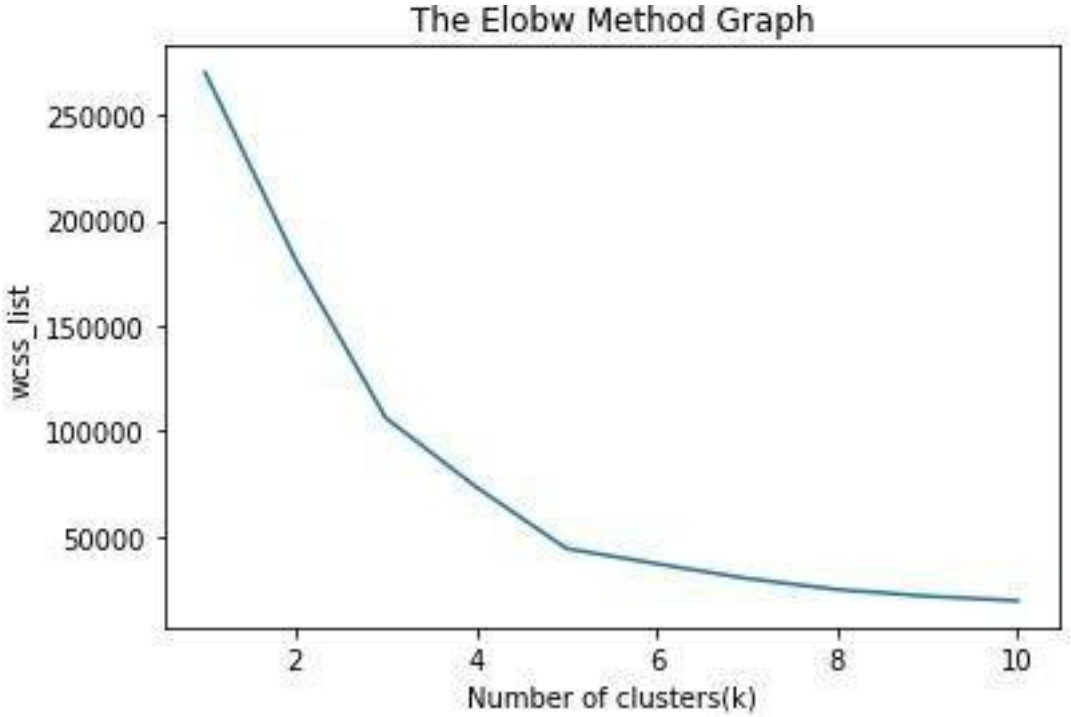
C: K-means clustering algorithm to group the customers based on their demographic detail using the given dataset.

Index	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72
10	11	Male	67	19	14
11	12	Female	35	19	99
12	13	Female	58	20	15
13	14	Female	24	20	77
14	15	Male	37	20	13
15	16	Male	22	20	79

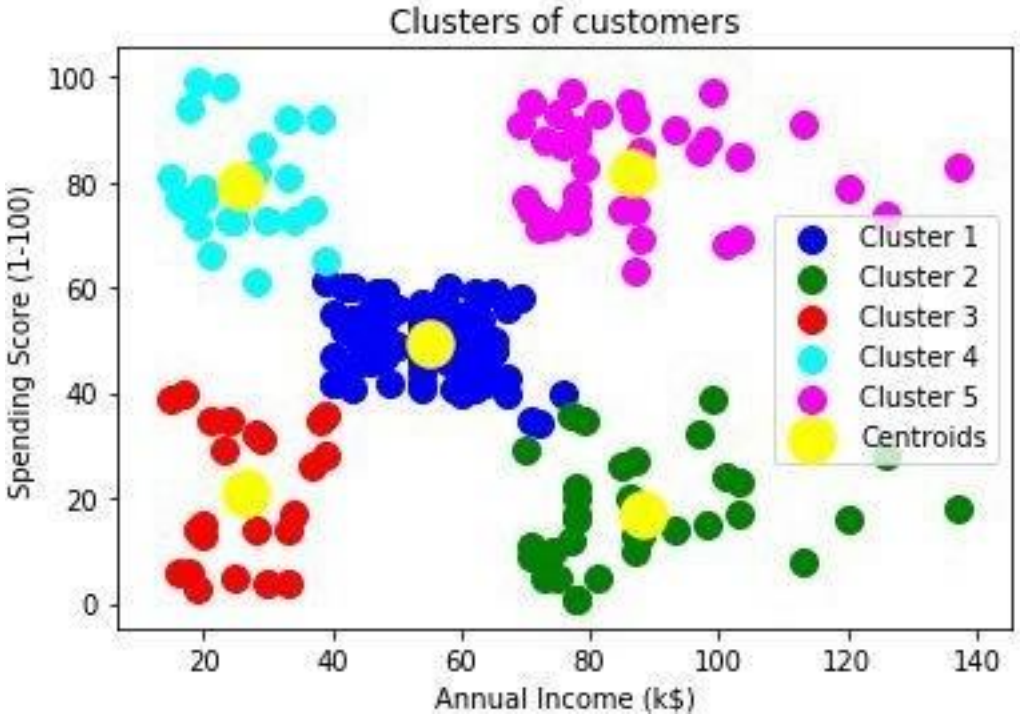
Code:

```
import numpy as nm
import matplotlib.pyplot as mtp import pandas as pd

dataset = pd.read_csv('Mall_Customers_data.csv')


from sklearn.cluster import KMeans wcss_list= []
#Using for loop for iterations from 1 to 10. for i in range(1, 11):

kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
kmeans.fit(x)
wcss_list.append(kmeans.inertia_)

mtp.plot(range(1, 11), wcss_list) mtp.title('The Elobw Method Graph')
mtp.xlabel('Number of clusters(k)') mtp.ylabel('wcss_list')
```

```
mtp.show()

kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)

mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue',
label = 'Cluster 1') #for first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green',
label = 'Cluster 2') #for second cluster

mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red',
label = 'Cluster 3') #for third cluster
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan',
label = 'Cluster 4') #for fourth cluster
```




```
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c =
'magenta', label = 'Cluster 5') #for fifth cluster
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
s = 300, c = 'yellow', label = 'Centroid')

mtp.title('Clusters of customers') mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)') mtp.legend()

mtp.show()
```



Result:

Hence, we successfully implemented Linear Regression, SVM and K-means, verified the output, and documented the result.