

**AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY  
AMITY UNIVERSITY  
NOIDA**

**MACHINE LEARNING LAB FILE**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Submitted by  
Utkarsh meena  
5 CSE 12-X  
A2305219699

Submitted to  
Sangeeta Chugh

## INDEX

S. No	Title	Remarks
1	To install Jupyter Notebook and implement small Python programs	
2	Implementation of logical rules in Python	
3	Implementation of linear regression using given values of constants	
4	Implementation of linear regression by finding the optimal values of theta0 and theta1	
5	EDA analysis of given dataset	
6	Implementation of Logistics Regression in Python	
7	Implementation of K-means Clustering in Python	
8	Implementation of KNN algorithm in Python	

# Experiment 1

## AIM

To install Jupyter Notebook and implement small python programs

## CODE

```
num = int(input("Enter a number: "))
if (num % 2) == 0:
    print("It is Even number")
else:
    print("It is Odd number")
```

```
i=1
num = int(input("Enter a limit: "))
while (i < num):
    print("Count: ",i)
    i = i+1
```

```
area = 0
len = int(input("Enter lenght: "))
bre = int(input("Enter breadth: "))
area = len*bre
print("Area: ",area)
```

```
i=0
num = int(input("Enter the limit: "))
for i in list(range(num)):
    print(i)
```

## OUTPUT

```
In [1]: num = int(input("Enter a number: "))
if (num % 2) == 0:
    print("It is Even number")
else:
    print("It is Odd number")

Enter a number: 20
It is Even number
```

```
In [2]: i=1
num = int(input("Enter a limit: "))
while (i < num):
    print("Count: ",i)
    i = i+1

Enter a limit: 4
Count: 1
Count: 2
Count: 3
```

```
In [1]: area = 0
len = int(input("Enter lenght: "))
bre = int(input("Enter breadth: "))
area = len*bre
print("Area: ",area)
```

```
Enter lenght: 10
Enter breadth: 20
Area: 200
```

```
In [4]: i=0
num = int(input("Enter the limit: "))
for i in list(range(num)):
    print(i)
```

```
Enter the limit: 5
0
1
2
3
4
```

## Experiment 2

### AIM

Implementation of logical rules in python

### CODE

```
str1 = print("It is raining ")
str2 = print("I carry an Umbrella")

a = int(input("Set the value for first String: "))
b = int(input("Set the value for second String: "))

if(a == 1):
    A = True
else:
    A = False

if(b == 1):
    B = True
else:
    B = False

if(A == True and B == True):
    print("Both the statements are True")
elif(A == True or B == True):
    print("One of the statement is True")
    if(A == True):
        print("It is raining")
    else:
        print("I carry an Umbrella")
else:
    print("Both statement are False")

if(not A or B):
    print("If it rains then I carry an Umbrella")

if((not A or B) and (not B or A)):
    print("If and only if it rains then I carry an Umbrella")
```

## OUTPUT

```
In [1]: str1 = print("It is raining ")
str2 = print("I carry an Umbrella")

a = int(input("Set the value for first String: "))
b = int(input("Set the value for second String: "))

if(a == 1):
    A = True
else:
    A = False

if(b == 1):
    B = True
else:
    B = False

if(A == True and B == True):
    print("Both the statements are True")
elif(A == True or B == True):
    print("One of the statement is True")
    if(A == True):
        print("It is raining")
    else:
        print("I carry an Umbrella")
else:
    print("Both statement are False")

if(not A or B):
    print("If it rains then I carry an Umbrella")

if((not A or B) and (not B or A)):
    print("If and only if it rains then I carry an Umbrella")

It is raining
I carry an Umbrella
Set the value for first String: 1
Set the value for second String: 1
Both the statements are True
If it rains then I carry an Umbrella
If and only if it rains then I carry an Umbrella
```

## Experiment 3

### AIM

Implementation of linear regression using given values of constants

### CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as py

train = pd.read_csv("usa_house.csv")

x_train = train[['avg_area_income']]
y_train = train[['house_price']]

py.scatter(x_train,y_train,c='blue')
py.xlabel('House price')
py.ylabel('Average income')

theta0 = 0
theta1 = 20
predicted = theta0 + theta1*x_train
py.scatter(x_train,y_train,c='blue')
py.plot(x_train,predicted,'-y')
py.xlabel('House price')
py.ylabel('Average income')
py.show()
```

### OUTPUT

```
In [13]: import pandas as pd
import numpy as np
import matplotlib.pyplot as py
```

```
In [11]: train = pd.read_csv("C:\\Users\\Chaitanya\\OneDrive\\Desktop\\usa_house.csv")
train
```

```
Out[11]:
```

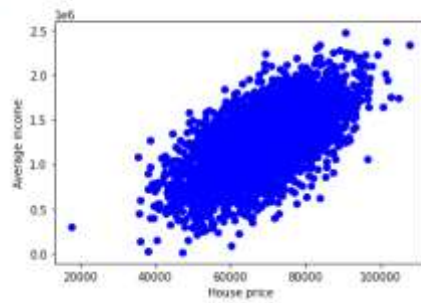
	avg_area_income	avg_house_age	avg_nb_rooms	avg_nb_bathrooms	area_population	house_price	address
0	79545.45857	5.882981	7.009188	4.09	23088.80950	1.059034e+06	208 Michael Ferry Apt. 674\\nLaurabury, NE 3701...
1	79248.64245	8.002900	8.730821	3.09	40173.07217	1.505891e+06	188 Johnson Viewz Suite 079\\nLake Kathleen, CA...
2	81287.06718	5.895890	8.512727	5.13	36882.15940	1.058988e+06	9127 Elizabeth Stravenue\\nDanielstown, WI 06482...
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260517e+06	US9 Barnet\\nFPO AP 44820
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05	USNS Raymond\\nFPO AE 09386
...	...	...	...	...	...	...	...
4029	60567.94414	7.830362	8.137356	3.48	22837.36103	1.090194e+06	USNS Williams\\nFPO AP 30153-7653
4030	78491.27543	8.999135	6.576763	4.02	25616.11548	1.482610e+06	PSC 9258, Box 8489\\nAPO AA 42991-3352
4031	63380.68689	7.250591	4.805081	2.13	33266.14548	1.030730e+06	4215 Tracy Garden Suite 078\\nJoshualand, VA 01...
4032	68001.33124	5.534388	7.130144	5.44	42625.62016	1.198657e+06	US9 Wallace\\nFPO AE 73316
4033	65510.58180	8.982305	6.782336	4.07	48501.28380	1.298950e+06	37778 George Ridges Apt. 509\\nEast Holly, NV 2...

4034 rows x 7 columns

```
In [12]: x_train = train[['avg_area_income']]
y_train = train[['house_price']]
```

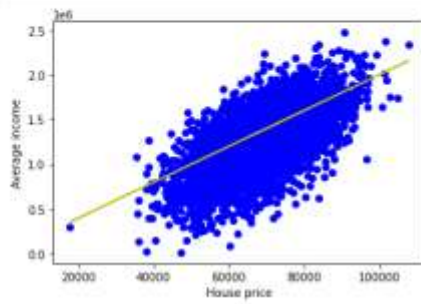
```
In [14]: py.scatter(x_train,y_train,c='blue')
py.xlabel('House price')
py.ylabel('Average income')
```

```
Out[14]: Text(0, 0.5, 'Average income')
```



```
In [15]: theta0 = 0
theta1 = 28

predicted = theta0 + theta1*x_train
py.scatter(x_train,y_train,c='blue')
py.plot(x_train,predicted,'-y')
py.xlabel('House price')
py.ylabel('Average income')
py.show()
```





## Experiment 4

### AIM

Implementation of linear regression by finding the optimal values of  $\theta_0$  and  $\theta_1$

### CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as py

train = pd.read_csv("usa_house.csv")

x_train = train[['avg_area_income']]
y_train = train[['house_price']]

py.scatter(x_train,y_train,c='blue')
py.xlabel('House price')
py.ylabel('Average income')

def val(x_train,y_train):
    theta1=theta0=0
    a=len(x_train)
    itr=1000
    lr=0.0000000001
    for i in range(itr):
        y_pre = theta1*x_train +theta0
        cost=(1/a) *sum ( [val**2 for val in (y_train-y_pre)])
        md= -(2/a) *sum (x_train*(y_train-y_pre))
        cd= -(2/a) *sum(y_train-y_pre)
        theta1=theta1-lr*md
        theta0=theta0-lr*cd
        print("m {}, {}, cost{},itr {}".format(theta1, theta0 ,cost, i))
    return theta1,theta0

py.scatter(x_train,y_train,c='blue')
py.xlabel('house price')
py.ylabel('average income')

theta1,theta0=val(x_train,y_train)
predicted = theta0 + theta1*x_train
py.scatter(x_train,y_train,c='blue')
py.plot(x_train,predicted,'-y')
py.xlabel('house price')
py.ylabel('average income')
py.show()
```

## OUTPUT

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
In [3]: train = pd.read_csv("C:\\Users\\Chaitanya\\OneDrive\\Desktop\\AI Lab\\usa_house.csv")
train
```

```
Out[3]:
```

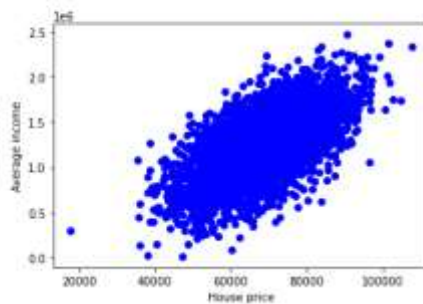
	avg_area_income	avg_house_age	avg_nb_rooms	avg_nb_bathrooms	area_population	house_price	address
0	79545.45857	5.682861	7.009188	4.09	23088.80050	1.059034e+06	208 Michael Ferry Apt. 674nLaurabury, NE 3701...
1	79248.84245	8.002900	8.730821	3.09	40173.07217	1.505891e+06	188 Johnson Views Suite 079nLake Kathleen, CA...
2	81287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06	9127 Elizabeth StravenueinDaneletown, WI 05482...
3	83345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06	USNS BarnettinFPO AP 44829
4	59982.10723	5.040555	7.839388	4.23	26354.10947	6.309435e+05	USNS RaymondinFPO AE 09386
...	...	...	...	...	...	...	...
4029	60567.94414	7.830362	6.137356	3.46	22837.38103	1.060194e+06	USNS WilliamsinFPO AP 30153-7653
4030	78491.27543	6.999135	6.576763	4.02	25616.11549	1.482618e+06	PSC 9258, Box 8489nAPO AA 42991-3352
4031	83390.68889	7.250591	4.805081	2.13	33266.14548	1.030730e+06	4215 Tracy Garden Suite 076nJoshuaLand, VA 01...
4032	68001.33124	5.534388	7.130144	5.44	42625.62016	1.198657e+06	USS WallaceinFPO AE 73316
4033	66519.98189	5.992305	6.792336	4.07	46501.28380	1.298950e+06	37778 George Ridges Apt. 509nEast Holly, NV 2...

4034 rows x 7 columns

```
In [4]: x_train = train[['avg_area_income']]
y_train = train[['house_price']]
```

```
In [5]: plt.scatter(x_train, y_train, c='blue')
plt.xlabel('House price')
plt.ylabel('Average Income')
```

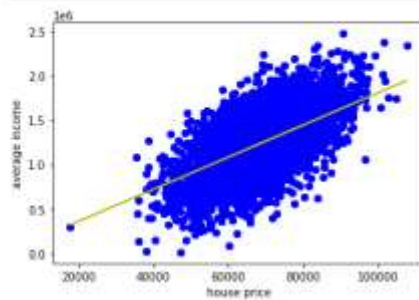
```
Out[5]: Text(0, 0.5, 'Average Income')
```



```
In [10]: def val(x_train,y_train):
    theta1=theta0=0
    a=len(x_train)
    itr=1000
    lr=0.0000000001
    for i in range(itr):
        y_pre = theta1*x_train+theta0
        cost=(1/a) *sum ( [val**2 for val in (y_train-y_pre)])
        nd= -(2/a) *sum (x_train*(y_train-y_pre))
        cd= -(2/a) *sum(y_train-y_pre)
        theta1=theta1-lr*nd
        theta0=theta0-lr*cd
    return theta1,theta0

py.scatter(x_train,y_train,c='blue')
py.xlabel('house price')
py.ylabel('average income')

theta1,theta0=val(x_train,y_train)
predicted = theta0 + theta1*x_train
py.scatter(x_train,y_train,c='blue')
py.plot(x_train,predicted,'-y')
py.xlabel('house price')
py.ylabel('average income')
py.show()
```



## Experiment 5

### AIM

EDA analysis of given dataset

### CODE

```
import pandas as pd
import matplotlib.pyplot as py
import seaborn
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt

data = pd.read_csv("nyc.csv")
data

x = data[['Food']]
y = data[['Price']]

print(data.info())
print(data.corr())

py.boxplot(x)
py.boxplot(y)

x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.3,random_state=0)

Lr = LinearRegression()
Lr.fit(x_train,y_train)
y_pred = Lr.predict(x_test)

rms = sqrt(mean_squared_error(y_test, y_pred))
print(rms)
```

## OUTPUT

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
In [2]: data = pd.read_csv("nyc.csv")
data
```

```
Out[2]:
```

	Food	Decor	Service	East	Price
0	22	18	20	0	.43
1	20	19	19	0	.32
2	21	13	18	0	.34
3	20	20	17	0	.41
4	24	19	21	0	.54
...	...	...	...	...	...
163	17	15	18	0	.31
164	20	16	17	0	.26
165	18	16	17	0	.31
166	22	17	21	0	.38
167	24	10	16	0	.34

168 rows x 5 columns

```
In [3]: x = data[["Food"]]
y = data[["Price"]]
```

```
In [4]: print(data.info())
```

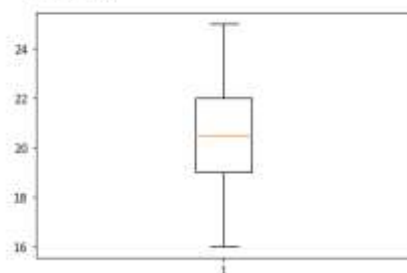
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 168 entries, 0 to 167
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  --
 0   Food    168 non-null     int64  
 1   Decor   168 non-null     int64  
 2   Service 168 non-null     int64  
 3   East    168 non-null     int64  
 4   Price   168 non-null     int64  
dtypes: int64(5)
memory usage: 6.7 KB
None
```

```
In [5]: print(data.corr())
```

	Food	Decor	Service	East	Price
Food	1.000000	0.503916	0.794525	0.188371	0.627843
Decor	0.503916	1.000000	0.645331	0.035749	0.724352
Service	0.794525	0.645331	1.000000	0.209094	0.641148
East	0.188371	0.035749	0.209094	1.000000	0.186638
Price	0.627843	0.724352	0.641148	0.186638	1.000000

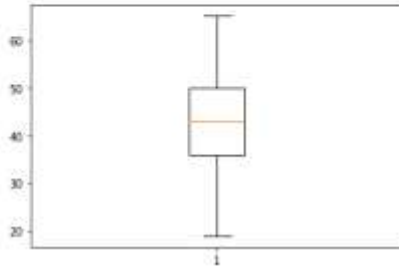
```
In [6]: py.boxplot(x)
```

```
Out[6]: {'whiskers': [matplotlib.lines.Line2D at 0x1799b559fa0],
<matplotlib.lines.Line2D at 0x1799d57a2b0>],
'caps': [matplotlib.lines.Line2D at 0x1799d57a648],
<matplotlib.lines.Line2D at 0x1799d57a9d0>],
'boxes': [matplotlib.lines.Line2D at 0x1799b559b50],
'medians': [matplotlib.lines.Line2D at 0x1799d57ad60],
'fliers': [matplotlib.lines.Line2D at 0x1799d584130],
'means': []}
```



```
In [7]: py.boxplot(y)
```

```
Out[7]: {'whiskers': [<matplotlib.lines.Line2D at 0x1799d65d220>,  
<matplotlib.lines.Line2D at 0x1799d65d5b8>],  
'caps': [<matplotlib.lines.Line2D at 0x1799d65d0940>,  
<matplotlib.lines.Line2D at 0x1799d65dcd0>],  
'boxes': [<matplotlib.lines.Line2D at 0x1799d64de50>],  
'medians': [<matplotlib.lines.Line2D at 0x1799d6880a8>],  
'fliers': [<matplotlib.lines.Line2D at 0x1799d668430>],  
'means': []}
```



```
In [8]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=0)
```

```
In [9]: lr = LinearRegression()  
lr.fit(x_train,y_train)  
y_pred = lr.predict(x_test)
```

```
In [10]: from sklearn.metrics import mean_squared_error  
from math import sqrt
```

```
In [11]: rms = sqrt(mean_squared_error(y_test, y_pred))  
print(rms)
```

```
6.464670171753509
```

## Experiment 6

### AIM

Implementation of Logistics Regression in Python

### CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

data = pd.read_csv("Social_Network_Ads.csv")
data

x = data.iloc[:, [2, 3]].values
y = data.iloc[:, 4].values
x_test, x_train, y_test, y_train =
train_test_split(x, y, test_size=0.25, random_state=1)

scaler = StandardScaler()
x_train_scd = scaler.fit_transform(x_train)
x_test_scd = scaler.transform(x_test)

LR = LogisticRegression()
LR.fit(x_train_scd, y_train)
y_pred = LR.predict(x_test_scd)

score = accuracy_score(y_test, y_pred)
print(score)
```

## OUTPUT

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

```
In [2]: data = pd.read_csv("Social_Network_Ads.csv")
data
```

```
Out[2]:
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15903246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15735018	Male	38	33000	0
399	15594041	Female	49	36000	1

400 rows x 5 columns

```
In [3]: x = data.iloc[:,[2,3]].values
y = data.iloc[:,4].values
x_test,x_train,y_test,y_train = train_test_split(x,y,test_size=0.25,random_state=1)
```

```
In [4]: scaler = StandardScaler()
x_train_scd = scaler.fit_transform(x_train)
x_test_scd = scaler.transform(x_test)
```

```
In [5]: LR = LogisticRegression()
LR.fit(x_train_scd,y_train)
y_pred = LR.predict(x_test_scd)
```

```
In [6]: Score = accuracy_score(y_test,y_pred)
print(Score)
```

0.86



## Experiment 7

### AIM

Implementation of K-means Clustering in Python

### CODE

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv("Mall_Customers.csv")
X = dataset.iloc[:, [3, 4]].values

# Using the elbow method to find the optimal no of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', n_init=10, max_iter=300, random_state=None)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('No of Clusters')
plt.ylabel('WCSS')
plt.show()

kmeans = KMeans(n_clusters=5, init='k-means++', n_init=10, max_iter=300, random_state=None)
y_kmeans = kmeans.fit_predict(X)

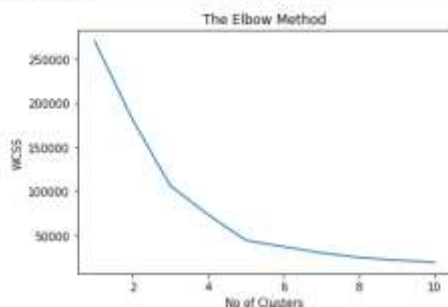
# Visualizing the clusters
plt.scatter(X[y_kmeans==0, 0], X[y_kmeans==0, 1], s=10, c='red', label='Cluster1')
plt.scatter(X[y_kmeans==1, 0], X[y_kmeans==1, 1], s=10, c='blue', label='Cluster2')
plt.scatter(X[y_kmeans==2, 0], X[y_kmeans==2, 1], s=10, c='green', label='Cluster3')
plt.scatter(X[y_kmeans==3, 0], X[y_kmeans==3, 1], s=10, c='cyan', label='Cluster4')
plt.scatter(X[y_kmeans==4, 0], X[y_kmeans==4, 1], s=10, c='magenta', label='Cluster5')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=30, c='yellow', label='Centroid')
plt.title('Cluster of Clients')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

## OUTPUT

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

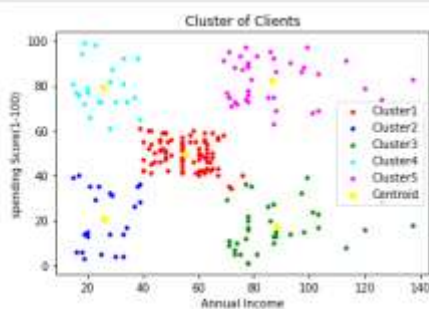
```
In [2]: dataset = pd.read_csv("Mail_Customers.csv")
X = dataset.iloc[:, [3,4]].values

# Using the elbow method to find the optimal no of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', n_init=10, max_iter=300, random_state=None)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11), wcss)
plt.title('The Elbow Method')
plt.xlabel('No of Clusters')
plt.ylabel('WCSS')
plt.show()
```



```
In [3]: kmeans = KMeans(n_clusters=5, init='k-means++', n_init=10, max_iter=300, random_state=None)
y_kmeans = kmeans.fit_predict(X)

#Visualizing the Clusters
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],s=10, c='red', label='Cluster1')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],s=10, c='blue', label='Cluster2')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],s=10, c='green', label='Cluster3')
plt.scatter(X[y_kmeans==3,0],X[y_kmeans==3,1],s=10, c='cyan', label='Cluster4')
plt.scatter(X[y_kmeans==4,0],X[y_kmeans==4,1],s=10, c='magenta', label='Cluster5')
plt.scatter(kmeans.cluster_centers_[0,0],kmeans.cluster_centers_[0,1],s=30, c='yellow', label='Centroid')
plt.title('Cluster of Clients')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score(1-100)')
plt.legend()
plt.show()
```



## Experiment 8

### AIM

Implementation of KNN algorithm in Python

### CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

data = pd.read_csv("Diabetic.csv")
data.head()

data.info()

data.describe()

plt.hist(data)

y = data["Outcome"].values
x = data.drop(["Outcome"],axis=1)

ss = StandardScaler()
data = ss.fit_transform(data)

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)

train_score = []
test_score = []
k_value = []
accuracy = []
y_p = []

for k in range(1,21):
    k_value.append(k)
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(x_train,y_train)

    y_pred = knn.predict(x_test)
    y_p.append(y_pred)

    a_score = accuracy_score(y_test,y_pred)
    accuracy.append(a_score)

    tr_score = knn.score(x_train,y_train)
    train_score.append(tr_score)

    te_score = knn.score(x_test,y_test)
    test_score.append(te_score)

plt.xlabel('Different values of K')
plt.ylabel('Model score')
plt.plot(k_value, train_score, color = 'r', label = "training score")
plt.plot(k_value, test_score, color = 'b', label = 'test score')
```

```

plt.xlabel('Different values of K')
plt.ylabel('Accuracy score')
plt.plot(k_value, accuracy, color = 'r', label = "accuracy")

print("Maximum training accuracy at: ",train_score.index(max(train_score))+1)
print("Accuracy: ",max(train_score))

print("Maximum testing accuracy at: ",test_score.index(max(test_score))+1)
print("Accuracy: ",max(test_score))

knn = KNeighborsClassifier(n_neighbors = 17)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_test)

print("Accuracy: ",accuracy_score(y_test,y_pred))

print(classification_report(y_test,y_pred))

```

## OUTPUT

```

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.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

In [2]: data = pd.read_csv("Diabetic.csv")
data.head()

Out[2]:

```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	0	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```

In [3]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  -
 0   Pregnancies         768 non-null   int64
 1   Glucose              768 non-null   int64
 2   BloodPressure        768 non-null   int64
 3   SkinThickness        768 non-null   int64
 4   Insulin              768 non-null   int64
 5   BMI                  768 non-null   float64
 6   DiabetesPedigreeFunction 768 non-null   float64
 7   Age                  768 non-null   int64
 8   Outcome              768 non-null   int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB

```

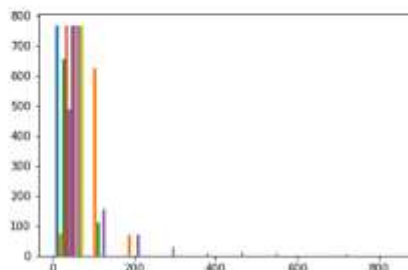
```
In [4]: data.describe()
```

```
Out[4]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471875	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
In [5]: plt.hist(data)
```

```
Out[5]: (array([[768., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
 [ 72., 625., 71., 0., 0., 0., 0., 0., 0., 0.],
 [656., 112., 0., 0., 0., 0., 0., 0., 0., 0.],
 [767., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
 [487., 155., 70., 30., 0., 9., 5., 1., 2., 1.],
 [768., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
 [768., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
 [768., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
 [768., 0., 0., 0., 0., 0., 0., 0., 0., 0.]]),
 array([ 0., 84.6, 169.2, 253.8, 338.4, 423., 507.6, 592.2, 676.8,
        761.4, 846. ]),
 <a list of 9 BarContainer objects>)
```



```
In [6]: y = data["Outcome"].values
x = data.drop(["Outcome"],axis=1)
```

```
In [7]: SS = StandardScaler()
data = SS.fit_transform(data)
```

```
In [8]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

```
In [9]: train_score = []
test_score = []
k_value = []
accuracy = []
y_p = []
```

```
In [10]: for k in range(1,21):
    k_value.append(k)
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(x_train,y_train)

    y_pred = knn.predict(x_test)
    y_p.append(y_pred)

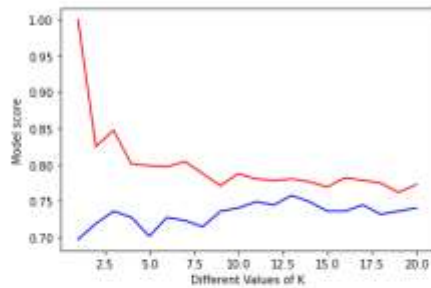
    a_score = accuracy_score(y_test,y_pred)
    accuracy.append(a_score)

    tr_score = knn.score(x_train,y_train)
    train_score.append(tr_score)

    te_score = knn.score(x_test,y_test)
    test_score.append(te_score)
```

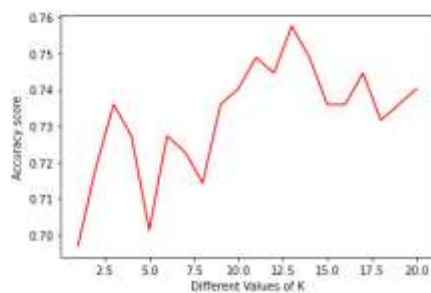
```
In [11]: plt.xlabel('Different Values of K')
plt.ylabel('Model score')
plt.plot(k_value, train_score, color = 'r', label = "training score")
plt.plot(k_value, test_score, color = 'b', label = 'test score')
```

```
Out[11]: [<matplotlib.lines.Line2D at 0x14856b289d0>]
```



```
In [12]: plt.xlabel('Different Values of K')
plt.ylabel('Accuracy score')
plt.plot(k_value, accuracy, color = 'r', label = "accuracy")
```

```
Out[12]: [<matplotlib.lines.Line2D at 0x14856ba2340>]
```



```
In [13]: print("Maximum training accuracy at: ",train_score.index(max(train_score))+1)
print("Accuracy: ",max(train_score))

Maximum training accuracy at: 1
Accuracy: 1.0
```

```
In [14]: print("Maximum testing accuracy at: ",test_score.index(max(test_score))+1)
print("Accuracy: ",max(test_score))

Maximum testing accuracy at: 13
Accuracy: 0.7575757575757576
```

```
In [15]: knn = KNeighborsClassifier(n_neighbors = 17)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_test)

print("Accuracy: ",accuracy_score(y_test,y_pred))

Accuracy: 0.7445887445887446
```

```
In [16]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.77	0.87	0.81	149
1	0.68	0.52	0.59	82
accuracy			0.74	231
macro avg	0.73	0.70	0.70	231
weighted avg	0.74	0.74	0.74	231