MASTER OF COMPUTER APPLICATIONS

PRACTICAL RECORD WORK
ON
20MCA241 – DATA SCIENCE LAB

Submitted by

NAME : RAHBAR ZAHID S

Reg. No. : VDA24MCA-2039

DEPARTMENT OF COMPUTER APPLICATIONS

COLLEGE OF ENGINEERING VADAKARA

(CAPE – GOVT. OF KERALA)

NOV 2025

DEPARTMENT OF COMPUTER APPLICATIONS

COLLEGE OF ENGINEERING VADAKARA

(CAPE – GOVT. OF KERALA)

CERTIFICATE

Certified that this is the bonafide record work on the practical course **20MCA241 DATA SCIENCE LAB** done and prepare **Mr. RAHBAR ZAHID S (Reg No.VDA24MCA-2039)**, 3rd Semester **MCA(2024-26 Batch)** student of Department of Computer Applications at College of Engineering Vadakara, in the partial fulfillment of the award of MCA Degree of APJ Abdul Kalam Technological University (KTU).

Date :

Faculty-in-Charge **Head of the Department**

(Office Seal)

Internal Examiners: **External Examiners:**

INDEX

EXPERIMENT NO 1

AIM:
Review of python programming - numpy, pandas, matplotlib.

NUMPY:

1. Program to multiply two given arrays of same size element-by-element.

Source code:

```
import numpy as np

nums1 = np.array([[2, 5, 2],[1, 5, 5]])
nums2 = np.array([[5, 3, 4],[3, 2, 5]])
print("Array1:")
print(nums1)
print("Array2:")
print(nums2)
print("\nMultiply said arrays of same size element-by-element:")
print(np.multiply(nums1, nums2))
```

Output:

```
Array1:
[[2 5 2]
[1 5 5]]
Array2:
[[5 3 4]
[3 2 5]]
Multiply said arrays of same size element-by-element:
[[10 15 8]
[ 3 10 25]]
```

2. Program to create an element-wise comparison (greater, greater equal, lesser).

Source code:

```
import numpy as np
x = np.array([3,5,1,2,3])
y = np.array([2,5,3,2,1])
print("Array A")
print(x)
print("\nArray B")
print(y)
print("\nA>B")
print(np.greater(x, y))
print("\nA>=B")
print(np.greater_equal(x, y))
```

Output:

```
Array A
[3 5 1 2 3]
Array B
[2 5 3 2 1]

A>B
[ True False False False True]
A>=B
[ True True False True True]
A<B
[False False True False False]
A<=B
[False True True True False]
```

3. Program to create an array of all the even integers from 30 to 70

Source code:

```
import numpy as np

x = np.arange(start=30, stop=71, step=2)
print(x)
```

Output:

```
[30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70]
```

4. program to create a 3x3 identity matrix.

Source code:

```
import numpy as np

x = np.identity(3)
print(x)
```

Output:

```
[[1. 0. 0.]
[0. 1. 0.]
[0. 0. 1.]]
```

5. Program to create a vector with values from 0 to 20 and change the sign.

Source code:

```
import numpy as np
```

```
x = np.arange(21)
print("Vectors ")
print(x)
print("\nAfter changing the sign of the numbers in the range from 9 to 15:")
x[(x >= 9) & (x <= 15)] *= -1
print(x)
```

Output:

```
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
After changing the sign of the numbers in the range from 9 to 15:
[ 0 1 2 3 4 5 6 7 8 -9 -10 -11 -12 -13 -14 -15 16 17 18 19 20]
```

6. Program to create a 5x5 zero matrix with elements on the main diagonal.

Source code:

```
import numpy as np
x = np.diag([1, 2, 3, 4, 5])
print(x)
```

Output:

```
[[1 0 0 0 0]
[0 2 0 0 0]
[0 0 3 0 0]
[0 0 0 4 0]
[0 0 0 0 5]]
```

7. program to compute sum of all elements, sum of each column and sum of each row.

Source code:

```
import numpy as np
x = np.array([[1,0],[0,1]])
print("Array")
print(x)
print("\nSum of all elements:")
print(np.sum(x))
print("\nSum of each column:")
print(np.sum(x, axis=0))
print("\nSum of each row:")
print(np.sum(x, axis=1))
```

Output:

```
Array
[[1 0]
[0 1]]
Sum of all elements: 2
Sum of each column: [1 1]
Sum of each row: [1 1]
```

8. Program to save a given array to a text file and load it.

Source code:

```
import numpy as np

x = np.arange(16).reshape(4,4)
print("Array:")
print(x)
header = 'C1 C2 C3 C4'
np.savetxt('array.txt', x, fmt="%d", header=header)
print("\nAfter loading, content of the text file:")
print(np.loadtxt('array.txt'))
```

Output:

```
Array:
[[ 0 1 2 3]
[ 4 5 6 7]
[ 8 9 10 11]
[12 13 14 15]]
After loading, content of the text file:
[[ 0. 1. 2. 3.]
[ 4. 5. 6. 7.]
[ 8. 9. 10. 11.]
[12. 13. 14. 15.]]
```

9. Program to check whether two arrays are equal (element wise) or no

Source code:

```
import numpy as np

nums1 = np.array([2,2,3,2,1])
nums2 = np.array([2,3,4,3,1])
print("Original arrays:")
print(nums1)
print(nums2)
print("\nTest said two arrays are equal (element wise) or not:?")
print(nums1 == nums2)
print(np.equal(nums1, nums2))
```

Output:

```
Original arrays:
[2 2 3 2 1]
[2 3 4 3 1]
```

Test said two arrays are equal (element wise) or not:?
[True False False False True]
[True False False False True]

10. Program to create a 4x4 array with random values, now create a new array after swapping first and last row.

Source code:

```
import numpy as np

nums = np.arange(16, dtype='int').reshape(-1, 4)
print("Original array:")
print(nums)
print("\nNew array after swapping first and last rows of the said array:")
nums = nums[[-1,1,2,0]]
print(nums)
```

Output:

```
Original array:
[[ 0 1 2 3]
[ 4 5 6 7]
[ 8 9 10 11]
[12 13 14 15]]
New array after swapping first and last rows of the said array:
[[12 13 14 15]
[ 4 5 6 7]
[ 8 9 10 11]
[ 0 1 2 3]]
```

PANDAS :

11. Write a python program to implement List-to-Series Conversion.

Source code:

```
import pandas as pd

names = ['asif','amal','rabee']
x = pd.Series(names)
print(x)
```

Output:

```
0    asif
1    amal
2   rabee
dtype: object
```

12. Write a python program to Generate the series of dates from 1st May, 2021 to 12th May, 2021 (both inclusive).

Source code:

```
import pandas as pd

sr = pd.Series(pd.date_range('2021-05-01','2021-05-12',freq = 'D'))
print(sr.to_string(index=False))
```

Output:

```
2021-05-01
2021-05-02
2021-05-03
2021-05-04
2021-05-05
2021-05-06
2021-05-07
2021-05-08
2021-05-09
2021-05-10
2021-05-11
2021-05-12
```

13. Given a dictionary, convert it into corresponding dataframe and display it.

Source code:

```
import pandas as pd

details = {
'Name' : ['a','b','c','d'],
'Age' : [24,25,26,27],
}
df = pd.DataFrame(details)
print(df)
```

Output:

```
  name  age
0   a    24
1   b    25
2   c    26
3   d    27
```

14. Given a 2D List, convert it into corresponding dataframe and display it.

Source code:

```
import pandas as pd

details = [[2,4],[1,5]]
df = pd.DataFrame(details)
print(df)
```

Output:

```
   0  1
0  2  4
1  1  5
```

15. Given a CSV file, read it into a dataframe and display it.

Source code:

```
import pandas as pd

df = pd.read_csv('name_mark.csv')
print(df.to_string(index=False))
```

Output:

```
Name  mark
 a     1
 b     2
 c     3
```

16. Given a dataframe, sort it by multiple columns.

Source code:

```
import pandas as pd

df = pd.DataFrame({'Name': ['e','a','a','b','c','d'],
                   'Age': [1,2,1,3,3,4],
                   'Rank': [0,1,2,3,4,5]})


print(df.to_string(index=False))
print('SORTED DATAFRAME')
df = df.sort_values(by=['Name','Age'], ascending=[True,True])
print(df.to_string(index=False))
```

Output:

```
Name  Age  Rank
   e    1     0
   a    2     1
   a    1     2
   b    3     3
   c    3     4
   d    4     5
SORTED DATAFRAME
Name  Age  Rank
   a    1     2
   a    2     1
   b    3     3
   c    3     4
   d    4     5
   e    1     0
```

17. Given a dataframe with custom indexing, convert and it to default indexing and display it.

Source code:

```
import pandas as pd

data={
   'name':['e','a','a','b','c','d'],
   'age':[20,1,2,3,45,5],
   'rank':[0,1,2,3,4,5]
}
index = ['a1','b1','c1','d1','e1','f1']
df=pd.DataFrame(data,index)
print(df.to_string())
df.reset_index(inplace=True,drop=True)
print(df.to_string())
```

Output:

```
    name  age  rank
a1    e   20     0
b1    a    1     1
c1    a    2     2
d1    b    3     3
e1    c   45     4
f1    d    5     5

  name  age  rank
0    e   20     0
1    a    1     1
2    a    2     2
3    b    3     3
4    c   45     4
5    d    5     5
```

18. Given a dataframe, select first 2 rows and output them.

Source code:

```
import pandas as pd

details = {
'Name' : ['a','b','c','d'],
'Age' : [24,25,26,27],
}
df = pd.DataFrame(details)
print(df[:2])
```

Output:

```
   Name Age
 0    a   24
 1    b   25
```

19. Given is a dataframe showing name, occupation,salary of people. Find the average salary per occupation.

Source code:

```
import pandas as pd

details = {
'Name' : ['a','b','c','d','e'],
'Occupation' : ['A1','A1','A1','B1','B1'],
'Salary' : [1000,2000,1500,500,300],
}
df = pd.DataFrame(details)
print(df.to_string(index=False))
average_age = df.groupby('Occupation')['Salary'].mean()
print("Average salary per occupation : ")
print(average_age)
```

Output:

```
name occupation  salary
   a         A1        1000
   b         A1        2000
   c         B1        1500
   d         B1        500
   e         A1        300

occupation
A1    1100.0
B1    1000.0

Name: salary, dtype: float64
```

20. Given a dataframe with NaN Values, fill the NaN values with 0.

Source code:

```
import pandas as pd

import numpy as np

data={'numbers':[1,2,4,6,5,np.nan,8,23,np.nan,23,4,9,np.nan]}

df=pd.DataFrame(data)

print(df)

df_fill=df.fillna(0)

print(df_fill.to_string(index=False))
```

Output:

```
    numbers
0       1.0
1       2.0
2       4.0
3       6.0
4       5.0
5       NaN
6       8.0
7      23.0
8       NaN
9      23.0
10      4.0
11      9.0
12      NaN
   numbers
    1.0
    2.0
    4.0
    6.0
    5.0
    0.0
    8.0
   23.0
    0.0
   23.0
    4.0
    9.0
    0.0
```

21. Given is a dataframe showing Company Names (cname) and corresponding Profits (profit). Convert the values of Profit column such that values in it greater than 0 are set to True and the rest are set to False.

Source code:

```
import pandas as pd

data={
   'cname':['a','b','c','d'],
   'profit':[24,56,0,-21]
}
df=pd.DataFrame(data)
print(df.to_string(index=False))
df['profit']=df['profit']>0
print(df.to_string(index=False))
```

Output:

```
cname  profit
   a      24
   b      56
   c       0
   d     -21
cname  profit
    a    True
    b    True
    c   False
    d   False
```

22. Given are 2 dataframes, with one dataframe containing Employee ID (eid), Employee Name (ename) and Stipend (stipend) and the other dataframe containing Employee ID (eid) and designation of the employee (designation). Output the Dataframe containing Employee ID (eid), Employee Name (ename), Stipend (stipend) and Position (position).

Source code:

```
import pandas as pd
d1={
   'eid' : [1,2,3,4],
   'ename' : ['a','b','c','d'],
   'stipend':[123,131,23,434],
}
d2={
   'eid' : [1,2,3,4],
   'designation' : ['aa','bb','cc','dd'],
}
df1=pd.DataFrame(d1)
df2=pd.DataFrame(d2)
df=pd.merge(df1,df2,on='eid')
print(df)
```

Output:

```
   eid ename  stipend designation
0    1     a      123          aa
1    2     b      131          bb
2    3     c       23          cc
3    4     d      434          dd
```

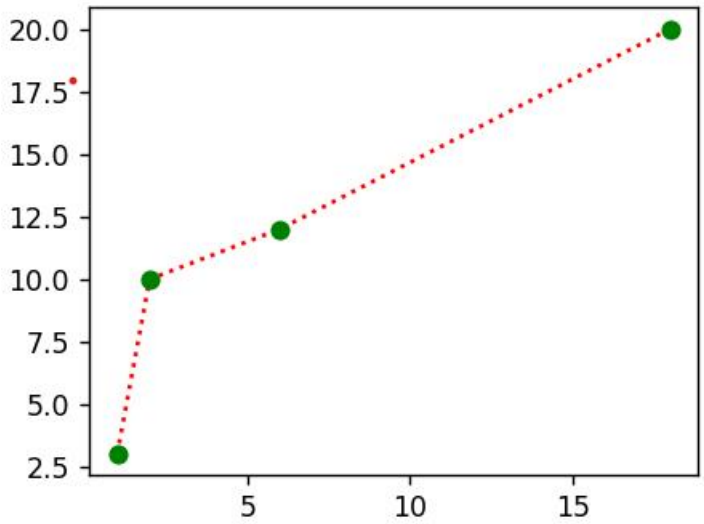
MATPLOTLIB

23. Draw a line in a diagram from position (1, 3) to (2, 10) then to (6, 12) and finally to position (18, 20). (Mark each point with a beautiful green colour and set line colour to red and line style dotted).

Source code:

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

x = np.array([1,2,6,18])
y = np.array([3,10,12,20])

plt.plot(x, y, marker = 'o', mfc='g', mec='g', c='r', linestyle='dotted')
plt.show()
```

Output:



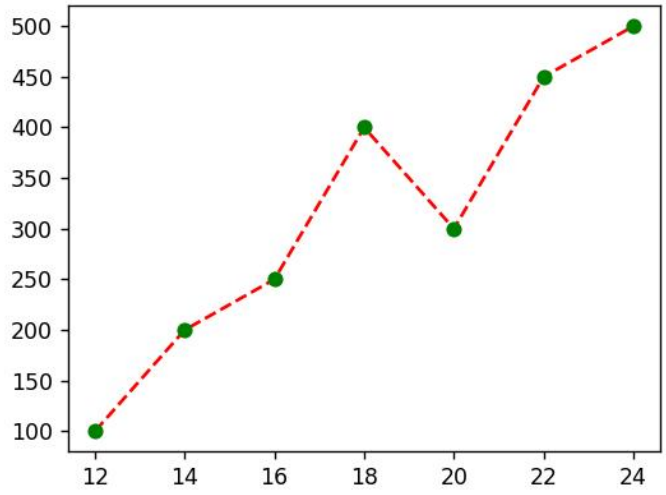
24. Draw a plot for the following data:

Temperature in degree	Celsius Sales
12	100
14	20
16	250
18	400
20	300
22	450
24	500

Source code:

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

x = np.array([12,14,16,18,20,22,24])
y = np.array([100,200,250,400,300,450,500])

plt.plot(x, y, marker = 'o', mfc='g', c='r', linestyle = 'dashed',mec='g')
plt.show()
```

Output:

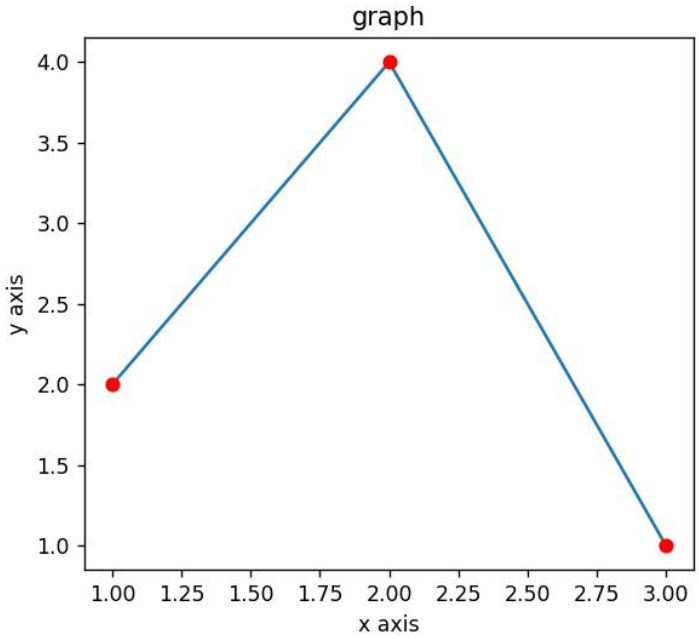


25 . Write a Python program to draw a line using given axis values taken from a text file, with suitable label in the x axis, y axis and a title.

Source code:

```
import matplotlib.pyplot as plt

with open('data_13.txt') as f:
    data = f.read()
data = data.split('\n')
x = [int(row.split(' ')[0]) for row in data]
y = [int(row.split(' ')[1]) for row in data]
plt.plot(x,y, marker='o', mfc='r', mec='r')
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.title("graph")
plt.show()
```

Output:


graph
4.0
3.5
3.0
2.5
2.0
1.5
1.0
y axis
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
x axis

EXPERIMENT NO 2

AIM:

Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm

Source Code

```
from sklearn.datasets import load_iris
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
import pandas as pd

iris= load_iris()
df=pd.DataFrame(iris.data, columns=iris.feature_names)
df['target']=iris.target
print(df.head())

x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
y_pred=knn.predict(x_test)
print(f"prediction of testing values:\n {y_pred}")
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
sample=[[2,2,2,2]]
pred=knn.predict(sample)
pred_v=[iris.target_names[p] for p in pred]
print(f"prediction of sample [2,2,2,2]:  {pred_v}")
print(f"confusion matrix:\n {confusion_matrix(y_test,y_pred)}")
```

Output:

```
C:\Users\rahba\AppData\Local\Programs\Python\Python312\python.exe D:\MCA\Sem3\DS\knn.py
   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  target
0                5.1               3.5  ...               0.2       0
1                4.9               3.0  ...               0.2       0
2                4.7               3.2  ...               0.2       0
3                4.6               3.1  ...               0.2       0
4                5.0               3.6  ...               0.2       0

[5 rows x 5 columns]
prediction of testing values:
 [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 1 2 2 0 1
 0 1 2 2 0 1 2 1]
Accuracy: 0.9777777777777777
prediction of sample [2,2,2,2]:  [np.str_('setosa')]
confusion matrix:
 [[14  0  0]
 [ 0 18  0]
 [ 0  1 12]]
```

EXPERIMENT NO 3

AIM:

Program to implement Naive Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

Source Code

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import StandardScaler

iris=load_iris()
df=pd.DataFrame(iris.data, columns=iris.feature_names)
print(df.head())

X,y=load_iris(return_X_y=True)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)

sc=StandardScaler()
sc.fit(X_train)
X_train=sc.transform(X_train)
X_test=sc.transform(X_test)

gnb=GaussianNB()
gnb.fit(X_train,y_train)
y_pred=gnb.predict(X_test)
print(f"prediction of testing values: \n {y_pred}")

X_new=[[5,5,4,4]]
y_new=gnb.predict(X_new)
print(f"predicted output of [5,5,4,4]:   {y_new}")
y_name=[iris.target_names[p] for p in y_new]
print(y_name)
print(f"naive bayes score:  {gnb.score(X_test,y_test)}")
print(f"confusion matrix:\n  {metrics.confusion_matrix(y_test,y_pred)}")
```

Output:

```
C:\Users\rahba\AppData\Local\Programs\Python\Python312\python.exe D:\MCA\Sem3\DS\bayes.py
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
prediction of testing values:
 [2 0 1 0 1 2 1 2 0 1 2 1 2 1 2 2 2 1 2 0 2 0 1 1 1 0 0 1 0 1 2 2 2 1 2 1 2
 1 0 1 2 0 1 2 2]
predicted output of [5,5,4,4]:   [2]
[np.str_('virginica')]
naive bayes score:  0.9555555555555556
confusion matrix:
  [[10  0  0]
 [ 0 17  2]
 [ 0  0 16]]

Process finished with exit code 0
```

EXPERIMENT NO 4

AIM:

Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance

Source Code

Linear Regression

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data=load_diabetes()
df=pd.DataFrame(data.data, columns=data.feature_names)
df['target']=data.target
print(df.head())

x=df.iloc[:, 7]
print("x feature s5")
print(x.head())

y=df.iloc[:, 6]
print("x feature s3")
print(y.head())
x=np.array(x).reshape(-1,1)
print(f"reshaped x: \n {x}")
y=np.array(y).reshape(-1,1)
print(f"reshaped y: {y}")

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

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

print(f"r2 score:  {r2_score(y_test,y_pred)}")
print(f"mean square error:  {mean_squared_error(y_test,y_pred)}")
print(f"coefficient:  {lr.coef_}")

plt.scatter(x_test,y_test,color='b',label="Actual")
```

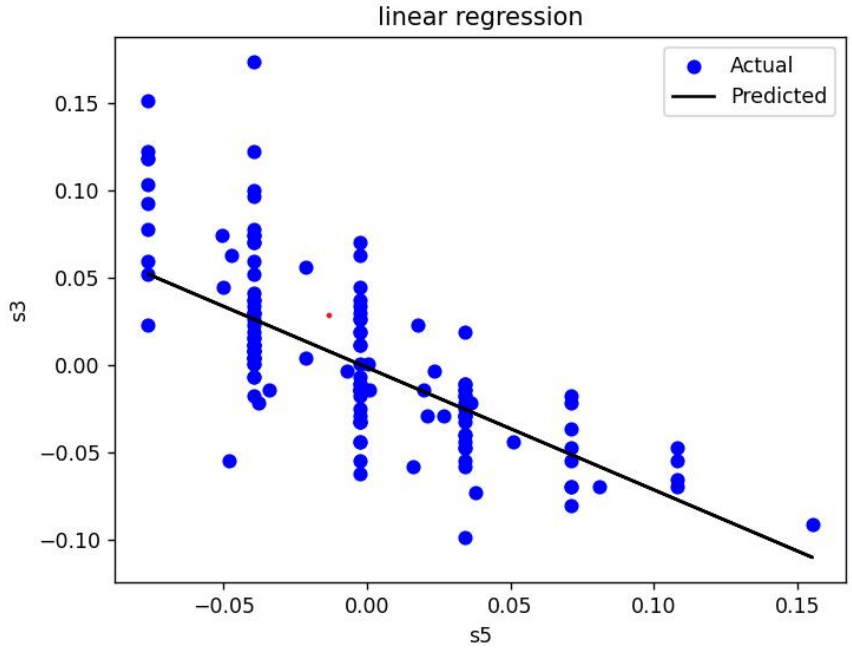
```
plt.plot(x_test,y_pred,color='k',label='Predicted')
plt.xlabel('s5')
plt.ylabel('s3')
plt.title('linear regression')
plt.legend()
plt.show()
```

Output:

```
C:\Users\rahba\AppData\Local\Programs\Python\Python312\python.exe D:\MCA\Sem3\DS\linear_regression.py
        age       sex       bmi        bp  ...        s4        s5        s6  target
0  0.038076  0.050680  0.061696  0.021872  ... -0.002592  0.019907 -0.017646   151.0
1 -0.001882 -0.044642 -0.051474 -0.026328  ... -0.039493 -0.068332 -0.092204    75.0
2  0.085299  0.050680  0.044451 -0.005670  ... -0.002592  0.002861 -0.025930   141.0
3 -0.089063 -0.044642 -0.011595 -0.036656  ...  0.034309  0.022688 -0.009362   206.0
4  0.005383 -0.044642 -0.036385  0.021872  ... -0.002592 -0.031988 -0.046641   135.0

[5 rows x 11 columns]
x feature s5
0   -0.002592
1   -0.039493
2   -0.002592
3    0.034309
4   -0.002592
Name: s4, dtype: float64
x feature s3
0   -0.043401
1    0.074412
2   -0.032356
3   -0.036038
4    0.008142
Name: s3, dtype: float64
```

```
reshaped x:
 [[-0.00259226]
 [-0.03949338]
 [-0.00259226]
 [ 0.03430886]
 [-0.00259226]
 [-0.0763945 ]
 [-0.03949338]
 [ 0.01770335]
 [-0.00259226]
 [-0.00259226]
 [-0.0763945 ]
 [ 0.07120998]
 [-0.03949338]
 [-0.00259226]
 [-0.03949338]
 [ 0.1081111 ]
 [-0.03949338]
 [ 0.03430886]
 [-0.00259226]
 [-0.03949338]
 [-0.03949338]
```

```
reshaped y: [[-0.04340085]
 [ 0.07441156]
 [-0.03235593]
 [-0.03603757]
 [ 0.00814208]
 [ 0.04127682]
 [ 0.00077881]
 [ 0.02286863]
 [-0.02867429]
 [-0.02499266]
 [-0.01394774]
 [-0.06549067]
 [ 0.04495846]
 [-0.00290283]
 [ 0.08177484]
 [-0.03971921]
 [ 0.07441156]
 [-0.03971921]
 [-0.02867429]
 [ 0.03759519]
 [ 0.00077881]
 [ 0.01550536]
 [ 0.02286863]
 [ 0.03071817]
 [ 0.03071817]
 [ 0.05787532]
 [-0.02359613]
 [ 0.03071817]
 [ 0.03071817]
 [ 0.05787532]
 [ 0.00356102]
 [ 0.00356102]]
r2 score:  0.6170542534730672
mean square error:  0.0007655734074880548
coefficient:  [[-0.73594371]]

Process finished with exit code 0
```



Multiple Regression

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score

iris=load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
y_pred_train=lr.predict(x_train)
print(y_pred)
print(y_pred_train)
print(f"r2 score:  {r2_score(y_test,y_pred)}")
print(f"mean square error:  {mean_squared_error(y_test,y_pred)}")
```

Output:

```
C:\Users\rahba\AppData\Local\Programs\Python\Python312\python.exe D:\MCA\Sem3\DS\multiple_regression.py
[-0.0253798   1.42566386  0.8990623   2.02967212  1.03866694  1.82620927
 -0.01620682 -0.20290979  1.64869649  1.15875699 -0.03774953  1.66220522
  1.7511773   0.85154421  1.36415556  0.02207576 -0.11752888 -0.13631914
  2.12740554  1.1483478   1.97978186  1.28481823  1.6985422  -0.02298965
  1.87205166  1.34581466  1.77116175 -0.13389939  1.9319344   1.76755767
  1.54570126  1.25608053 -0.04372325 -0.03855611  2.0336433  -0.15110608
  1.9008339   1.13482666 -0.05254633  1.16953074 -0.127933    0.81645584
  1.70371187  1.47068727 -0.01862177]
[ 0.93012762  0.09071426  1.15313543  0.96406458  2.15970707  1.69851006
  1.56287102  0.1677265   1.63426424 -0.01381187 -0.01540531  0.83321571
  1.71208307  0.87948519 -0.0577036  -0.06687405 -0.13712319  1.16072485
  1.25686485 -0.06286087  1.95067707 -0.146641    1.01632271  1.9817548
  1.28722811  1.30795917  2.22631012 -0.03054948  1.57444149  0.05599299
  2.16813825  1.3821319   1.982991   -0.09002486  1.54773672  1.50222675
 -0.04095613  1.53255788  1.47632123  1.71410358  1.86014877  1.12921018
  1.30361835  1.19902237  1.26924444  1.31073615 -0.08921827 -0.10756172
 -0.05290356  1.08290603  1.66977237 -0.11671976  1.7946946   1.40927362
  1.99579767  2.07790466  2.04603275  1.14914198  0.03329153  1.21217386
  1.23736013  1.12323646  1.8868386   1.10489556  1.8816616  -0.00624472
  0.89026881  1.37260168  0.2167879   1.36655811  1.94587957 -0.07684122
 -0.00944879  1.56883487 -0.01025283 -0.06846749  0.97403175  1.03307272
  1.56091049  1.71208307  1.75637227  2.04357833 -0.03695027  1.13517403
  0.06675687  1.13162006  0.01014806  0.84397706  1.14877742  1.07178745
  1.27920907 -0.07764781 -0.09678543  0.04360099 -0.09919531  0.13735337
  1.14442674 -0.01621189 -0.08165112  1.72765939  1.83533772  1.70289034
  1.1918651   1.98099579  0.0281416 ]
r2 score:  0.9334879512610564
mean square error:  0.04847989330305221

Process finished with exit code 0
```

EXPERIMENT NO 5

AIM:

Program to implement text classification using Support vector machine.

Source Code

```
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import pandas as pd

iris=load_iris()
df=pd.DataFrame(iris.data, columns=iris.feature_names)
df['target']=iris.target
print(df.head())

x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

svm=SVC(kernel='linear')
svm.fit(x_train,y_train)
y_pred=svm.predict(x_test)
print(f"prediction of testing variables:\n {y_pred}")

s=[[5,5,4,4]]
pred=svm.predict(s)
pred_v=[iris.target_names[p] for p in pred]
print(f"prediction of sample [5,5,4,4]: {pred_v}")
print(f"Accuracy score:  {accuracy_score(y_test,y_pred)}")
print(f"confusion matrix:\n  {confusion_matrix(y_test,y_pred)}")
```

Output:

```
C:\Users\rahba\AppData\Local\Programs\Python\Python312\python.exe D:\MCA\Sem3\DS\svm.py
   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  target
0                5.1               3.5  ...               0.2       0
1                4.9               3.0  ...               0.2       0
2                4.7               3.2  ...               0.2       0
3                4.6               3.1  ...               0.2       0
4                5.0               3.6  ...               0.2       0

[5 rows x 5 columns]
prediction of testing variables:
 [2 2 1 1 1 2 0 1 1 1 2 0 1 1 2 2 2 1 2 0 2 0 0 1 2 2 0 1 0 2 0 0 2 0 0 2 2
 1 2 2 2 0 0 1 1]
prediction of sample [5,5,4,4]: [np.str_('virginica')]
Accuracy score:  0.9777777777777777
confusion matrix:
  [[13  0  0]
 [ 0 14  1]
 [ 0  0 17]]

Process finished with exit code 0
```

EXPERIMENT NO 6

AIM:

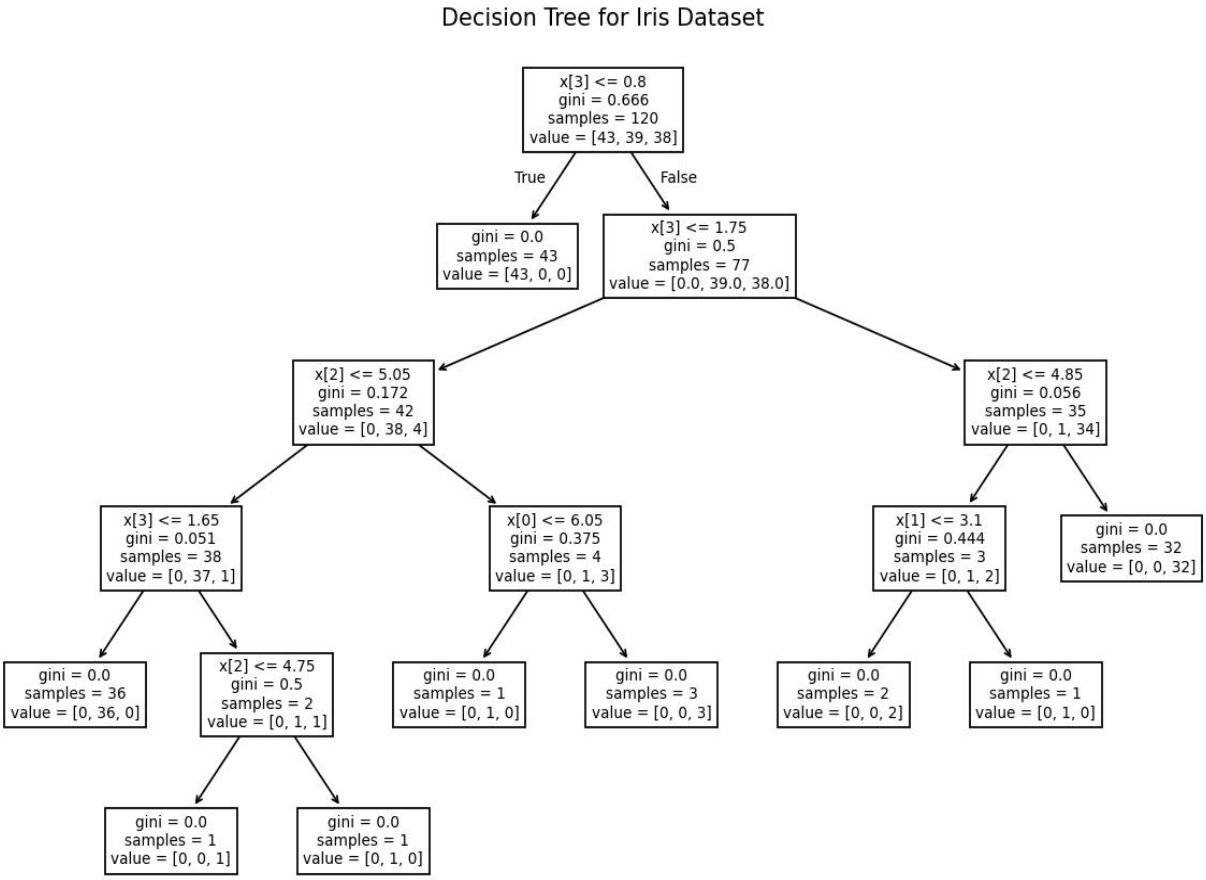
Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.

Source Code

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import matplotlib.pyplot as plt

iris=load_iris()
df=pd.DataFrame(iris.data, columns=iris.feature_names)
df['target']=iris.target
print(df.head())

x = iris.data
y = iris.target
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20)

classifier = DecisionTreeClassifier()
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print(f"prediction of testing values: \n {y_pred}")
ac = accuracy_score(y_test, y_pred)
print(f"Accuracy: ",ac)

plt.figure(figsize=(12, 8))
tree.plot_tree(classifier)
plt.title("Decision Tree for Iris Dataset")
plt.show()
```

Output:

```
C:\Users\rahba\AppData\Local\Programs\Python\Python312\python.exe "D:\MCA\Sem3\DS\decision tree.py"
   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  target
0                5.1               3.5  ...               0.2       0
1                4.9               3.0  ...               0.2       0
2                4.7               3.2  ...               0.2       0
3                4.6               3.1  ...               0.2       0
4                5.0               3.6  ...               0.2       0

[5 rows x 5 columns]
prediction of testing values:
 [2 1 1 0 2 1 1 2 2 1 1 0 0 1 1 2 0 1 1 0 2 2 2 2 0 1 0 1 2 2]
Accuracy:  0.9666666666666667
```


Decision Tree for Iris Dataset
x[3] <= 0.8
gini = 0.666
samples = 120
value = [43, 39, 38]
True
False
gini = 0.0
samples = 43
value = [43, 0, 0]
x[3] <= 1.75
gini = 0.5
samples = 77
value = [0.0, 39.0, 38.0]
x[2] <= 5.05
gini = 0.172
samples = 42
value = [0, 38, 4]
x[2] <= 4.85
gini = 0.056
samples = 35
value = [0, 1, 34]
x[3] <= 1.65
gini = 0.051
samples = 38
value = [0, 37, 1]
x[0] <= 6.05
gini = 0.375
samples = 4
value = [0, 1, 3]
x[1] <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
gini = 0.0
samples = 32
value = [0, 0, 32]
gini = 0.0
samples = 36
value = [0, 36, 0]
x[2] <= 4.75
gini = 0.5
samples = 2
value = [0, 1, 1]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 3
value = [0, 0, 3]
gini = 0.0
samples = 2
value = [0, 0, 2]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 1
value = [0, 0, 1]
gini = 0.0
samples = 1
value = [0, 1, 0]

EXPERIMENT NO 7

AIM:

Program to implement k-means clustering technique using any standard dataset available in the public Domain.

Source Code

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import pandas as pd


iris=load_iris()
df=pd.DataFrame(iris.data, columns=iris.feature_names)
df['target']=iris.target
print(df.head())


x=df.iloc[:, :4]
print(x.head())
km=KMeans(n_clusters=3)
print("fitting model")
print(km.fit(x))
y=km.predict(x)
print("predicted clustres")
print(y)
centroid=km.cluster_centers_
print("cluster centroids")
print(centroid)
```

Output:

```
C:\Users\rahba\AppData\Local\Programs\Python\Python312\python.exe D:\MCA\Sem3\DS\kmean.py
   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  target
0                5.1               3.5  ...               0.2       0
1                4.9               3.0  ...               0.2       0
2                4.7               3.2  ...               0.2       0
3                4.6               3.1  ...               0.2       0
4                5.0               3.6  ...               0.2       0

[5 rows x 5 columns]
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
fitting model
KMeans(n_clusters=3)
predicted clustres
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 0 0 0 2 0 0 0 0
 0 0 2 2 0 0 0 0 2 0 2 0 2 0 0 2 2 0 0 0 0 0 2 0 0 0 0 2 0 0 0 2 0 0 0 2 0
 0 2]
cluster centroids
[[6.85       3.07368421 5.74210526 2.07105263]
 [5.006      3.428      1.462      0.246     ]
 [5.9016129  2.7483871  4.39354839 1.43387097]]

Process finished with exit code 0
```