MASTER OF COMPUTER APPLICATIONS

PRACTICAL RECORD WORK

ON

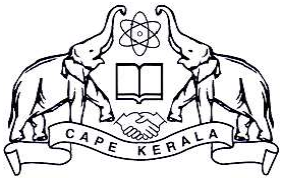
20MCA241 DATA SCIENCE LAB

Submitted

By

……..

(Reg. No. : ……..)



DEPARTMENT OF COMPUTER APPLICATIONS
COLLEGE OF ENGINEERING VADAKARA
(CAPE - GOVT. OF KERALA)

DECEMBER - 2021

DEPARTMENT OF COMPUTER APPLICATIONS
COLLEGE OF ENGINEERING VADAKARA
(CAPE - GOVT. OF KERALA)

CERTIFICATE

Certified that this is a bonafide record of the practical work on the course 20MCA241 DATA SCIENCE LAB done by Mr./Ms. ……. (Reg.No.: ……) Third Semester MCA student of Department of Computer Applications at College of Engineering Vatakara in the partial fulfilment for the award of the degree of Master of Computer Applications (MCA) of APJ Abdul Kalam Technological University (KTU)

(Ms. NIDHINA K)
FACULTY-IN-CHARGE

HEAD OF THE DEPARTMENT

DATE:

EXAMINERS:

Experiment No 1

Review of python programming, Matrix operations, Programs using matplotlib / plotly / bokeh / seaborn for data visualisation and programs to handle data using pandas.

Source Code

Review of python programming:

1. Write a program to copy a text file to another file.

```
f1=open("demo1.txt","r")
f2=open("demo2.txt","w")
for i in f1:
    f2.write(i)
f1.close()
f2.close()
```

Output:

```
Process finished with exit code 0
```

2. Write a program to count the number of lines in a file.

```
f1=open("demo1.txt","r")
count=0
for i in f1:
    if i !="\n":
        count += 1
f1.close()
print(count)
```

Output:

```
4

Process finished with exit code 0
```

3. Write a program to count number of words

```
from collections import Counter
def word_count(fname):
    with open(fname) as f:
        return Counter(f.read().split())
print("number of words:",word_count("demo1.txt"))
```

Output:

```
number of words: Counter({'Aswin': 2, 'Kailas': 2, 'Dilshad': 1})

Process finished with exit code 0
```

4. Write a program to append a file with the contents of another file.

```
f1=open("demo1.txt","r")
data=f1.read()
f2=open("demo2.txt","a")
f2.write(data)
f1.close()
f2.close()
```

Output:

```
Process finished with exit code 0
```

5. Write a program to compare two files

```
f1=open("demo1.txt","r")
f2=open("demo2.txt","r")
i=0
for j in f1:
    i += 1;
    for k in f2:
        if j == k:
            print("Both the lines are same")
        else:
            print("Both  the lines are not same")
f1.close()
f2.close()
```

Output:

```
Both the lines are same

Process finished with exit code 0
```

6. Write a program to delete a sentence from the specified position in a file

```
f_f1=open("demo1.txt","r")
line=f1.readlines()
f1.close()

f2=open("demo1.txt","w")
for lines in line:
    if lines.strip("\n") != "Vyshak":
        f2.write(lines)

f2.close()
```

Output:

```
Process finished with exit code 0
```

Matrix Operations:

1. Write a program to get maximum and minimum number from given matrix.

```
import numpy as np
m=int(input("Enter no of rows : "))
n=int(input("Enter no of cols : "))
print("Enter ",m*n," elements : ")
arr=np.array([])
for i in range(m*n):
    e=int(input())
    arr=np.append(arr,e)
arry = arr.astype(int)

print("The Matrix is  : ")

mat=arry.reshape(m,n)
print(mat)

minm=np.min(mat)

print("The minimum value from the given matrix is : ",minm)

maxm=np.max(mat)

print("The minimum value from the given matrix is : ",maxm)
```

Output:

```
Enter no of rows : 3
Enter no of cols : 3
Enter  9  elements :
9
3
4
5
6
7
8
2
4
The Matrix is  :
[[9 3 4]
 [5 6 7]
 [8 2 4]]
The minimum value from the given matrix is :  2
The minimum value from the given matrix is :  9

Process finished with exit code 0
```

2. Write a program to find the number of rows and columns of a given matrix using numpy

```
import numpy as np
m=int(input("Enter no of rows : "))
n=int(input("Enter no of cols : "))
print("Enter ",m*n," elements : ")
arr=np.array([])
for i in range(m*n):
    e=int(input())
    arr=np.append(arr,e)
arry = arr.astype(int)

print("The Matrix is  : ")

mat=arry.reshape(m,n)
print(mat)

print("Number of rows from the matrix is : ")
print(mat.shape[0])

print("Number of columns from the matrix is : ")
print(mat.shape[1])
```

Output:

```
Enter no of rows : 3
Enter no of cols : 2
Enter  6  elements :
1
2
3
4
5
6
The Matrix is  :
[[1 2]
 [3 4]
 [5 6]]
Number of rows from the matrix is :
3
Number of columns from the matrix is :
2

Process finished with exit code 0
```

3. Write a program to find sum of values in a matrix

```
import numpy as np
m=int(input("Enter no of rows : "))
n=int(input("Enter no of cols : "))
print("Enter ",m*n," elements : ")
arr=np.array([])
for i in range(m*n):
    e=int(input())
    arr=np.append(arr,e)
arry = arr.astype(int)

print("The Matrix is  : ")

mat=arry.reshape(m,n)
print(mat)

sumv=np.sum(mat)
print("Sum of values in the matrix is : ",sumv)
```

Output:

```
Enter no of rows : 2
Enter no of cols : 2
Enter  4  elements :
1
2
3
4
The Matrix is  :
[[1 2]
 [3 4]]
Sum of values in the matrix is :  10

Process finished with exit code 0
```

4. Write a program to calculate the sum of diagonal elements of a numpy array.

```
import numpy as np
m=int(input("Enter no of rows : "))
n=int(input("Enter no of cols : "))
print("Enter ",m*n," elements : ")
arr=np.array([])
for i in range(m*n):
    e=int(input())
    arr=np.append(arr,e)
arry = arr.astype(int)

print("The Matrix is  : ")

mat=arry.reshape(m,n)
print(mat)

tracem=np.trace(mat)
print("Sum of diagonal elements in the matrix is : ",tracem)
```

Output:

```
Enter no of rows : 3
Enter no of cols : 3
Enter  9  elements :
1
2
3
4
5
6
7
8
9
The Matrix is  :
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Sum of diagonal elements in the matrix is :  15

Process finished with exit code 0
```

5. Write a program to demonstrate matrix addition,subtraction and division

```
import numpy as np
m=int(input("Enter no of rows : "))
n=int(input("Enter no of cols : "))
print("Enter ",m*n," elements for first matrix : ")
arr=np.array([])
for i in range(m*n):
    e=int(input())
    arr=np.append(arr,e)
arry = arr.astype(int)

print("Enter ",m*n," elements for second matrix : ")
arra=np.array([])
for j in range(m*n):
    v=int(input())
    arra=np.append(arra,v)
arri = arra.astype(int)

print("The first Matrix is  : ")

mat1=arry.reshape(m,n)
print(mat1)

print("The second Matrix is  : ")

mat2=arri.reshape(m,n)
print(mat2)

add=np.add(mat1,mat2)
print("Addition of two matrix is : ",add)


print("Substraction of two matrix is : ",np.subtract(mat1,mat2))

print("Division of two matrix is : ",np.divide(mat1,mat2))
```

Output:

```
Enter no of rows : 3
Enter no of cols : 2
Enter  6  elements for first matrix :
1
2
3
4
5
6
Enter  6  elements for second matrix :
7
8
9
2
3
5
The first Matrix is  :
[[1 2]
 [3 4]
 [5 6]]
The second Matrix is  :
[[7 8]
 [9 2]
 [3 5]]
Addition of two matrix is :  [[ 8 10]
 [12  6]
 [ 8 11]]
Substraction of two matrix is :  [[-6 -6]
 [-6  2]
 [ 2  1]]
Division of two matrix is :  [[0.14285714 0.25      ]
 [0.33333333 2.        ]
 [1.66666667 1.2       ]]

Process finished with exit code 06
```

Programs using matplotlib / plotly / bokeh / seaborn for data visualisation and programs to handle data using pandas.

In [1]:

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

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

print(data)
```

```
     sepal.length  sepal.width  petal.length  petal.width    variety
0             5.1          3.5           1.4          0.2     Setosa
1             4.9          3.0           1.4          0.2     Setosa
2             4.7          3.2           1.3          0.2     Setosa
3             4.6          3.1           1.5          0.2     Setosa
4             5.0          3.6           1.4          0.2     Setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  Virginica
146           6.3          2.5           5.0          1.9  Virginica
147           6.5          3.0           5.2          2.0  Virginica
148           6.2          3.4           5.4          2.3  Virginica
149           5.9          3.0           5.1          1.8  Virginica

[150 rows x 5 columns]
```

In [2]:

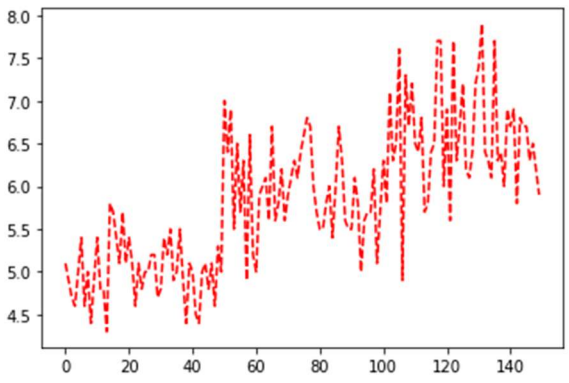
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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

plt.plot(data["sepal.length"],"r--")
plt.show
```

Out[2]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

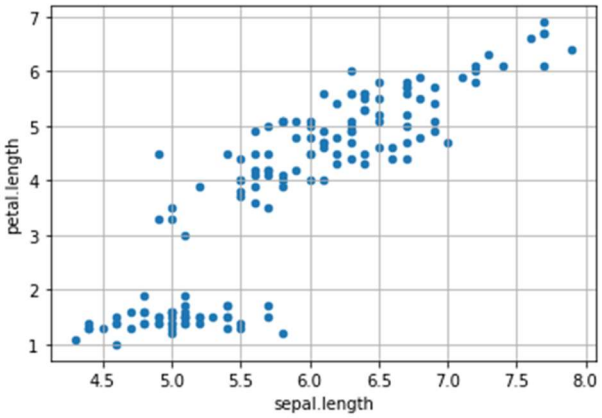


In [3]:

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

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

data.plot(kind="scatter",x="sepal.length",y="petal.length")
plt.grid()
plt.show()
```



In [4]:

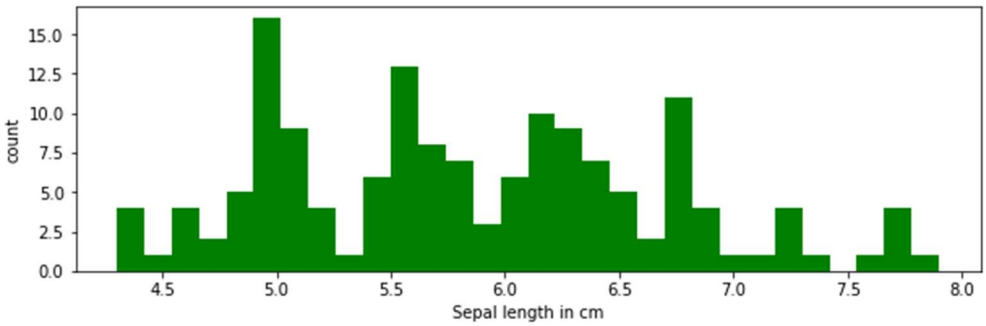
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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

plt.figure(figsize=(10,3))
x=data["sepal.length"]
plt.hist(x,bins=30,color="green")
plt.xlabel("Sepal length in cm")
plt.ylabel("count")
plt.show
```

Out[4]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [5]:

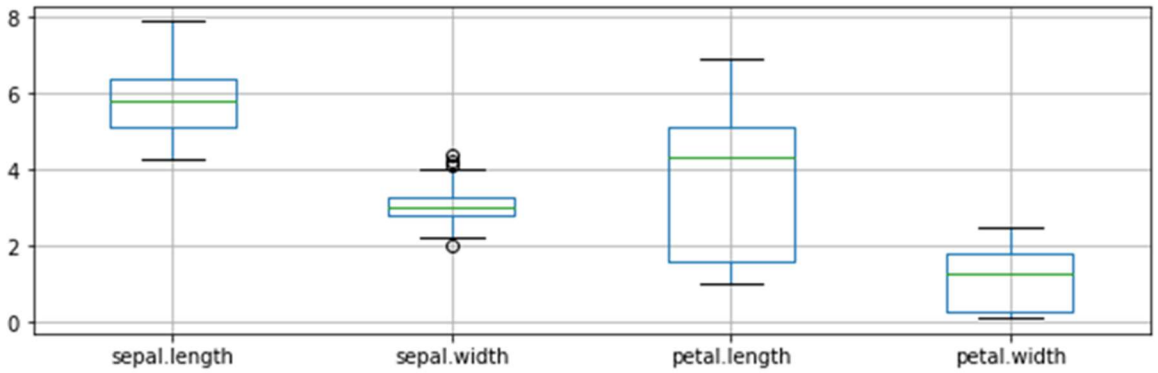
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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

plt.figure(figsize=(10,3))
data.boxplot()
```

Out[5]:

```
<AxesSubplot:>
```


8
6
4
2
0
sepal.length
sepal.width
petal.length
petal.width

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

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
```

In [3]:
```
data=pd.read_csv('iris.csv')
```

In [5]:
```
data.head()
```

Out[5]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [7]:
```
x=data.iloc[:,:4]
x.head()
```

Out[7]:

	sepal.length	sepal.width	petal.length	petal.width
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

In [8]:
```
y=data.iloc[:,-1]
y.head()
```

Out[8]:
```
0    Setosa
1    Setosa
2    Setosa
3    Setosa
4    Setosa
Name: variety, dtype: object
```

In [9]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20)
```

In [10]:
```
x_train.head()
```

Out[10]:

	sepal.length	sepal.width	petal.length	petal.width
89	5.5	2.5	4.0	1.3
63	6.1	2.9	4.7	1.4
114	5.8	2.8	5.1	2.4
146	6.3	2.5	5.0	1.9
134	6.1	2.6	5.6	1.4

In [12]:
```
x_test.head()
```

Out[12]:

	sepal.length	sepal.width	petal.length	petal.width
141	6.9	3.1	5.1	2.3
76	6.8	2.8	4.8	1.4
91	6.1	3.0	4.6	1.4
103	6.3	2.9	5.6	1.8
121	5.6	2.8	4.9	2.0

In [13]:
```
sc=StandardScaler()
sc.fit(x_train)
x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
```

In [42]:
```
classifier=KNeighborsClassifier(n_neighbors=5)
```

In [43]:
```
classifier.fit(x_train,y_train)
```

Out[43]:
```
KNeighborsClassifier()
```

In [44]:
```
y_pred=classifier.predict(x_test)
```

In [45]:
```
y_pred
```

Out[45]:
```
array(['Virginica', 'Versicolor', 'Versicolor', 'Virginica', 'Virginica',
       'Versicolor', 'Setosa', 'Versicolor', 'Setosa', 'Virginica',
       'Setosa', 'Versicolor', 'Virginica', 'Versicolor', 'Virginica',
       'Virginica', 'Setosa', 'Virginica', 'Virginica', 'Setosa',
       'Versicolor', 'Versicolor', 'Setosa', 'Versicolor', 'Virginica',
       'Virginica', 'Virginica', 'Versicolor', 'Setosa', 'Versicolor'],
      dtype=object)
```

```
In [46]: y_test

Out[46]: 141     Virginica
         76     Versicolor
         91     Versicolor
         103     Virginica
         121     Virginica
         98     Versicolor
         29         Setosa
         58     Versicolor
         23         Setosa
         119     Virginica
         27         Setosa
         53     Versicolor
         124     Virginica
         87     Versicolor
         147     Virginica
         137     Virginica
         38         Setosa
         127     Virginica
         109     Virginica
         42         Setosa
         73     Versicolor
         96     Versicolor
         26         Setosa
         94     Versicolor
         72     Versicolor
         129     Virginica
         144     Virginica
         71     Versicolor
         46         Setosa
         88     Versicolor
         Name: variety, dtype: object

In [47]: from sklearn.metrics import confusion_matrix, accuracy_score
         cm=confusion_matrix(y_test,y_pred)
         ac=accuracy_score(y_test,y_pred)

In [48]: cm

Out[48]: array([[ 7,  0,  0],
                [ 0, 11,  1],
                [ 0,  0, 11]], dtype=int64)

In [49]: ac

Out[49]: 0.9666666666666667
```

Result: Successfully implemented k-NN classification using **Iris** dataset and the accuracy of the algorithm is **0.9666666666666667.**

Experiment No 3

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

Source Code

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
```

In [3]:
```
data=pd.read_csv('iris.csv')
```

In [5]:
```
data.head()
```

Out[5]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [7]:
```
x=data.iloc[:,:4]
x.head()
```

Out[7]:

	sepal.length	sepal.width	petal.length	petal.width
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

In [8]:
```
y=data.iloc[:,-1]
y.head()
```

Out[8]:
```
0    Setosa
1    Setosa
2    Setosa
3    Setosa
4    Setosa
Name: variety, dtype: object
```

In [9]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20)
```

In [10]:
```
x_train.head()
```

Out[10]:

	sepal.length	sepal.width	petal.length	petal.width
89	5.5	2.5	4.0	1.3
63	6.1	2.9	4.7	1.4
114	5.8	2.8	5.1	2.4
146	6.3	2.5	5.0	1.9
134	6.1	2.6	5.6	1.4

In [12]:
```
x_test.head()
```

Out[12]:

	sepal.length	sepal.width	petal.length	petal.width
141	6.9	3.1	5.1	2.3
76	6.8	2.8	4.8	1.4
91	6.1	3.0	4.6	1.4
103	6.3	2.9	5.6	1.8
121	5.6	2.8	4.9	2.0

In [13]:
```
sc=StandardScaler()
sc.fit(x_train)
x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
```

In [34]:
```
classifier=GaussianNB()
```

In [35]:
```
classifier.fit(x_train,y_train)
```

Out[35]:
```
GaussianNB()
```

In [36]:
```
y_pred=classifier.predict(x_test)
```

In [37]:
```
y_pred
```

Out[37]:
```
array(['Virginica', 'Versicolor', 'Versicolor', 'Virginica', 'Virginica',
       'Versicolor', 'Setosa', 'Versicolor', 'Setosa', 'Versicolor',
       'Setosa', 'Versicolor', 'Virginica', 'Versicolor', 'Virginica',
       'Virginica', 'Setosa', 'Virginica', 'Virginica', 'Setosa',
       'Versicolor', 'Versicolor', 'Setosa', 'Versicolor', 'Versicolor',
       'Virginica', 'Virginica', 'Versicolor', 'Setosa', 'Versicolor'],
      dtype='<U10')
```

```
In [38]: y_test

Out[38]: 141     Virginica
         76     Versicolor
         91     Versicolor
         103     Virginica
         121     Virginica
         98     Versicolor
         29         Setosa
         58     Versicolor
         23         Setosa
         119     Virginica
         27         Setosa
         53     Versicolor
         124     Virginica
         87     Versicolor
         147     Virginica
         137     Virginica
         38         Setosa
         127     Virginica
         109     Virginica
         42         Setosa
         73     Versicolor
         96     Versicolor
         26         Setosa
         94     Versicolor
         72     Versicolor
         129     Virginica
         144     Virginica
         71     Versicolor
         46         Setosa
         88     Versicolor
         Name: variety, dtype: object

In [39]: from sklearn.metrics import confusion_matrix, accuracy_score
         cm=confusion_matrix(y_test,y_pred)
         ac=accuracy_score(y_test,y_pred)

In [40]: cm

Out[40]: array([[ 7,  0,  0],
                [ 0, 12,  0],
                [ 0,  1, 10]], dtype=int64)

In [41]: ac

Out[41]: 0.9666666666666667
```

Result: Successfully implemented Naïve Bayes Algorithm using **Iris** dataset and the accuracy of the algorithm is **0.9666666666666667.**

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

In [27]:

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

In [17]:

```
data=pd.read_csv('dia.csv')
```

In [18]:

```
data.head()
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [19]:

```
x=data.iloc[:,7]
x.head()
```

Out[19]:

```
0    50
1    31
2    32
3    21
4    33
Name: Age, dtype: int64
```

In [20]:

```
y=data.iloc[:,6]
y.head()
```

Out[20]:

```
0    0.627
1    0.351
2    0.672
3    0.167
4    2.288
Name: DiabetesPedigreeFunction, dtype: float64
```

In [21]:

```
x=np.array(x).reshape(-1,1)
```

In [22]:

```
x
```

```
       [42],
       [45],
       [38],
       [25],
       [22],
       [22],
       [22],
       [34],
       [22],
       [24],
       [22],
       [53],
       [28],
       [21],
       [42],
       [21],
       [42],
       [48],
       [26],
       [22],
```

In [23]:

```
y=np.array(y).reshape(-1,1)
```

In [24]:

```
y
```

Out[24]:

```
array([[0.627],
       [0.351],
       [0.672],
       [0.167],
       [2.288],
       [0.201],
       [0.248],
       [0.134],
       [0.158],
       [0.232],
       [0.191],
       [0.537],
       [1.441],
       [0.398],
       [0.587],
       [0.484],
       [0.551],
       [0.254],
```

In [25]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20)
```

In [28]:

```
classifier=LinearRegression()
```

In [29]:

```
classifier.fit(x_train,y_train)
```

Out[29]:

```
LinearRegression()
```

In [30]:

```
y_pred=classifier.predict(x_test)
```

In [31]:

```
y_pred
```

Out[31]:

```
array([[0.45921739],
       [0.47215969],
       [0.46784559],
       [0.48606067],
       [0.46065542],
       [0.45730001],
       [0.45730001],
       [0.4582587 ],
       [0.45777936],
       [0.46065542],
       [0.46544887],
       [0.47215969],
       [0.471201  ],
       [0.45730001],
       [0.46017608],
       [0.46640756],
       [0.46209346],
       [0.45777936],
```

In [34]:

```
r2_score(y_test,y_pred)
```

Out[34]:

```
-0.013647232579697022
```

In [36]:

```
mean_squared_error(y_test,y_pred)
```

Out[36]:

```
0.12172988403486278
```

In [38]:

```
classifier.coef_
```

Out[38]:

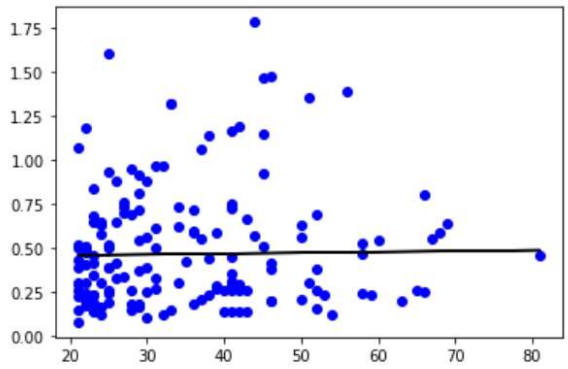
```
array([[0.00047934]])
```

In [39]:

```
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
```

Out[39]:

```
[<matplotlib.lines.Line2D at 0x180f950a560>]
```



In [ ]:

Multiple Regression

In [1]:

```
#Linear_regression

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

irisData = load_iris()

X = irisData.data
Y = irisData.target

x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size=7, random_state=42)

model = linear_model.LinearRegression()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
y_pred_train = model.predict(x_train)
print("Mean square error (0.2<x<0.5) =",mean_squared_error(y_test, y_pred))
print("r2Score (x>0.7) =",r2_score(y_test,y_pred))
```

```
Mean square error (0.2<x<0.5) = 0.0445766362839052
r2Score (x>0.7) = 0.8907872411044322
```

Result: Successfully implemented implement linear and multiple regression techniques.

Experiment No 5

Aim: Program to implement text classification using Support vector machine.

Source Code

In [22]:

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

In [23]:

```
data=pd.read_csv('iris.csv')
```

In [24]:

```
data.head()
```

Out[24]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [25]:

```
x=data.iloc[:,:-4]
y=data.iloc[:,-1]
```

In [26]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20)
```

In [27]:

```
classifier=SVC(kernel='linear')
```

In [28]:

```
classifier.fit(x_train,y_train)
```

Out[28]:

```
SVC(kernel='linear')
```

In [29]:

```
y_pred=classifier.predict(x_test)
```

In [30]:

```
y_pred
```

Out[30]:

```
array(['Virginica', 'Setosa', 'Virginica', 'Setosa', 'Virginica',
       'Setosa', 'Setosa', 'Setosa', 'Virginica', 'Virginica',
       'Virginica', 'Setosa', 'Virginica', 'Virginica', 'Setosa',
       'Versicolor', 'Virginica', 'Versicolor', 'Virginica', 'Virginica',
       'Setosa', 'Versicolor', 'Setosa', 'Virginica', 'Versicolor',
       'Setosa', 'Versicolor', 'Virginica', 'Setosa', 'Virginica'],
      dtype=object)
```

In [31]:

```
cm=confusion_matrix(y_test,y_pred)
```

In [32]:

```
cm
```

Out[32]:

```
array([[ 9,  0,  0],
       [ 1,  5,  2],
       [ 1,  0, 12]], dtype=int64)
```

In [33]:

```
ac=accuracy_score(y_test,y_pred)
ac
```

Out[33]:

```
0.8666666666666667
```

In [ ]:

```

```

Result: Successfully implemented text classification using Support vector machine using **Iris** dataset and the accuracy of the algorithm is **0.8666666666666667.**

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

In [2]:

```
import pandas as pd
import numpy as np
import sklearn
from sklearn import tree
import matplotlib as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

In [3]:

```
data=pd.read_csv('iris.csv')
```

In [4]:

```
data.head()
```

Out[4]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [5]:

```
x=data.iloc[:,:4]
x.head()
```

Out[5]:

	sepal.length	sepal.width	petal.length	petal.width
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

In [6]:

```
y=data.iloc[:,-1]
y.head()
```

Out[6]:

```
0    Setosa
1    Setosa
2    Setosa
3    Setosa
4    Setosa
Name: variety, dtype: object
```

In [7]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20)
```

In [8]:

```
classifier=DecisionTreeClassifier()
```

In [9]:

```
classifier.fit(x_train,y_train)
```

Out[9]:

```
DecisionTreeClassifier()
```

In [10]:

```
y_pred=classifier.predict(x_test)
```

In [11]:

```
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_test,y_pred)
```

In [12]:

```
ac
```

Out[12]:

```
0.9
```

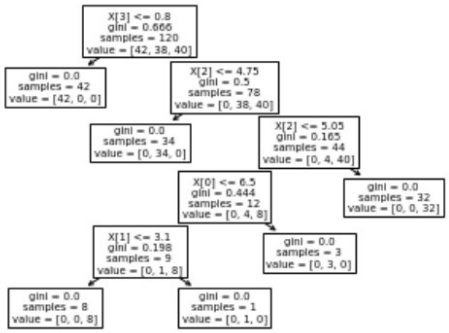
In [13]:

```
from sklearn import tree

tree.plot_tree(classifier)
```

Out[13]:

```
[Text(0.3333333333333333, 0.9166666666666666, 'X[3] <= 0.8\ngini = 0.666\nsa
mples = 120\nvalue = [42, 38, 40]'),
 Text(0.16666666666666666, 0.75, 'gini = 0.0\nsamples = 42\nvalue = [42, 0,
0]'),
 Text(0.5, 0.75, 'X[2] <= 4.75\ngini = 0.5\nsamples = 78\nvalue = [0, 38, 4
0]'),
 Text(0.3333333333333333, 0.5833333333333334, 'gini = 0.0\nsamples = 34\nval
ue = [0, 34, 0]'),
 Text(0.6666666666666666, 0.5833333333333334, 'X[2] <= 5.05\ngini = 0.165\ns
amples = 44\nvalue = [0, 4, 40]'),
 Text(0.5, 0.4166666666666667, 'X[0] <= 6.5\ngini = 0.444\nsamples = 12\nval
ue = [0, 4, 8]'),
 Text(0.3333333333333333, 0.25, 'X[1] <= 3.1\ngini = 0.198\nsamples = 9\nval
ue = [0, 1, 8]'),
 Text(0.16666666666666666, 0.08333333333333333, 'gini = 0.0\nsamples = 8\nva
lue = [0, 0, 8]'),
 Text(0.5, 0.08333333333333333, 'gini = 0.0\nsamples = 1\nvalue = [0, 1,
0]'),
 Text(0.6666666666666666, 0.25, 'gini = 0.0\nsamples = 3\nvalue = [0, 3,
0]'),
 Text(0.8333333333333334, 0.4166666666666667, 'gini = 0.0\nsamples = 32\nval
ue = [0, 0, 32]')]
```



Result: Successfully implemented decision trees using **Iris** dataset and the accuracy of the algorithm is **0.9.**

Experiment No 7

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

Source Code

In [10]:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

In [11]:

```
data=pd.read_csv('iris.csv')
data.head()
```

Out[11]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [12]:

```
x=data.iloc[:,:4]
x.head()
```

Out[12]:

	sepal.length	sepal.width	petal.length	petal.width
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

In [13]:

```
km=KMeans(n_clusters=3)
```

In [14]:

```
km.fit(x)
```

Out[14]:

```
KMeans(n_clusters=3)
```

In [15]:

```
y=km.predict(x)
```

In [16]:

```
y
```

Out[16]:

```
array([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, 0, 2, 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, 2, 2, 0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 0, 2, 2, 0, 0,
       0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2])
```

In [24]:

```
centroid=km.cluster_centers_
centroid
```

Out[24]:

```
array([[6.85      , 3.07368421, 5.74210526, 2.07105263],
       [5.006     , 3.428     , 1.462     , 0.246     ],
       [5.9016129 , 2.7483871 , 4.39354839, 1.43387097]])
```

Result: Successfully implemented k-means clustering technique using **Iris** dataset**.**

Experiment No 8

Aim: Program on convolutional neural network to classify images from any standard dataset in the public domain using Keras framework.

Source Code

In [17]:

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
```

In [2]:

```
(X_train,y_train) , (X_test,y_test)=mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-dat
asets/mnist.npz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/mnist.npz)
11493376/11490434 [==============================] - 35s 3us/step
11501568/11490434 [==============================] - 35s 3us/step
```

In [3]:

```
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], X_train.shape[2], 1))
X_test = X_test.reshape((X_test.shape[0],X_test.shape[1],X_test.shape[2],1))
```

In [5]:

```
print(X_train.shape)
print(X_test.shape)
```

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
```

In [6]:

```
X_train=X_train/255
X_test=X_test/255
```

In [7]:

```
model=Sequential()
```

In [8]:

```
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
```

In [9]:

```
model.add(MaxPool2D(2,2))
```

In [10]:

```
model.add(Flatten())
```

In [11]:

```
model.add(Dense(100,activation='relu'))
```

In [12]:

```
model.add(Dense(10,activation='softmax'))
```

In [13]:

```
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [14]:

```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 30s 12ms/step - loss: 0.1682 -
accuracy: 0.9504
Epoch 2/10
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0569 -
accuracy: 0.9830
Epoch 3/10
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0373 -
accuracy: 0.9886
Epoch 4/10
1875/1875 [==============================] - 23s 12ms/step - loss: 0.0249 -
accuracy: 0.9923
Epoch 5/10
1875/1875 [==============================] - 24s 13ms/step - loss: 0.0173 -
accuracy: 0.9942
Epoch 6/10
1875/1875 [==============================] - 25s 13ms/step - loss: 0.0136 -
accuracy: 0.9955
Epoch 7/10
1875/1875 [==============================] - 23s 12ms/step - loss: 0.0098 -
accuracy: 0.9969
Epoch 8/10
1875/1875 [==============================] - 26s 14ms/step - loss: 0.0071 -
accuracy: 0.9976
Epoch 9/10
1875/1875 [==============================] - 26s 14ms/step - loss: 0.0063 -
accuracy: 0.9981
Epoch 10/10
1875/1875 [==============================] - 25s 13ms/step - loss: 0.0051 -
accuracy: 0.9983
```

Out[14]:

```
<keras.callbacks.History at 0x2bdba4be3b0>
```

In [15]:

```
model.evaluate(X_test,y_test)
```

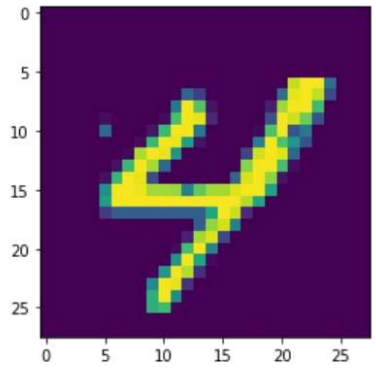
```
313/313 [==============================] - 2s 5ms/step - loss: 0.0532 - accu
racy: 0.9858
```

Out[15]:

```
[0.05316377058625221, 0.98580002784729]
```

In [20]:

```
plt.imshow(X_train[89, :])
plt.show()
```



In [26]:

```
y_pred=model.predict(X_test[89, :].reshape(1,28,28,1))
y_pred
```

Out[26]:

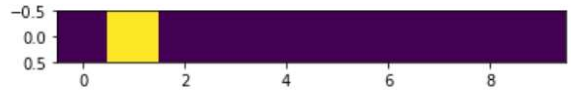
```
array([[9.6334700e-09, 9.9998164e-01, 3.5556809e-06, 1.1106345e-06,
        2.0665028e-07, 2.1412344e-10, 2.0225615e-09, 9.0603744e-06,
        4.4473359e-06, 2.2387640e-08]], dtype=float32)
```

In [28]:

```
plt.imshow(y_pred)
```

Out[28]:

```
<matplotlib.image.AxesImage at 0x2bdd7705360>
```



Result: Successfully implemented convolutional neural network to classify images using Keras framework.