**MOUNT ZION COLLEGE OF ENGINEERING**

**Kadammanitta – Pathanamthitta Kerala 689649**

***(Affiliated to APJ Abdul Kalam Technological University)***



**20MCA241 DATA SCIENCE LAB**

**LABORATORY RECORD**

**SECOND YEAR**

**Submitted by**

**P J SREEDEEP**

**MZC21MCA-2021**

***Submitted in partial fulfillment of the requirement for the***

***Award of the Degree***

***of***

**MASTER OF COMPUTER APPLICATIONS**

**(2021-2023)**

**Department of Computer Applications**

**MOUNT ZION COLLEGE OF ENGINEERING, KADAMMANITTA**

**MOUNT ZION COLLEGE OF ENGINEERING**

**Kadammanitta – Pathanamthitta Kerala 689649**

***-(Affiliated to APJ Abdul Kalam Technological University)***



**CERTIFICATE**

***Certified that this is a bonafide record of practical work done in Data Science Lab (20MCA241) Laboratory by P J SREEDEEP Reg No:MZC21MCA-2021 of Mount Zion College of Engineering, Kadammanitta – Pathanamthitta during the academic year 2021-2023.***

**Head of the department Staff member in-charge**

**Internal Examiner**

**INDEX PAGE**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL.NO** | **List Of Experiments** | **Page No:** | **Date** |
| 1 | Review of python programming, Matrix operations, matplotlib, Sentiment Analysis. | 04 |  |
| 2 | Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm. | 24 |  |
| 3 | Program to implement Naive Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm. | 29 |  |
| 4 | Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance. | 31 |  |
| 5 | Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm. | 36 |  |
| 6 | Program to implement linear regression for stock market prediction using stock price dataset of any stock. | 38 |  |
| 7 | Program on Natural Language Toolkit. | 40 |  |
| 8 | Program to implement text classification using support vector machine. | 43 |  |
| 9 | Program to implement a simple web crawler and scrapping web pages. | 45 |  |
| 10 | Program to implement k-means clustering technique using any standard dataset available in the public domain. | 49 |  |

**PROGRAM NO. 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab: Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title**: Matrix Operation :

Dot product,Transpose,Inverse,Trace,Rank,Eigen,Determinant,Sentimental Analysis.

**Objectives**: Review of the python programming, matrix operations.

**DOT PRODUCT OF MATRIX:**

**Input**:

import numpy as np

def create\_matrix(mc):

print("ARRAY"+str(mc)+" Elements:")

array\_1=map(int,input().split())

array\_1=np.array(list(array\_1))

print("ARRAY"+str(mc)+", ROW COLUMN:")

row,column=map(int,input().split())

if(len(array\_1)!=(row\*column)):

print("\nRow and Column size not match with total elements !! retry")

return create\_matrix(mc)

array\_1=array\_1.reshape(row,column)

print("\nARRAY"+str(mc))

print(array\_1)

return(array\_1)

arr1=create\_matrix(1)

arr2=create\_matrix(2)

if(arr1.shape==arr2.shape):

print("\nDot product")

print(np.dot(arr1,arr2))

else:

print("\nDimensions not matching!")

**Output:**

ARRAY1 Elements: 1 2 3 4 5 6 7 8 9

ARRAY 1, ROW COLUMN:

3 3

ARRAY1

[[1 2 3]

[4 5 6]

[7 8 9]]

ARRAY2 Elements:

1 2 3 4 5 6 7 8 9

ARRAY 2, ROW COLUMN:

3 3

ARRAY2

[[1 2 3]

[4 5 6]

[7 8 9]]

Dot product

[[ 30 36 42]

[ 66 81 96]

[102 126 150]]

**Result/Observation**

Successfully completed the data science program and output is obtained.

**Mark:**

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**TRANSPOSE OF MATRIX:**

**Input:**

import numpy as np

def create\_matrix(mc):

print("\nARRAY"+str(mc)+" Elements:")

array\_1=map(int,input().split())

array\_1=np.array(list(array\_1))

print("ARRAY"+str(mc)+", ROW COLUMN:")

row,column=map(int,input().split())

if(len(array\_1)!=(row\*column)):

print("\nRow and Column size not match with total elements !! retry")

return create\_matrix(mc)

array\_1=array\_1.reshape(row,column)

print("\nARRAY"+str(mc))

print(array\_1)

print("\nTranspose:")

return(array\_1)

print(create\_matrix(1).transpose())

**Output:**

ARRAY 1 Elements: 1 2 3 4 5 6 7 8 9

ARRAY 1, ROW COLUMN: 3 3

ARRAY 1 : [[1 2 3]

[4 5 6]

[7 8 9]]

Transpose: [[1 4 7]

[2 5 8]

[3 6 9]]

Result/Observation

Successfully completed the data science program and output is obtained.

**Mark:**

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**RANK OF MATRIX:**

**Input:**

import numpy as np

def create\_matrix(mc):

print("\nARRAY"+str(mc)+"Elements:")

array=map(int,input().split())

array=np.array(list(array))

print("\n ARRAY"+str(mc)+"ROW COLUMN:")

row,column=map(int,input().split())

if(len(array)!=(row\*column)):

print("\n Row and column size not match with total elements!!retry")

return create\_matrix(mc)

array=array.reshape(row,column)

print("\n ARRAY"+str(mc))

print(array)

print("\n Rank:")

return array

print(np.linalg.matrix\_rank(create\_matrix(1)))

**Output:**

ARRAY 1 Elements: 1 2 3 4

ARRAY 1 ROW COLUMN : 2 2

ARRAY 1

[[1 2]

[3 4]]

Rank:2

Result/Observation

Successfully completed the data science program and output is obtained.

**Mark:**

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**INVERSE OF MATRIX:**

**Input:**

import numpy as np

def create\_matrix(mc):

print("\nARRAY"+str(mc)+" Elements:")

array\_1=map(int,input().split())

array\_1=np.array(list(array\_1))

print("ARRAY"+str(mc)+", ROW COLUMN:")

row,column=map(int,input().split())

if(len(array\_1)!=(row\*column)):

print("\nRow and Column size not match with total elements !! retry")

return create\_matrix(mc)

array\_1=array\_1.reshape(row,column)

print("\nARRAY"+str(mc))

print(array\_1)

print("\nInverse:")

return(array\_1)

print(np.linalg.inv(create\_matrix(1)))

**Output:**

ARRAY 1 Elements: 1 2 3 4

ARRAY1, ROW COLUMN: 2 2

ARRAY 1 [[1 2]

[3 4]]

Inverse: [[-2. 1. ]

[ 1.5 -0.5]]

Result/Observation

Successfully completed the data science program and output is obtained.

**Mark:**

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**DETERMINANT OF MATRIX:**

**Input:**

import numpy as np

def create\_matrix(mc):

print("\nARRAY"+str(mc)+" Elements:")

array\_1=map(int,input().split())

array\_1=np.array(list(array\_1))

print("ARRAY"+str(mc)+", ROW COLUMN:")

row,column=map(int,input().split())

if(len(array\_1)!=(row\*column)):

print("\nRow and Column size not match with total elements !! retry")

return create\_matrix(mc)

array\_1=array\_1.reshape(row,column)

print("\nARRAY"+str(mc))

print(array\_1)

print("\nDeterminant:")

return(array\_1)

print(np.linalg.det(create\_matrix(1)))

**Output:**

ARRAY1 Elements: 1 2 3 4 5 6 7 8 9

ARRAY1, ROW COLUMN: 3 3

ARRAY 1:

[[1 2 3]

[4 5 6]

[7 8 9]]

Determinant: 6.66133814775094e-16

Result/Observation

Successfully completed the data science program and output is obtained.

**Mark:**

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**TRACE OF MATRIX:**

**Input:**

import numpy as np

def create\_matrix(mc):

print("\nARRAY"+str(mc)+" Elements:")

array\_1=map(int,input().split())

array\_1=np.array(list(array\_1))

print("ARRAY"+str(mc)+", ROW COLUMN:")

row,column=map(int,input().split())

if(len(array\_1)!=(row\*column)):

print("\nRow and Column size not match with total elements !! retry")

return create\_matrix(mc)

array\_1=array\_1.reshape(row,column)

print("\nARRAY"+str(mc))

print(array\_1)

print("\nTrace:")

return(array\_1)

print(create\_matrix(1).trace())

**Output:**

ARRAY1 Elements: 1 2 3 4

ARRAY1, ROW COLUMN: 2 2

ARRAY1

[[1 2]

[3 4]]

Trace: 5

Result/Observation

Successfully completed the data science program and output is obtained.

**Mark:**

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**EIGEN VALUES AND EIGEN VECTORS OF MATRIX:**

**Input:**

import numpy as np

def create\_matrix(mc):

print("\nARRAY"+str(mc)+" Elements:")

array\_1=map(int,input().split())

array\_1=np.array(list(array\_1))

print("ARRAY"+str(mc)+", ROW COLUMN:")

row,column=map(int,input().split())

if(len(array\_1)!=(row\*column)):

print("\nRow and Column size not match with total elements !! retry")

return create\_matrix(mc)

array\_1=array\_1.reshape(row,column)

print("\nARRAY"+str(mc))

print(array\_1)

return array\_1

x,y=np.linalg.eig(create\_matrix(1))

print("\nE-value : ")

print(x)

print("\nE-vector : ")

print(y)

**Output:**

ARRAY 1 Elements: 1 2 3 4 5 6 7 8 9

ARRAY1, ROW COLUMN: 3 3

ARRAY 1: [[1 2 3]

[4 5 6]

[7 8 9]]

E-value : [ 1.61168440e+01 -1.11684397e+00 -4.22209278e-16]

E-vector : [[-0.23197069 -0.78583024 0.40824829]

[-0.52532209 -0.08675134 -0.81649658] [-0.81867350.61232756 0.40824829]]

Result/Observation

Successfully completed the data science program and output is obtained.

**Mark:**

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**SENTIMENTAL ANALYSIS :**

**Input:**

l1=['good','fine','nice','happy','positive','well']

l2=['bad','sad','tired','frustrated','not']

str1=input('Enter your response')

flag=0

ncount=0

pcount=0

t=str1.split()

for i in range(len(t)):

for j in range(len(l1)):

if t[i]==l1[j]:

flag=1

pcount+=1

for k in range(len(l2)):

if t[i]==l2[k]:

flag=1

ncount+=1

if flag==0:

print('you are in another mood')

elif ncount%2==0:

print('positive response')

else:

print('negative response')

**Output:**

Enter your response: bad

negative response

"c:/Users/sree/AppData/Local/Programs/Python/Python38/ML/matrix\_analysis":

Enter your response: good

positive response

Result/Observation

Successfully completed the data science program and output is obtained.

**Mark:**

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**PROGRAM NO. 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab: Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title:** KNN classification

**Objectives:** Program to implement a K-NN classification using any dataset.

**Dataset**: Iris.csv

**Input:**

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_iris

irisdata=load\_iris()

print(irisdata.data)

x=irisdata.data

y=irisdata.target

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,random\_state=42)

knn=KNeighborsClassifier(n\_neighbors=3)

knn.fit(x\_train,y\_train)

print(knn.predict(x\_test))

print(knn.score(x\_test,y\_test))

**Output:**

[[5.1 3.5 1.4 0.2]

[4.4 3.2 1.3 0.2]

[5. 3.5 1.6 0.6]

[5.1 3.8 1.9 0.4]

[4.8 3. 1.4 0.3]

[5.1 3.8 1.6 0.2]

[4.6 3.2 1.4 0.2]

[5.3 3.7 1.5 0.2]

[5. 3.3 1.4 0.2]

[7. 3.2 4.7 1.4]

[6.4 3.2 4.5 1.5]

[6.9 3.1 4.9 1.5]

[5.5 2.3 4. 1.3]

[6.5 2.8 4.6 1.5]

[5.7 2.8 4.5 1.3]

[6.3 3.3 4.7 1.6]

[4.9 2.4 3.3 1. ]

[6.6 2.9 4.6 1.3]

[5.2 2.7 3.9 1.4]

[5. 2. 3.5 1. ]

[5.9 3. 4.2 1.5]

[6. 2.2 4. 1. ]

[6.1 2.9 4.7 1.4]

[5.4 3.4 1.7 0.2]

[5.1 3.7 1.5 0.4]

[4.6 3.6 1. 0.2]

[5.1 3.3 1.7 0.5]

[4.8 3.4 1.9 0.2]

[5. 3. 1.6 0.2]

[5. 3.4 1.6 0.4]

[5.2 3.5 1.5 0.2]

[5.2 3.4 1.4 0.2]

[4.7 3.2 1.6 0.2]

[4.8 3.1 1.6 0.2]

[5.4 3.4 1.5 0.4]

[5.2 4.1 1.5 0.1]

[5.5 4.2 1.4 0.2]

[4.9 3.1 1.5 0.2]

[5. 3.2 1.2 0.2]

[5.5 3.5 1.3 0.2]

[4.9 3.6 1.4 0.1]

[4.4 3.1.3 2.0.2]

[5.1 3.4 1.5 0.2]

[5. 3.5 1.3 0.3]

[4.5 2.3 1.3 0.3]

[4.9 3. 1.4 0.2]

[4.7 3.2 1.3 0.2]

[4.6 3.1 1.5 0.2]

[5. 3.6 1.4 0.2]

[5.4 3.9 1.7 0.4]

[4.6 3.4 1.4 0.3]

[5. 3.4 1.5 0.2]

[4.4 2.9 1.4 0.2]

[4.9 3.1 1.5 0.1]

[5.4 3.7 1.5 0.2]

[4.8 3.4 1.6 0.2]

[4.8 3. 1.4 0.1]

[4.3 3. 1.1 0.1]

[5.8 4. 1.2 0.2]

[5.7 4.4 1.5 0.4]

[5.4 3.9 1.3 0.4]

[5.1 3.5 1.4 0.3]

[5.7 3.8 1.7 0.3]

[5.1 3.8 1.5 0.3]

[6. 3.4 4.5 1.6]

[6.7 3.1 4.7 1.5]

[6.3 2.3 4.4 1.3]

[5.6 3. 4.1 1.3]

[5.5 2.5 4. 1.3]

[5.5 2.6 4.4 1.2]

[6.1 3. 4.6 1.4]

[5.8 2.6 4. 1.2]

[5. 2.3 3.3 1. ]

[5.6 2.7 4.2 1.3]

[5.7 3. 4.2 1.2]

[5.7 2.9 4.2 1.3]

[6.2 2.9 4.3 1.3]

[5.1 2.5 3. 1.1]

[5.7 2.8 4.1 1.3]

[6.3 3.3 6. 2.5]

[5.8 2.7 5.1 1.9]

[7.1 3. 5.9 2.1]

[6.3 2.9 5.6 1.8]

[6.5 3. 5.8 2.2]

[7.6 3. 6.6 2.1]

[4.9 2.5 4.5 1.7]

[5.6 2.9 3.6 1.3]

[7.3 2.9 6.3 1.8]

[6.7 2.5 5.8 1.8]

[7.2 3.6 6.1 2.5]

[6.5 3.2 5.1 2. ]

[6.4 2.7 5.3 1.9]

[6.8 3. 5.5 2.1]

[5.7 2.5 5. 2. ]

[5.8 2.8 5.1 2.4]

[6.4 3.2 5.3 2.3]

[6.5 3. 5.5 1.8]

[7.7 3.8 6.7 2.2]

[7.7 2.6 6.9 2.3]

[6. 2.2 5. 1.5]

[6.9 3.2 5.7 2.3]

[5.6 2.8 4.9 2. ]

[7.7 2.8 6.7 2. ]

[6.3 2.7 4.9 1.8]

[6.7 3.3 5.7 2.1]

[7.2 3.2 6. 1.8]

[6.2 2.8 4.8 1.8]

[6.1 3. 4.9 1.8]

[6.4 2.8 5.6 2.1]

[6.7 3.1 4.4 1.4]

[5.6 3. 4.5 1.5]

[5.8 2.7 4.1 1. ]

[6.2 2.2 4.5 1.5]

[5.6 2.5 3.9 1.1]

[5.9 3.2 4.8 1.8]

[6.1 2.8 4. 1.3]

[6.3 2.5 4.9 1.5]

[6.1 2.8 4.7 1.2]

[6.4 2.9 4.3 1.3]

[6.6 3. 4.4 1.4]

[6.8 2.8 4.8 1.4]

[6.7 3. 5. 1.7]

[6. 2.9 4.5 1.5]

[5.7 2.6 3.5 1. ]

[5.5 2.4 3.8 1.1]

[5.5 2.4 3.7 1. ]

[5.8 2.7 3.9 1.2]

[6. 2.7 5.1 1.6]

[5.4 3. 4.5 1.5]

[7.2 3. 5.8 1.6]

[7.4 2.8 6.1 1.9]

[7.9 3.8 6.4 2. ]

[6.4 2.8 5.6 2.2]

[6.3 2.8 5.1 1.5]

[6.1 2.6 5.6 1.4]

[7.7 3. 6.1 2.3]

[6.3 3.4 5.6 2.4]

[6.4 3.1 5.5 1.8]

[6. 3. 4.8 1.8]

[6.9 3.1 5.4 2.1]

[6.7 3.1 5.6 2.4]

[6.9 3.1 5.1 2.3]

[5.8 2.7 5.1 1.9]

[6.8 3.2 5.9 2.3]

[6.7 3.3 5.7 2.5]

[6.7 3. 5.2 2.3]

[6.3 2.5 5. 1.9]

[6.5 3. 5.2 2. ]

[6.2 3.4 5.4 2.3]

[5.9 3. 5.1 1.8]]

[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1 0]

1.0

**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**PROGRAM NO:3**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab: Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title:** Naive Bayes Algorithm

**Objectives:** Program to implement Naive Bayes classification using any dataset

**Input:**

weather=['sunny','sunny','overcast','rainy','rainy','rainy','overcast','sunny','sunny','rainy','sunny','overcast','overcast','rainy']

temp=['hot','hot','hot','mild','cool','cool','cool','mild','cool','mild','mild','mild','hot','mild']

play=['no','no','yes','yes','yes','no','yes','no','yes','yes','yes','yes','yes','no']

from cProfile import label

from cgi import MiniFieldStorage

from heapq import merge

from sklearn import preprocessing

le=preprocessing.LabelEncoder()

weather\_encoded=le.fit\_transform(weather)

print("Weather",weather\_encoded)

temp\_encoded=le.fit\_transform(temp)

label=le.fit\_transform(play)

print("Temp",temp\_encoded)

print("Play",label)

features=list(zip(weather\_encoded,temp\_encoded))

print(features)

from sklearn.naive\_bayes import GaussianNB

model=GaussianNB()

model.fit(features,label)

predicted=model.predict([[0,2]])

print("Predicted value",predicted)

**Output:**

Weather [2 2 0 1 1 1 0 2 2 1 2 0 0 1]

Temp [1 1 1 2 0 0 0 2 0 2 2 2 1 2]

Play [0 0 1 1 1 0 1 0 1 1 1 1 1 0]

[(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2), (2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]

Predicted value [1]

**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**PROGRAM NO:4**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab: Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title:** Regression Technique

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

LINEAR REGRESSION:

**Input:**

Input: import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets,linear\_model

from sklearn.metrics import mean\_squared\_error,r2\_score

df=datasets.load\_diabetes()

df=['feature\_names']

diabetes\_x,diabetes\_y=datasets.load\_diabetes(return\_X\_y=True)

diabetes\_x.shape

diabetes\_y.shape

diabetes\_x=diabetes\_x[:,np.newaxis,2]

diabetes\_x.shape

diabetes\_x\_train=diabetes\_x[:-20]

diabetes\_x\_test=diabetes\_x[-20:]

diabetes\_y\_train=diabetes\_y[:-20]

diabetes\_y\_test=diabetes\_x[-20:]

regr=linear\_model.LinearRegression()

regr.fit(diabetes\_x\_train,diabetes\_y\_train)

diabetes\_y\_pred=regr.predict(diabetes\_x\_test)

print("coefficients:\n",regr.coef\_)

print("Mean squared error:%.2f"%mean\_squared\_error(diabetes\_y\_test,diabetes\_y\_pred))

print("coefficient ofc determination:%2f"%r2\_score(diabetes\_y\_test,diabetes\_y\_pred))

plt.scatter(diabetes\_x\_test,diabetes\_y\_test,color="black")

plt.plot(diabetes\_x\_test,diabetes\_y\_pred,color="blue",linewidth=3)

plt.xlabel("age")

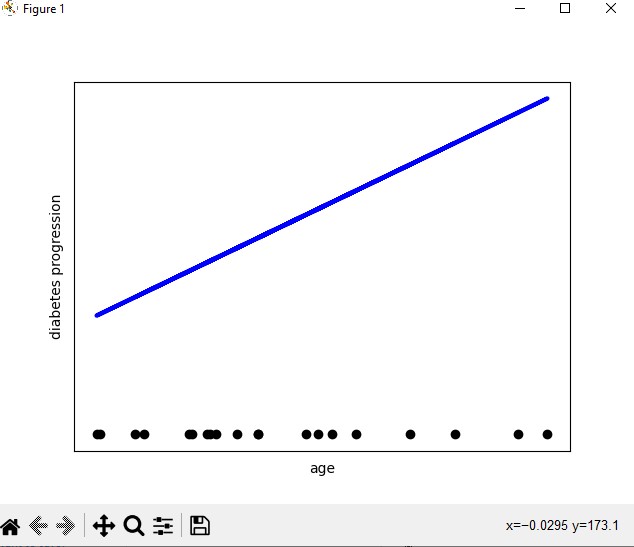
plt.ylabel("diabetes progression")

plt.xticks(())

plt.yticks(())

plt.show()

**Output:**



**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

MULTIPLE REGRESSION:

**Dataset**:Real Estate.csv

**Input:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

from sklearn import preprocessing

df = pd.read\_csv('Real-estate1.csv')

df.drop('No', inplace=True, axis=1)

print(df.head())

print(df.columns)

sns.scatterplot(x='X4 number of convenience stores',y='Y house price of unit area', data=df)

plt.show()

X = df.drop('Y house price of unit area', axis=1)

y = df['Y house price of unit area']

print(X)

print(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=101)

model = LinearRegression()

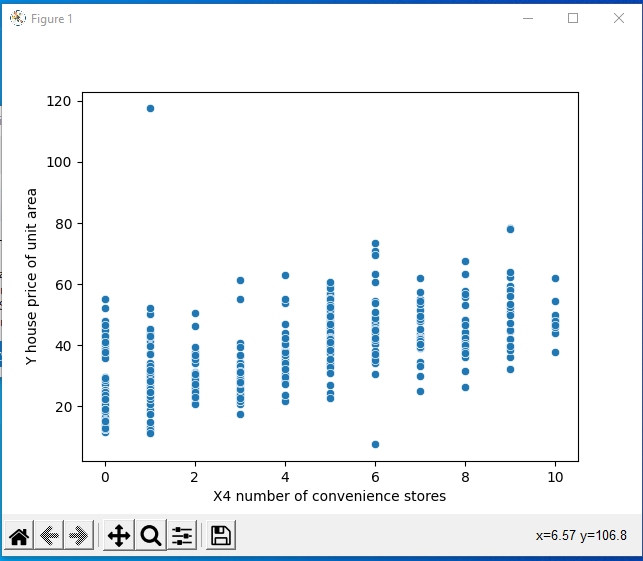
model.fit(X\_train, y\_train)

predictions = model.predict(X\_test)

print('mean\_squared\_error : ', mean\_squared\_error(y\_test, predictions))

print('mean\_absolute\_error : ', mean\_absolute\_error(y\_test, predictions))

**Output:**

****

**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**PROGRAM NO:5**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab:Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title:** Decision Tree

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

**Input:**

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn import tree

iris=load\_iris()

X,y=iris.data,iris.target

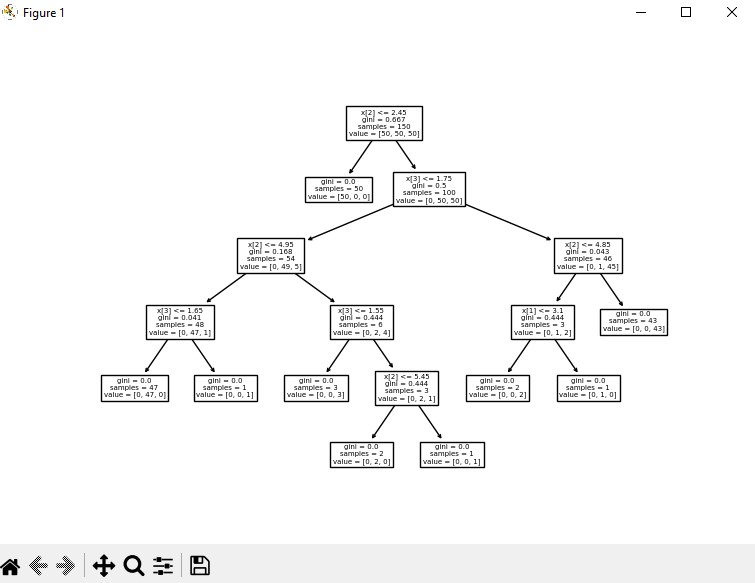
clf=tree.DecisionTreeClassifier()

clf=clf.fit(X,y)

print(tree.plot\_tree(clf))

plt.show()

**Output:**

****

**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**PROGRAM NO:6**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab:Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title:** Linear Regression

**Objectives:** Program to implement linear regression for stock market prediction using stock price dataset of any stock.

**Dataset**: price.xlsx

**Input:**

from sklearn.linear\_model import LinearRegression

from email import header

import matplotlib.pyplot as pit

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn import datasets, linear\_model

from sklearn.metrics import mean\_squared\_error, r2\_score

import pandas as pd

data = pd.read\_excel('price.xlsx',index\_col=None,na\_values=['NA'],usecols="B,E")

df\_binary = pd.DataFrame(data)

x=np.array(df\_binary['OPEN']).reshape(-1,1)

y=np.array(df\_binary['CLOSE']).reshape(-1,1)

df\_binary.dropna(inplace=True)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=0)

reg=LinearRegression().fit(x\_train,y\_train.reshape((-1,1)))

newvalue=float(input('enter todays opening '))

y\_pred = reg.intercept\_ + reg.coef\_ \*newvalue

print(y\_pred)

**Output:**

PS G:\ML exam> & C:/Users/sree/AppData/Local/Programs/Python/Python311/python.exe "g:/ML exam/Dataset/nahar.py"

enter todays opening : 100

[[99.37377025]]

**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**PROGRAM NO:7**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab:Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title:** NLTK

**Objectives:** Program on Natural Language Toolkit

**Input:**

import nltk

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')

from nltk.tokenize import sent\_tokenize,word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from nltk.stem.wordnet import WordNetLemmatizer

text="""Hello Mr. Smith, how are you doing today? The weather is great, and city is awesome.

The sky is pinkish-blue. You shouldn't eat cardboard"""

tokenized\_word=word\_tokenize(text)

print(tokenized\_word)

stop\_words=set(stopwords.words("english"))

print(stop\_words)

filtered\_word=[]

for w in tokenized\_word:

if w not in stop\_words:

filtered\_word.append(w)

print("Tokenized Sentence:",tokenized\_word)

print("Filterd Sentence:",filtered\_word)

ps = PorterStemmer()

stemmed\_words=[]

for w in filtered\_word:

stemmed\_words.append(ps.stem(w))

print("Filtered Sentence:",filtered\_word)

print("Stemmed Sentence:",stemmed\_words)

lem = WordNetLemmatizer()

stem = PorterStemmer()

word = "flying"

print("Lemmatized Word:",lem.lemmatize(word,"v"))

print("Stemmed Word:",stem.stem(word))

**Output:**

PS C:\Users\Technosoft> & C:/Users/Technosoft/AppData/Local/Programs/Python/Python311/python.exe d:/MCA/DS/lab/nlp.py

[nltk\_data] Downloading package stopwords to

[nltk\_data] C:\Users\Technosoft\AppData\Roaming\nltk\_data...

[nltk\_data] Package stopwords is already up-to-date!

[nltk\_data] Downloading package punkt to

[nltk\_data] C:\Users\Technosoft\AppData\Roaming\nltk\_data...

[nltk\_data] Package punkt is already up-to-date!

[nltk\_data] Downloading package averaged\_perceptron\_tagger to

[nltk\_data] C:\Users\Technosoft\AppData\Roaming\nltk\_data...

[nltk\_data] Package averaged\_perceptron\_tagger is already up-to-

[nltk\_data] date!

[('Hello', 'NNP'), ('.', '.')]

[('MCA', 'NNP'), ('S3', 'NNP'), ('fantastic', 'JJ'), ('.', '.')]

[('We', 'PRP'), ('learn', 'VBP'), ('many', 'JJ'), ('new', 'JJ'), ('concepts', 'NNS'), ('implement', 'JJ'), ('practical', 'JJ'), ('exams', 'NN'), ('.', '.')][('1st', 'CD'), ('data', 'NNS'), ('science', 'NN'), ('new', 'JJ'), ('paper', 'NN'), ('.', '.')]

**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**PROGRAM NO:8**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab:Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title:** Support Vector Machine

**Objectives:** Program to implement text classification using support vector machine

**Input:**

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

from sklearn import svm

cancer=datasets.load\_breast\_cancer()

x\_train,x\_test,y\_train,y\_test=train\_test\_split(cancer.data,cancer.target,test\_size=0.3,random\_state=109)

clf=svm.SVC(kernel='linear')

clf.fit(x\_train,y\_train)

y\_pred=clf.predict(x\_test)

print("Actual values",y\_test)

print("Predicted values",y\_pred)

print("Accuracy:",metrics.accuracy\_score(y\_test,y\_pred))

print("Precision:",metrics.precision\_score(y\_test,y\_pred))

print("Recall:",metrics.recall\_score(y\_test,y\_pred))

**Output:**

Actual values:

[1 1 0 0 1 0 1 1 1 0 0 0 1 0 1 1 0 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 1

0 1 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0 1 1 1 1 0 1 1 1

0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 1 0 1 1

0 1 1 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 0

0 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1]

Predicted values :

[1 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 1

0 1 1 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0 1 1 1 1 0 1 1 1

0 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0 1 1 0 1 0 1 1

0 1 1 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 1 0 1 1 1 0 1 1 0 0 1 0

0 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1]

Accuracy: 0.9649122807017544

Precision: 0.9811320754716981

Recall: 0.9629629629629629

**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**PROGRAM NO:9**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab:Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title:** Web Crawler

**Objectives:** Program to implement a simple web crawler and scrapping web pages

**Input:**

import requests

from bs4 import BeautifulSoup

from xlwt import \*

Workbook= Workbook(encoding='utf-8')

table=Workbook.add\_sheet('data')

table.write(0,0,'Number')

table.write(0,1,'Title')

table.write(0,2,'Company')

table.write(0,3,'Location')

line=1

URL = "https://realpython.github.io/fake-jobs/"

page = requests.get(URL)

soup = BeautifulSoup(page.content, "html.parser")

results = soup.find(id="ResultsContainer")

print(results.prettify())

num=0

job\_elements = results.find\_all("div", class\_="card-content")

for job\_element in job\_elements:

print(job\_element, end="\n"\*2)

for job\_element in job\_elements:

title\_element = job\_element.find("h2", class\_="title")

company\_element = job\_element.find("h3", class\_="company")

location\_element = job\_element.find("p", class\_="location")

print(title\_element.text.strip())

print(company\_element.text.strip())

print(location\_element.text.strip())

num+=1

print()

table.write(line,0,num)

table.write(line,1,title\_element.text.strip())

table.write(line,2,company\_element.text.strip())

table.write(line,3,location\_element.text.strip())

**Output:**

Textile designer

English as a second language teacher

Parker, Murphy and Brooks

Mitchellburgh, AE

Surgeon

Cruz-Brown

West Jessicabury, AA

Equities trader

Macdonald-Ferguson

Maloneshire, AE

Newspaper journalist

Williams, Peterson and Rojas

Johnsonton, AA

Materials engineer

Smith and Sons

South Davidtown, AP

Python Programmer (Entry-Level)

Moss, Duncan and Allen

Port Sara, AE

Gomez-Carroll

Marktown, AA

Senior Python Developer

Meyers-Johnson

Port Jonathan, AE

Television floor manager

Hughes-Williams

Osbornetown, AE

Waste management officer

Jones, Williams and Villa

Scotttown, AP

Software Engineer (Python)

Garcia PLC

Ericberg, AE

Interpreter

Gregory and Sons

Ramireztown, AE

Architect

Clark, Garcia and Sosa

Figueroaview, AA

Meteorologist

Bush PLC

Kelseystad, AA

Audiological scientist

Salazar-Meyers

Williamsburgh, AE

Payne, Roberts and Davis

Stewartbury, AA

Energy engineer

Vasquez-Davidson

Christopherville, AA

Legal executive

Jackson, Chambers and

Port Ericaburgh, AA

Fitness centre manager

Savage-Bradley

East Seanview, AP

Product manager

Ramirez Inc

North Jamieview, AP

Medical technical officer

Rogers-Yates

Davidville, AP

Physiological scientist

Kramer-Klein

South Christopher, AE

Radiographer, diagnostic

Holder LLC

Jacobshire, AP

Database administrator

Yates-Ferguson

Port Susan, AE

**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**

**PROGRAM NO:10**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name: P J Sreedeep** | **Roll No: 20** | **Name of Lab:Data Science Lab** | **Period:** |
| **Class: S3, MCA** | **Date:** | **Nature of Lab Work: Practical** | **Batch:**  **2021-2023** |

**Title:** k-means clustering technique

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

**Dataset:** Mall\_Customer.csv

**Input:**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import sklearn

dataset = pd.read\_csv('Mall\_Customers.csv')

X = dataset.iloc[:, [3, 4]].values

from sklearn.cluster import KMeans

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)

kmeans.fit(X)

wcss.append(kmeans.inertia\_)

plt.plot(range(1, 11), wcss)

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()

kmeans = KMeans(n\_clusters = 5, init = "k-means++", random\_state = 42)

y\_kmeans = kmeans.fit\_predict(X)

print(y\_kmeans)

plt.scatter(X[y\_kmeans == 0, 0], X[y\_kmeans == 0, 1], s = 50, c = 'red', label = 'Cluster1')

plt.scatter(X[y\_kmeans == 1, 0], X[y\_kmeans == 1, 1], s = 50, c = 'blue', label = 'Cluster2')

plt.scatter(X[y\_kmeans == 2, 0], X[y\_kmeans == 2, 1], s = 50, c = 'green', label = 'Cluster3')

plt.scatter(X[y\_kmeans == 3, 0], X[y\_kmeans == 3, 1], s = 50, c = 'violet', label = 'Cluster4')

plt.scatter(X[y\_kmeans == 4, 0], X[y\_kmeans == 4, 1], s = 50, c = 'yellow', label = 'Cluster5')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s = 100,marker='x', c = 'red', label = 'Centroids')

plt.xlabel('Annual Income ')

plt.ylabel('Spending Score (1-100)')

plt.legend()

plt.show()

**Output:**

PS G:\ML exam> & C:/Users/sree/AppData/Local/Programs/Python/Python311/python.exe "g:/ML exam/kmeans.py"

[2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2

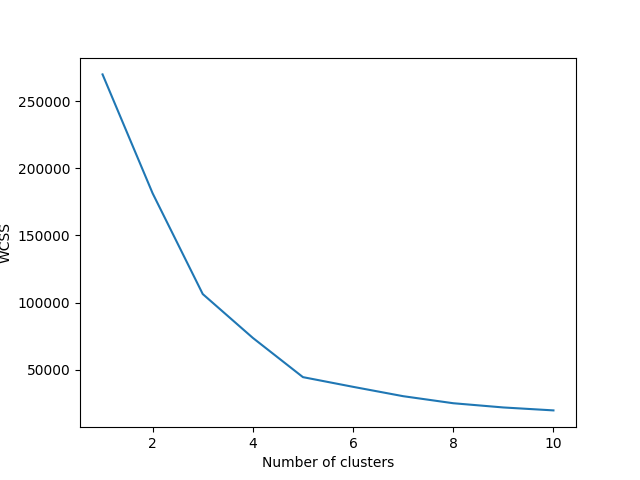
3 2 3 2 3 2 0 2 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

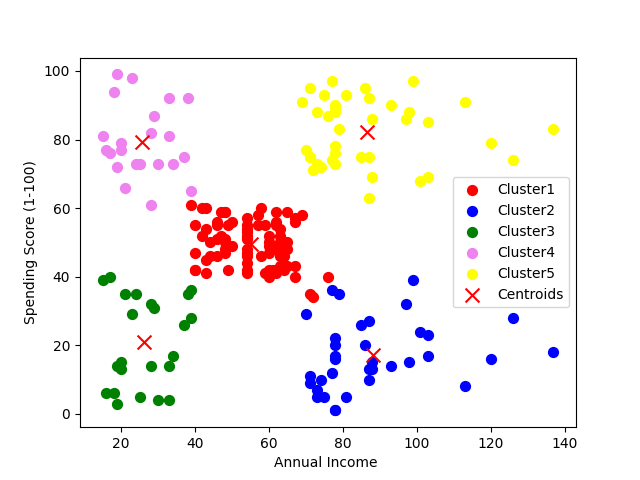
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 4 1 4 0 4 1 4 1 4 0 4 1 4 1 4 1 4 1 4 0 4 1 4 1 4

1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1 4 1

4 1 4 1 4 1 4 1 4 1 4 1 4 1 4]





**Result/Observation**

Successfully completed the data science program and output is obtained.

|  |  |  |
| --- | --- | --- |
| **Viva(5)** | **Performance(5)** | **Total(10)** |
|  |  |  |

**Assessor:**