DEPARTMENT OF COMPUTER APPLICATIONS

MUTHOOT INSTITUTE OF TECHNOLOGY & SCIENCE

ERNAKULAM DISTRICT

MASTER OF COMPUTER APPLICATIONS

SEMESTER III

LAB RECORD

20MCA241 DATA SCIENCE LAB

Name: Gokul Raj C

Register Number: MUT24MCA-2029

Month: November **Year:** 2025

Muthoot
Institute of Technology & Science

CERTIFICATE

Certified that this is the Bonafide Record of Practical work done by ***Reg.No*** *MUT24MCA-2029* ***Name:*** *Gokul Raj C for the partial fulfillment of the requirements for the award of the degree of Master of Computer Applications during the year* ***2025****.*

Head of the Department   Faculty in Charge

Dr Saritha K

University Exam Reg. No: MUT24MCA-2029

Date of Examination: …………………………………

Internal Examiner   External Examiner

Muthoot
Institute of Technology & Science

Vision Of Institute

To be a centre of excellence for learning and research in engineering and technology, producing intellectually well-equipped and socially committed citizens possessing an ethical value system.

Mission Of Institute

- Offer well-balanced programme of instruction, practical exercise and opportunities in technology.
- Foster innovation and ideation of technological solutions on sustainable basis.
- Nurture a value system in students and engender in them a spirit of inquiry.

INDEX

<table>
<tr><td colspan="3"><b>CO 3: Use different packages and frameworks to implement text classification using SVM and clustering using k-means</b></td></tr>
<tr><td>14</td><td>Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.</td><td>10/09/2025</td></tr>
<tr><td>15</td><td>Program to implement k-means clustering technique using any standard dataset available in the public domain.</td><td>17/09/2025</td></tr>
<tr><td>16</td><td>Program to implement text classification using Support vector machine using any standard dataset available in the public domain and evaluate its performance.</td><td>07/10/2025</td></tr>
<tr><td colspan="3"><b>CO 4: Implement convolutional neural network algorithm using Keras framework.</b></td></tr>
<tr><td>17</td><td>Programs on convolutional neural network to classify images from any standard dataset in the public domain.</td><td>09/10/2025</td></tr>
<tr><td colspan="3"><b>CO 5: Implement programs for web data mining and natural language processing using NLTK</b></td></tr>
<tr><td>18</td><td>For a given text

1) perform word and sentence tokenization.
2) Remove the stop words from the given text
3) Perform Part of Speech tagging
4) create n-grams for different values of n=2,4.</td><td>17/10/2025</td></tr>
<tr><td>19</td><td>Implement a program to scrap the web page of any popular website – suggested python package is scrappy (ensure ethical conduct).</td><td>20/10/2025</td></tr>
</table>

COURSE OUTCOME 1

Use different Python packages to perform numerical calculations, statistical computations and data visualization

PROGRAM 1

AIM: Program to review the fundamentals of python.

1. To create a 3x3 matrix with values ranging from 2 to 10.

PROGRAM

```
# Write your code here
import numpy as np
x=np.arange(2,11)
print(x)
y=x.reshape(3,3)
print(y)
```

OUTPUT

```
[ 2  3  4  5  6  7  8  9 10]
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

2. Write a program to change the dimension of an array (say my_arr =[1, 2, 3, 4, 5, 6, 7, 8, 9]) into a 3 X 3 (3 rows and 3 columns) array and convert this NumPy array into a list.

PROGRAM

```
# Write your code here
my_arr=np.arange(1,10)
print(my_arr)
reshape=my_arr.reshape(3,3)
print(reshape)
print(type(reshape))
list=reshape.tolist()
print(list)
print(type(list))
```

OUTPUT

```
[1 2 3 4 5 6 7 8 9]
[[1 2 3]
 [4 5 6]
 [7 8 9]]
<class 'numpy.ndarray'>
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
<class 'list'>
```

3. Write a program to Print the square root of numbers in the list.

PROGRAM

```
list1=[4,16,9,1,25]
print(list1)
list2=np.sqrt(list1)
x=np.array((list2),dtype=int).tolist()
print(x)
```

OUTPUT

```
[4, 16, 9, 1, 25]
[2, 4, 3, 1, 5]
```

4. Write a program to create Null array of size 10 and update the sixth value to 11.

PROGRAM

```
import numpy as np
null_arr=np.zeros(10,dtype=int).tolist()
print(null_arr)
null_arr[5]=11
print(null_arr)
```

OUTPUT

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 11, 0, 0, 0, 0]
```

5. Write a program that populates a list by numbers that lies in the range of 0 - 49 and also divisible by 5. Use List Comprehension method.

PROGRAM

```
# Write a program to populate a number list divisible by 5 in a range 0 - 49
list=[x for x in range(0,49) if x%5==0]
print(list)
```

OUTPUT

```
[0, 5, 10, 15, 20, 25, 30, 35, 40, 45]
```

6. Write a program to convert a list of numeric values into a one-dimensional NumPy array.

PROGRAM

```
# Program to convert a list into one dimensional NumPy array
mylist=[1.23,23.32,300,16.37]
print(mylist)
print(type(mylist))
numpy_array=np.array(mylist)
print(numpy_array)
print(type(numpy_array))
```

OUTPUT

```
[1.23, 23.32, 300, 16.37]
<class 'list'>
[  1.23  23.32 300.    16.37]
<class 'numpy.ndarray'>
```

PROGRAM 2

AIM: Program to perform arithmetic operations on a 2D Matrices.

PROGRAM

```
import numpy as np
a1=np.array([[1,2,3],[4,5,6],[7,8,9]])
a2=np.array([[1,2,3],[4,5,6],[7,8,9]])
print("Matrix 1")
print(a1)
print("Matrix 2")
print(a2)
print("Sum")
print(np.add(a1,a2))
print("Difference")
print(np.subtract(a1,a2))
print("Product")
print(np.multiply(a1,a2))
print("Transpose of A")
print(a1.T)
print("Transpose of B")
print(a2.T)
print("Sum of diagonals of A")
print(sum(np.diag(a1)))
print("Sum of diagonals of b")
print(sum(np.diag(a2)))
```

OUTPUT

```
Matrix 1
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Matrix 2
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Sum
[[ 2  4  6]
 [ 8 10 12]
 [14 16 18]]
```

```
Difference
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Product
[[ 1  4  9]
 [16 25 36]
 [49 64 81]]
Transpose of A
[[1 4 7]
 [2 5 8]
 [3 6 9]]
Transpose of B
[[1 4 7]
 [2 5 8]
 [3 6 9]]
Sum of diagonals of A
15
Sum of diagonals of b
15
```

PROGRAM 3

AIM: Program to find the inverse, rank, determinant, eigen values of a given matrix.

PROGRAM

```
import numpy as np
a1 = np.array([[4, 2, 1],
               [3, 5, 2],
               [2, 1, 3]])

print("Matrix:")
print(a1)

# Determinant
det = np.linalg.det(a1)
print("\nDeterminant of Matrix:", round(det, 3))

# Rank
rank = np.linalg.matrix_rank(a1)
print("Rank of Matrix:", rank)

# Check if inverse exists
if det != 0:
    inv = np.linalg.inv(a1)
    print("\nInverse of Matrix:")
    print(np.round(inv, 3))
else:
    print("\nInverse does not exist (Determinant is 0).")

# Eigenvalues and Eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(a1)
print("\nEigenvalues of Matrix:")
print(np.round(eigenvalues, 3))

print("\nEigenvectors of Matrix:")
print(np.round(eigenvectors, 3))
```

OUTPUT

```
Matrix:
[[4 2 1]
 [3 5 2]
 [2 1 3]]

Determinant of Matrix: 35.0
Rank of Matrix: 3

Inverse of Matrix:
[[ 0.371 -0.143 -0.029]
 [-0.143  0.286 -0.143]
 [-0.2    0.     0.4  ]]

Eigenvalues of Matrix:
[7.858+0.j    2.071+0.407j 2.071-0.407j]

Eigenvectors of Matrix:
[[ 0.501+0.j    -0.253+0.356j -0.253-0.356j]
 [ 0.783+0.j    -0.218-0.395j -0.218+0.395j]
 [ 0.368+0.j     0.779+0.j     0.779-0.j   ]]
```

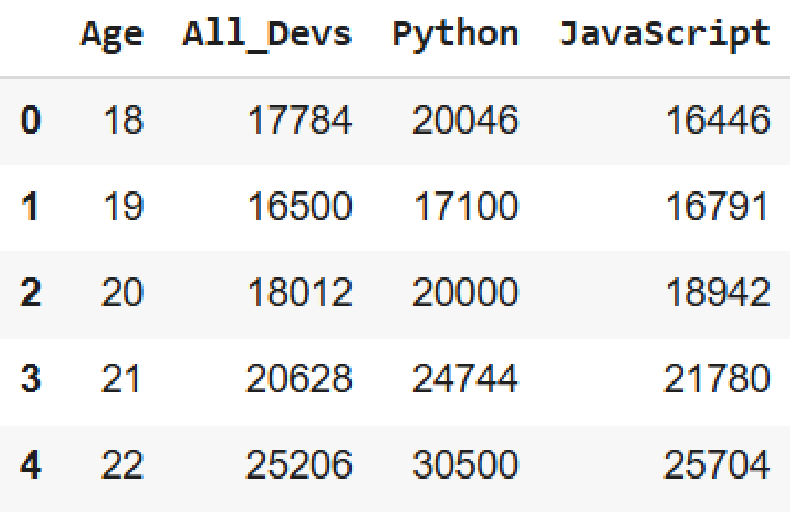
PROGRAM 4

AIM: Program to create customized line plot for comparing the Age-wise annual salary variations for Python developer with JavaScript developer from the dataset of the average annual salary of developers of various programming languages.

PROGRAM

```
# Step 1: Import necessary modules to create dataframe and line plots
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

# Step 2: Create a Dataframe and store it in a variable from the given dataset
data_f=pd.read_csv('https://raw.githubusercontent.com/CoreyMSchafer/code_snippets
/master/Python/Matplotlib/10-Subplots/data.csv')

# Print the first 5 rows in the DataFrame
data_f.head()
```

OUTPUT

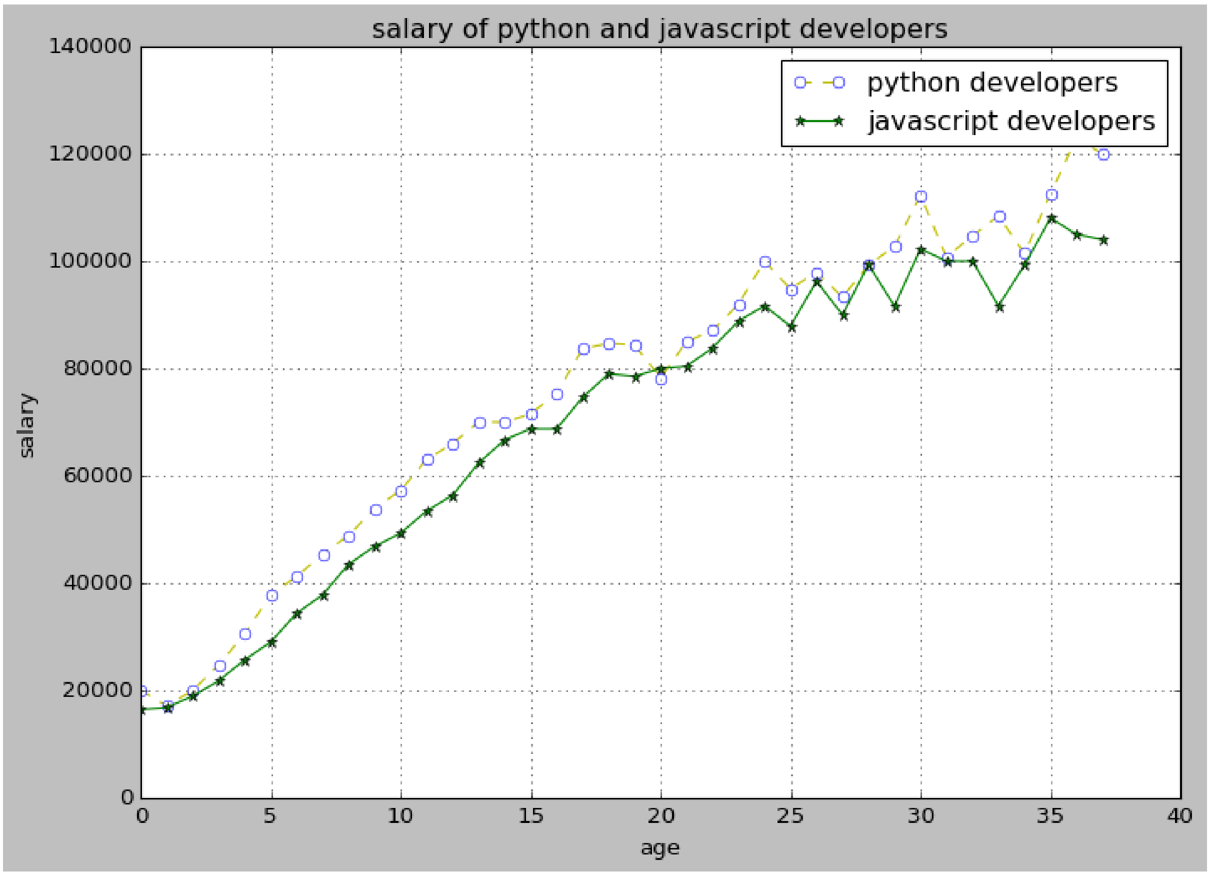


	Age	All_Devs	Python	JavaScript
0	18	17784	20046	16446
1	19	16500	17100	16791
2	20	18012	20000	18942
3	21	20628	24744	21780
4	22	25206	30500	25704

PROGRAM

```
# Step 3: Create a customised line plot for comparing the Age-wise annual salary
variations for Python developer with JavaScript developer.  Use the 'seaborn-dark'
style

plt.style.use('classic')
plt.figure(figsize=(10,7))
plt.title('salary of python and javascript developers')
plt.plot(data_f['Python'],'y--o',label='python developers',mfc='white',mec='blue')
plt.plot(data_f['JavaScript'],'g-*',label='javascript developers')
plt.xlabel('age')
plt.ylabel('salary')
plt.legend()
plt.grid()
plt.show()
```

OUTPUT



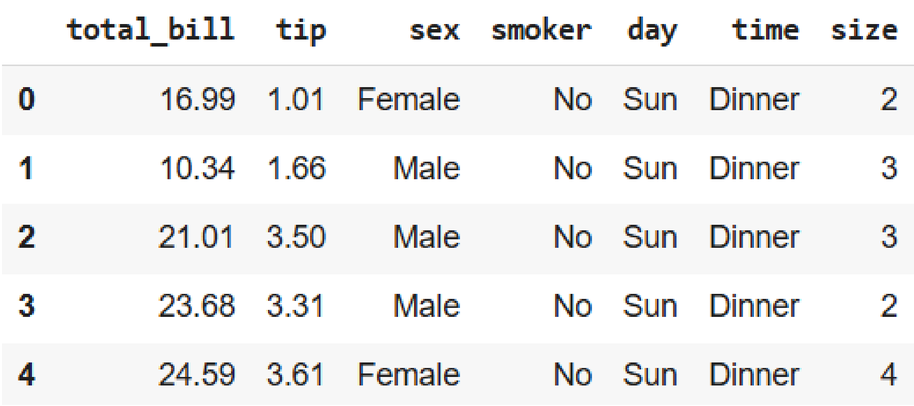
PROGRAM 5

AIM: Program to create a gender wise count plot by using the values in the sex column dataset of tips taken on the total bill amount in restaurants.

PROGRAM

```
# Create a pandas DataFrame
tip_f=pd.read_csv('https://raw.githubusercontent.com/jissgithub123/tips/main/
tips.csv)
tip_f.head()
```

OUTPUT

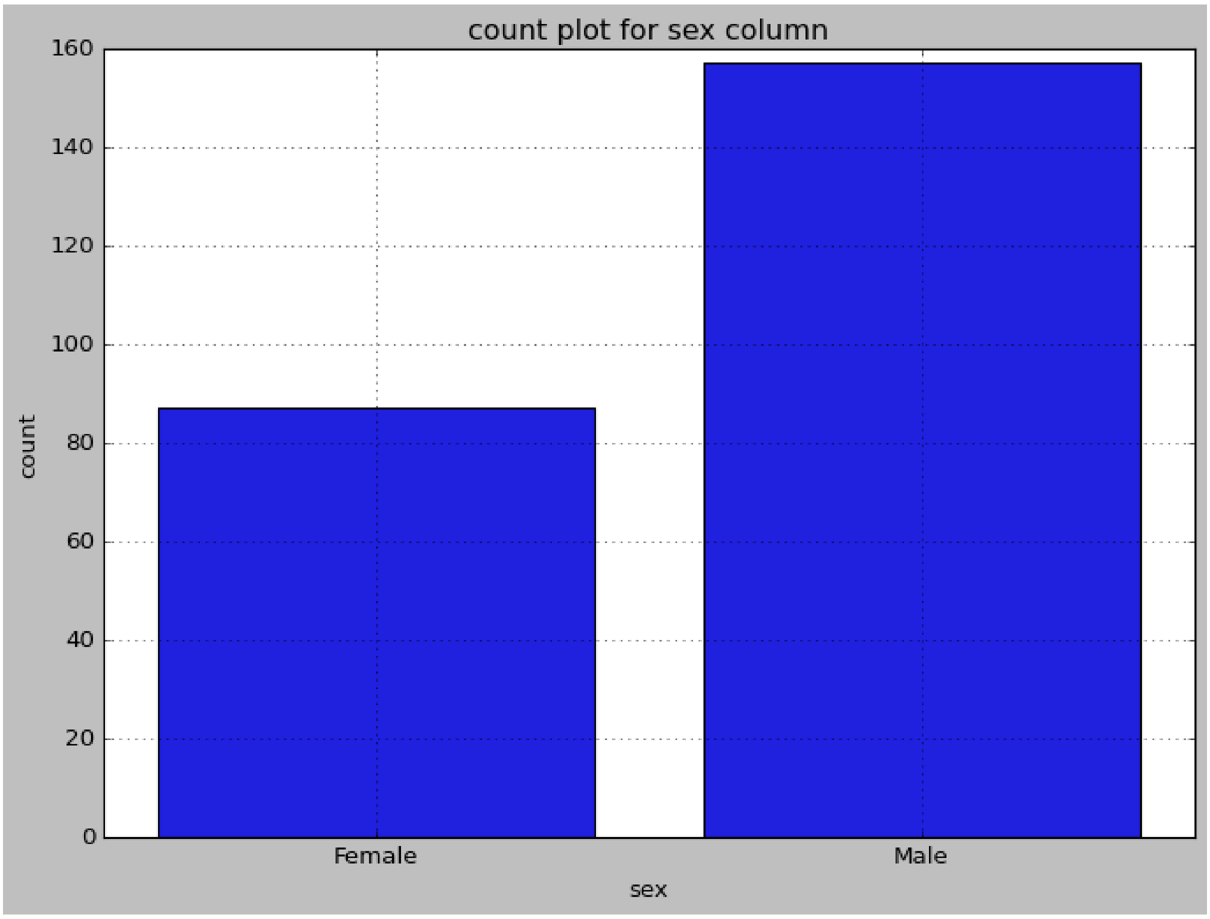


	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

PROGRAM

```
# Gender wise count plot for the 'sex' values in the 'tip_df' DataFrame on the x-axis.
plt.style.use('classic')
plt.figure(figsize=(10,7))
plt.title('count plot for sex column')
#sb.countplot(x='sex',data=tip_f)
sb.countplot(x=tip_f['sex'])
plt.grid()
plt.show()
```

OUTPUT


count plot for sex column
count
0
20
40
60
80
100
120
140
160
Female
Male
sex

PROGRAM 6

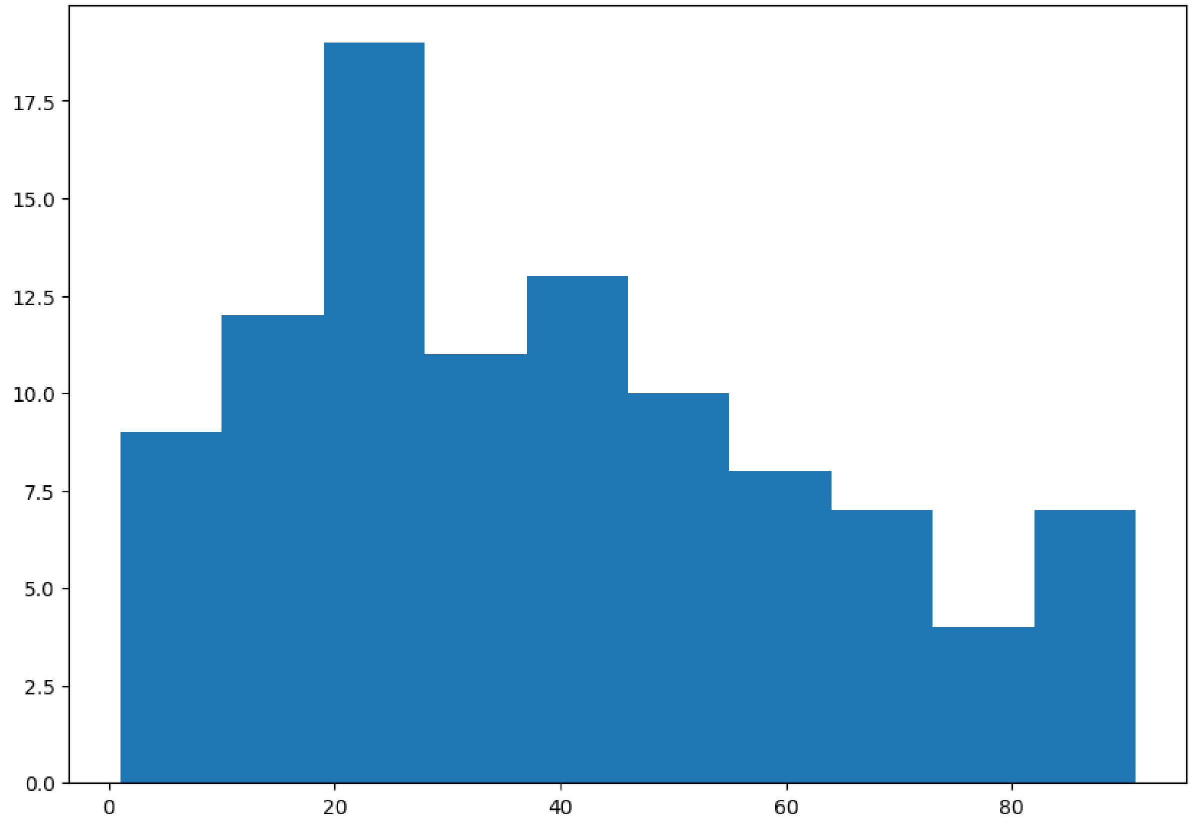
AIM: Program to create a histogram of a list of random age of 100 individuals in a range between 1 and 91.

```
age_list = [1,1,2,3,3,5,7,8,9,10,
    10,11,11,13,13,15,16,17,18,18,
    18,19,20,21,21,23,24,24,25,25,
    25,25,26,26,26,27,27,27,27,27,
    29,30,30,31,33,34,34,34,35,36,
    36,37,37,38,38,39,40,41,41,42,
    43,44,45,45,46,47,48,48,49,50,
    51,52,53,54,55,55,56,57,58,60,
    61,63,64,65,66,68,70,71,72,74,
    75,77,81,83,84,87,89,90,90,91
    ]
```

PROGRAM

```
# Import the 'matplotlib.pyplot' module.
import matplotlib.pyplot as plt
# Set the size of the plot using the 'figsize' attribute of the 'figure()' function.
plt.figure(figsize=(10,7))
age_list = [1,1,2,3,3,5,7,8,9,10,
    10,11,11,13,13,15,16,17,18,18,
    18,19,20,21,21,23,24,24,25,25,
    25,25,26,26,26,27,27,27,27,27,
    29,30,30,31,33,34,34,34,35,36,
    36,37,37,38,38,39,40,41,41,42,
    43,44,45,45,46,47,48,48,49,50,
    51,52,53,54,55,55,56,57,58,60,
    61,63,64,65,66,68,70,71,72,74,
    75,77,81,83,84,87,89,90,90,91
    ]
# Pass the 'age_list' list inside the 'hist()' function and set 'bins = 10'.
plt.hist(age_list,bins=10)
# Display the histogram using the 'show()' function of the 'matplotlib.pyplot' module.
plt.show()
```

OUTPUT



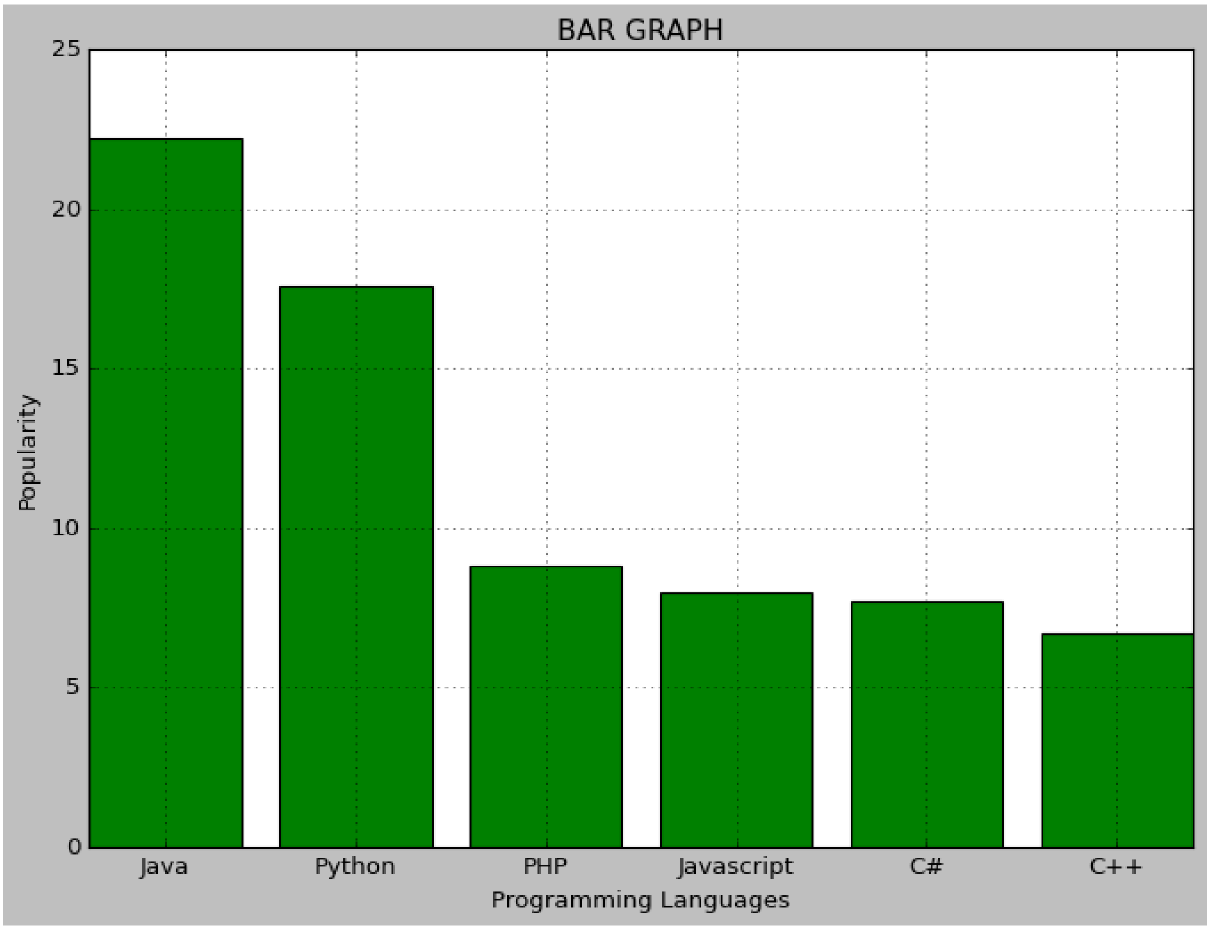
PROGRAM 7

AIM: Program to create a bar chart of the popularity of programming languages.

PROGRAM

```
plt.style.use('classic')
plt.figure(figsize=(10,7))
xval=['Java','Python','PHP','Javascript','C#','C++']
yval=[22.2,17.6,8.8,8,7.7,6.7]
plt.bar(xval,yval,color='g',edgecolor='black')
plt.xlabel('Programming Languages')
plt.ylabel('Popularity')
plt.title('BAR GRAPH')
plt.grid()
plt.show()
```

OUTPUT



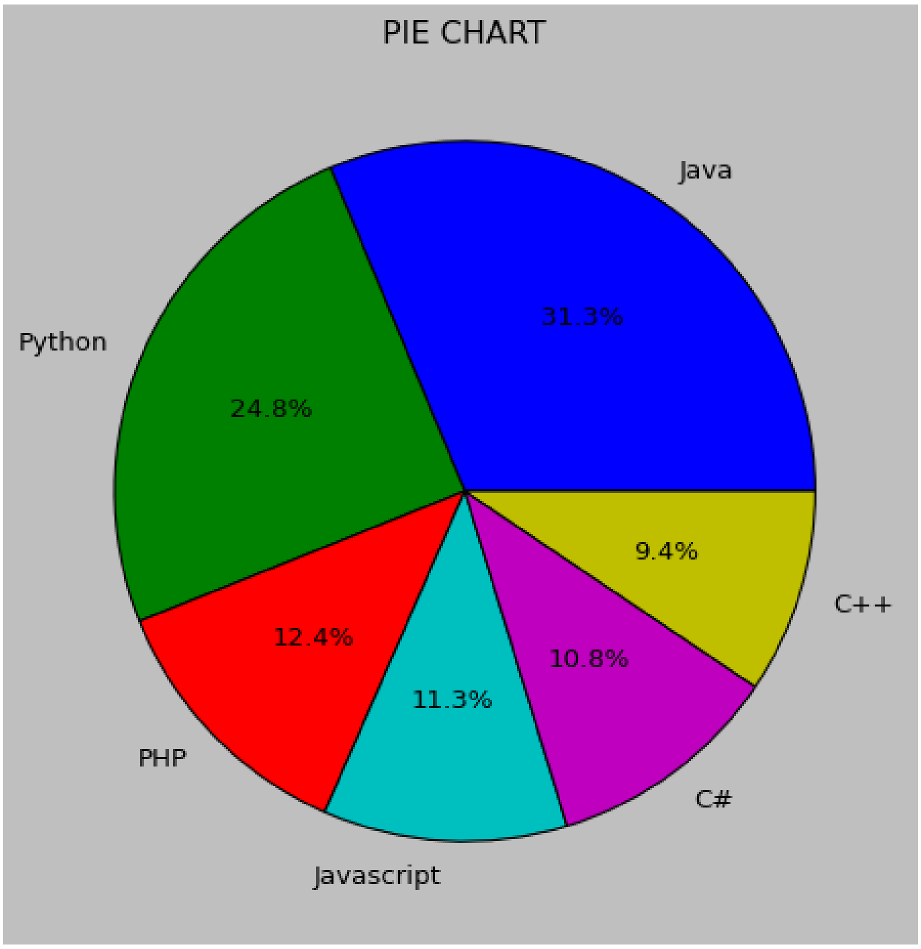
PROGRAM 8

AIM: Program to create a pie chart of the popularity of programming languages.

PROGRAM

```
plt.style.use('classic')
plt.figure(figsize=(10,7))
Lang=['Java','Python','PHP','Javascript','C#','C++']
pop=[22.2,17.6,8.8,8,7.7,6.7]
plt.pie(pop,labels=Lang,autopct='%1.1f%%')
plt.title('PIE CHART')
plt.show()
```

OUTPUT



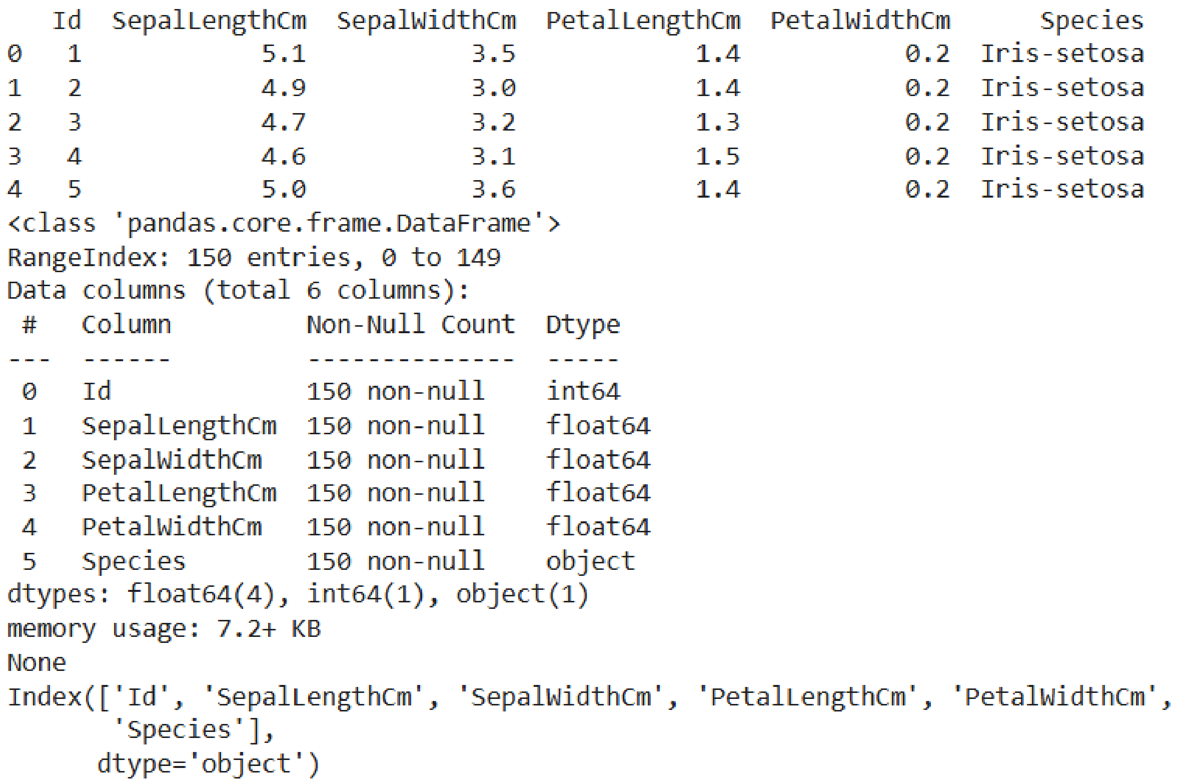
PROGRAM 9

AIM: Program to create a scatter plot between the SepalLength & SepalWidth columns. Differentiate between the data points of different classes.

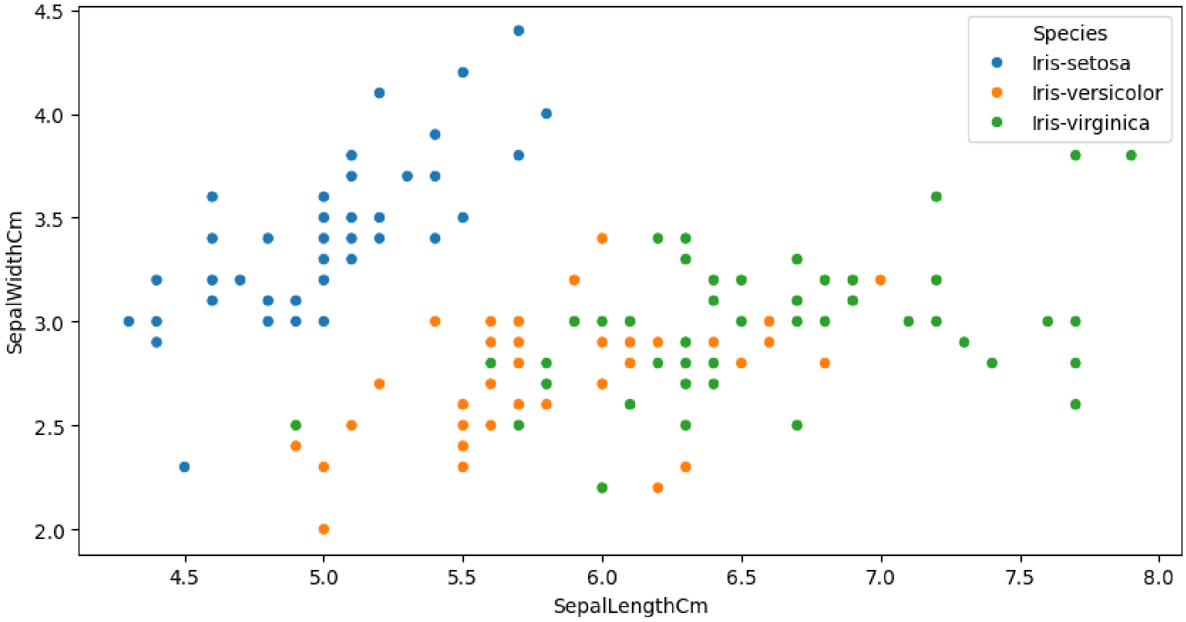
PROGRAM

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
iris_df=pd.read_csv(“https://raw.githubusercontent.com//gokul-raj-c/datasets/refs/heads/main/IRIS.csv”)
print(iris_df.head())
print(iris_df.info())
print(iris_df.columns)
plt.figure(figsize=(10,5))
sb.scatterplot(data=iris_df,x="SepalLengthCm",y="SepalWidthCm",hue="Species")
plt.show()
```

OUTPUT



```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```


Species
Iris-setosa
Iris-versicolor
Iris-virginica
SepalWidthCm
SepalLengthCm
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

COURSE OUTCOME 2

Use different packages and frameworks to implement regression and classification algorithms.

PROGRAM 10

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

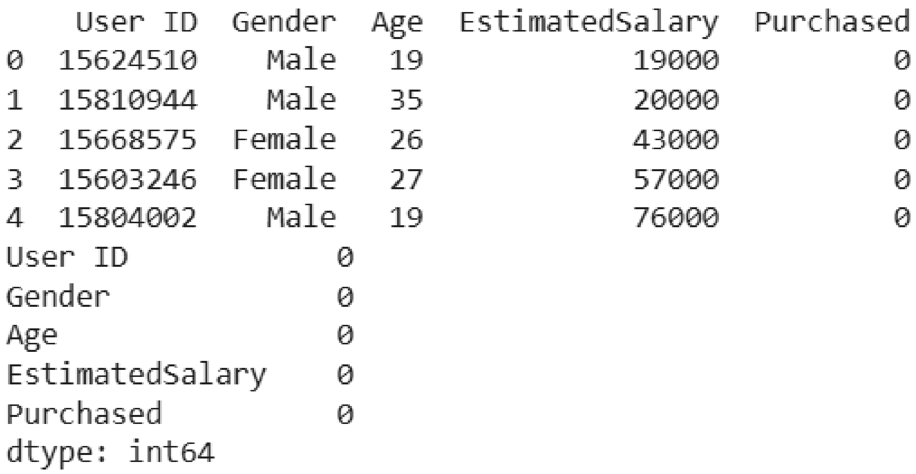
Activity 1: Import Modules and Read Data

Import the necessary Python packages.
Read the data from a CSV file to create a Pandas DataFrame.
Dataset--> social-network-ads.csv
Also, print the first five rows of the dataset. Check for null values and treat them accordingly.

PROGRAM

```
# Import all the necessary packages
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score

# Load the dataset
df=pd.read_csv("https://raw.githubusercontent.com/gokul-raj-
c/datasets/refs/heads/main/social-network-ads.csv")

# Print first five rows using head() function
print(df.head())

# Check if there are any null values. If any column has null values, treat them
accordingly
print(df.isnull().sum())
```

OUTPUT



```
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

Activity 2: Perform Train-Test Split

In this dataset, Purchased is the target variable and all other columns other than Purchased are feature variables.

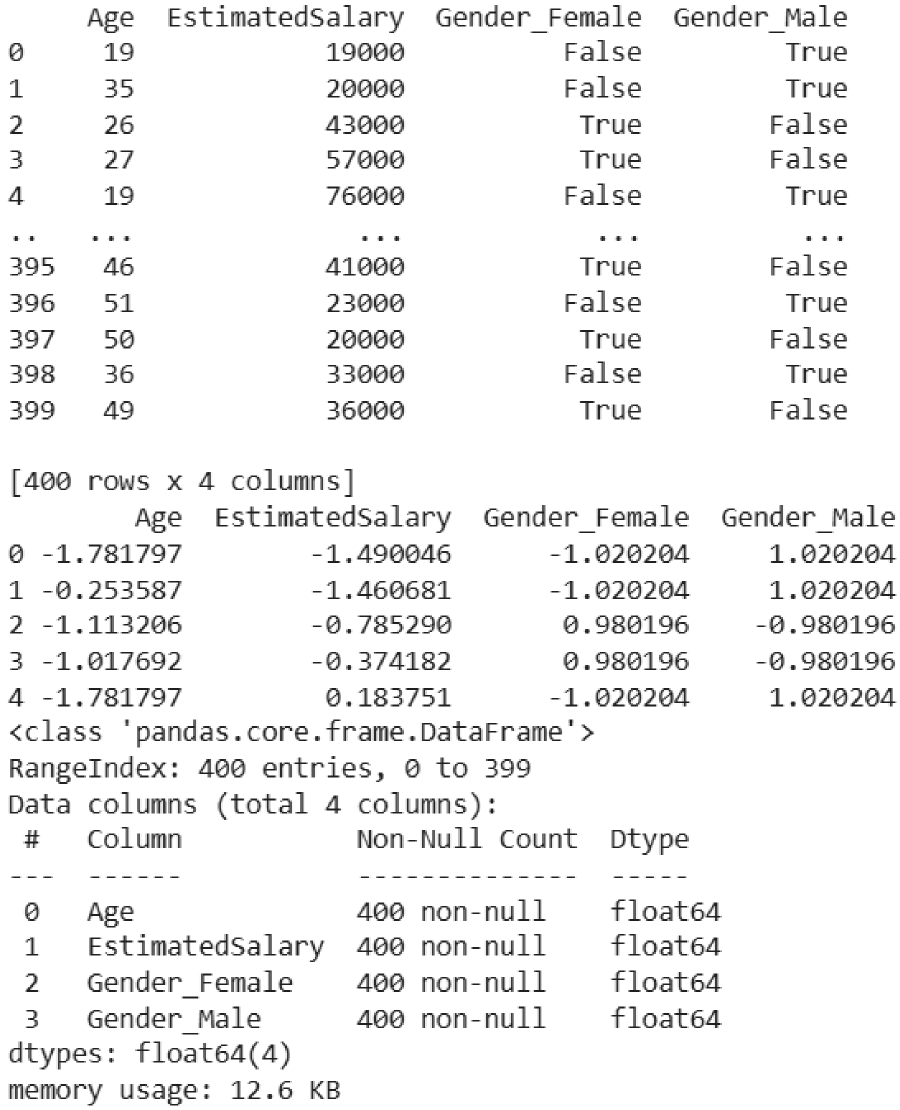
Create two separate DataFrames, one containing the feature variables and the other containing the target variable. Also, drop the User ID column from the features DataFrame as it is of no use.

Print the summary of features DataFrame to determine the data type of each feature variable.

PROGRAM

```
# Split the dataset into dependent and independent features
X = df.drop(['User ID', 'Purchased'], axis=1)
y=df['Purchased']
# Use 'get_dummies()' function to convert each categorical column in a DataFrame to numerical.
X=pd.get_dummies(X)
print(X)

ob=StandardScaler()
scaled=ob.fit_transform(X)
X_scaled=pd.DataFrame(scaled)
X_scaled.columns=X.columns
print(X_scaled.head())

# Use 'info()' function with the features DataFrame.
X_scaled.info()
```

OUTPUT

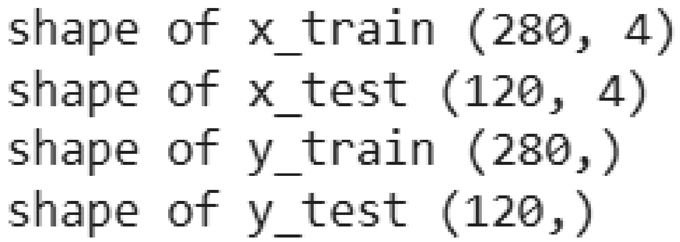


```
     Age  EstimatedSalary  Gender_Female  Gender_Male
0     19            19000          False         True
1     35            20000          False         True
2     26            43000           True        False
3     27            57000           True        False
4     19            76000          False         True
..   ...              ...            ...          ...
395   46            41000           True        False
396   51            23000          False         True
397   50            20000           True        False
398   36            33000          False         True
399   49            36000           True        False

[400 rows x 4 columns]
        Age  EstimatedSalary  Gender_Female  Gender_Male
0 -1.781797        -1.490046      -1.020204     1.020204
1 -0.253587        -1.460681      -1.020204     1.020204
2 -1.113206        -0.785290       0.980196    -0.980196
3 -1.017692        -0.374182       0.980196    -0.980196
4 -1.781797         0.183751      -1.020204     1.020204
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Age              400 non-null    float64
 1   EstimatedSalary  400 non-null    float64
 2   Gender_Female    400 non-null    float64
 3   Gender_Male      400 non-null    float64
dtypes: float64(4)
memory usage: 12.6 KB
```

PROGRAM

```
# Split the DataFrame into the train and test sets.
# Perform train-test split using 'train_test_split' function.
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.3,random_state=42)
# Print the shape of the train and test sets.
print("shape of x_train",X_train.shape)
print("shape of x_test",X_test.shape)
print("shape of y_train",y_train.shape)
print("shape of y_test",y_test.shape)
```

OUTPUT



```
shape of x_train (280, 4)
shape of x_test (120, 4)
shape of y_train (280,)
shape of y_test (120,)
```

Activity 4: Build kNN Classifier Model

Deploy the kNN Classifier model for the optimal value of k using the steps given below:

1. Import the KNeighborsClassifier class from the sklearn.neighbors module (if not imported yet).

2. Create an object of KNeighborsClassifier and pass the optimal k value as 5 to its constructor.

3. Call the fit() function using the classifier object and pass the train set as inputs to this function.

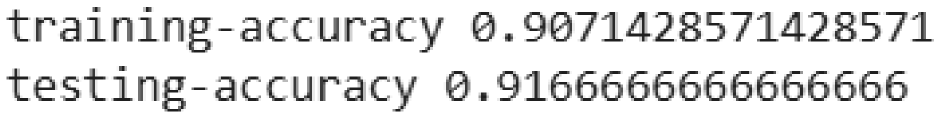
4. Perform prediction for train and test sets using the predict() function.

5. Also, determine the accuracy score of the train and test sets using the score() function.

PROGRAM

```
# Train kNN Classifier model
model=KNeighborsClassifier(n_neighbors=5)
model.fit(X_train,y_train)
# Perform prediction using 'predict()' function.
y_train_predict=model.predict(X_train)
y_test_predict=model.predict(X_test)

# Call the 'score()' function to check the accuracy score of the train set and test set.
from sklearn.metrics import accuracy_score
print("training-accuracy",accuracy_score(y_train,y_train_predict))
print("testing-accuracy",accuracy_score(y_test,y_test_predict))
```

OUTPUT



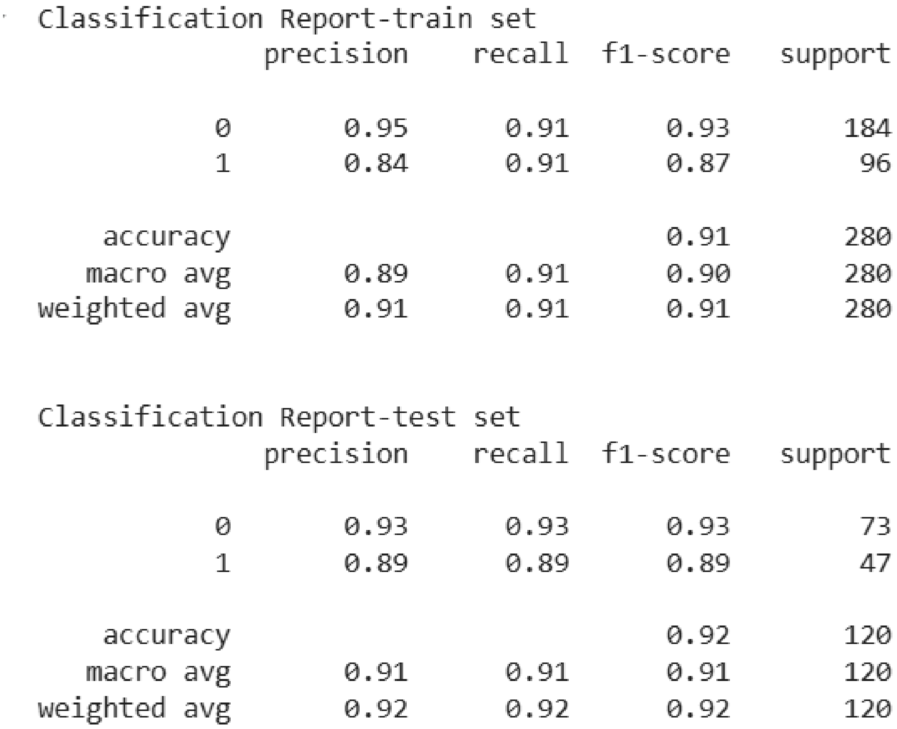
```
training-accuracy 0.9071428571428571
testing-accuracy 0.9166666666666666
```

Print the classification report to get an in-depth overview of the classifier performance using the classification_report() function of sklearn.metrics module.

PROGRAM

```
# Display the precision, recall, and f1-score values.
from sklearn.metrics import classification_report
print("Classification Report-train set")
print(classification_report(y_train, y_train_predict))
print()
print("Classification Report-test set")
print(classification_report(y_test, y_test_predict))
```

OUTPUT



```
Classification Report-train set
              precision    recall  f1-score   support

           0       0.95      0.91      0.93       184
           1       0.84      0.91      0.87        96

    accuracy                           0.91       280
   macro avg       0.89      0.91      0.90       280
weighted avg       0.91      0.91      0.91       280


Classification Report-test set
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        73
           1       0.89      0.89      0.89        47

    accuracy                           0.92       120
   macro avg       0.91      0.91      0.91       120
weighted avg       0.92      0.92      0.92       120
```

PROGRAM 11

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

Bayes‘ Theorem provides a way that we can calculate the probability of a piece of data belonging to a given class, given our prior knowledge. Bayes‘ Theorem is stated as:

P(classldata) = (P(datalclass) * P(class)) / P(data)

Where P(class|data) is the probability of class given the provided data.

We are using Iris Dataset. The Iris Flower Dataset involves predicting the flower species given measurements of iris flowers.

It is a multiclass classification problem. The number of observations for each class is balanced. There are 150 observations with 4 input variables and 1 output variable. The variable names are as follows:

Sepal length in cm.

Sepal width in cm.

Petal length in cm.

Petal width in cm.

Class.

Algorithm:

Step 1: Separate By Class.

Step 2: Summarize Dataset.

Step 3: Summarize Data By Class.

Step 4: Gaussian Probability Density Function.

Step 5: Class Probabilities.

PROGRAM

```
#import Modules
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.naive_bayes import GaussianNB
```

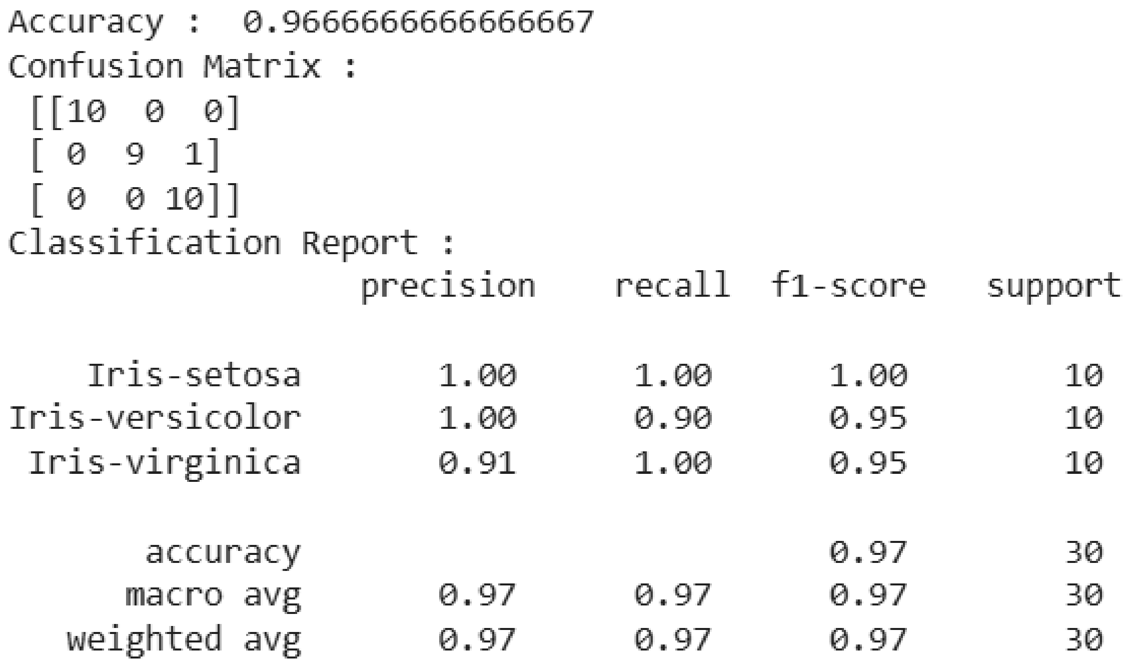
```
#load iris_dataset and do train_test_split
df=pd.read_csv('https://raw.githubusercontent.com/gokul-raj-
c/datasets/refs/heads/main/IRIS.csv')
X = df.drop('species', axis=1)
y = df['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
```

In this step, we introduce the class GaussianNB that is used from the skIearn.naive_bayes library. Here, we have used a Gaussian model, there are several other models such as Bernoulli, Categorical and Multinomial. Here, we assign the GaussianNB class to the variable classifier and fit the X train and y train values to it for training purpose.

PROGRAM

```
#implement naive bayes
classifier=GaussianNB()
classifier.fit(X_train,y_train)

#predict values for test data
y_pred=classifier.predict(X_test)

#Display accuracy score and display confusion matrix and classification report
pred=classifier.predict(X_test)
print("Accuracy : ",accuracy_score(y_test,y_pred))
print("Confusion Matrix : \n",confusion_matrix(y_test,y_pred))
print("Classification Report : \n",classification_report(y_test,pred))
```

OUTPUT



```
Accuracy :  0.9666666666666667
Confusion Matrix :
 [[10  0  0]
 [ 0  9  1]
 [ 0  0 10]]
Classification Report :
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      0.90      0.95        10
 Iris-virginica       0.91      1.00      0.95        10

       accuracy                           0.97        30
      macro avg       0.97      0.97      0.97        30
   weighted avg       0.97      0.97      0.97        30
```

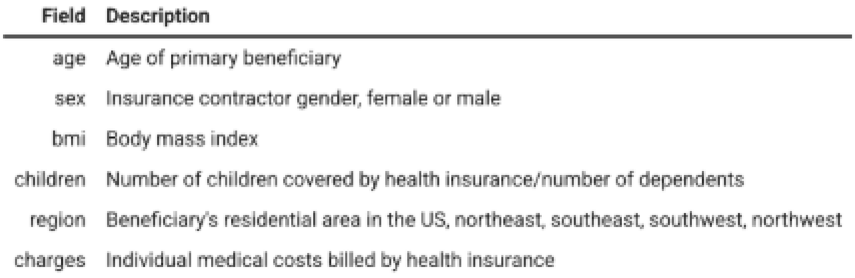
PROGRAM 12

AIM: Program to implement linear regression algorithm using any standard dataset available in the public domain and evaluate its performance.

Activity 1: Analysing the Dataset

Create a Pandas DataFrame for Insurance dataset using the below link. This dataset consists of following columns:

Field Description:



Field	Description
age	Age of primary beneficiary
sex	Insurance contractor gender, female or male
bmi	Body mass index
children	Number of children covered by health insurance/number of dependents
region	Beneficiary's residential area in the US, northeast, southeast, southwest, northwest
charges	Individual medical costs billed by health insurance

Source: https://www.kaggle.com/bmarco/health-insurance-data

Dataset Link: insurance_dataset.csv

Print the first five rows of the dataset. Check for null values and treat them accordingly.

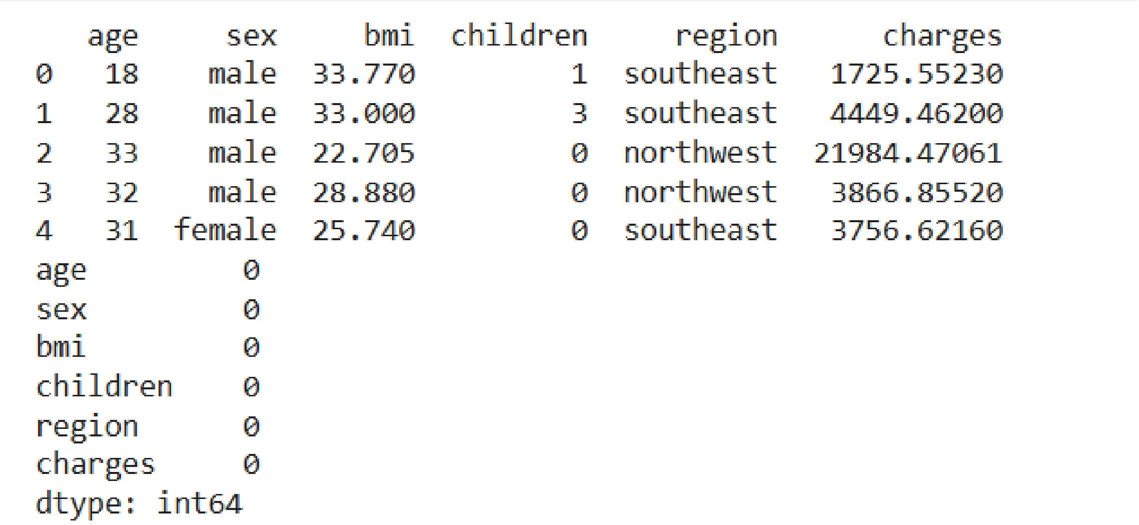
Create a regression plot with age on X-axis and charges on Y-axis to identify the relationship between these two attributes.

PROGRAM

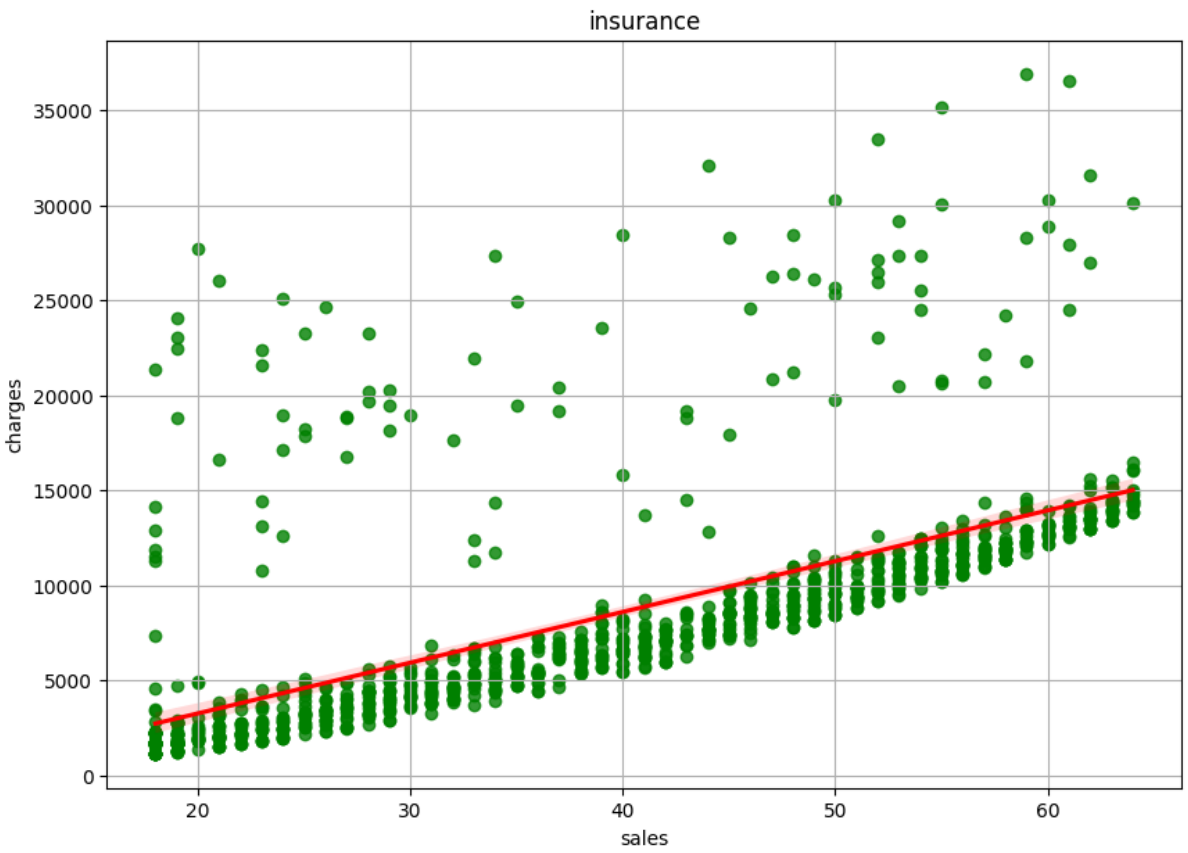
```
#import modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

#loading the datasets
df=pd.read_csv("https://raw.githubusercontent.com/gokul-raj-
c/datasets/refs/heads/main/insurance_dataset%20(1).csv")

# Print first five rows using head() function
print(df.head())
```

```
#Check if there are any null values. If any column has null values, treat them accordingly
print(df.isnull().sum())

# Create a regression plot between 'age' and 'charges'
plt.figure(figsize=(10,7))
plt.title("insurance")
sns.regplot(x="age",y="charges",data=df,color="green",line_kws={"color":"red"})
plt.xlabel("sales")
plt.ylabel("charges")
plt.grid()
plt.show()
```

OUTPUT



```
   age     sex     bmi  children     region      charges
0   18    male  33.770         1  southeast   1725.55230
1   28    male  33.000         3  southeast   4449.46200
2   33    male  22.705         0  northwest  21984.47061
3   32    male  28.880         0  northwest   3866.85520
4   31  female  25.740         0  southeast   3756.62160
age         0
sex         0
bmi         0
children    0
region      0
charges     0
dtype: int64
```



Activity 2: Train-Test Split

We have to determine the effect of age on insurance charges. Thus, age is the feature variable and charges is the target variable.

Split the dataset into training set and test set such that the training set contains 67% of the instances and the remaining instances will become the test set.

PROGRAM

```
# Split the DataFrame into the training and test sets
from sklearn.model_selection import train_test_split
X=df[["age"]]
y=df["charges"]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

Activity 3: Model Training

Implement simple linear regression using sklearn module in the following way:

1. Reshape the feature and the target variable arrays into two-dimensional arrays by using reshape(-1, 1) function of numpy module.

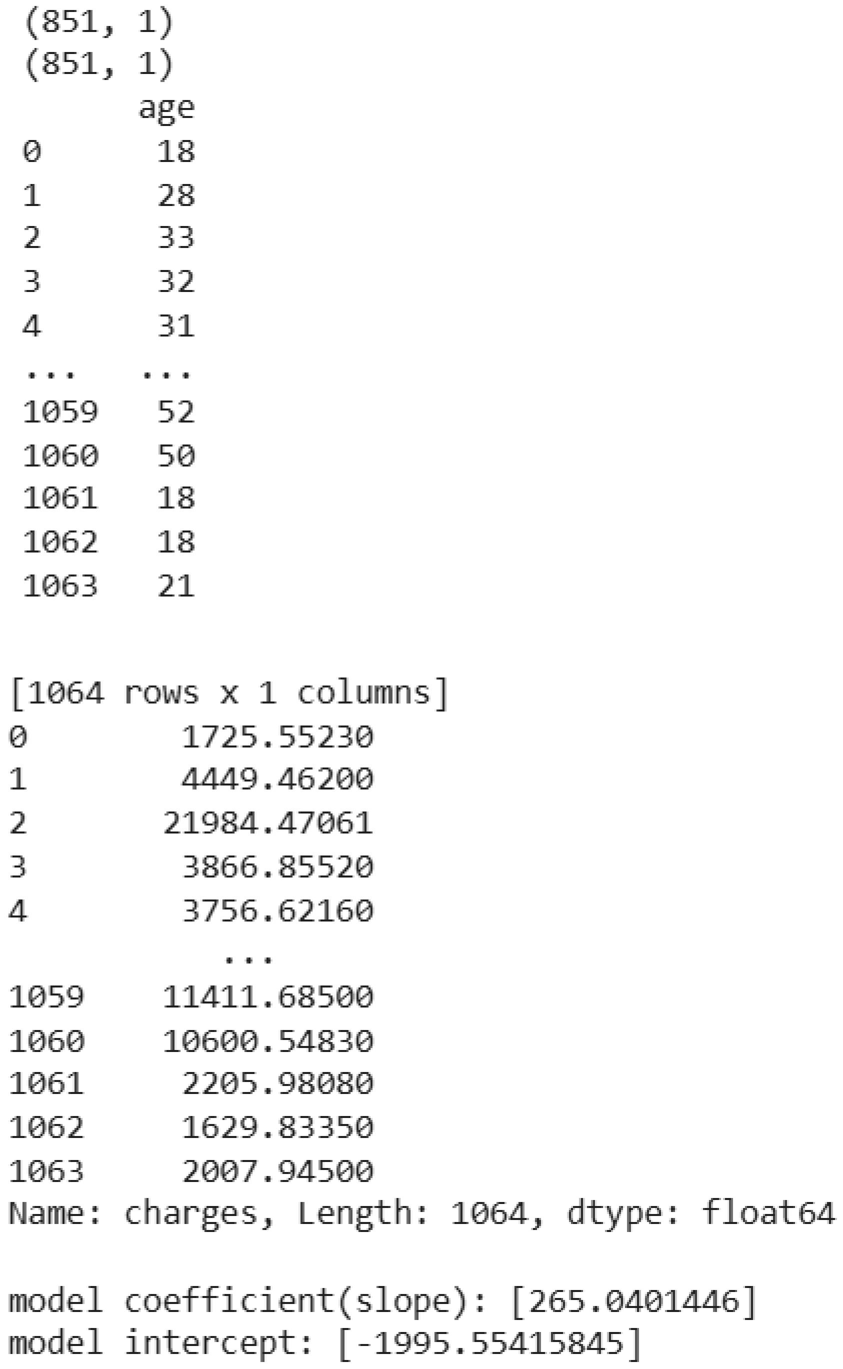
2. Deploy the model by importing the LinearRegression class and create an object of this class.

3. Call the fit() function on the LinearRegression object and print the slope and intercept values of the best line.

PROGRAM

```
# Create two-dimensional NumPy arrays for the feature and target
X_train=np.reshape(X_train,(-1,1))
X_test=np.reshape(X_test,(-1,1))
y_train=np.reshape(y_train,(-1,1))

# Print the shape or dimensions of these reshaped arrays
print(X_train.shape)
print(y_train.shape)
print(X)
print(y)

#model training
# 2. Deploy linear regression model using the 'sklearn.linear_model' module.
from sklearn.linear_model import LinearRegression

# Create an object of the 'LinearRegression' class.
model=LinearRegression()

# 3. Call the 'fit()' function
model.fit(X_train,y_train)
y_predicted=model.predict(X_test)

# Print the slope and intercept values
print()
print("model coefficient(slope):",model.coef_[0])
print("model intercept:",model.intercept_)
print()
```

OUTPUT



```
 (851, 1)
 (851, 1)
       age
 0      18
 1      28
 2      33
 3      32
 4      31
 ...   ...
 1059   52
 1060   50
 1061   18
 1062   18
 1063   21

[1064 rows x 1 columns]
0        1725.55230
1        4449.46200
2       21984.47061
3        3866.85520
4        3756.62160
            ...
1059    11411.68500
1060    10600.54830
1061     2205.98080
1062     1629.83350
1063     2007.94500
Name: charges, Length: 1064, dtype: float64

model coefficient(slope): [265.0401446]
model intercept: [-1995.55415845]
```

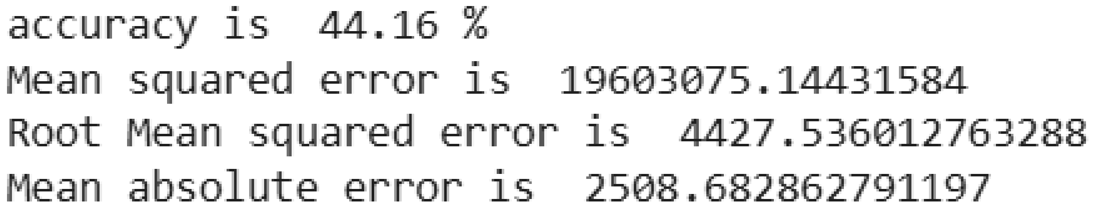
Activity 4: Model Prediction and Evaluation

Predict the values for both training and test sets by calling the predict() function on the LinearRegression object. Also, calculate the ,MSE, RMSE and MAE values to evaluate the accuracy of your model.

PROGRAM

```
# Predict the target variable values for both training set and test set
y_pred=model.predict(X_test)

# Call 'r2_score', 'mean_squared_error' & 'mean_absolute_error' functions of the 'sklearn' module.
from sklearn.metrics import mean_squared_error,mean_absolute_error,root_mean_squared_error,r2_score

# Print these values for both training set and test set
res=r2_score(y_test,y_pred)
print("accuracy is ",round(res*100,2),'%')
print("Mean squared error is ",mean_squared_error(y_test,y_pred))
print("Root Mean squared error is ",root_mean_squared_error(y_test,y_pred))
print("Mean absolute error is ",mean_absolute_error(y_test,y_pred))
```

OUTPUT



```
accuracy is  44.16 %
Mean squared error is  19603075.14431584
Root Mean squared error is  4427.536012763288
Mean absolute error is  2508.682862791197
```

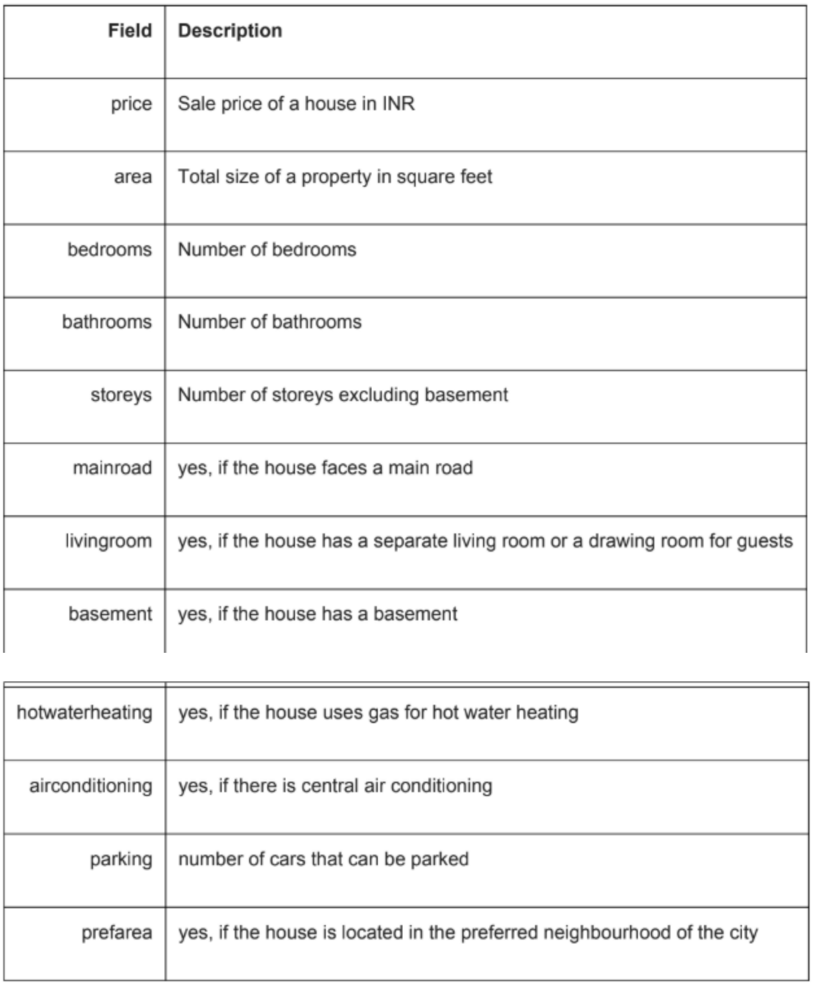
PROGRAM 13

AIM: Program to implement multiple regression algorithm using any standard dataset available in the public domain and evaluate its performance.

A real estate company wishes to analyse the prices of properties based on various factors such as area, number of rooms, bathrooms, bedrooms, etc. Create a multiple linear regression model which is capable of predicting the sale price of houses based on multiple factors and evaluate the accuracy of this model.

Activity 1: Analysing the Dataset

Create a Pandas DataFrame for Housing dataset using the below link. This dataset consists of following columns:



Field	Description
price	Sale price of a house in INR
area	Total size of a property in square feet
bedrooms	Number of bedrooms
bathrooms	Number of bathrooms
storeys	Number of storeys excluding basement
mainroad	yes, if the house faces a main road
livingroom	yes, if the house has a separate living room or a drawing room for guests
basement	yes, if the house has a basement
hotwaterheating	yes, if the house uses gas for hot water heating
airconditioning	yes, if there is central air conditioning
parking	number of cars that can be parked
prefarea	yes, if the house is located in the preferred neighbourhood of the city

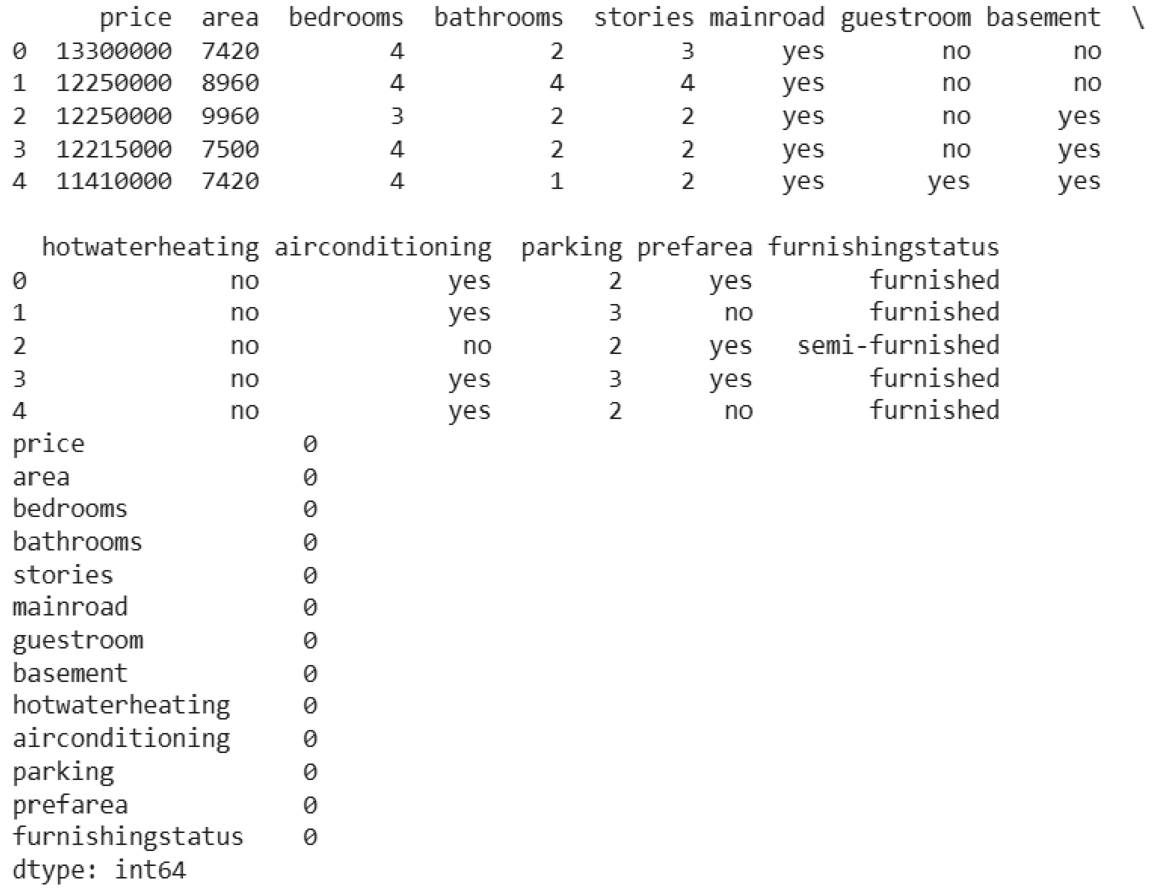
Dataset Link: house-prices.csv

Print the first five rows of the dataset. Check for null values and treat them accordingly.

PROGRAM

```
# Import modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df=pd.read_csv("https://raw.githubusercontent.com/gokul-raj
c/datasets/refs/heads/main/Housing_Price.csv")

# Print first five rows using head() function
print(df.head())

# Check if there are any null values. If any column has null values, treat them
accordingly
print(df.isnull().sum())
```

OUTPUT



```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no             yes        2      yes        furnished
1              no             yes        3       no        furnished
2              no              no        2      yes   semi-furnished
3              no             yes        3      yes        furnished
4              no             yes        2       no        furnished
price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
prefarea            0
furnishingstatus    0
dtype: int64
```

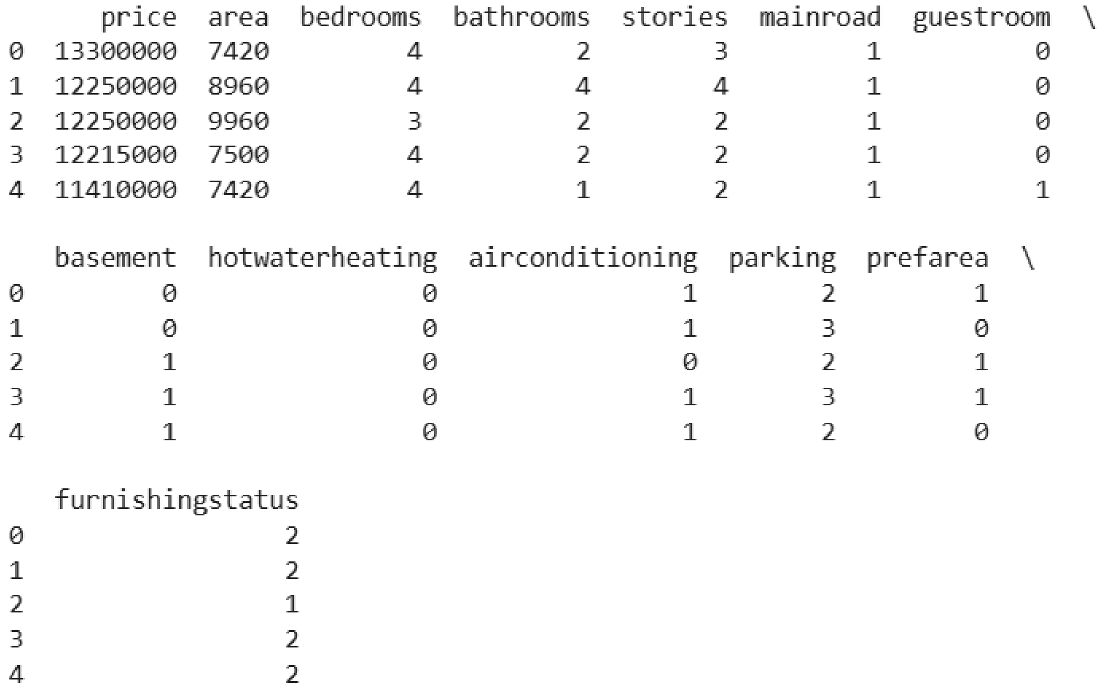
Activity 2: Data Preparation

This dataset contains many columns having categorical data i.e. values 'Yes' or 'No'. However for linear regression, we need numerical data. So you need to convert all 'Yes' and 'No' values to 1s and 0s, where
• 1 means 'Yes'
• 0 means 'No'
Similarly, replace
• unfurnished with 0
• semi-furnished with 1
• furnished with 2
Hint: To replace all 'Yes' values with 1 and 'No' values with 0, use replace() function of the DataFrame object.

PROGRAM

```
# Replace all non-numeric values with numeric values.
df.replace(to_replace="yes", value=1, inplace=True)
df.replace(to_replace="no", value=0, inplace=True)
df.replace(to_replace="unfurnished", value=0, inplace=True)
df.replace(to_replace="semi-furnished", value=1, inplace=True)
df.replace(to_replace="furnished", value=2, inplace=True)
print(df.head())
```

OUTPUT



```
      price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
0  13300000  7420         4          2        3         1          0
1  12250000  8960         4          4        4         1          0
2  12250000  9960         3          2        2         1          0
3  12215000  7500         4          2        2         1          0
4  11410000  7420         4          1        2         1          1

   basement  hotwaterheating  airconditioning  parking  prefarea  \
0         0                0                1        2         1
1         0                0                1        3         0
2         1                0                0        2         1
3         1                0                1        3         1
4         1                0                1        2         0

   furnishingstatus
0                 2
1                 2
2                 1
3                 2
4                 2
```

Activity 3: Train-Test Split

You need to predict the house prices based on several factors. Thus, price is the target variable and other columns except price will be feature variables. Split the dataset into training set and test set such that the training set contains 67% of the instances and the remaining instances will become the test set.

PROGRAM

```
# Split the DataFrame into the training and test sets.
from sklearn.model_selection import train_test_split
x=df.drop('price',axis=1)
y=df['price']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

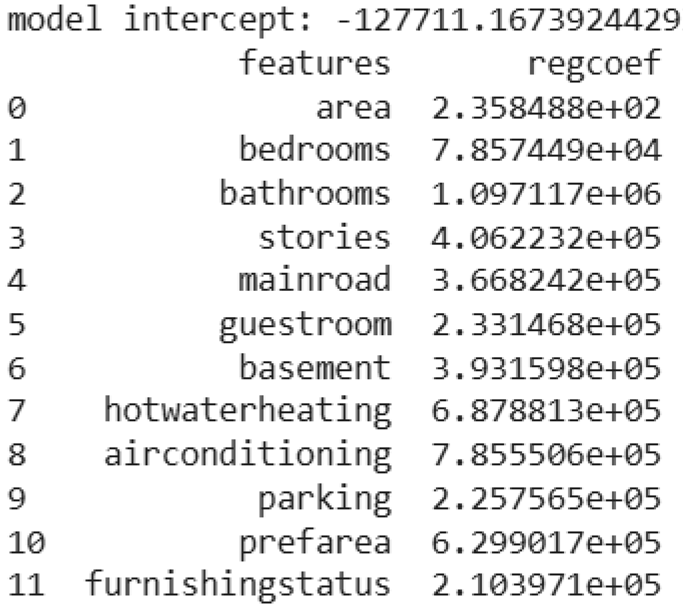
Activity 4: Model Training

Implement multiple linear regression using sklearn module in the following way:
1. Deploy the model by importing the LinearRegression class and create an object of this class.
2. Call the fit() function on the LinearRegression object.

PROGRAM

```
# Build linear regression model
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
# Print the value of the intercept
print("model intercept:",model.intercept_)
# Print the names of the features along with the values of their corresponding
coefficients.
coef = model.coef_
f=x.columns
d=pd.DataFrame({'features':f,'regcoef':coef})
print(d)
```

OUTPUT



```
model intercept: -127711.16739244293
            features       regcoef
0               area  2.358488e+02
1           bedrooms  7.857449e+04
2          bathrooms  1.097117e+06
3            stories  4.062232e+05
4           mainroad  3.668242e+05
5          guestroom  2.331468e+05
6           basement  3.931598e+05
7     hotwaterheating  6.878813e+05
8     airconditioning  7.855506e+05
9            parking  2.257565e+05
10          prefarea  6.299017e+05
11  furnishingstatus  2.103971e+05
```

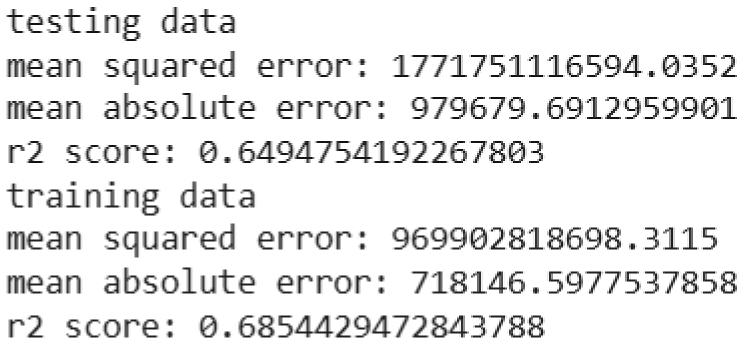
Activity 5: Model Prediction and Evaluation

Predict the values for both training and test sets by calling the predict() function on the LinearRegression object. Also, calculate the R2, MSE, RMSE and MAE values to evaluate the accuracy of your model.

PROGRAM

```
# Predict the target variable values for training and test set
y_predicted=model.predict(x_test)

# Evaluate the linear regression model using the 'r2_score','mean_squared_error' &
'mean_absolute_error' functions of the 'sklearn' module.
from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error
print("testing data")
y_predicted=model.predict(x_test)
print("mean squared error:",mean_squared_error(y_test,y_predicted))
print("mean absolute error:",mean_absolute_error(y_test,y_predicted))
print("r2 score:",r2_score(y_test,y_predicted))
print("training data")
y_predicted1=model.predict(x_train)
print("mean squared error:",mean_squared_error(y_train,y_predicted1))
print("mean absolute error:",mean_absolute_error(y_train,y_predicted1))
print("r2 score:",r2_score(y_train,y_predicted1))
```

OUTPUT



```
testing data
mean squared error: 1771751116594.0352
mean absolute error: 979679.6912959901
r2 score: 0.6494754192267803
training data
mean squared error: 969902818698.3115
mean absolute error: 718146.5977537858
r2 score: 0.6854429472843788
```

COURSE OUTCOME 3

Use different packages and frameworks to implement text classification using SVM and clustering using k-means

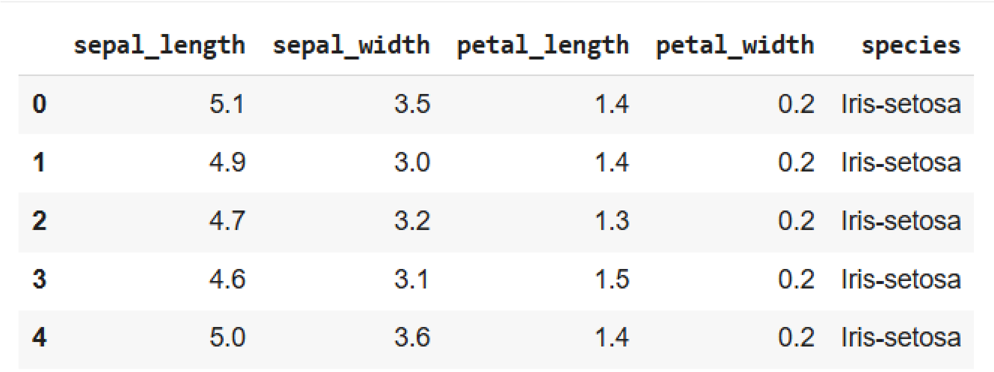
PROGRAM 14

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

PROGRAM

```
#import the modules
import graphviz
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier, plot_tree, export_graphviz
from sklearn.model_selection import train_test_split

#load dataset
df=pd.read_csv('https://raw.githubusercontent.com/gokul-raj-c/datasets/refs/heads/main/IRIS.csv')
df.head()
```

OUTPUT

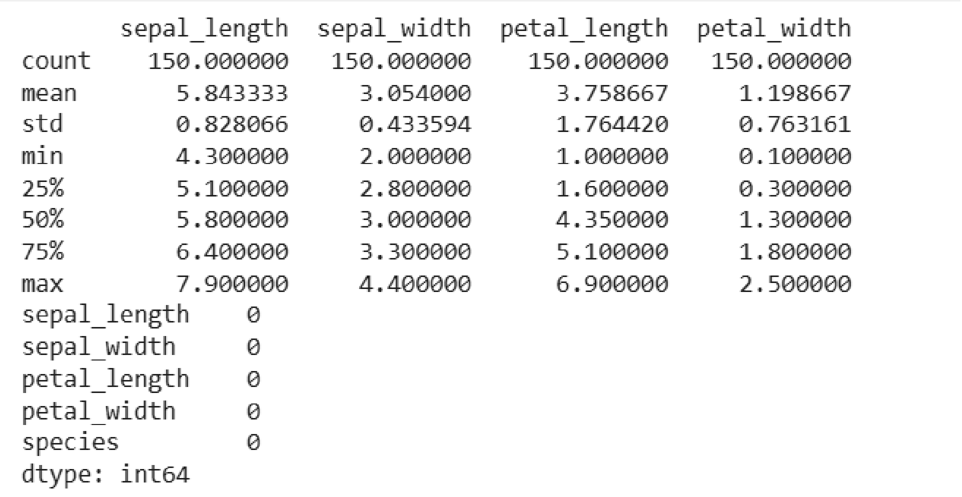


	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

PROGRAM

```
#Display the number of rows & columns in dataframe
print(df.describe())
print(df.isnull().sum())
```

OUTPUT

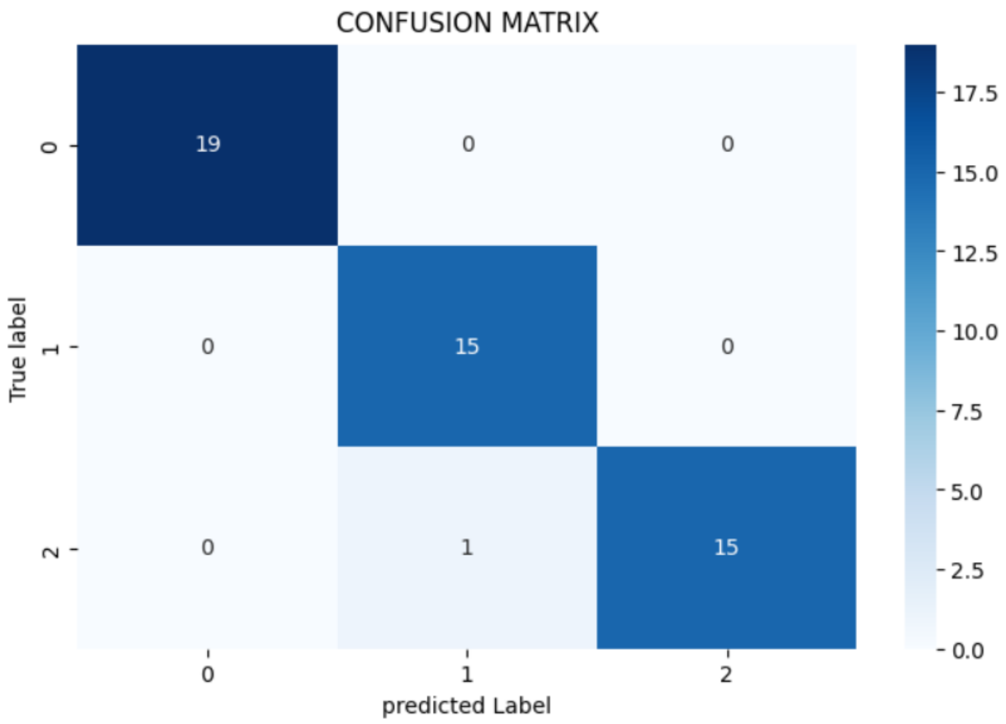


```
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

PROGRAM

```
#Perform Train Test Split
x=df[['sepal_length','sepal_width','petal_length','petal_width']]
y=df['species']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)

#Construct decission tree classifier with criterion='entropy' with min_samples_split to 50. Default value is 2
dt_model=DecisionTreeClassifier(criterion='entropy',min_samples_split=50,min_samples_leaf=3)
dt_model.fit(x_train,y_train)

#Display Accuracy on test data
y_predict=dt_model.predict(x_test)
cm=confusion_matrix(y_test,y_predict)
plt.figure(figsize=(8,5))
sns.heatmap(cm,annot=True,cmap='Blues')
plt.xlabel('predicted Label')
plt.ylabel('True label')
plt.title('CONFUSION MATRIX')
plt.show()
plt.figure(figsize=(15,15))
plot_tree(dt_model,filled=True,fontsize=10)
plt.show()
```

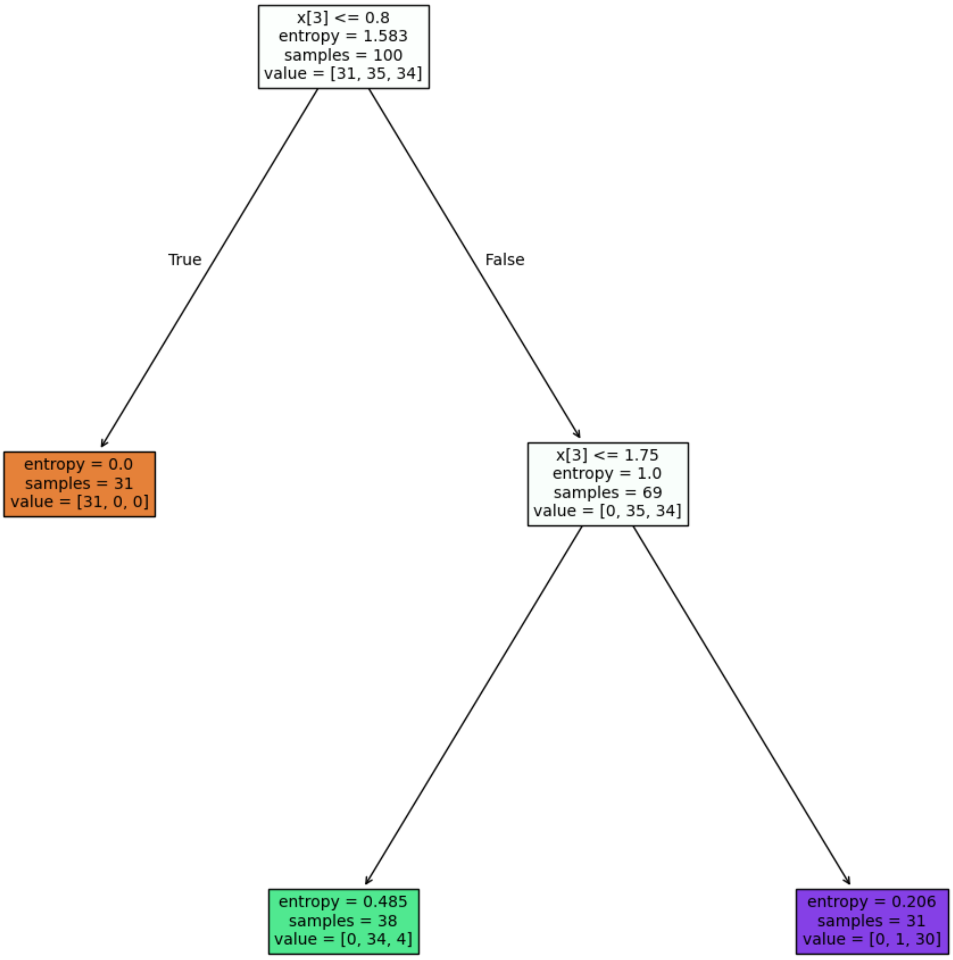
OUTPUT



PROGRAM

```
fname=['sepal_length','sepal_width','petal_length','petal_width']
tname=y.unique().tolist()
dot_data=export_graphviz(dt_model,out_file=None,feature_names=fname,class_nam
es=tname,filled=True,rounded=True,special_characters=True)
graph=graphviz.Source(dot_data)
graph.render("iris_decision",view=True,format='png')
print("")
```

OUTPUT


x[3] <= 0.8
entropy = 1.583
samples = 100
value = [31, 35, 34]
True
False
entropy = 0.0
samples = 31
value = [31, 0, 0]
x[3] <= 1.75
entropy = 1.0
samples = 69
value = [0, 35, 34]
entropy = 0.485
samples = 38
value = [0, 34, 4]
entropy = 0.206
samples = 31
value = [0, 1, 30]

PROGRAM 15

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

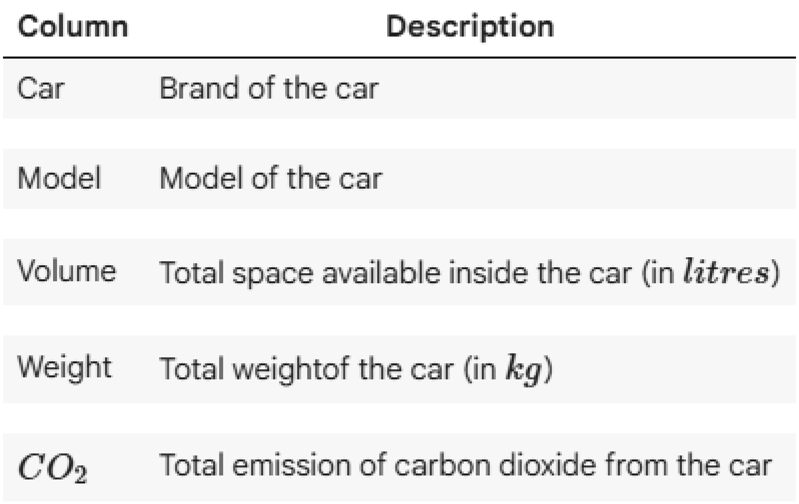
Problem Statement

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

Dataset Description

In this project, we will be using the dataset holding the information of carbon dioxide emission from different car models.

The dataset includes 36 instances with 5 columns which can be briefed as:



Column	Description
Car	Brand of the car
Model	Model of the car
Volume	Total space available inside the car (in $litres$)
Weight	Total weightof the car (in kg)
CO_2	Total emission of carbon dioxide from the car

Activity 1: Import Modules and Read Data

Import the necessary Python modules along with the following modules:

- KMeans - For clustering using K-means.
- re - To remove unwanted rows using regex.

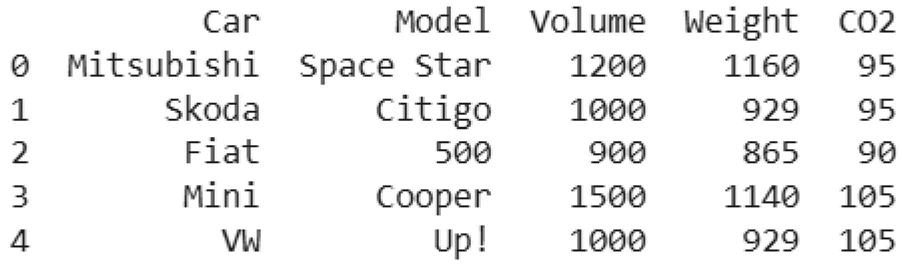
Read the data from a CSV file to create a Pandas DataFrame and go through the necessary data-cleaning process (if required).

Dataset link: https://raw.githubusercontent.com/jiss-sngce/CO_3/main/jkcars.csv

PROGRAM

```
# Import the modules and Read the data.
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import seaborn as sns

# Print the first five records
df = pd.read_csv("https://raw.githubusercontent.com/jiss-
sngce/CO_3/main/jkcars.csv")
print(df.head())
```

OUTPUT

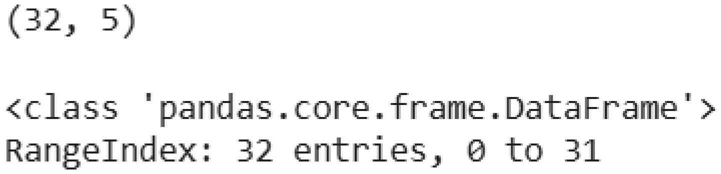


```
          Car       Model  Volume  Weight  CO2
0  Mitsubishi  Space Star    1200    1160   95
1       Skoda      Citigo    1000     929   95
2        Fiat         500     900     865   90
3        Mini      Cooper    1500    1140  105
4          VW         Up!    1000     929  105
```

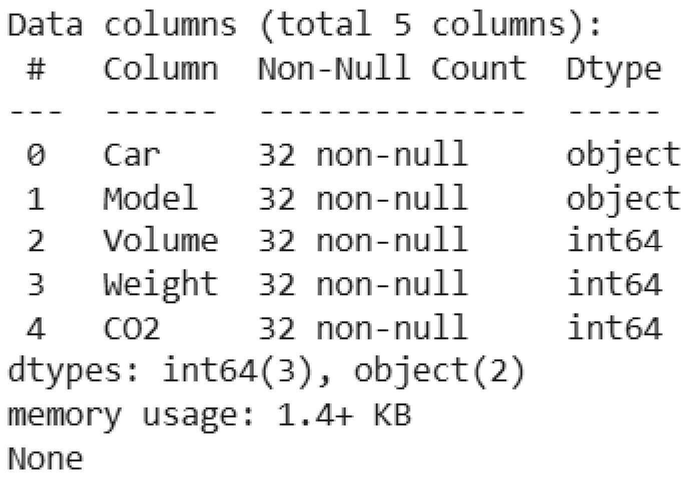
PROGRAM

```
# Get the total number of rows and columns, data types of columns and missing
values (if exist) in the dataset.
print(df.shape)
print()
print(df.info())
```

OUTPUT



```
(32, 5)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
```



```
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Car     32 non-null     object
 1   Model   32 non-null     object
 2   Volume  32 non-null     int64 
 3   Weight  32 non-null     int64 
 4   CO2     32 non-null     int64 
dtypes: int64(3), object(2)
memory usage: 1.4+ KB
None
```

Activity 3: Find Optimal value of K

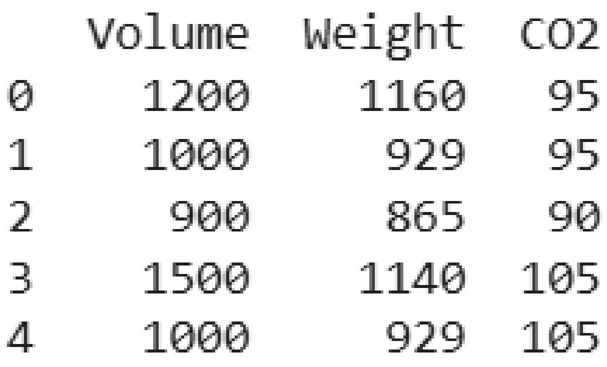
In this activity, you need to find the optimal value of K using the silhouette score.

1. Create a subset of the dataset consisting of three columns i.e Volume, Weight, and CO2.

PROGRAM

```
# Create a new DataFrame consisting of three columns 'Volume', 'Weight', 'CO2'.
new_data = df[['Volume', 'Weight', 'CO2']]

# Print the first 5 rows of this new DataFrame.
print(new_data.head(5))
```

OUTPUT



```
   Volume  Weight  CO2
0    1200    1160   95
1    1000     929   95
2     900     865   90
3    1500    1140  105
4    1000     929  105
```

2. Compute K-Means clustering for the 3D dataset data_3d by varying K from 2 to 10 clusters. Also, for each K, calculate silhouette score using silhouette_score function.

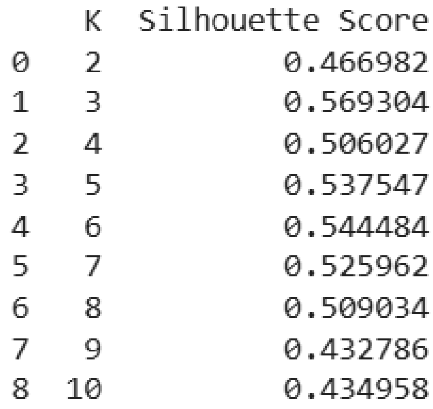
Steps to Follow

- Create an empty list to store silhouette scores obtained for each K (let's say sil_scores).
- Initiate a for loop that ranges from 2 to 10.
- Perform K-means clustering for the current value of K inside for loop.
- Use fit() and predict() to create clusters.
- Calculate silhouette score for current K value using silhouette_score() function and append it to the empty list sil_scores.
- Create a DataFrame with two columns. The first column must contain K values from 2 to 10 and the second column must contain silhouette values obtained after the for loop.

PROGRAM

```
# Calculate inertia for different values of 'K'.
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Create an empty list to store silhouette scores obtained for each 'K'
sil_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=10, n_init=10)
    kmeans.fit(new_data)
    score = silhouette_score(new_data, kmeans.labels_)
    sil_scores.append(score)
silhoutte_df = pd.DataFrame({'K': range(2, 11), 'Silhouette Score': sil_scores})
print(silhoutte_df)
```

OUTPUT



```
    K  Silhouette Score
0   2          0.466982
1   3          0.569304
2   4          0.506027
3   5          0.537547
4   6          0.544484
5   7          0.525962
6   8          0.509034
7   9          0.432786
8  10          0.434958
```

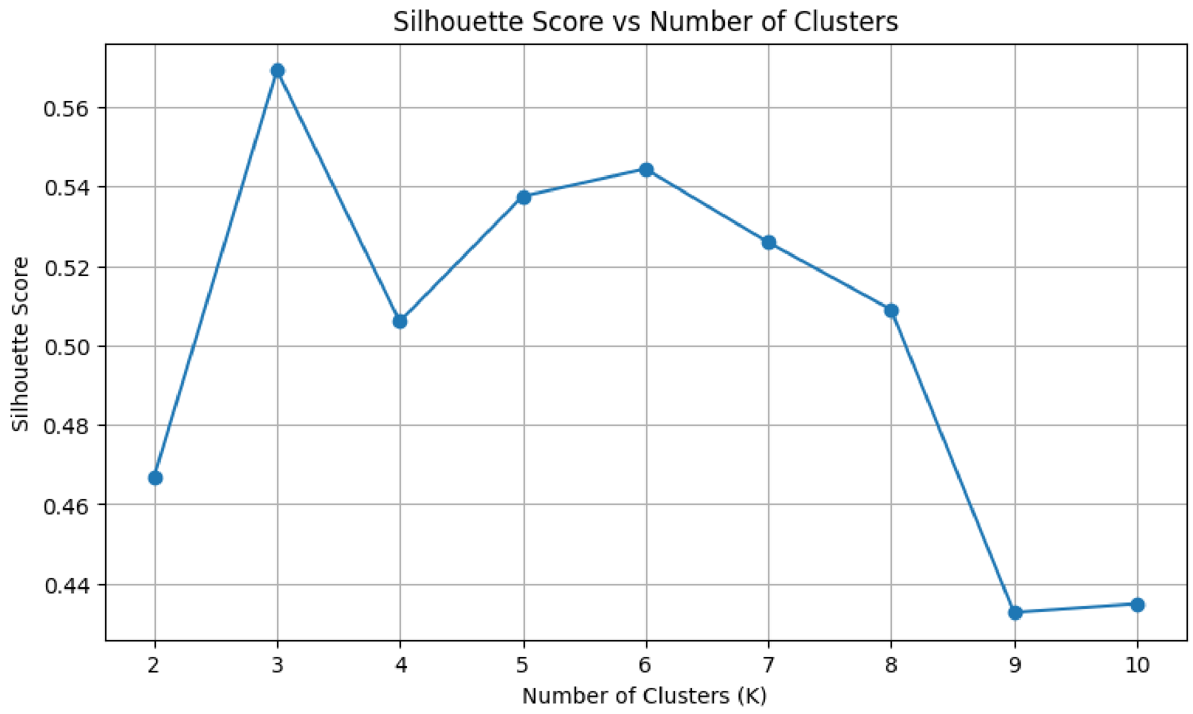
Activity 4: Plot silhouette Scores find optimal value for K

Create a line plot with K ranging from 2 to 10 on the x-axis and the silhouette scores stored in sil_scores list on the y-axis.

PROGRAM

```
# Plot silhouette scores vs number of clusters.
plt.figure(figsize=(9, 5))
plt.plot(silhoutte_df['K'], silhoutte_df['Silhouette Score'], marker='o')
plt.title('Silhouette Score vs Number of Clusters')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Silhouette Score')
plt.grid(True)
plt.show()
```

OUTPUT

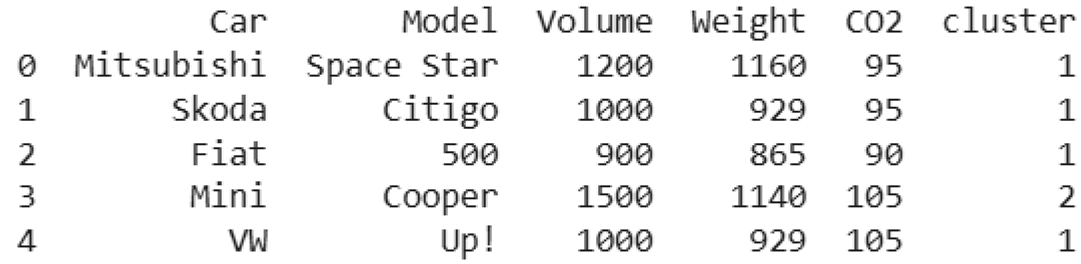


PROGRAM

```
# Clustering the dataset for K = 3
# Perform K-Means clustering with n_clusters = 3 and random_state = 10
k_means = KMeans(n_clusters=3, random_state=10)
```

```
# Fit the model to the scaled_df
k_means.fit(new_data)

# Make a series using predictions by K-Means
clusters = pd.Series(k_means.predict(new_data))

# Create a DataFrame with cluster labels for cluster visualisation
df['cluster'] = clusters
print(df.head())
```

OUTPUT



```
          Car       Model  Volume  Weight  CO2  cluster
0  Mitsubishi  Space Star    1200    1160   95        1
1       Skoda      Citigo    1000     929   95        1
2        Fiat         500     900     865   90        1
3        Mini      Cooper    1500    1140  105        2
4          VW         Up!    1000     929  105        1
```

PROGRAM 16

AIM: Program to implement text classification using Support vector machine using any standard dataset available in the public domain and evaluate its performance.

PROGRAM

```
#SIMPLE PROGRAM
#import libraries
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

# Define the small dataset
# We will classify texts as tech(0) or finance(1)
data = [
    "Apple launched a new iPhone with better neural engine.",  # tech
    "The stock market saw huge gains after the quarterly report.", # finance
    "Google's machine learning model achieved 90% accuracy.",  # tech
    "Investors are worried about rising interest rates and inflation.", # finance
    "Python libraries like scikit-learn are great for ML.", # tech
    "Bonds and treasury yields are highly volatile this week." # finance
]
#0 for tech 1 for finance
labels=[0,1,0,1,0,1]
target_names=['tech','finance']

# 2. Split Data
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2,
random_state=42)
print("Total Data Points:", len(data))
print("Training Data Points:", len(X_train))
print("Testing Data Points:", len(X_test))
```

OUTPUT

```
Total Data Points: 6
Training Data Points: 4
Testing Data Points: 2
```

PROGRAM

```
# 3. Feature Extraction (TF-IDF)
vectorizer=TfidfVectorizer(stop_words="english")

# Convert training data
X_train_vectors=vectorizer.fit_transform(X_train)

# Convert testing data
X_test_vectors=vectorizer.transform(X_test)

# 4. Initialize and Train the SVM
svm_classifier=SVC(kernel="linear",random_state=42)
svm_classifier.fit(X_train_vectors,y_train)

# 5. Predict and Evaluate
y_pred=svm_classifier.predict(X_test_vectors)
accuracy=accuracy_score(y_test,y_pred)
print("Accuracy:",accuracy)
```

OUTPUT

```
Accuracy: 0.5
```

PROGRAM

```
# 6. Simple Prediction
text = "OpenAI’s new model delivers more natural and context-aware responses."
text_vectors = vectorizer.transform([text])
prediction = svm_classifier.predict(text_vectors)
print(f'Prediction: {target_names[prediction[0]]}')
```

OUTPUT

```
Prediction: tech
```

COURSE OUTCOME 4

Implement convolutional neural network algorithm using Keras framework.

PROGRAM 17

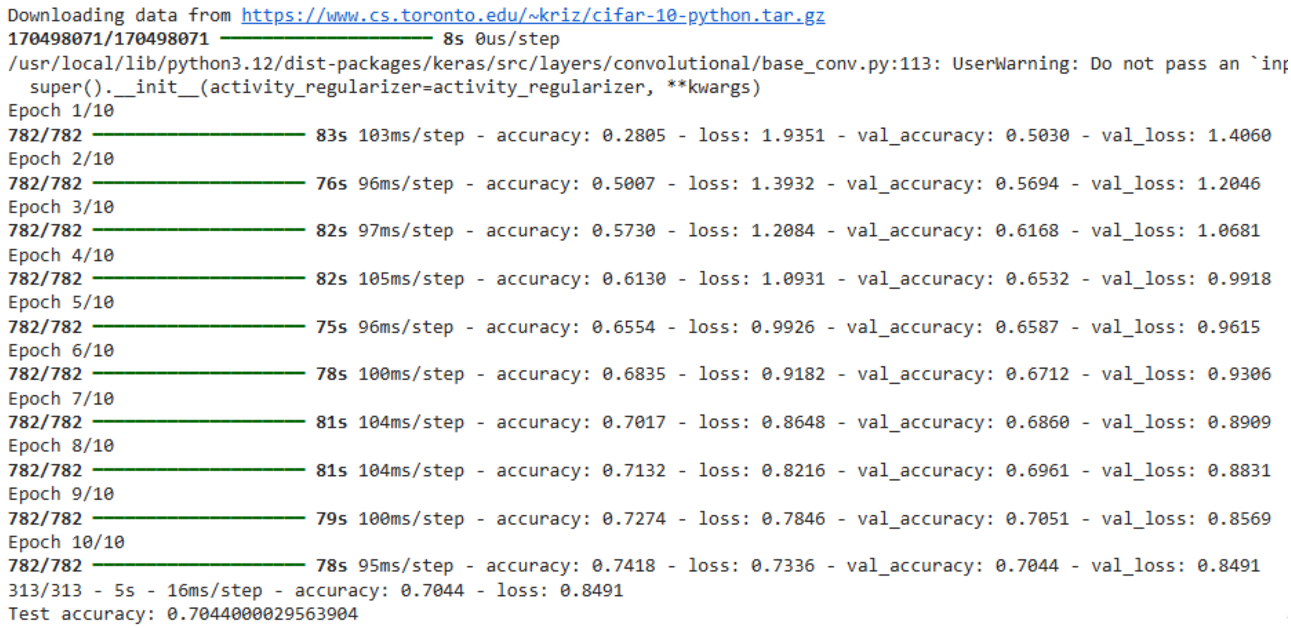
AIM: Programs on convolutional neural network to classify images from any standard dataset in the public domain.

PROGRAM

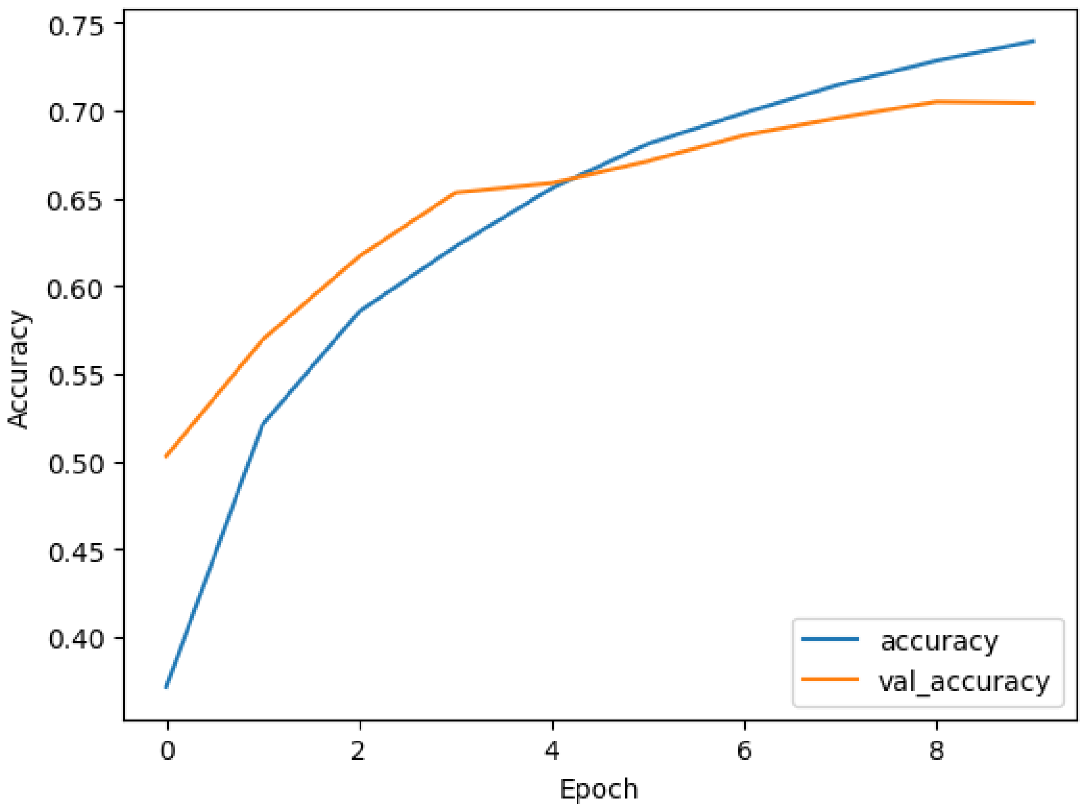
```
# Import necessary libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,Dropout
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets import cifar10
import matplotlib.pyplot as plt

# Load the CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Preprocess the data
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalize pixel values to be
between 0 and 1
y_train = to_categorical(y_train, 10)  # Convert labels to one-hot encoding
y_test = to_categorical(y_test, 10)

# Define the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')  # 10 classes for CIFAR-10
])
```

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Train the model
history = model.fit(x_train, y_train, epochs=10, batch_size=64,
validation_data=(x_test, y_test))

# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f'Test accuracy: {test_acc}')

# Plot training and validation accuracy over epochs
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()
```

OUTPUT



```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 8s 0us/step
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `inp
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 103ms/step - accuracy: 0.2805 - loss: 1.9351 - val_accuracy: 0.5030 - val_loss: 1.4060
Epoch 2/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 76s 96ms/step - accuracy: 0.5007 - loss: 1.3932 - val_accuracy: 0.5694 - val_loss: 1.2046
Epoch 3/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 97ms/step - accuracy: 0.5730 - loss: 1.2084 - val_accuracy: 0.6168 - val_loss: 1.0681
Epoch 4/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 105ms/step - accuracy: 0.6130 - loss: 1.0931 - val_accuracy: 0.6532 - val_loss: 0.9918
Epoch 5/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 75s 96ms/step - accuracy: 0.6554 - loss: 0.9926 - val_accuracy: 0.6587 - val_loss: 0.9615
Epoch 6/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 78s 100ms/step - accuracy: 0.6835 - loss: 0.9182 - val_accuracy: 0.6712 - val_loss: 0.9306
Epoch 7/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 104ms/step - accuracy: 0.7017 - loss: 0.8648 - val_accuracy: 0.6860 - val_loss: 0.8909
Epoch 8/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 104ms/step - accuracy: 0.7132 - loss: 0.8216 - val_accuracy: 0.6961 - val_loss: 0.8831
Epoch 9/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 79s 100ms/step - accuracy: 0.7274 - loss: 0.7846 - val_accuracy: 0.7051 - val_loss: 0.8569
Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 78s 95ms/step - accuracy: 0.7418 - loss: 0.7336 - val_accuracy: 0.7044 - val_loss: 0.8491
313/313 - 5s - 16ms/step - accuracy: 0.7044 - loss: 0.8491
Test accuracy: 0.7044000029563904
```

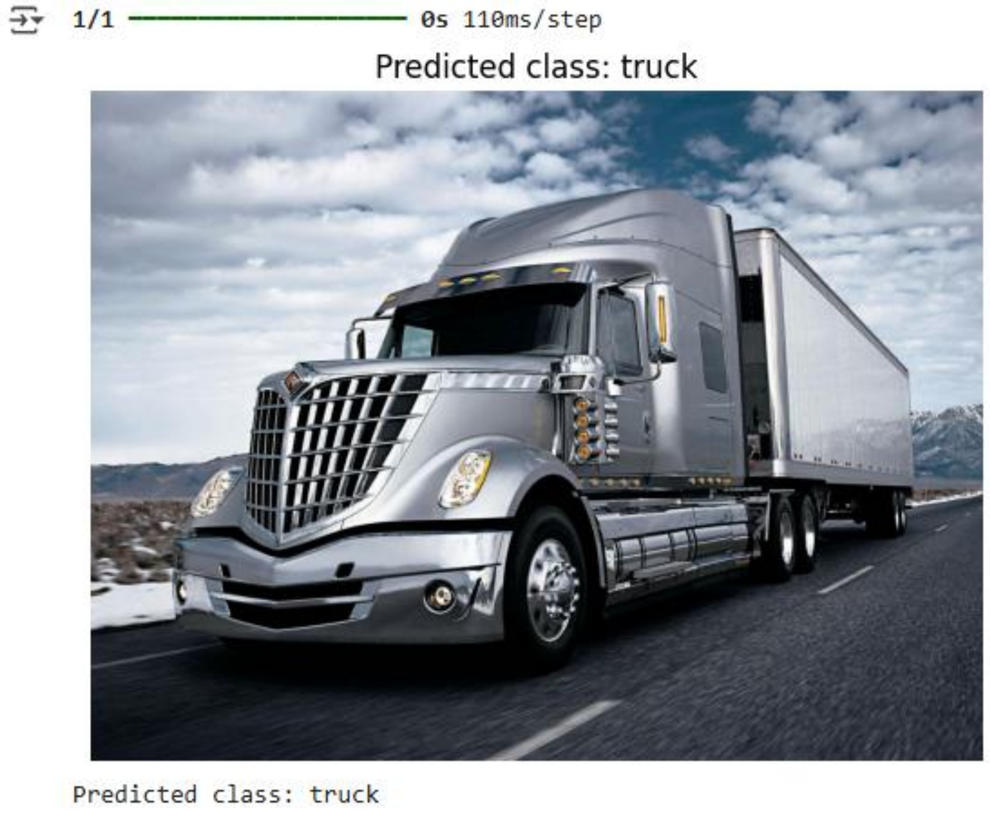


PROGRAM

```
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt

# Path to the new image you want to classify
image_path = 'test.jpg'

# Load and preprocess the image
img = load_img(image_path, target_size=(32, 32))  # Resize the image to 32x32
img_array = img_to_array(img)  # Convert the image to a numpy array
img_array = img_array / 255.0  # Normalize the image array to [0, 1] range
img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension for
prediction

# Make a prediction
predictions = model.predict(img_array)
predicted_class = np.argmax(predictions, axis=1)  # Get the index of the highest
probability
```

```
# Define class names for CIFAR-10 dataset
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

# Display the image and prediction
plt.imshow(load_img(image_path))
plt.title(f'Predicted class: {class_names[predicted_class[0]]}')
plt.axis('off')
plt.show()

print(f"Predicted class: {class_names[predicted_class[0]]}")
```

OUTPUT



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
```

```
Predicted class: truck
```

COURSE OUTCOME 5

Implement programs for natural language processing using NLTK

PROGRAM 18

AIM: For a given text:

1.Perform word and sentence tokenization.

PROGRAM

```
import nltk
from nltk import word_tokenize,sent_tokenize
text="The data set given satisfies the requirement for model generation. This is used in Data Science Lab"
print("Text:")
print(text)
print("Word Tokenization:")
print(nltk.word_tokenize(text))
print("Sentence Tokenization:")
print(nltk.sent_tokenize(text))
```

OUTPUT

```
C:\Users\MITS\PycharmProjects\PythonProject3\.venv\Scripts\python.exe
C:\Users\MITS\PycharmProjects\PythonProject3\question1.py
Text:
The data set given satisfies the requirement for model generation. This is used in Data Science Lab
Word Tokenization:
['The', 'data', 'set', 'given', 'satisfies', 'the', 'requirement', 'for', 'model', 'generation', '.', 'This', 'is', 'used', 'in', 'Data', 'Science', 'Lab']
Sentence Tokenization:
['The data set given satisfies the requirement for model generation.', 'This is used in Data Science Lab']
Process finished with exit code 0
```

2. Remove the stop words from the given text.

PROGRAM

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
```

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
text="The data set given satisfies the requirement for model generation. This is used in Data Science Lab"
tokens=word_tokenize(text)
stop_words=set(stopwords.words('english'))
filterd_tokens=[word for word in tokens if word.lower() not in stop_words]
print("Text before:")
print(text)
print("Text after removing stopwords:")
print(' '.join(filterd_tokens))
```

OUTPUT

```
C:\Users\MITS\PycharmProjects\PythonProject3\.venv\Scripts\python.exe
C:\Users\MITS\PycharmProjects\PythonProject3\question2.py
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\MITS\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\MITS\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
Text before:
The data set given satisfies the requirement for model generation. This is used in Data Science Lab
Text after removing stopwords:
data set given satisfies requirement model generation . used Data Science Lab
Process finished with exit code 0
```

3. Perform Part of Speech tagging

PROGRAM

```
import nltk
from nltk import word_tokenize
from nltk import pos_tag
text="The data set given satisfies the requirement for model generation. This is used in Data Science Lab"
print("Text:")
print(text)
tokenized_text=nltk.word_tokenize(text)
```

```
tags=token_tags=nltk.pos_tag(tokenized_text)
print("Part of Speech tagging")
for word,token_tags in tags:
print(f"{word}: {token_tags}")
```

OUTPUT

```
C:\Users\MITS\PycharmProjects\PythonProject3\.venv\Scripts\python.exe
C:\Users\MITS\PycharmProjects\PythonProject3\question3.py
Text:
The data set given satisfies the requirement for model generation. This is used in Data
Science Lab
Part of Speech tagging
The: DT
data: NN
set: NN
given: VBN
satisfies: VBZ
the: DT
requirement: NN
for: IN
model: NN
generation: NN
.: .
This: DT
is: VBZ
used: VBN
in: IN
Data: NNP
Science: NNP
Lab: NNP
Process finished with exit code 0
```

4. create n-grams for different values of n=2,4

PROGRAM

```
def generate_ngrams(text,n):
tokens=text.split()
ngrams=[tuple(tokens[i:i+n]) for i in range(len(tokens)-n+1)]
return ngrams
text="The data set given satisfies the requirement for model generation. This is used
```

```
in Data Science Lab"
print("Text:")
print(text)
ntwo=generate_ngrams(text,2)
nfour=generate_ngrams(text,4)
print("N=2:",ntwo)
print("N=4:",nfour)
```

OUTPUT

```
C:\Users\MITS\PycharmProjects\PythonProject3\.venv\Scripts\python.exe
C:\Users\MITS\PycharmProjects\PythonProject3\question4.py
Text:
The data set given satisfies the requirement for model generation. This is used in Data
Science Lab
N=2: [('The', 'data'), ('data', 'set'), ('set', 'given'), ('given', 'satisfies'), ('satisfies', 'the'),
('the', 'requirement'), ('requirement', 'for'), ('for', 'model'), ('model', 'generation.'),
('generation.', 'This'), ('This', 'is'), ('is', 'used'), ('used', 'in'), ('in', 'Data'), ('Data',
'Science'), ('Science', 'Lab')]
N=4: [('The', 'data', 'set', 'given'), ('data', 'set', 'given', 'satisfies'), ('set', 'given',
'satisfies', 'the'), ('given', 'satisfies', 'the', 'requirement'), ('satisfies', 'the', 'requirement',
'for'), ('the', 'requirement', 'for', 'model'), ('requirement', 'for', 'model', 'generation.'),
('for', 'model', 'generation.', 'This'), ('model', 'generation.', 'This', 'is'), ('generation.',
'This', 'is', 'used'), ('This', 'is', 'used', 'in'), ('is', 'used', 'in', 'Data'), ('used', 'in', 'Data',
'Science'), ('in', 'Data', 'Science', 'Lab')]
Process finished with exit code 0
```

PROGRAM 19

AIM: Implement a program to scrap the web page of any popular website – suggested python package is scrappy (ensure ethical conduct).

PROGRAM

```
#Program to implement a simple web crawler and scrapping web pages.
import requests
from bs4 import BeautifulSoup
URL = "https://realpython.github.io/fake-jobs/"
page = requests.get(URL)
print(page.text)
soup = BeautifulSoup(page.content, "html.parser")
results = soup.find(id="ResultsContainer")
job_elements = results.find_all("div", class_="card-content")
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())
  print()
```

OUTPUT

```
Senior Python Developer
Payne, Roberts and Davis
Stewartbury, AA

Energy engineer
Vasquez-Davidson
Christopherville, AA

Legal executive
Jackson, Chambers and Levy
Port Ericaburgh, AA

Fitness centre manager
Savage-Bradley
East Seanview, AP
```