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) Create a 3x3 matrix with values ranging from 2 to 10.
   Program

```
import numpy as np
a1=np.arange(2,11).reshape(3,3)
print(a1)
```

   Output

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

2) Write a program to change the dimension of an array into a 3 X 3 array and convert this NumPy array into a list.
   Program

```
import numpy as np
arr=np.arange(1,10)
a=arr.reshape(3,3)
a2=a.tolist()
print(a2)
```

   Output

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

3) Print the square root of numbers in the list.
   Program

```
l1=[4,16,9,1,25]
arr=np.array(l1)
l2=np.sqrt(arr)
print(l2)
```

   Output

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

4) Create a null NumPy array of size 10 and update the sixth value to 11.

Program

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

Output

```
[ 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

```
l1=[i for i in range(0,50) if i%5==0]
print(l1)
```

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

```
l2=[1.23, 23.32, 300, 16.37]
arr=np.array(l2)
print(type(arr))
```

Output

```
<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([[11,12,13],[14,15,16],[17,18,19]])
print("Addition:")
print(np.add(a1,a2))
print("Subtraction:")
print(np.subtract(a1,a2))
print("Multiplication:")
print(np.multiply(a1,a2))
print("Division:")
print(np.divide(a1,a2))
print("Transpose of a1:")
print(a1.T)
print("Transpose of a2:")
print(a2.T)
print("Sum of diagonal of a1:")
print(sum(np.diag(a1)))
print("Sum of diagonal of a2:")
print(sum(np.diag(a2)))
X=np.array([[1,2,3],[4,5,6],[7,8,9]])
print("Display each element of the matrix to different
powers:\n",np.power(X,[[2,4,5],[1,2,4],[5,2,3]]))
print("Identity matrix\n",np.identity(3,dtype=int))
```

Output

```
Addition:
[[12 14 16]
 [18 20 22]
 [24 26 28]]
Subtraction:
[[-10 -10 -10]
 [-10 -10 -10]
 [-10 -10 -10]]
Multiplication:
[[ 11  24  39]
 [ 56  75  96]
```

```
 [119 144 171]]
Division:
[[0.09090909 0.16666667 0.23076923]
 [0.28571429 0.33333333 0.375     ]
 [0.41176471 0.44444444 0.47368421]]
Transpose of a1:
[[1 4 7]
 [2 5 8]
 [3 6 9]]
Transpose of a2:
[[11 14 17]
 [12 15 18]
 [13 16 19]]
Sum of diagonal of a1:
15
Sum of diagonal of a2:
45
Display each element of the matrix to different powers:
 [[    1    16  243]
 [    4    25  1296]
 [16807    64  729]]
Identity matrix
 [[1 0 0]
 [0 1 0]
 [0 0 1]]
```

Program 3

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

Program

```
import numpy as np
a=np.random.randint(9,size=(3,3))
print("Random matrix: \n",a)
print("Inverse:")
print(np.linalg.inv(a))
print("Rank:",np.linalg.matrix_rank(a))
print("Determinant:",np.linalg.det(a))
v,u=np.linalg.eig(a)
print("Eigen  Value:\n",v)
print("Eigen vector:\n",u)
```

Output

```
Random matrix:
 [[6 5 2]
 [5 3 1]
 [8 0 3]]
Inverse:
[[-0.31034483  0.51724138  0.03448276]
 [ 0.24137931 -0.06896552 -0.13793103]
 [ 0.82758621 -1.37931034  0.24137931]]
Rank: 3
Determinant: -28.99999999999999
Eigen Value:
 [11.42792375 -1.33243696  1.90451322]
Eigen vector:
 [[-0.64558921 -0.44994433 -0.13046829]
 [-0.45571791  0.32750236 -0.27424154]
 [-0.61280973  0.83083832  0.95276944]]
```

Program 4

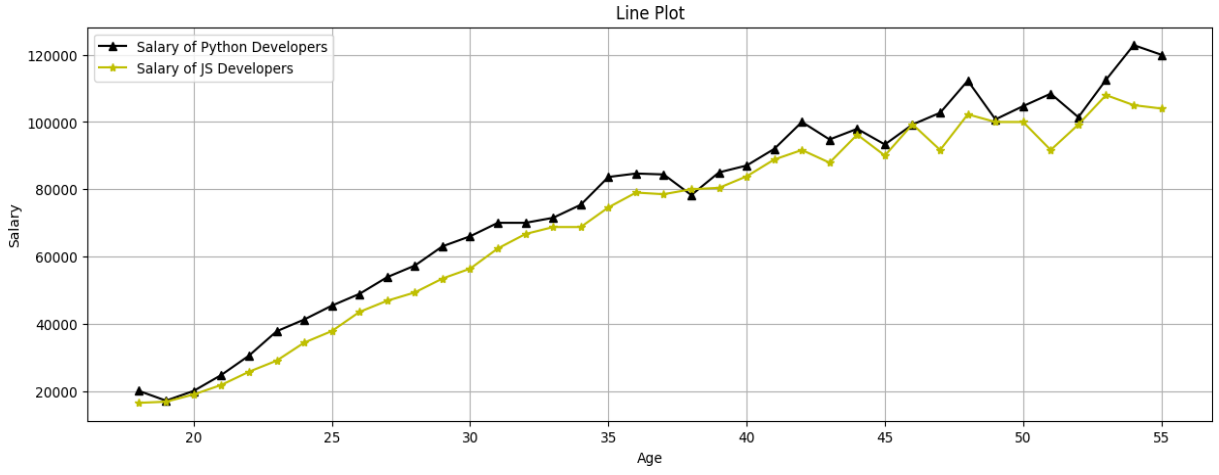
Aim: Program to create customized line plot for comparing the Age-wise annual salary variations for Python developer with JavaScript developer from the dataset of the average annual salary of developers of various programming languages.
Dataset:https://raw.githubusercontent.com/CoreyMSchafer/code_snippets/master/Python/Matplotlib/10-Subplots/data.csv

Program

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv("https://raw.githubusercontent.com/CoreyMSchafer/code_snippets/master/Python/Matplotlib/10-Subplots/data.csv")
df.head()
plt.figure(figsize=(15,5))
plt.plot(df["Age"],df["Python"],"k-^",label="Salary of Python Developers")
plt.plot(df["Age"],df["JavaScript"],"y--*",label="Salary of JS Developers")
plt.legend()
plt.title("Line Plot")
plt.xlabel("Age")
plt.ylabel("Salary")
plt.grid()
plt.show()
```

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 5

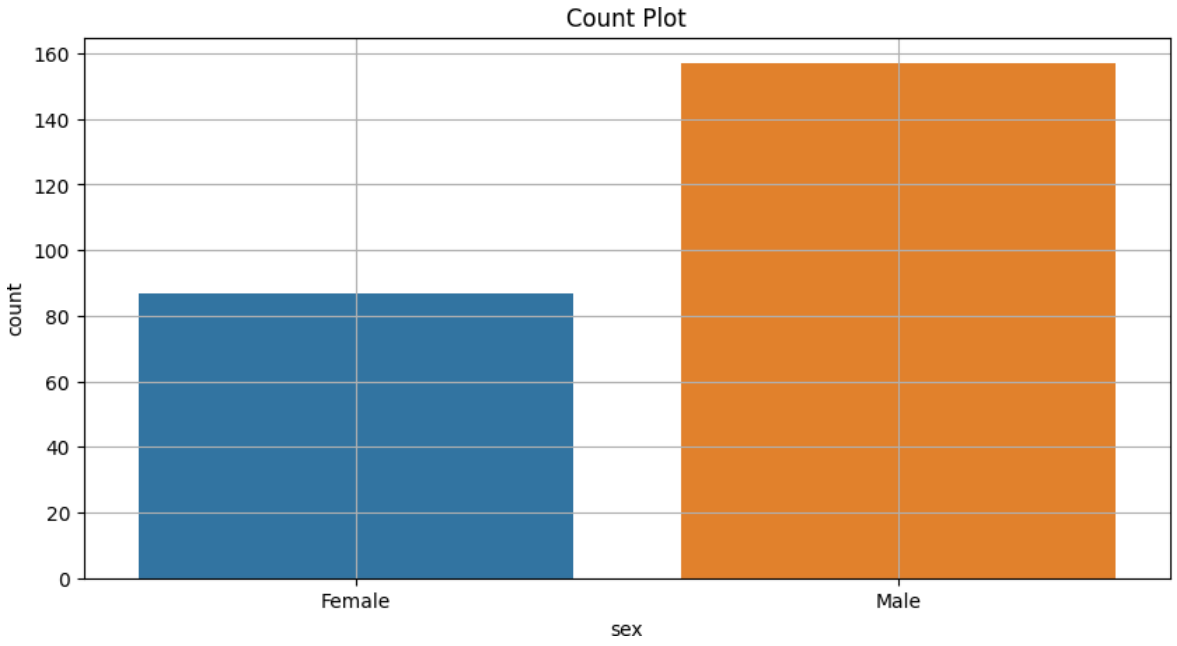
Aim: Program to create a gender wise count plot by using the values in the sex column dataset of tips taken on the total bill amount in restaurants.

Program

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
tip_df=pd.read_csv("https://raw.githubusercontent.com/jiss-github123/tips/main/tips.csv")
tip_df.head()
plt.figure(figsize=(10,5))
plt.title("Count Plot")
sb.countplot(x='sex',data=tip_df,hue="sex")
plt.grid()
plt.show()
```

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



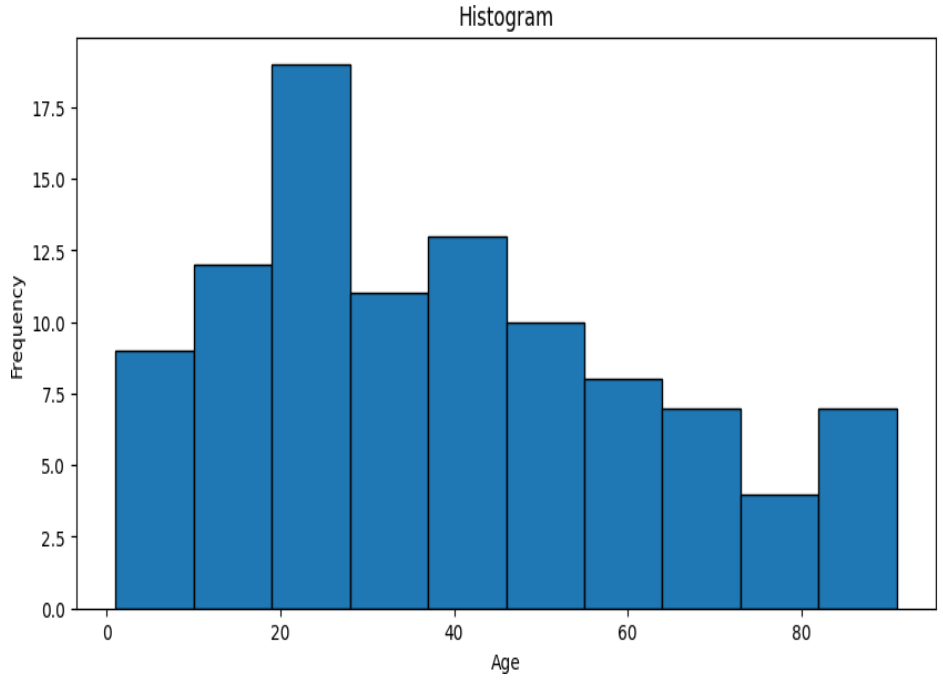
Program 6

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

Program

```
import matplotlib.pyplot as plt
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
    ]
plt.figure(figsize=(10,5))
plt.title("Histogram")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.hist(age_list,bins=10,ec='black')
plt.show()
```

Output



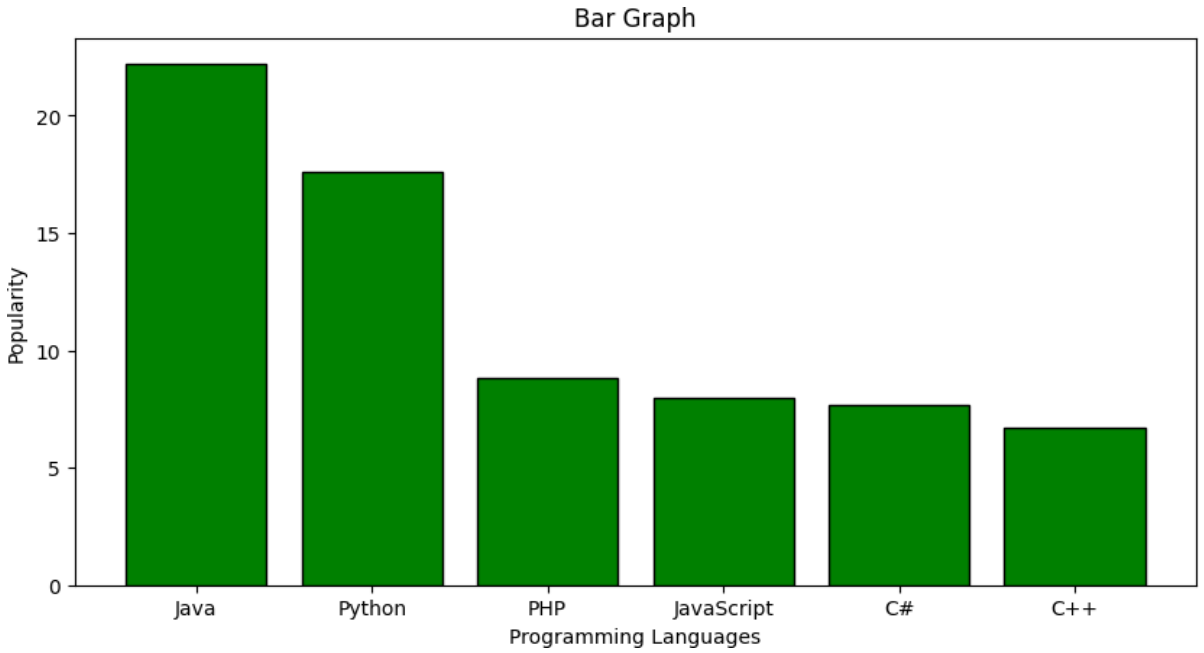
Program 7

Aim: Program to create a bar chart of the popularity of programming languages.
Programming languages: Java Python PHP JavaScript C# C++
Popularity : 22.2 17.6 8.8 8 7.7 6.7

Program

```
plt.figure(figsize=(10,5))
lang=np.array(['Java','Python','PHP','JavaScript','C#','C++'])
popu=np.array([22.2, 17.6, 8.8, 8, 7.7, 6.7])
plt.title("Bar Graph")
plt.xlabel("Programming Languages")
plt.ylabel("Popularity")
plt.bar(lang,popu,color='Green',edgecolor='black')
plt.show()
```

Output



Program 8

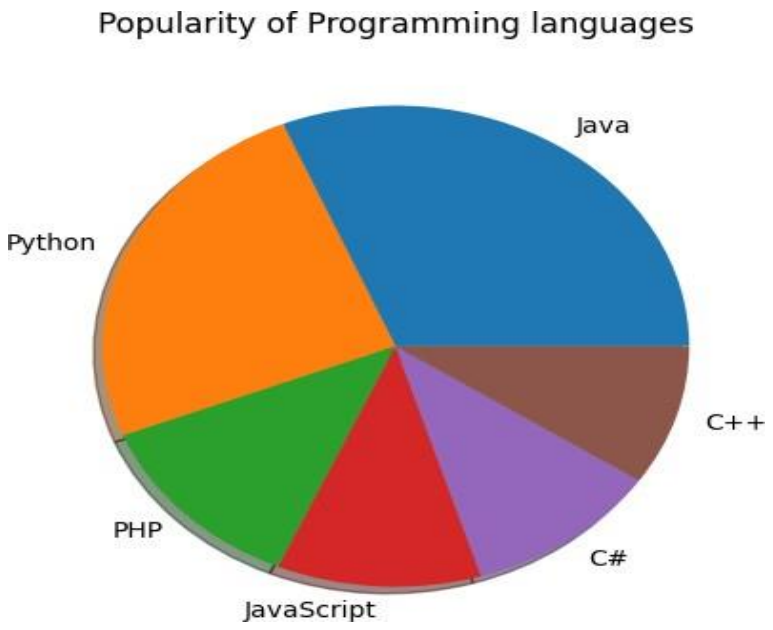
Aim: Program to create a pie chart of the popularity of programming languages.

Programming languages:	Java	Python	PHP	JavaScript	C#	C++
Popularity :	22.2	17.6	8.8	8	7.7	6.7

Program

```
plt.figure(figsize=(20,60))
lang=np.array(['Java','Python','PHP','JavaScript','C#','C++'])
popu=np.array([22.2,17.6,8.8,8,7.7,6.7])
plt.figure(figsize=(10,5))
plt.title("Popularity of Programming languages")
plt.pie(popu,labels=lang,shadow=True)
plt.show()
```

Output



Program 9

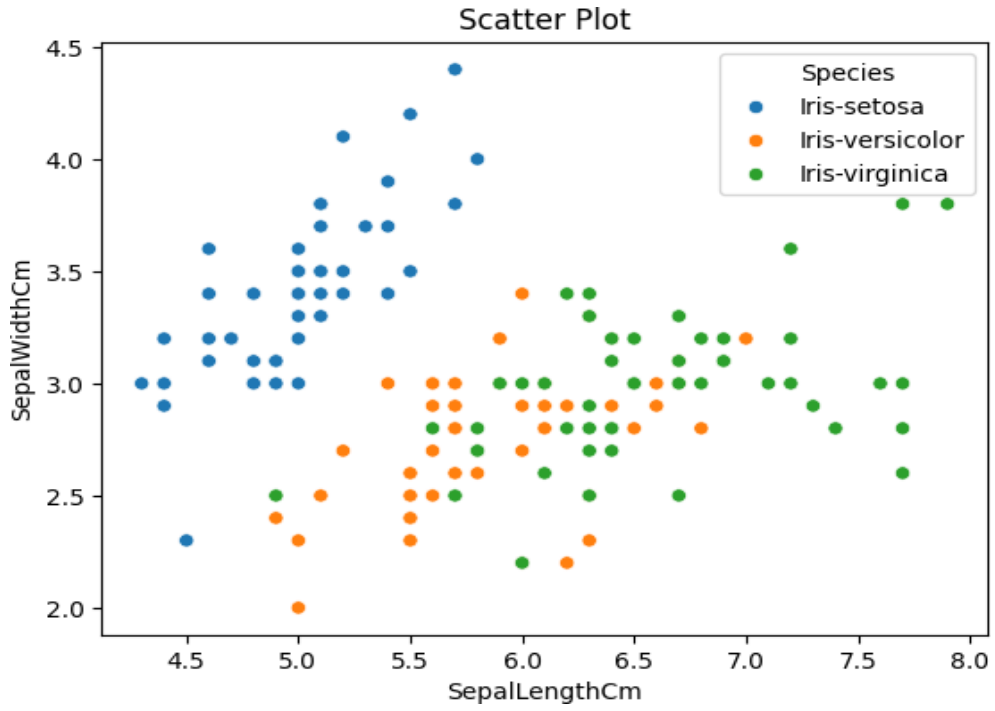
Aim: Program to create a scatter plot between the 'Sepal Length' & 'Sepal Width' columns & Differentiate between the data points of different classes.

Program

```
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
Iris_df=pd.read_csv("https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/main/Iris.csv")
Iris_df.head()
plt.title("Scatter Plot")
sn.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



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.
Problem Statement:
Nowadays, social media advertising is one of the popular forms of advertising. Advertisers can utilise user's demographic information and target their ads accordingly. You are given a dataset having the following attributes:

Field	Description
UserID	Unique ID
Gender	Male or Female
Age	Age of a person
EstimatedSalary	Salary of a person
Purchased	'0' or '1'. '0' means not purchased and '1' means purchased.

Dataset: https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/refs/heads/main/Social_Network_Ads.csv

Activity 1: Import Modules and Read Data

Program

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
ad_df=pd.read_csv('https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/refs/heads/main/Social_Network_Ads.csv')
ad_df.head()
ad_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

	0
User ID	0
Gender	0
Age	0
EstimatedSalary	0
Purchased	0

dtype: int64

Activity 2: Perform Train-Test Split

Program

```
features_df=ad_df.drop(['Purchased'],axis=1)
target_df=ad_df['Purchased']
features_df=pd.get_dummies(features_df)
features_df.head()
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(features_df,target_df,test_size=0.3,
random_state=42)
print("\nShapes of the train and test datasets:")
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Output

	User ID	Age	EstimatedSalary	Gender_Female	Gender_Male
0	15624510	19	19000	False	True
1	15810944	35	20000	False	True
2	15668575	26	43000	True	False
3	15603246	27	57000	True	False
4	15804002	19	76000	False	True

Shapes of the train and test datasets:
(280, 4)
(120, 4)
(280,)
(120,)

Activity 3: Build kNN Classifier Model

Program

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
kml=KNeighborsClassifier(n_neighbors=3)
kml.fit(x_train,y_train)
y_pred=kml.predict(x_test)
a=accuracy_score(y_test,y_pred)*100
print("Accuracy of the model on the test data: ",a,"%")
train_accuracy = kml.score(x_train, y_train)
print("Accuracy of the model on the training data: ", train_accuracy)
test_accuracy = kml.score(x_test, y_test)
print("Accuracy of the model on the test data: ", test_accuracy)from sklearn.metrics import classification_report
print("Classification Report:\n",classification_report(y_test,y_pred))
```

Output

Accuracy of the model on the test data: 72.5 %
Accuracy of the model on the training data: 0.8571428571428571
Accuracy of the model on the test data: 0.725
Classification Report:

	precision	recall	f1-score	support
0	0.73	0.86	0.79	73
1	0.71	0.51	0.59	47
accuracy			0.72	120
macro avg	0.72	0.69	0.69	120
weighted avg	0.72	0.72	0.71	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.

Program

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
df=pd.read_csv('https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/main/Iris.csv')
df.head()
features=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
x=df[features]
y=df['Species']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.7,random_state=47,stratify=y)
print("\nShapes of the train and test datasets:")
print("x_train:",x_train.shape)
print("x_test:",x_test.shape)
print("y_train:",y_train.shape)
print("y_test:",y_test.shape)
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
classifier=GaussianNB()
classifier.fit(x_train,y_train)
y_pred = classifier.predict(x_test)
p=classifier.predict([[4.9,3,1.4,0.2]])
print("Predicted species:",p)
print("Accuracy of the model:",accuracy_score(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
print("Classification Report:\n",classification_report(y_test,y_pred))
```

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

Shapes of the train and test datasets:
x_train: (45, 4)
x_test: (105, 4)
y_train: (45,)
y_test: (105,)
Predicted species: ['Iris-setosa']
Accuracy of the model:0.95
Confusion Matrix:
[[40 0 0]
[0 38 2]
[0 4 36]]
Classification Report:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	40
Iris-versicolor	0.90	0.95	0.93	40
Iris-virginica	0.95	0.90	0.92	40
accuracy			0.95	120
macro avg	0.95	0.95	0.95	120
weighted avg	0.95	0.95	0.95	120

Program 12

Aim: Program to implement linear regression algorithm using any standard dataset available in the public domain and evaluate its performance.
Problem Statement:
As an owner of a startup, you wish to forecast the sales of your product to plan how much money should be spent on advertisements. This is because the sale of a product is usually proportional to the money spent on advertisements. Predict the impact of TV advertising on your product sales by performing simple linear regression analysis.
Dataset : advertisement.csv

Activity 1: Analysing the Dataset

Program

```
import numpy as np
import pandas as pd
df=pd.read_csv("advertising.csv")
df.head()
```

Output

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

Activity 2: Train-Test Split

Program

```
from sklearn.model_selection import train_test_split
X = df['TV']
y = df['Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Output

```
(140,)
(60,)
(140,)
(60,)
```

Activity 3: Model Training

Program

```
from sklearn.linear_model import LinearRegression
X_train_reshape = X_train.values.reshape(-1, 1)
X_test_reshape = X_test.values.reshape(-1, 1)
y_train_reshape = y_train.values.reshape(-1, 1)
y_test_reshape = y_test.values.reshape(-1, 1)
print(X_train_reshape.shape)
print(X_test_reshape.shape)
print(y_train_reshape.shape)
print(y_test_reshape.shape)
li_reg = LinearRegression()
li_reg.fit(X_train_reshape, y_train_reshape)
print("Intercept(c):",li_reg.intercept_)
print("Slope(m):",li_reg.coef_)
```

Output

```
(140, 1)
(60, 1)
(140, 1)
(60, 1)
Intercept(c): [7.20655455]
Slope(m): [[0.05483488]]
```

Activity 4: Model Prediction

Program

```
def sales_predicted(tv_budget):
    return li_reg.predict([[tv_budget]])[0][0]
predicted_sales = sales_predicted(50) * 1000
print("Predicted Sales for $50,000 TV advertising:",predicted_sales)
```

Output

```
Predicted Sales for $50,000 TV advertising: 9948.298737937223
```

Program 13

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

Problem Statement:

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..

Dataset : https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/main/house-prices%20-%20house-prices.csv

Activity 1: Analysing the Dataset

Program

```
import numpy as np
import pandas as pd
df=pd.read_csv("https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/main/house-prices%20-%20house-prices.csv")
df.head()
```

Output

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished

Activity 2: Data Preparation

Program

```
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)
df.head()
```

Output

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	1	0	0	0	1	2	1	2
1	12250000	8960	4	4	4	1	0	0	0	1	3	0	2
2	12250000	9960	3	2	2	1	0	1	0	0	2	1	1
3	12215000	7500	4	2	2	1	0	1	0	1	3	1	2
4	11410000	7420	4	1	2	1	1	1	0	1	2	0	2

Activity 3: Train-Test Split

Program

```
from sklearn.model_selection import train_test_split
X=df.drop(columns="price")
y=df["price"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Output

```
(365, 12)
(180, 12)
(365,)
(180,)
```

Activity 4: Model Training

Program

```
from sklearn.linear_model import LinearRegression
y_train_reshaped=y_train.values.reshape(-1,1)
y_test_reshaped=y_test.values.reshape(-1,1)
lin_reg=LinearRegression()
lin_reg.fit(X_train,y_train_reshaped)
print("Intercept:", lin_reg.intercept_)
print("Coefficients:", lin_reg.coef_)
print("\n")
features = X_train.columns
for feature, coef in zip(features,lin_reg.coef_.flatten()):
print("feature:",coef)
```

Output

```
Intercept: [-276654.3971631]
Coefficients: [[2.51340200e+02 9.27166053e+04 1.12647938e+06 3.96248428e+05
  4.10635156e+05 3.20496711e+05 4.84622279e+05 6.23047393e+05
  6.78375342e+05 2.92410463e+05 5.24417243e+05 2.00615357e+05]]

feature: 251.34019992678213
feature: 92716.60526930424
feature: 1126479.3774358043
```

feature: 396248.42774732463
feature: 410635.1556971093
feature: 320496.7112104647
feature: 484622.2788531302
feature: 623047.392903681
feature: 678375.3422621795
feature: 292410.4631406697
feature: 524417.2428236584
feature: 200615.35703557188

Activity 5: Model Prediction and Evaluation

Program

```
from sklearn.metrics import r2_score,mean_squared_error
y_train_pred=lin_reg.predict(X_train)
y_test_pred=lin_reg.predict(X_test)
print("r2_score of test:",r2_score(y_test_reshaped,y_test_pred))
print("r2_score of train:",r2_score(y_train_reshaped,y_train_pred))
print("mean_squared_error of test:",mean_squared_error(y_test_reshaped,y_test_pred))
print("mean_squared_error of
train:",mean_squared_error(y_train_reshaped,y_train_pred))
print("mean_absolute_error of
test:",np.sqrt(mean_squared_error(y_test_reshaped,y_test_pred)))
print("mean_absolute_error of
train:",np.sqrt(mean_squared_error(y_train_reshaped,y_train_pred)))
print("mean_absolute_error of
train:",np.sqrt(mean_squared_error(y_train_reshaped,y_train_pred)))
```

Output

r2_score of test: 0.655707070748526
r2_score of train: 0.68603602364727
mean_squared_error of test: 1475542475754.55
mean_squared_error of train: 971946527815.6641
mean_absolute_error of test: 1214719.0933522657
mean_absolute_error of train: 985873.4846904364
mean_absolute_error of train: 985873.4846904364

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 text classification using Support vector machine using any standard dataset available in the public domain and evaluate its performance.
Dataset : https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/main/Iris.csv

Activity 1: Analysing the Dataset

Program

```
import numpy as np
import pandas as pd
df=pd.read_csv("https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/main/Iris.csv")
df.head()
```

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

Activity 2: Train-Test Split

Program

```
from sklearn.model_selection import train_test_split
X=df.drop(['Id','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)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

Output

```
(120, 4)
(30, 4)
(120,)
(30,)
```

Activity 3: Model Training

Program

```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train, y_train)
```

Output

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

Activity 4: Model Prediction and Evaluation

Program

```
from sklearn.metrics import accuracy_score, classification_reportOutput
y_pred = svm_classifier.predict(X_test)
print("Accuracy of the model:",accuracy_score(y_test,y_pred))
print("Classification Report:\n",classification_report(y_test,y_pred))
```

Output

Accuracy of the model: 0.9666666666666667

Classification Report:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolor	1.00	0.89	0.94	9
Iris-virginica	0.92	1.00	0.96	11
accuracy			0.97	30
macro avg	0.97	0.96	0.97	30
weighted avg	0.97	0.97	0.97	30

Program 15

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

Short notes

Decision tree is a type of supervised learning algorithm (having a predefined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables.

Dataset: https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/refs/heads/main/Iris.csv

Activity 1: Analysing the Dataset

Program

```
import pandas as pd
import numpy as np
df=pd.read_csv("https://raw.githubusercontent.com/Akhilas11/DS-ML-Lab/refs/heads/main/Iris.csv")
df.head()
```

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

Activity 2: Train-Test Split

Program

```
from sklearn.model_selection import train_test_split
X=df.drop(["Id","Species"],axis=1)
y=df["Species"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=15)
print("Shape of the train and test dataset:")
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
```

print(y_test.shape)

Output

Shape of the train and test dataset:
(112, 4)
(38, 4)
(112,)
(38,)

Activity 3: Model Training

Program

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion="entropy", min_samples_split=50)
dtc.fit(X_train,y_train)
```

Activity 4: Model Prediction and Evaluation

Program

```
y_pred=dtc.predict(X_test)
accuracy_score = dtc.score(X_test,y_test)
print(accuracy_score)
```

Output

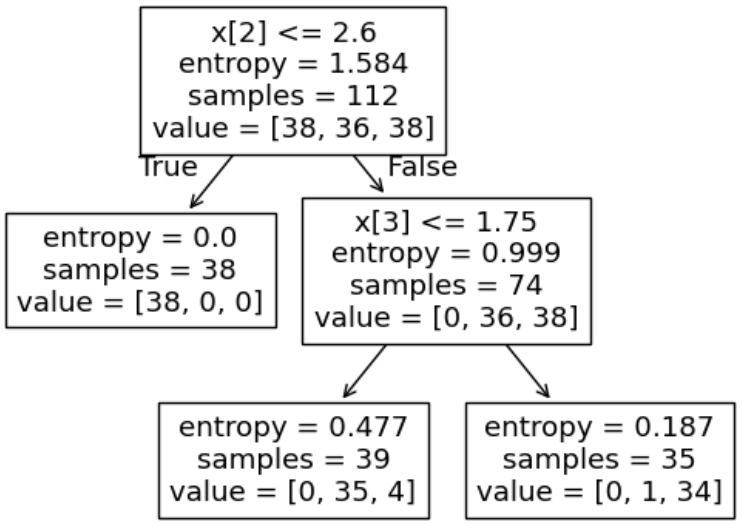
0.9736842105263158

Activity 5: Visualise the decision tree

Program

```
from sklearn import tree
from matplotlib import pyplot as plt
tree.plot_tree(dtc)
plt.show()
```

Output



Program 16

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

Activity 1: Analysing the Dataset

Program

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
df = pd.read_csv('https://raw.githubusercontent.com/jiss-sngce/CO_3/main/jkcars.csv')
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

Activity 2: Data Cleaning

Program

```
print(df.isnull().sum())
```

Output

```
Car       0
Model     0
Volume    0
Weight    0
CO2       0
dtype: int64
```

Activity 3: Find Optimal value of K

Program

```
data_3d = df[['Volume', 'Weight', 'CO2']]
data_3d.head()
sil_scores = []
K_values = range(2, 11)
for K in K_values:
    kmeans = KMeans(n_clusters=K, random_state=10)
    labels = kmeans.fit_predict(data_3d)
    sil_score = silhouette_score(data_3d, labels)
    sil_scores.append(sil_score)
silhouette_df = pd.DataFrame({'K': K_values, 'Silhouette Score': sil_scores})
max_sil_score = silhouette_df['Silhouette Score'].max()
best_k = silhouette_df.loc[silhouette_df['Silhouette Score'].idxmax(), 'K']
print("Maximum Silhouette Score:",max_sil_score,"Optimal K:" ,best_k)
```

Output

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

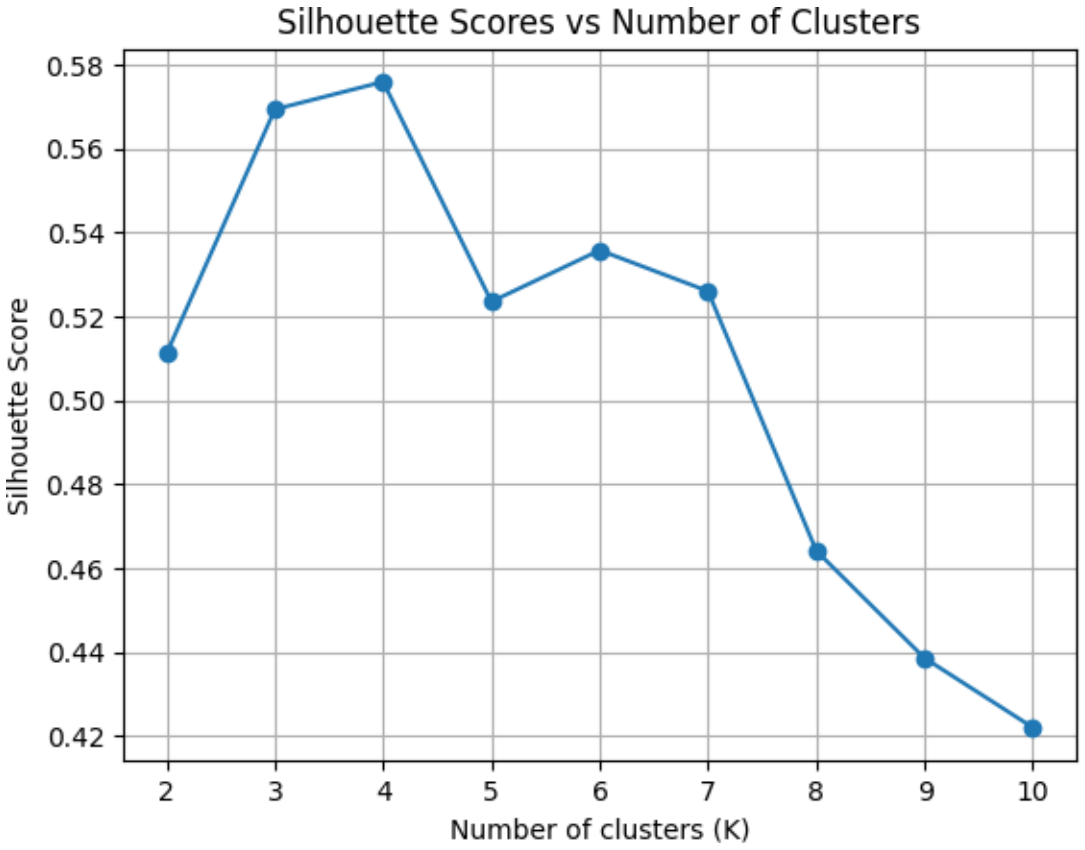
Maximum Silhouette Score: 0.5758722401446237 Optimal K: 4

Activity 4: Plot silhouette Scores find optimal value for K

Program

```
plt.plot(K_values, sil_scores, marker='o')
plt.xlabel('Number of clusters (K)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Scores vs Number of Clusters')
plt.grid(True)
plt.show()
kmeans = KMeans(n_clusters=best_k, random_state=10)
cluster_labels  =  kmeans.fit_predict(data_3d)
df['Cluster'] = cluster_labels
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

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.

Activity 1: Analysing the Dataset

Program

```
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  y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

Activity 2: Define and Compile the CNN model

Program

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

Activity 3: Train the model

Program

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

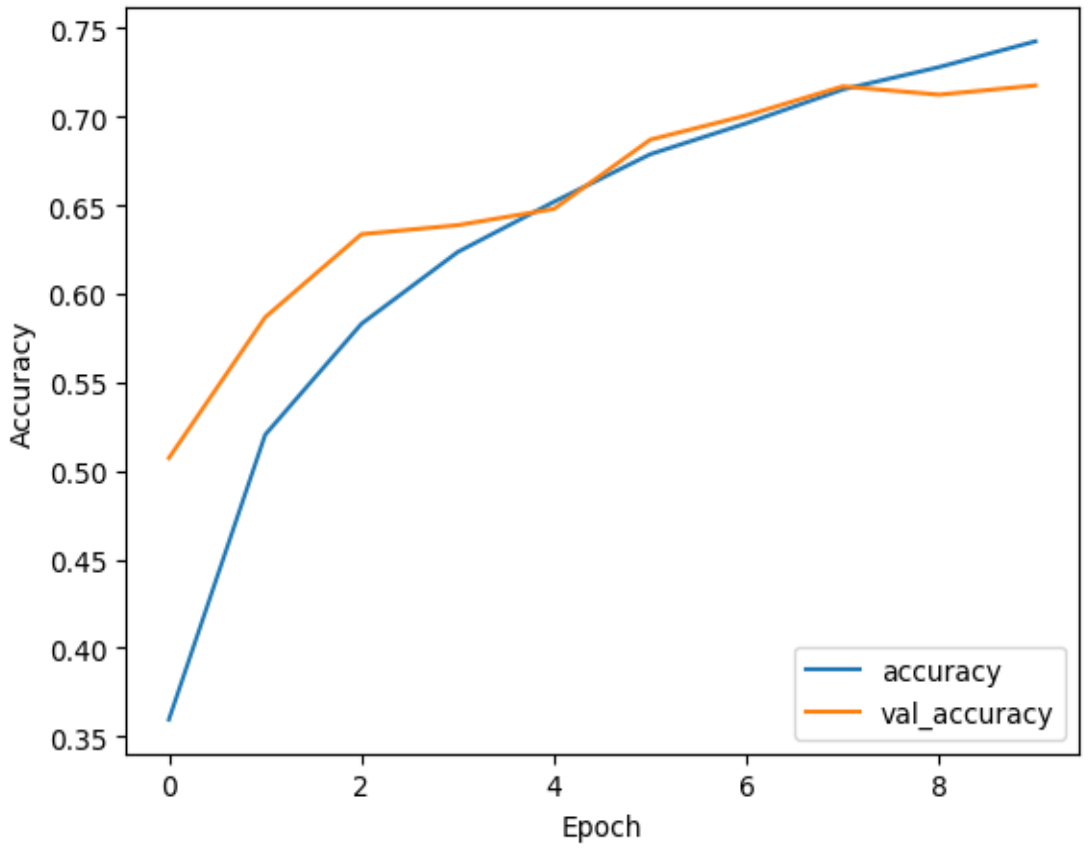
Activity 4: Evaluate the model on the test data

Program

```
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

Test accuracy: 0.7177000045776367



Activity 5: Load and preprocess the image

Program

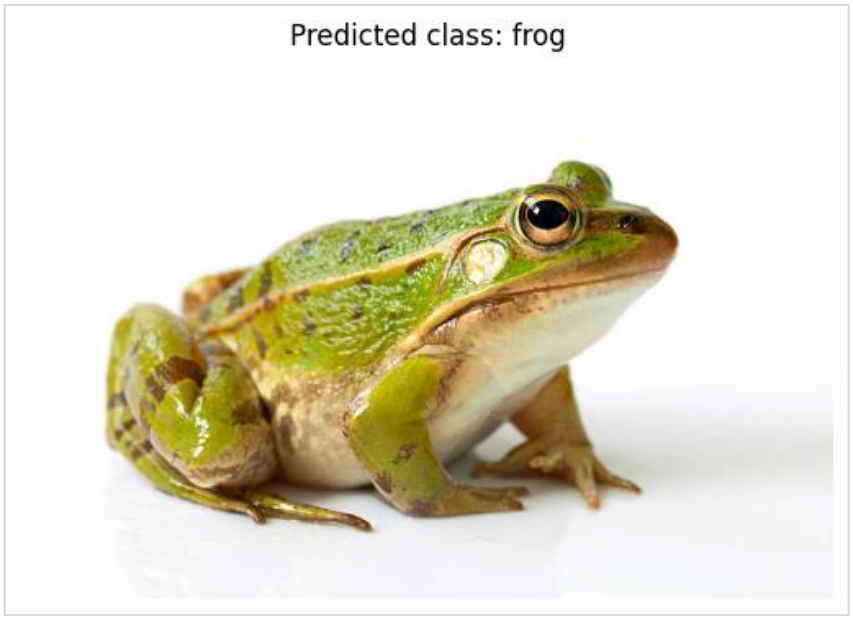
```
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt
image_path = '/content/image1.jpg'
img = load_img(image_path, target_size=(32, 32))
img_array = img_to_array(img)
img_array = img_array / 255.0
img_array = np.expand_dims(img_array, axis=0)
```

Activity 6: Make a prediction

Program

```
predictions = model.predict(img_array)
predicted_class = np.argmax(predictions, axis=1)
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
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



Predicted class: frog

COURSE OUTCOME 5

Implement programs for web data mining and natural language processing using NLTK

Program 18

Aim: Implement natural language processing using NLTK.
For given text: "The data set given satisfies the requirement for model generation.This is used in Data Science Lab."

1) Perform word and sentence tokenization.

Program

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk import pos_tag
from nltk.util import ngrams
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
text = "The data set given satisfies the requirement for model generation. This is used in Data Science Lab."
sentence = sent_tokenize(text)
print("Sentence Tokenization:")
print(sentence)
word = word_tokenize(text)
print("Word Tokenization:")
print(word)
```

Output

```
Sentence Tokenization:
['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', '.']
```

2) Remove the stop words from the given text

Program

```
stop = set(stopwords.words('english'))
```

```
rev_words = [w for w in word if w not in stop]
print("Remove Stop Words:")
print(rev_words)
```

Output

```
Remove Stop Words:
['The', 'data', 'set', 'given', 'satisfies', 'requirement', 'model', 'generation', '.', 'This', 'used',
'Data', 'Science', 'Lab', '.']
```

3) Perform Part of Speech tagging

Program

```
pos_tags = pos_tag(word)
print("Perform Part of Speech Tagging:")
print(pos_tags)
```

Output

```
Perform 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'), ('.','.')]
```

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

Program

```
tgrams = list(ngrams(word, 2))
print("(n=2):")
print(tgrams)
fgrams = list(ngrams(word, 4))
print("(n=4):")
print(fgrams)
```

Output

```
(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'), ('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', '.'), ('model', 'generation', '.',
'This'), ('generation', '.', 'This', 'is'), ('.', 'This', 'is','used'), ('This', 'is', 'used', 'in'), ('is', 'used', 'in', 'Data'),
('used', 'in', 'Data', 'Science'), ('in','Data', 'Science', 'Lab'), ('Data', 'Science', 'Lab', '.')]
```

Program 19

Aim: Implement a program to scrap the web page of any popular website.

Program

```
import requests
from bs4 import BeautifulSoup
URL = "https://realpython.github.io/fake-jobs/"
page = requests.get(URL)
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

Product manager
Ramirez Inc
North Jamieview, AP
```

Medical technical officer
Rogers-Yates
Davidville, AP

Physiological scientist
Kramer-Klein
South Christopher, AE

Textile designer
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