Module 1

PROGRAM 3:

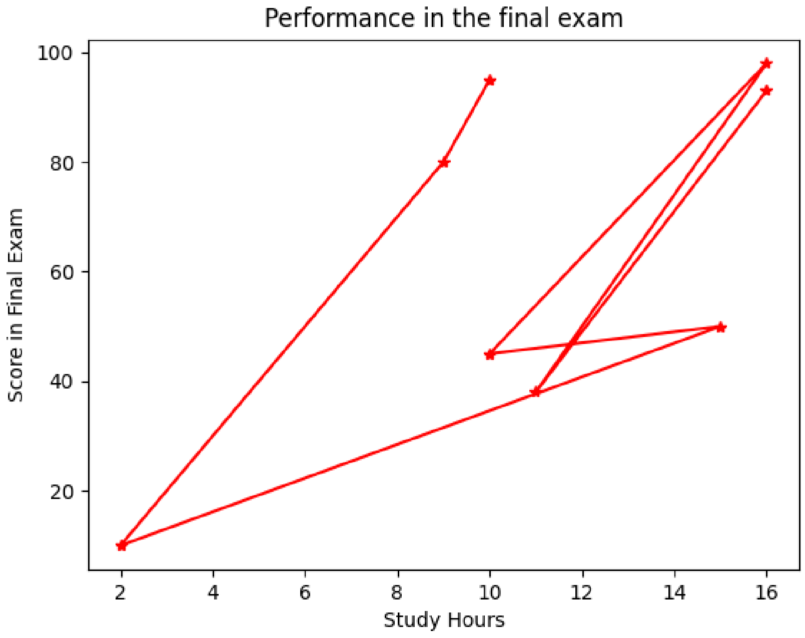
A study was conducted to understand the effect of the number of hours the students spent studying on their performance in the final exams. Write a code to plot a line chart with the number of hours spent studying on x-axis and score in final exam on y-axis. Use a red ‘*’ as the point character, label the axes and give the plot a title.

Number of hrs spent studying (x)	10	9	2	15	10	16	11	16
Score in the final exam (0 – 100) (y)	95	80	10	50	45	98	38	93

PROGRAM:

```
from matplotlib import pyplot as plt
num_of_hrs=[10,9,2,15,10,16,11,16]
score=[95,80,10,50,45,98,38,93]
plt.plot(num_of_hrs, score, color='red', marker='*', linestyle='solid')
plt.title("Performance in the final exam")
plt.xlabel("Study Hours")
plt.ylabel("Score in Final Exam")
plt.show()
```

OUTPUT:



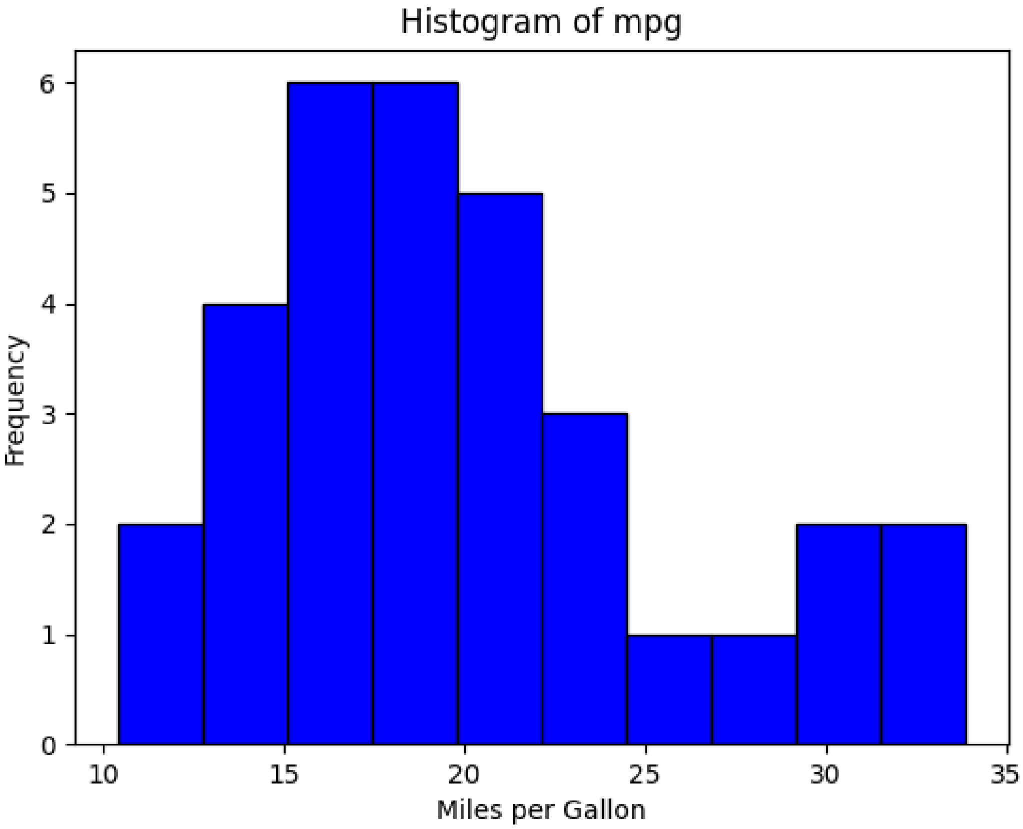
PROGRAM 4:

For the given dataset mtcars.csv (www.kaggle.com/ruiromanini/mtcars), plot a histogram to check the frequency distribution of the variable ‘mpg’ (Miles per gallon)

PROGRAM:

```
from matplotlib import pyplot as plt
import pandas as pd
from collections import Counter
mtcars=pd.read_csv("mtcars.csv")
plt.hist(mtcars['mpg'],bins=10,color='blue',edgecolor='black')
plt.xlabel("Miles per Gallon")
plt.ylabel("Frequency")
plt.title("Histogram of mpg")
plt.show()
```

OUTPUT:



MODULE 2

PROGRAM 5:

Consider the books dataset BL-Flickr-Images-Book.csv from Kaggle (https://www.kaggle.com/adeyoyintemidayo/publication-of-books) which contains information about books. Write a program to demonstrate the following.

- Import the data into a DataFrame
- Find and drop the columns which are irrelevant for the book information.
- Change the Index of the DataFrame
- Tidy up fields in the data such as date of publication with the help of simple regular expression.
- Combine str methods with NumPy to clean columns

Program:

```
import pandas as pd
import numpy as np

df = pd.read_csv('BL-Flickr-Images-Book.csv')

# Display the first few rows of the DataFrame
print("Original DataFrame:")
print(df.head())

# Find and drop the columns which are irrelevant for the book information
irrelevant_columns = ['Edition Statement', 'Corporate Author', 'Corporate Contributors',
'Former owner', 'Engraver', 'Contributors', 'Issuance type', 'Shelfmarks']
df.drop(columns=irrelevant_columns, inplace=True)

# Change the Index of the DataFrame
df.set_index('Identifier', inplace=True)

# Tidy up fields in the data such as date of publication with the help of simple regular
expression
df['Date of Publication'] = df['Date of Publication'].str.extract(r'^(\d{4})', expand=False)

# Combine str methods with NumPy to clean columns
df['Place of Publication'] = np.where(df['Place of Publication'].str.contains('London'), 'London',
df['Place of Publication'].str.replace('-', ' '))

# Display the cleaned DataFrame
print("\nCleaned DataFrame:")
print(df.head())
```

Output:

```
Original DataFrame:
   Identifier             Edition Statement      Place of Publication  \
0         206                           NaN                    London
1         216                           NaN  London; Virtue & Yorston
2         218                           NaN                    London
3         472                           NaN                    London
4         480  A new edition, revised, etc.                    London

  Date of Publication              Publisher  \
0         1879 [1878]       S. Tinsley & Co.
1                1868           Virtue & Co.
2                1869  Bradbury, Evans & Co.
3                1851          James Darling
4                1857   Wertheim & Macintosh

                                               Title     Author  \
0                  Walter Forbes. [A novel.] By A. A      A. A.
1  All for Greed. [A novel. The dedication signed...  A., A. A.
2  Love the Avenger. By the author of “All for Gr...  A., A. A.
3  Welsh Sketches, chiefly ecclesiastical, to the...  A., E. S.
4  [The World in which I live, and my place in it...  A., E. S.

                                   Contributors  Corporate Author  \
0                               FORBES, Walter.               NaN
1  BLAZE DE BURY, Marie Pauline Rose - Baroness               NaN
2  BLAZE DE BURY, Marie Pauline Rose - Baroness               NaN
3                     Appleyard, Ernest Silvanus.             NaN
4                            BROOME, John Henry.              NaN

  Corporate Contributors Former owner  Engraver Issuance type  \
0                    NaN          NaN       NaN   monographic
1                    NaN          NaN       NaN   monographic
2                    NaN          NaN       NaN   monographic
3                    NaN          NaN       NaN   monographic
4                    NaN          NaN       NaN   monographic

                                          Flickr URL  \
0  http://www.flickr.com/photos/britishlibrary/ta...
1  http://www.flickr.com/photos/britishlibrary/ta...
2  http://www.flickr.com/photos/britishlibrary/ta...
3  http://www.flickr.com/photos/britishlibrary/ta...
4  http://www.flickr.com/photos/britishlibrary/ta...

                            Shelfmarks
0    British Library HMNTS 12641.b.30.
1    British Library HMNTS 12626.cc.2.
2    British Library HMNTS 12625.dd.1.
3  British Library HMNTS 10369.bbb.15.
```

```
4     British Library HMNTS 9007.d.28.

Cleaned DataFrame:
           Place of Publication   Date of Publication              Publisher  \
Identifier
206                      London                  1879       S. Tinsley & Co.
216                      London                  1868           Virtue & Co.
218                      London                  1869  Bradbury, Evans & Co.
472                      London                  1851          James Darling
480                      London                  1857   Wertheim & Macintosh

                                                        Title     Author  \
Identifier
206                           Walter Forbes. [A novel.] By A. A      A. A.
216         All for Greed. [A novel. The dedication signed...  A., A. A.
218         Love the Avenger. By the author of “All for Gr...  A., A. A.
472         Welsh Sketches, chiefly ecclesiastical, to the...  A., E. S.
480         [The World in which I live, and my place in it...  A., E. S.

                                                   Flickr URL
Identifier
206         http://www.flickr.com/photos/britishlibrary/ta...
216         http://www.flickr.com/photos/britishlibrary/ta...
218         http://www.flickr.com/photos/britishlibrary/ta...
472         http://www.flickr.com/photos/britishlibrary/ta...
480         http://www.flickr.com/photos/britishlibrary/ta...
```

MODULE 3

PROGRAM 6:

Train a regularized logistic regression classifier on the iris dataset (https://archive.ics.uci.edu/ml/machine-learning-databases/iris/ or the inbuilt iris dataset) using sk-learn. Train the model with the following hyper parameter C = 1e4 and report the best classification accuracy.

Program:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a pipeline with StandardScaler and LogisticRegression with regularization
pipeline = make_pipeline(StandardScaler(), LogisticRegression(C=1e4, max_iter=1000))

# Train the model
pipeline.fit(X_train, y_train)

# Calculate the accuracy on the testing set
accuracy = pipeline.score(X_test, y_test)
print("Classification accuracy:", accuracy)
```

Output:

```
Classification accuracy: 1.0
```

PROGRAM 7:

Train an SVM classifier on the iris dataset using sk-learn. Try different kernels and the associated hyper parameters. Train model with the following set of hyperparameters RBFkernel, gamma=0.5, one-vs-rest classifier, no-feature-normalization. Also try C=0.01,1,10C=0.01,1,10. For the above set of hyper parameters, find the best classification accuracy along with total number of support vectors on the test data.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Set of hyperparameters to try
hyperparameters = [
   {'kernel': 'rbf', 'gamma': 0.5, 'C': 0.01},
   {'kernel': 'rbf', 'gamma': 0.5, 'C': 1},
   {'kernel': 'rbf', 'gamma': 0.5, 'C': 10}
]

best_accuracy = 0
best_model = None
best_support_vectors = None

# Train SVM models with different hyperparameters and find the best accuracy
for params in hyperparameters:
   model = SVC(kernel=params['kernel'], gamma=params['gamma'], C=params['C'],
decision_function_shape='ovr')
   model.fit(X_train, y_train)
   accuracy = model.score(X_test, y_test)
   support_vectors = model.n_support_.sum()
   print(f"For hyperparameters: {params}, Accuracy: {accuracy}, Total Support Vectors:
{support_vectors}")
   if accuracy > best_accuracy:
      best_accuracy = accuracy
      best_model = model
      best_support_vectors = support_vectors

print("\nBest accuracy:", best_accuracy)
print("Total support vectors on test data:", best_support_vectors)
```

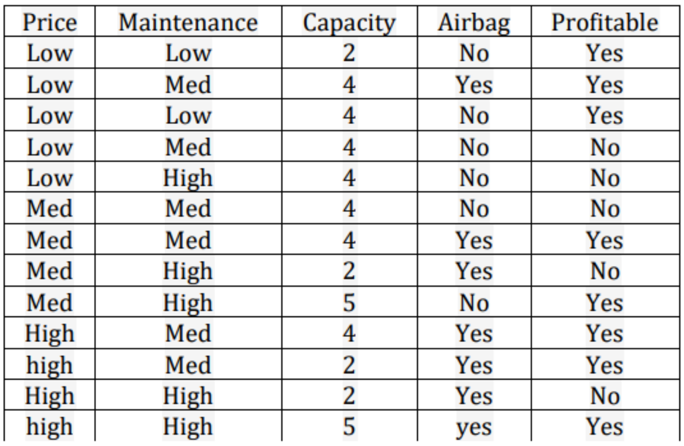
Output:

```
For hyperparameters: {'kernel': 'rbf', 'gamma': 0.5, 'C': 0.01}, Accuracy: 0.3, Total
Support Vectors: 120
For hyperparameters: {'kernel': 'rbf', 'gamma': 0.5, 'C': 1}, Accuracy: 1.0, Total
Support Vectors: 39
For hyperparameters: {'kernel': 'rbf', 'gamma': 0.5, 'C': 10}, Accuracy: 1.0, Total
Support Vectors: 31

Best accuracy: 1.0
Total support vectors on test data: 39
```

MODULE 4

PROGRAM 8

Consider the following dataset. Write a program to demonstrate the working of the decision tree based ID3 algorithm.



Price	Maintenance	Capacity	Airbag	Profitable
Low	Low	2	No	Yes
Low	Med	4	Yes	Yes
Low	Low	4	No	Yes
Low	Med	4	No	No
Low	High	4	No	No
Med	Med	4	No	No
Med	Med	4	Yes	Yes
Med	High	2	Yes	No
Med	High	5	No	Yes
High	Med	4	Yes	Yes
high	Med	2	Yes	Yes
High	High	2	Yes	No
high	High	5	yes	Yes

Program:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Define the dataset
data = {
   'Price': ['Low', 'Low', 'Low', 'Low', 'Low', 'Med', 'Med', 'Med', 'Med', 'High', 'High', 'High', 'High'],
   'Maintenance': ['Low', 'Med', 'Low', 'Med', 'High', 'Med', 'Med', 'High', 'High', 'Med', 'Med', 'High', 'High'],
   'Capacity': ['2', '4', '4', '4', '4', '4', '4', '2', '5', '4', '2', '2', '5'],
   'Airbag': ['No', 'Yes', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes'],
   'Profitable': [1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1]
}

# Convert the dictionary to a pandas DataFrame
df = pd.DataFrame(data)

# Convert categorical variables into numerical ones using one-hot encoding
df = pd.get_dummies(df, columns=['Price', 'Maintenance', 'Airbag'])

# Separate features (X) and target variable (y)
X = df.drop('Profitable', axis=1)
y = df['Profitable']

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a decision tree classifier with 'entropy' criterion
```

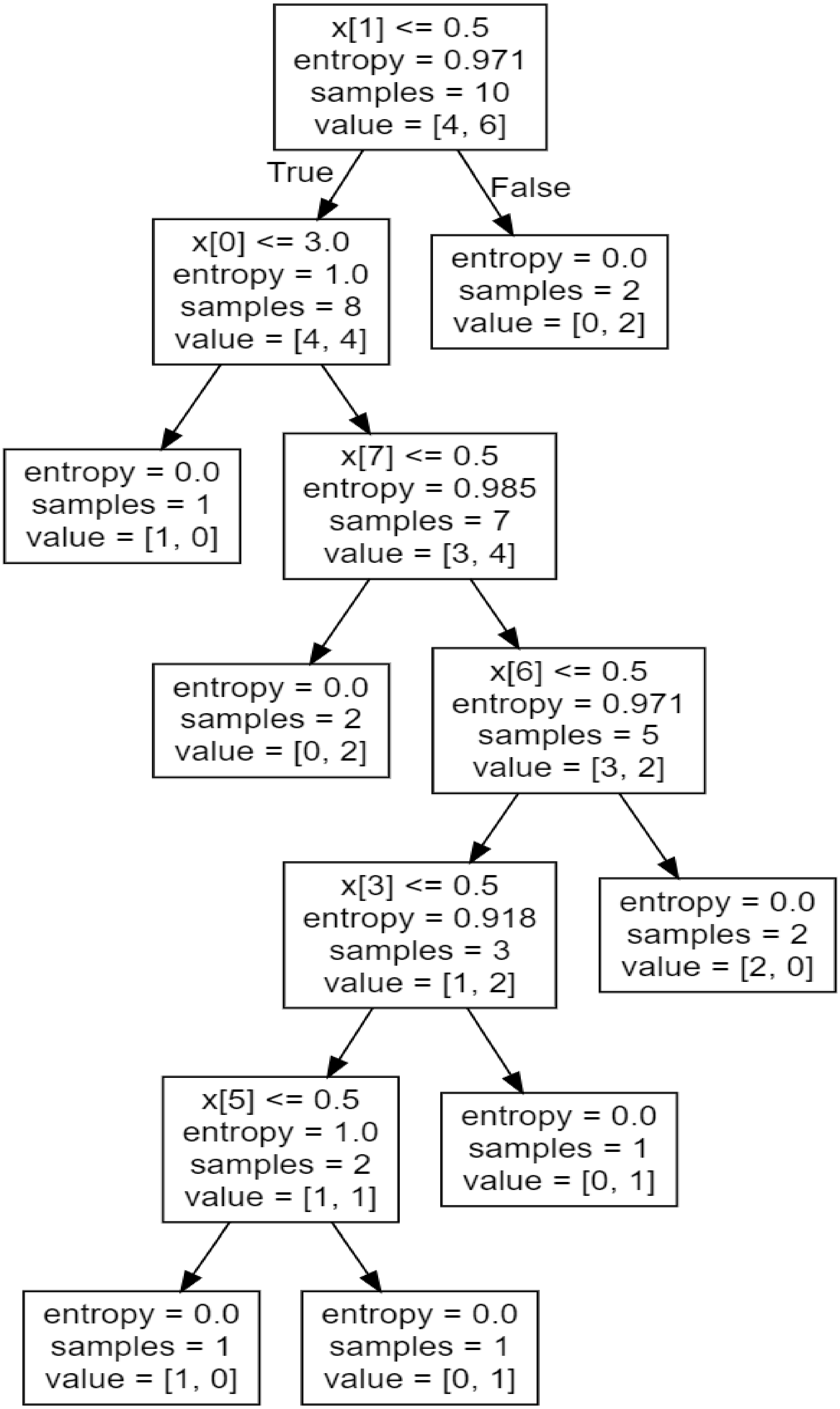
```
clf = DecisionTreeClassifier(criterion='entropy')

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Predict on the testing data
y_pred = clf.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Output:

```
Accuracy: 1.0
```

Graphviz : **https://dreampuf.github.io/GraphvizOnline**

```
digraph Tree {
node [shape=box, fontname="helvetica"] ;
edge [fontname="helvetica"] ;
0 [label="x[1] <= 0.5\nentropy = 0.971\nsamples = 10\nvalue = [4, 6]"] ;
1 [label="x[0] <= 3.0\nentropy = 1.0\nsamples = 8\nvalue = [4, 4]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="entropy = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
1 -> 2 ;
3 [label="x[7] <= 0.5\nentropy = 0.985\nsamples = 7\nvalue = [3, 4]"] ;
1 -> 3 ;
4 [label="entropy = 0.0\nsamples = 2\nvalue = [0, 2]"] ;
3 -> 4 ;
5 [label="x[6] <= 0.5\nentropy = 0.971\nsamples = 5\nvalue = [3, 2]"] ;
3 -> 5 ;
6 [label="x[3] <= 0.5\nentropy = 0.918\nsamples = 3\nvalue = [1, 2]"] ;
5 -> 6 ;
7 [label="x[5] <= 0.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]"] ;
6 -> 7 ;
8 [label="entropy = 0.0\nsamples = 1\nvalue = [1, 0]"] ;
7 -> 8 ;
9 [label="entropy = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
7 -> 9 ;
10 [label="entropy = 0.0\nsamples = 1\nvalue = [0, 1]"] ;
6 -> 10 ;
11 [label="entropy = 0.0\nsamples = 2\nvalue = [2, 0]"] ;
5 -> 11 ;
12 [label="entropy = 0.0\nsamples = 2\nvalue = [0, 2]"] ;
0 -> 12 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
}
```


x[1] <= 0.5
entropy = 0.971
samples = 10
value = [4, 6]
True
False
x[0] <= 3.0
entropy = 1.0
samples = 8
value = [4, 4]
entropy = 0.0
samples = 2
value = [0, 2]
entropy = 0.0
samples = 1
value = [1, 0]
x[7] <= 0.5
entropy = 0.985
samples = 7
value = [3, 4]
entropy = 0.0
samples = 2
value = [0, 2]
x[6] <= 0.5
entropy = 0.971
samples = 5
value = [3, 2]
x[3] <= 0.5
entropy = 0.918
samples = 3
value = [1, 2]
entropy = 0.0
samples = 2
value = [2, 0]
x[5] <= 0.5
entropy = 1.0
samples = 2
value = [1, 1]
entropy = 0.0
samples = 1
value = [0, 1]
entropy = 0.0
samples = 1
value = [1, 0]
entropy = 0.0
samples = 1
value = [0, 1]

PROGRAM 9:

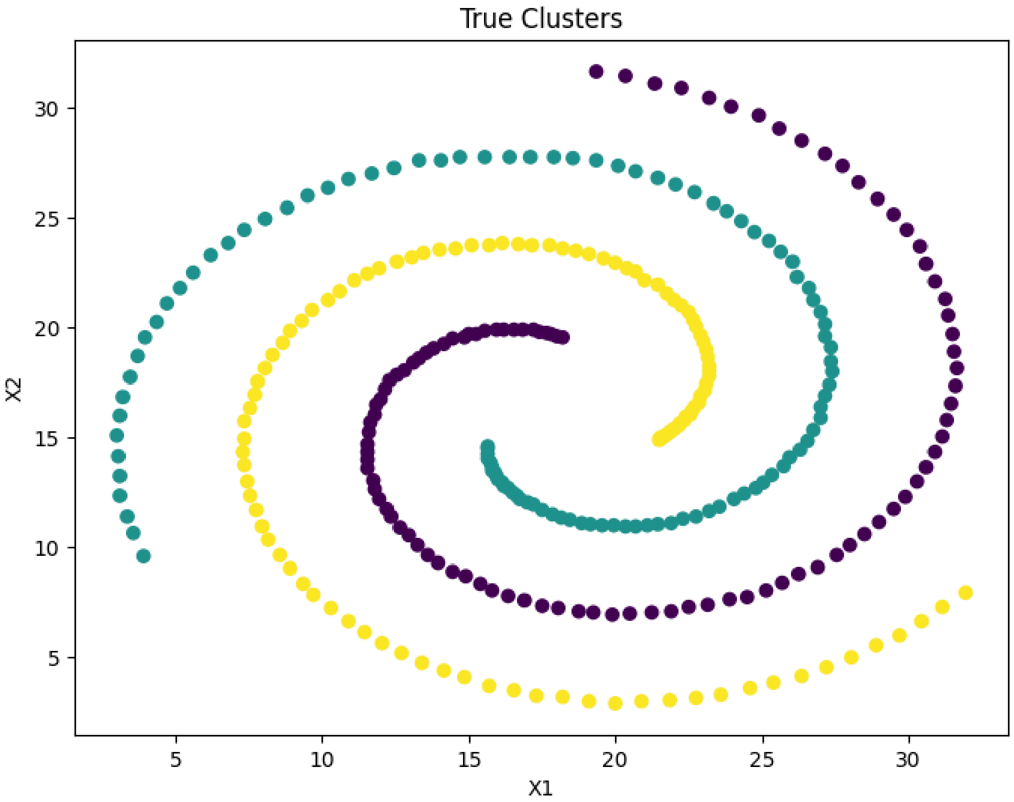
Consider the dataset spiral.txt (https://bit.ly/2Lm75Ly). The first two columns in the dataset correspond to the coordinates of each data point. The third column corresponds to the actual cluster label. Compute the rand index for the following methods.

- K – means Clustering
- Single – link Hierarchical Clustering
- Complete link hierarchical clustering.
- Also visualize the dataset and which algorithm will be able to recover the true clusters

Program:

```
import numpy as np
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import adjusted_rand_score
import matplotlib.pyplot as plt

# Load the dataset
data = np.loadtxt("Spiral.txt", delimiter="\t", skiprows=1)
X = data[:, :2]  # Features
y_true = data[:, 2]  # Actual cluster labels

# Visualize the dataset
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=y_true, cmap='viridis')
plt.title('True Clusters')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()

# K-means clustering
# kmeans = KMeans(n_clusters=3, random_state=42)
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
kmeans_clusters = kmeans.fit_predict(X)

# Single-link Hierarchical Clustering
single_link = AgglomerativeClustering(n_clusters=3, linkage='single')
single_link_clusters = single_link.fit_predict(X)

# Complete-link Hierarchical Clustering
complete_link = AgglomerativeClustering(n_clusters=3, linkage='complete')
complete_link_clusters = complete_link.fit_predict(X)

# Compute the Rand Index
rand_index_kmeans = adjusted_rand_score(y_true, kmeans_clusters)
rand_index_single_link = adjusted_rand_score(y_true, single_link_clusters)
rand_index_complete_link = adjusted_rand_score(y_true, complete_link_clusters)
```

```
print("Rand Index for K-means Clustering:", rand_index_kmeans)
print("Rand Index for Single-link Hierarchical Clustering:", rand_index_single_link)
print("Rand Index for Complete-link Hierarchical Clustering:",
rand_index_complete_link)

# This code will compute the Rand Index for each clustering method and provide a
visualization of the true clusters.
# The Rand Index ranges from 0 to 1, where 1 indicates perfect clustering agreement
with the true clusters.
# The method with a higher Rand Index is better at recovering the true clusters.
```



PROGRAM 10:

Mini Project – Simple web scraping in social media