

RNS FIRST GRADE COLLEGE AUTONOMOUS

(Affiliated to Bangalore University and NAAC Accredited with ‘A’ Grade) Dr. Vishnuvardhan Road, Channasandra, R R Nagara, Bengaluru – 560 098

**Department of Computer Science Machine Learning Lab Manual (BCA 6th Sem)**

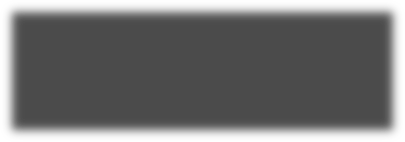
## LIST OF PROGRAMS

|  |  |
| --- | --- |
| **1.** | **Install and set up Python and essential libraries like NumPy and pandas** |
| **2.** | **Introduce scikit-learn as a machine learning library.** |
| **3.** | **Install and set up scikit-learn and other necessary tools.** |
| **4.** | **Write a program to Load and explore the dataset of .CVS and excel files using pandas.** |
| **5.** | **Write a program to Visualize the dataset to gain insights using Matplotlib or Seaborn by plotting scatter plots, bar charts.** |
| **6.** | **Write a program to Handle missing data, encode categorical variables, and perform feature scaling.** |
| **7.** | **Write a program to implement a k-Nearest Neighbours (k-NN) classifier using scikitlearn and Train the classifier on the dataset and evaluate its performance.** |
| **8.** | **Write a program to implement a linear regression model for regression tasks and Train the model on a dataset with continuous target variables.** |
| **9.** | **Write a program to implement a decision tree classifier using scikit-learn and visualize the decision tree and understand its splits** |
| **10.** | **Write a program to Implement K-Means clustering and Visualize clusters.** |

1. **Install and set up Python and essential libraries like NumPy and pandas.**

Installation of Python Step 1: Search for Python

Click on the official website link: <https://www.python.org/downloads/>

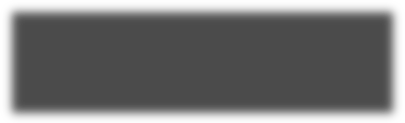


Step 2: Select Version to Install Python

Choose the latest versions for windows.

Step 3: Downloading the Python Installer

* + Once you have downloaded the installer, open the .exe file.
  + Enable users to run Python from the command line by checking the Add python.exe to PATH checkbox
  + Click on **Install Now** to start installation.

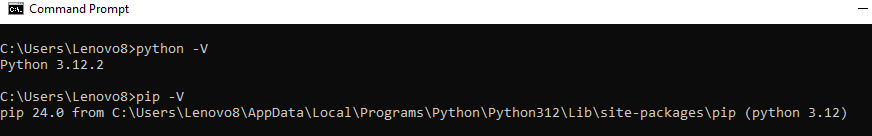
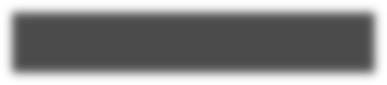


Step 4: Verify the Python Installation in Windows

Go to **Command Prompt**, type the command “**python -V” or “python --version**”. You can see installed version of Python on your system.

Step5: Check the Pip version

Go to **Command Prompt**, type the command “**pip -V” or “pip --version**”. You can see installed version of pip on your system.



Installation of essential packages Numpy and Pandas.

Install numpy and pandas package.

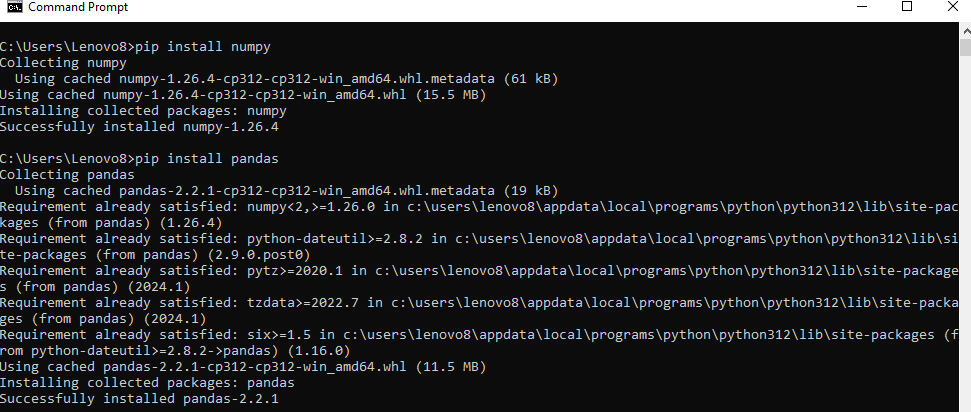
**NumPy** is an open-source Python library that facilitates efficient numerical operations on large quantities of data. There are a few functions that exist in NumPy that we use on pandas DataFrames. The most important part about NumPy is that pandas is built on top of it which means Numpy is required for operating the Pandas.

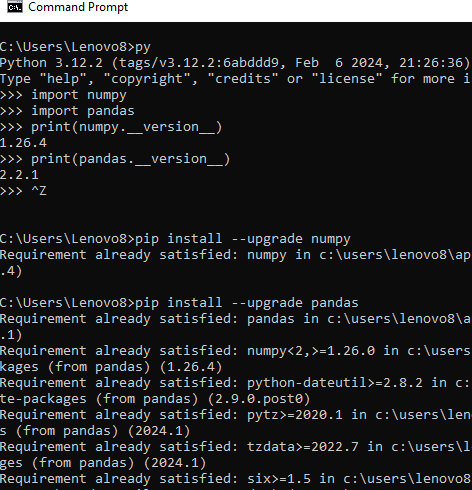
**Pandas** is a very popular library for working with data (its goal is to be the most powerful and flexible open source tool, and in our opinion, it has reached that goal). DataFrames are at the center of pandas. A DataFrame is structured like a table or spreadsheet. The rows and the columns both have indexes, and can perform operations on rows or columns separately.

**Step 1:** Open command prompt, CMD.

**Step 2:** Type the command,

pip install numpy pip install pandas



**Step3:** Print the versions of NumPy and Pandas that were installed.

Go to python script or jupyter notebook and type.

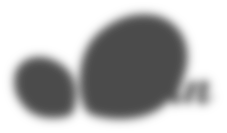
import numpy import pandas

print(numpy. version ) print(pandas. version )

**Step4:** Check for any updates on packages Type the command:

pip install --upgrade numpy pip install --upgrade pandas

# Introduce sci-kit-learn as a machine learning library.



**Scikit-learn (Sklearn)** is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical

modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Features

Rather than focusing on loading, manipulating and summarizing data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows −

**Supervised Learning algorithms** − Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit- learn.

**Unsupervised Learning algorithms** − On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.

**Clustering** − This model is used for grouping unlabeled data.

**Cross Validation** − It is used to check the accuracy of supervised models on unseen data.

# Install and set up scikit-learn and other necessary tools.

**Scikit-learn (Sklearn)** is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python

Install scikit-learn using pip: Open your terminal or command prompt and run the following command:

pip install -U scikit-learn

To verify your installation, you can use the following commands:

python -m pip show scikit-learn

To see which version and where scikit-learn is installed

python -m pip freeze

To see all packages installed

import sklearn import numpy import pandas

print(sklearn. version ) print(numpy. version ) print(pandas. version )

# Write a program to Load and explore the dataset of .CVS and excel files using pandas.

import pandas as pd def load\_data(file):

if file.endswith('.csv'): df = pd.read\_csv(file)

elif file.endswith('.xlsx'): df = pd.read\_excel(file)

else:

print("Unsupported file format. Please provide a CSV or Excel file.") return

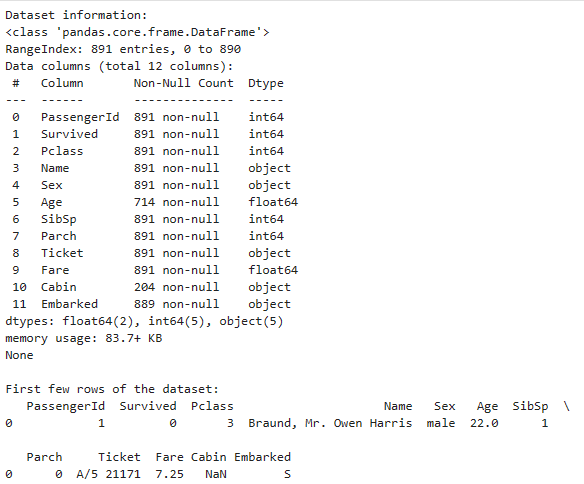
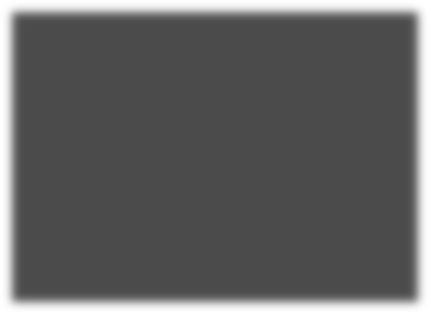
print("Dataset information:") print(df.info())

print("\nTop rows of the dataset:") print(df.head(1))

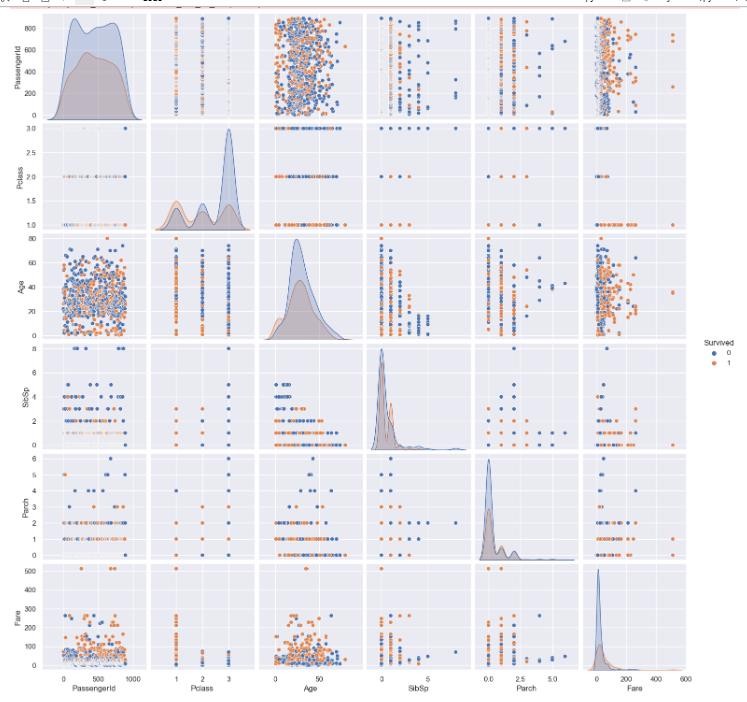
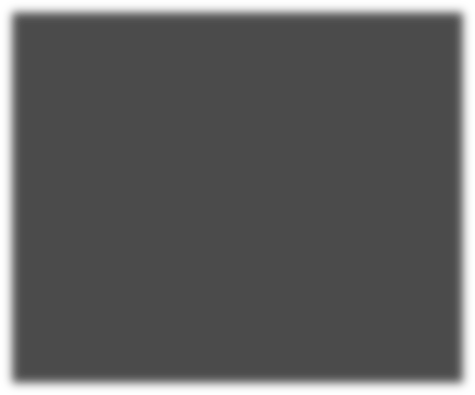
file = 'train.csv'

***# Change this to the path of your CSV or Excel file***

load\_data(file)



# Write a program to Visualize the dataset to gain insights using Matplotlib or Seaborn by plotting scatter plots, and bar charts.



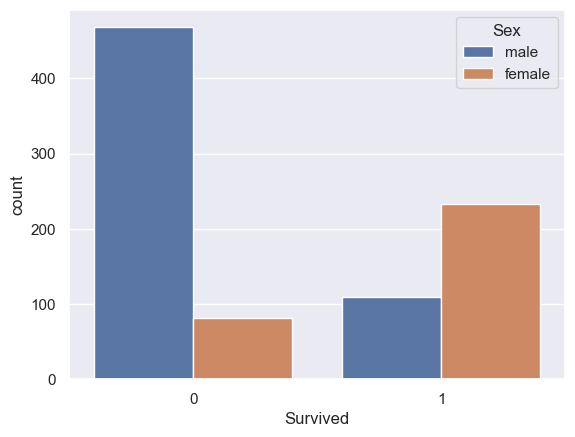
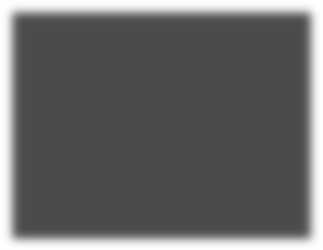
import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

df=pd.read\_csv('train.csv') df.head(2)

#plotting pairchart sns.pairplot(df,hue='Survived') sns.set\_theme(style="darkgrid") plt.show()

sns.countplot(x='Survived',data=df,hue = 'Sex')



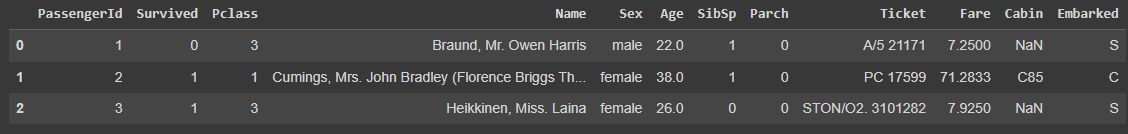
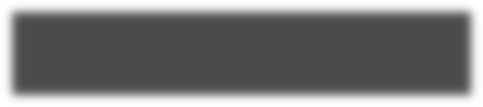
# Write a program to Handle missing data, encode categorical variables, and perform feature scaling.

***#importing necessary libraries***

import numpy as np import pandas as pd

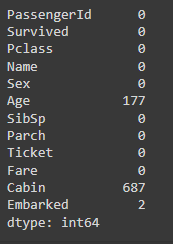
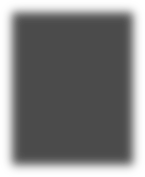
from sklearn.preprocessing import LabelEncoder from sklearn.preprocessing import StandardScaler from sklearn.model\_selection import train\_test\_split

***#Reading the dataset* df=pd.read\_csv(‘train.csv') df.head(3)**



***#HANDLING MISSING VALUES***

df.isnull().sum()



***#Dropping the “Cabin” column as it contains more null values***

df = df.drop(columns='Cabin', axis=1)

***#Replacing the missing values in the “Age” column with the mean value***

df['Age'].fillna(df['Age'].mean(), inplace=True)

***#Replacing the missing values in the “Age” column with the mode value***

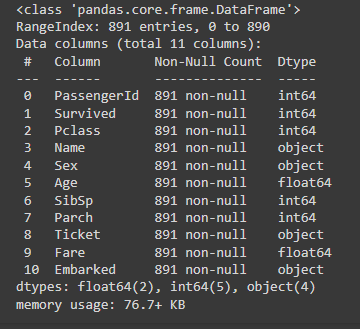
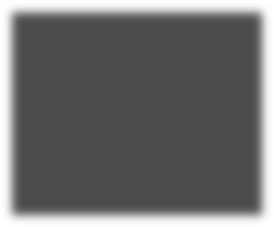
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True) df.isnull().sum().sum()

output:

0

***#ENCODING CATEGORICAL FEATURE***

df.info()



***#Droping unnessary columns***

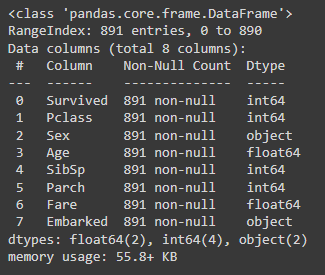
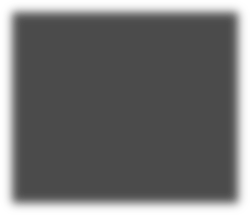
df= df.drop(columns = ['PassengerId','Name','Ticket'],axis=1)

***#Using labelEncoder to impute categorical features***

le=LabelEncoder()

df['Sex']= le.fit\_transform(df['Sex']) df['Embarked']=le.fit\_transform(df['Embarked'])

df.info()

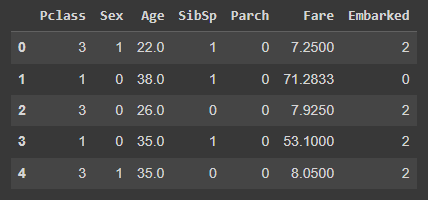
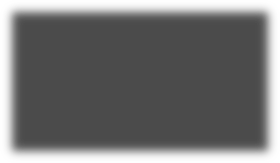


***#FEATURE SCALING***

***#spliting input and output***

X = df.drop(columns = ['Survived'],axis=1) y=df['Survived']

X.head()



***#Train-test split***

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.2, random\_state=42)

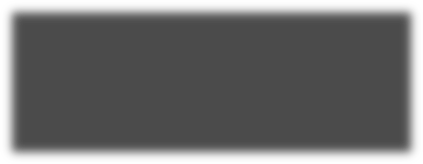
***Using Standarscalar to scale the features***

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

***#Displaying scaled data as dataframes***

scaled\_df = pd.DataFrame(X\_train, columns=X.columns) scaled\_df.head()



# Write a program to implement a k-Nearest Neighbours (k-NN) classifier using scikit-learn and Train the classifier on the dataset and evaluate its performance

***#importing necessary libraries***

import numpy as np import pandas as pd

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

from sklearn.metrics import accuracy\_score, classification\_report

***#importing iris dataset from sklearn and spliting input and output***

from sklearn.datasets import load\_iris iris = load\_iris()

X = iris.data y = iris.target

***#Train-test split***

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

***#Implementing Knn Classifier model***

knn\_model = KNeighborsClassifier(n\_neighbors=3) knn\_model.fit(X\_train, y\_train)

y\_pred = knn\_model.predict(X\_test)

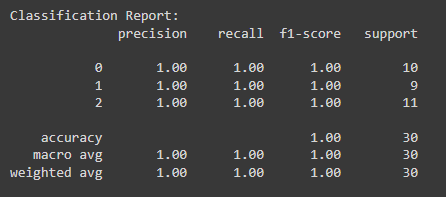
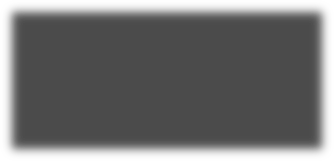
***#Checking performance matrices***

acc = accuracy\_score(y\_test, y\_pred) print("Accuracy:", acc)

Output:

Accuracy: 1.0

print("Classification Report:") print(classification\_report(y\_test, y\_pred))



# Write a program to implement a linear regression model for regression tasks and Train the model on a dataset with continuous target variables.

#importing necessary libraries import numpy as np

import pandas as pd

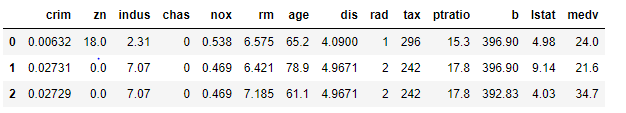
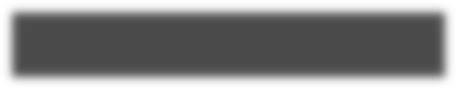
import matplotlib.pyplot as plt import seaborn as sns

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

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

***#Reading dataset* df=pd.read\_csv('Boston.csv') df.head(3)**

***#spliting input and output***



X = df.drop(columns = ['medv'],axis=1) y=df['medv']

***#Train-test split***

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.2, random\_state=42)

|  |  |  |
| --- | --- | --- |
| **print(X\_train.shape)** | **output:** | (404, 13) |
| **print(y\_train.shape)** |  | (404,) |
| **print(X\_test.shape)** |  | (102, 13) |
| **print(y\_test.shape)** |  | (102,) |

***#Performing simple linear regression***

model = LinearRegression()

***#Fitting model***

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

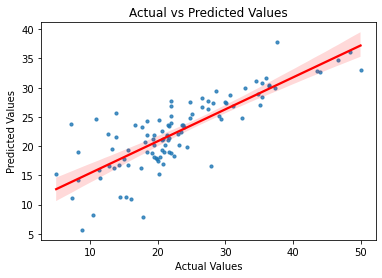
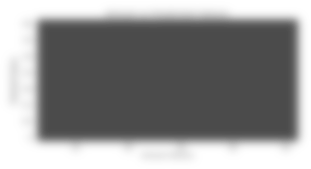
***#Prediction***

y\_pred = model.predict(X\_test)

***#Scatter plot for actual Vs predicted datapoints***

sns.regplot(x=y\_test, y=y\_pred, scatter\_kws={'s': 10}, line\_kws={'color': 'red'}) plt.xlabel('Actual Values')

plt.ylabel('Predicted Values') plt.title('Actual vs Predicted Values') plt.show()



#Calculating error rate via performance metrics

print('Root Mean Squared error:(RMSE)',np.sqrt(mean\_squared\_error(Y\_test,y\_pred))) print('R2-Square:',r2\_score(Y\_test,y\_pred))

Root Mean Squared error:(RMSE) 4.300630200615773 R2-Square: 0.7789207451814409

# Write a program to implement a decision tree classifier using scikit-learn and visualize the decision tree and understand its splits.

***#importing necessary libraries***

import numpy as np import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split from sklearn.tree import DecisionTreeClassifier, plot\_tree from sklearn.metrics import accuracy\_score

***#importing iris dataset from sklearn and spliting input and output***

from sklearn.datasets import load\_iris iris = load\_iris()

X = iris.data y = iris.target

***#Train-test split***

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.2, random\_state=42)

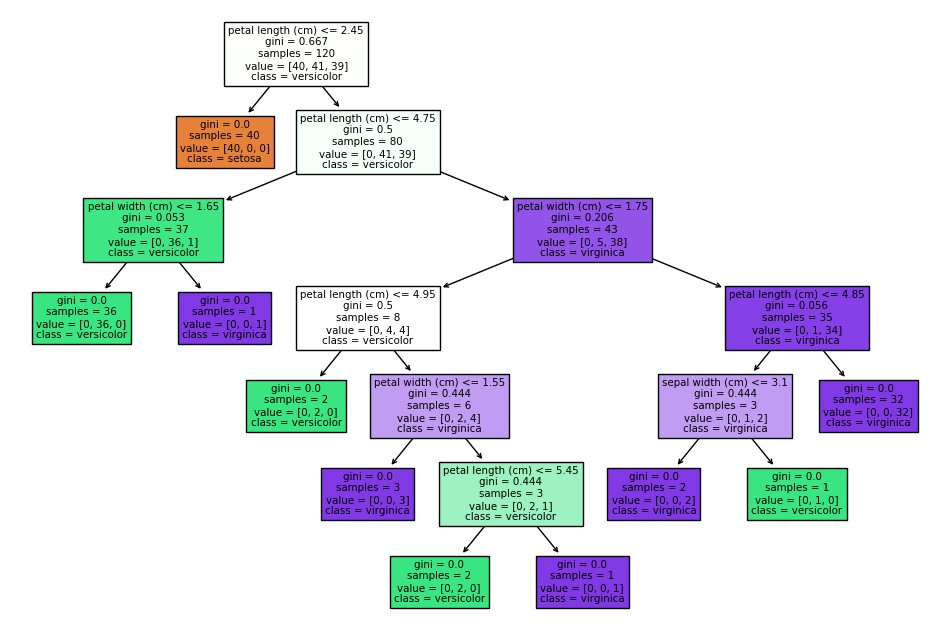
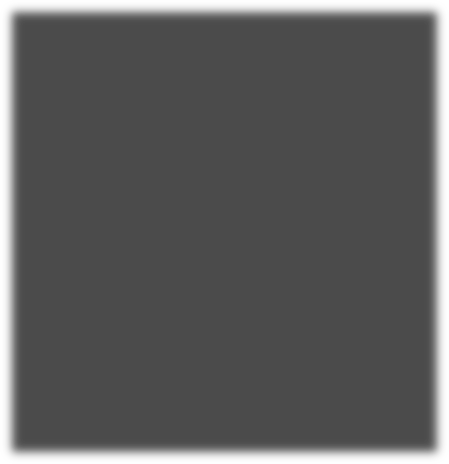
***Performing Decision Tree Classifier* dtc = DecisionTreeClassifier() dtc.fit(X\_train, y\_train) y\_pred=dtc.predict(X\_test)**

***#Checking accuracy* acc=accuracy\_score(y\_test,y\_pred) print("Accuracy of model=", acc)** Output:

Accuracy: 1.0

#Visualizing decision tree plt.figure(figsize=(12, 8))

plot\_tree(dtc, feature\_names=iris.feature\_names, class\_names=iris.target\_names, filled=True) plt.show()



# Write a program to Implement K-Means clustering and Visualize clusters

*make\_blobs* is a synthetic data generator, especially useful for clustering and classification algorithms. It generates isotropic Gaussian blobs. An isotropic Gaussian blob essentially means that the data points are distributed in a circular (spherical, for multi-dimensional data) shape around the centroid.

***#importing necessary libraries***

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs from sklearn.cluster import KMeans

***# Generate sample data***

X, y = make\_blobs(n\_samples=500, centers=4, cluster\_std=0.8, random\_state=42)

***# Create a K-Means clusterer with 4 clusters* kmeans = KMeans(n\_clusters=4, random\_state=42) kmeans.fit(X)**

***# Get cluster labels***

labels= kmeans.predict(X)

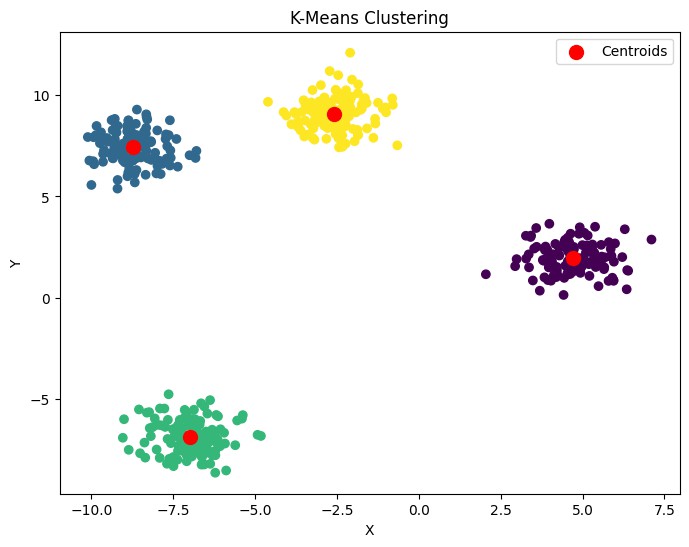
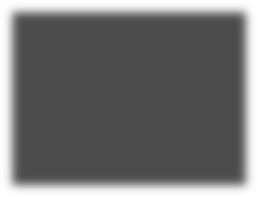
***#Plotting the data with cluster labels***

plt.figure(figsize=(8, 6))

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')

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

plt.title('K-Means Clustering') plt.xlabel('X')



plt.ylabel('Y') plt.legend() plt.show()