#Line plot

Import matplotlib.pyplot as plt

Import seaborn as sns

X = [1, 2, 3, 4, 5]

Y = [10, 20, 25, 30, 40]

Plt.plot(x, y, marker=’o’, linestyle=’—‘, color=’r’, label=”Sales”)

Plt.xlabel(“Days”)

Plt.ylabel(“Revenue”)

Plt.title(“Sales Growth Over Time”)

Plt.legend()

Plt.show()

#Python SQL Database

Import sqlite3

Import mysql.connector

Import psycopg2

Conn = sqlite3.connect(“my\_database.db”)

Cursor = conn.cursor()

Print(“Database connected successfully”)

Cursor.execute(“””

CREATE TABLE IF NOT EXISTS employees (

Id INTEGER PRIMARY KEY AUTOINCREMENT,

Name TEXT NOT NULL,

Age INTEGER,

Department TEXT )

“””)

Conn.commit() # Save changes

Print(“Table created successfully”)

Cursor.execute(“””

INSERT INTO employees (name, age, department)

VALUES (‘Alice’, 30, ‘HR’)

“””)

Conn.commit()

Print(“Data inserted successfully”)

Cursor.execute(“SELECT \* FROM employees”)

Rows = cursor.fetchall()

For row in rows:

Print(row)

Cursor.execute(“UPDATE employees SET age = 31 WHERE name = ‘Alice’”)

Conn.commit()

Print(“Data updated successfully”)

Cursor.execute(“DELETE FROM employees WHERE name =’Alice’”)

Conn.commit()

Print(“Data deleted successfully”)

Cursor.close()

Conn.close()

Print(“Connection closed”)

#pandas

Import pandas as pd

Data = {

“Name”: [“Alice”, “Bob”, “Charlie”],

“Age”: [25, 30, 35],

“City”: [“New York”, “Paris”, “London”] }

# Converting dictionary to DataFrame

Df = pd.DataFrame(data)

Print(df)

#Numpy

Import numpy as np

#1D Array (Vector)

Arr = np.array([1, 2, 3, 4, 5])

Print(arr)

Print(“”)

#2D Array (Matrix)

Matrix = np.array([[1, 2, 3], [4, 5, 6]])

Print(matrix)

Print(“”)

#3D Array (Tensor)

Tensor = np.array([[[1, 2], [3, 4]],[[5, 6], [7, 8]]])

Print(tensor)

# Check for missing values

Print(df.isnull())

Print(df.isnull().sum())

#Removing Missing Values

Df\_cleaned = df.dropna() # Remove rows with missing values

Print(df\_cleaned)

#Filling Missing Values

Df[“Age”].fillna(df[“Age”].mean(), inplace=True)

Df[“City”].fillna(“Unknown”, inplace=True)

Print(df)

Print(df.duplicated()) # Check for duplicates

Df\_unique = df.drop\_duplicates()

Print(df\_unique)

#Standardization (Mean = 0, Std Dev = 1)

From sklearn.preprocessing import StandardScaler

Scaler = StandardScaler()

Df[[“Age”]] = scaler.fit\_transform(df[[“Age”]])

#Encoding Categorical Variables

Df = pd.get\_dummies(df, columns=[“City”])

#Bar Graph

Categories = [“A”, “B”, “C”]

Values = [10, 20, 15]

Plt.bar(categories, values, color=”skyblue”)

Plt.title(“Bar Chart Example”)

Plt.show()

#Scatter Plot (Finding relationships)

Import numpy as np

X = np.random.rand(50)

Y = np.random.rand(50)

Plt.scatter(x, y, color=”red”)

Plt.title(“Scatter Plot Example”)

Plt.show()

Import seaborn as sns

Import pandas as pd

#Histogram (Distribution of data)

Data = np.random.randn(1000)

Plt.hist(data, bins=30, color=”purple”, alpha=0.7)

Plt.title(“Histogram Example”)

Plt.show()

#Heatmap (Correlation between variables)

Data = np.random.rand(5, 5)

Sns.heatmap(data, annot=True, cmap=”coolwarm”)

Plt.title(“Heatmap Example”)

Plt.show()

#Machine Learning

#Logistic Regression

Import sklearn

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

From sklearn.datasets import load\_iris

#Logistic Regression

# Load dataset

Iris = load\_iris()

X, y = iris.data, iris.target # Features (X) and labels (y)

# Split data into training (80%) and testing (20%)

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

#Preprocessing Data

From sklearn.preprocessing import StandardScaler

Scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train) # Fit & transform

#training data

X\_test\_scaled = scaler.transform(X\_test)

From sklearn.linear\_model import LogisticRegressio

# Create and train model

Model = LogisticRegression()

Model.fit(X\_train\_scaled, y\_train)

# Make predictions

Y\_pred = model.predict(X\_test\_scaled)

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

# Compute accuracy

Accuracy = accuracy\_score(y\_test, y\_pred)

Print(f”Accuracy: {accuracy:.2f}”)

# Detailed performance report

Print(classification\_report(y\_test, y\_pred))

Import seaborn as sns

# Scatter plot of the dataset

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

Sns.scatterplot(x=X\_test[:, 0], y=X\_test[:, 1], hue=y\_test,

Palette=”coolwarm”)

Plt.xlabel(“Feature 1”)

Plt.ylabel(“Feature 2”)

Plt.title(“Logistic Regression Classification”)

Plt.show()

#Linear Regression

Import numpy as np

Import pandas as pd

Import matplotlib.pyplot as plt

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

From sklearn.linear\_model import LinearRegression

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

# Generating random data

Np.random.seed(42)

X = 2 \* np.random.rand(100, 1) # Independent variable

Y = 4 + 3 \* X + np.random.randn(100, 1) # Dependent variable (y= 4 + 3X + nois

#Split Data into Training and Testing Sets

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

#Train the Linear Regression Model

Model = LinearRegression() # Create model

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

#Make Predictions

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

#Evaluate Model Performance

Mse = mean\_squared\_error(y\_test, y\_pred)

R2 = r2\_score(y\_test, y\_pred)

Print(f”Mean Squared Error: {mse:.2f}”)

Print(f”R-squared Score: {r2:.2f}”)

#Visualizing the Regression Line

Plt.scatter(X\_test, y\_test, color=”blue”, label=”Actual data”)

Plt.plot(X\_test, y\_pred, color=”red”, linewidth=2,

Label=”Regression Line”)

Plt.xlabel(“X (Independent Variable)”)

Plt.ylabel(“y (Dependent Variable)”)

Plt.title(“Linear Regression Model”)

Plt.legend()

Plt.show()

#Regularization

Ridge\_model = Ridge(alpha=1.0)