Internal Practical Exam:

Name:Bhos ViveK

Class:FYMCA

Div:A

Roll: 51005

Course:LP-II

Elective:Machine Learning

1)To write a program to create a network connection between 2 nodes using UDP

udp\_server.py

*import* socket

def **start\_server**():

*# Create a UDP socket*

    server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

*# Bind the socket to a specific address and port*

    server\_address = ('localhost', 12345)

    server\_socket.bind(server\_address)

    print(f"UDP Server is listening on {server\_address}")

*while* True:

*try*:

*# Receive data from client*

            data, client\_address = server\_socket.recvfrom(1024)

            print(f"Received message from {client\_address}: {data.decode()}")

*# Send response back to client*

            response = f"Server received: {data.decode()}"

            server\_socket.sendto(response.encode(), client\_address)

*except* Exception *as* e:

            print(f"Error: {e}")

*break*

    server\_socket.close()

*if* \_\_name\_\_ == "\_\_main\_\_":

    start\_server()

udp\_client.py

*import* socket

def **start\_client**():

*# Create a UDP socket*

    client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

*# Server address*

    server\_address = ('localhost', 12345)

*try*:

*while* True:

*# Get message from user*

            message = input("Enter message to send (or 'quit' to exit): ")

*if* message.lower() == 'quit':

*break*

*# Send data to server*

            client\_socket.sendto(message.encode(), server\_address)

*# Receive response from server*

            data, \_ = client\_socket.recvfrom(1024)

            print(f"Server response: {data.decode()}")

*except* Exception *as* e:

        print(f"Error: {e}")

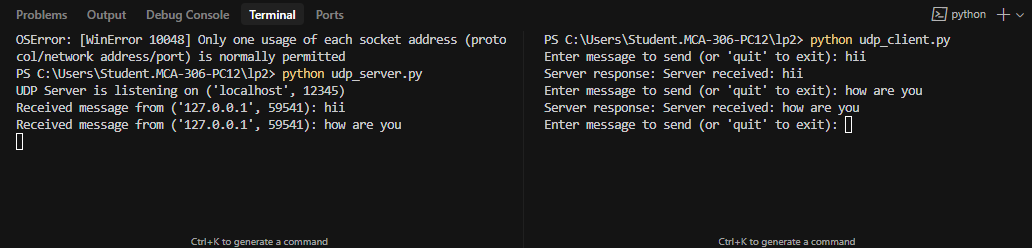
*finally*:

        client\_socket.close()

*if* \_\_name\_\_ == "\_\_main\_\_":

    start\_client()

Output:



2) *import* pandas *as* pd

*from* sklearn.model\_selection *import* train\_test\_split

*import* numpy *as* np

def **split\_dataset**(*data\_path*, *test\_size*=0.2, *random\_state*=42):

    """

    Split a dataset into training and test sets.

    Parameters:

    data\_path (str): Path to the dataset file (CSV format)

    test\_size (float): Proportion of the dataset to include in the test split (default: 0.2)

    random\_state (int): Random seed for reproducibility (default: 42)

    Returns:

    tuple: (X\_train, X\_test, y\_train, y\_test)

    """

*try*:

*# Read the dataset*

        df = pd.read\_csv(*data\_path*)

*# Assuming the last column is the target variable*

        X = df.iloc[:, :-1]  *# Features (all columns except the last one)*

        y = df.iloc[:, -1]   *# Target variable (last column)*

*# Split the dataset*

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(

            X, y, *test\_size*=*test\_size*, *random\_state*=*random\_state*

        )

        print("\nDataset splitting summary:")

        print(f"Total samples: {len(df)}")

        print(f"Training samples: {len(X\_train)} ({(1-*test\_size*)\*100}%)")

        print(f"Testing samples: {len(X\_test)} ({*test\_size*\*100}%)")

*return* X\_train, X\_test, y\_train, y\_test

*except* Exception *as* e:

        print(f"Error: {str(e)}")

*return* None

*# Example usage*

*if* \_\_name\_\_ == "\_\_main\_\_":

*# Using our sample dataset*

    data\_path = "sample\_dataset.csv"

*try*:

        X\_train, X\_test, y\_train, y\_test = split\_dataset(data\_path)

*if* X\_train is not None:

            print("\nShape of split datasets:")

            print(f"X\_train shape: {X\_train.shape}")

            print(f"X\_test shape: {X\_test.shape}")

            print(f"y\_train shape: {y\_train.shape}")

            print(f"y\_test shape: {y\_test.shape}")

            print("\nFeature names:")

            print(X\_train.columns.tolist())

*except* FileNotFoundError:

        print(f"\nError: Could not find {data\_path}. Please make sure the file exists in the current directory.")

Output:

