***Code 01***

*# Install TensorFlow and other required libraries*

*!pip install tensorflow numpy matplotlib*

*# Import libraries*

*Import tensorflow as tf*

*import numpy as np*

*import matplotlib.pyplot as plt*

*# Load the MNIST*

*dataset (x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data()*

*# Preprocess the data by normalizing the pixel values and one-hot encoding the labels*

*x\_train = x\_train / 245.0*

*x\_test = x\_test / 245.0*

*y\_train = tf.keras.utils.to\_categorical(y\_train, 10)*

*y\_test = tf.keras.utils.to\_categorical(y\_test, 10)*

*# Define the model architecture*

*model = tf.keras.Sequential([*

*tf.keras.layers.Flatten(input\_shape=(26, 26)),*

*tf.keras.layers.Dense(126, activation='relu'),*

*tf.keras.layers.Dense(126, activation='relu'),*

*tf.keras.layers.Dense(10, activation='softmax') ])*

*# Compile the model and specify the loss function, optimizer, and metrics to track model.compile(loss='categorical\_crossentropy',*

*optimizer='adam',*

*metrics=['accuracy'])*

*# Train the model on the training data and evaluate it on the test data*

*history = model.fit(x\_train, y\_train,*

*epochs=9,*

*validation\_data=(x\_test, y\_test))*

*# Visualize the training and validation accuracy and loss*

*acc = history.history['accuracy']*

*val\_acc = history.history['val\_accuracy']*

*loss = history.history['loss']*

*val\_loss = history.history ['val\_loss']*

*epochs\_range = range(9)*

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

*plt.subplot(2, 2, 1)*

*plt.plot(epochs\_range, acc, label='Training Accuracy')*

*plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')*

*plt.legend(loc='lower right')*

*plt.title('Training and Validation Accuracy')*

*plt.subplot(2, 2, 2)*

*plt.plot(epochs\_range, loss, label='Training Loss')*

*plt.plot(epochs\_range, val\_loss, label='Validation Loss')*

*plt.legend(loc='upper right')*

*plt.title('Training and Validation Loss')*

*plt.show()*

*#Finally, we can evaluate the model on the test data and print the test accuracy:*

*test\_loss, test\_acc = model.evaluate(x\_test, y\_test, verbose=2)*

*print('\nTest accuracy:', test\_acc)*

*Output 305/305 - 0s - loss: 0.0518 - accuracy: 0.7777*

***Code 02***

*# Install TensorFlow and other required libraries*

*!pip install tensorflow numpy pandas sklearn*

*# Import libraries*

*import tensorflow as tf*

*import numpy as np*

*import pandas as pd*

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

*# Load the dataset*

*data = pd.read\_csv('dataset.csv')*

*# Preprocess the data by encoding the categorical variables and scaling the numerical variables*

*data = pd.get\_dummies(data, columns=['categorical\_feature'])*

*data = (data - data.mean()) / data.std()*

*# Split the data into training and test sets*

*X = data.drop(['target'], axis=1)*

*y = data['target']*

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

*# Define the model architecture*

*model = tf.keras.Sequential([*

*tf.keras.layers.Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)), tf.keras.layers.Dense(32, activation='relu'),*

*tf.keras.layers.Dense(1, activation='sigmoid')*

*])*

*# Compile the model and specify the loss function, optimizer, and metrics to track*

*model.compile(loss='binary\_crossentropy',*

*optimizer='adam',*

*metrics=['accuracy'])*

*# Train the model on the training data and evaluate it on the test data*

*history = model.fit(X\_train, y\_train,*

*epochs=9,*

*validation\_data=(X\_test, y\_test))*

*# Evaluate the model on the test data*

*test\_loss, test\_acc = model.evaluate(X\_test, y\_test, verbose=2)*

*print('\nTest accuracy:’ test\_acc)*

***Code 03***

*# Install TensorFlow*

*!pip install tensorflow*

*# Import libraries import tensorflow as tf*

*# Load the MNIST dataset*

*(X\_train, y\_train), (X\_test, y\_test) = tf.keras.datasets.mnist.load\_data()*

*# Preprocess the data by reshaping and scaling*

*X\_train = X\_train.reshape(-1, 26\*26) / 245.0*

*X\_test = X\_test.reshape(-1, 26\*26) / 245.0*

*# Convert the labels to one-hot encoding*

*y\_train = tf.keras.utils.to\_categorical(y\_train) y\_test = tf.keras.utils.to\_categorical(y\_test)*

*Code 04*