# 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=10, 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)

In this code, we first load and preprocess the dataset, which includes encoding the categorical variables and scaling the numerical variables. We then split the data into training and test sets using the train\_test\_split function from the scikit-learn library. Next, we define the model architecture using the Sequential class from the TensorFlow library. The model consists of three fully-connected layers with 64, 32, and 1 nodes, respectively. The activation function for the output layer is set to sigmoid , which is suitable for binary classification. We then compile the model by specifying the loss function, optimizer, and metrics to track. In this case, we use the binary\_crossentropy loss function and the adam optimizer. We also specify the accuracy metric, which will be used to evaluate the model's performance. Finally, we train the model on the training data using the fit method and evaluate it on the test data using the evaluate method. We print the test accuracy to see how well the model performs on unseen data