Bank

import pandas as pd

import tensorflow as tf

from tensorflow import keras

from sklearn.metrics import accuracy\_score, confusion\_matrix

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

from sklearn.preprocessing import MinMaxScaler

data = pd.read\_csv('/content/Churn\_Modelling.csv')

X = data.drop(columns=['Exited', 'CustomerId', 'Surname', 'RowNumber'])  # Exclude columns

y = data['Exited']  # Target

# Step 3: Data Preprocessing

# Handle missing values and encode categorical variables

# Removing rows with missing values:

data = data.drop(['CustomerId', 'Surname', 'RowNumber'], axis = 1)

print(data.columns)

# Replacing missing values with a specific value (e.g., mean):

# data['column\_name'].fillna(data['column\_name'].mean(), inplace=True)

# You need to ensure that the columns 'Geography' and 'Gender' are present in the DataFrame X

# Add additional error handling to verify the column names

columns\_to\_encode = ['Geography', 'Gender']

for column in columns\_to\_encode:

    if column not in X.columns:

        raise ValueError(f"Column '{column}' not found in the DataFrame X.")

# You need to encode categorical variables like "Geography" and "Gender" into numerical format using one-hot encoding.

X = pd.get\_dummies(X, columns=['Geography', 'Gender'], drop\_first=True)

scaler = MinMaxScaler()

X = scaler.fit\_transform(X)

# Step 5: Initialize and Build the Model

model = keras.Sequential([

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

    keras.layers.Dense(32, activation='relu'),

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

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Split the data into training and testing sets

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

# Train the model

model.fit(X\_train, y\_train, epochs=20, batch\_size=32, verbose=1)

# Step 6: Evaluate the Model

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

y\_pred = (y\_pred > 0.5).astype(int)  # Convert to binary prediction

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

confusion = confusion\_matrix(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

print("Confusion Matrix:")

print(confusion)