**#ml3 bank churn**

**import pandas as pd**

**df = pd.read\_csv('Churn\_Modelling.csv')**

**df.shape**

**print(df.head())**

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

**# Define features and target**

**X = df.drop(columns=['Exited', 'CustomerId'])  # 'Exited' is the target variable**

**y = df['Exited']**

**# Convert categorical variables to dummy/indicator variables**

**X = pd.get\_dummies(X, drop\_first=True)**

**# Split the dataset**

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

**# Output the shape of the datasets**

**print(f"Training set shape: {X\_train.shape}, Test set shape: {X\_test.shape}")**

**from sklearn.preprocessing import StandardScaler**

**# Initialize the StandardScaler**

**scaler = StandardScaler()**

**# Fit and transform the training data, transform the test data**

**X\_train = scaler.fit\_transform(X\_train)**

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

**# Output the mean and variance of the scaled training data**

**print(f"Mean of X\_train: {X\_train.mean(axis=0)}, Variance of X\_train: {X\_train.var(axis=0)}")**

**!pip install tensorflow**

**import tensorflow as tf**

**from tensorflow import keras**

**from tensorflow.keras import layers**

**# Build the model**

**model = keras.Sequential([**

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

**layers.Dense(16, activation='relu'),**

**layers.Dense(1, activation='sigmoid')  # For binary classification])**

**# Compile the model**

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

**# Train the model**

**history = model.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_split=0.2)**

**# Output training history**

**print(history.history)**

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

**# Make predictions**

**y\_pred = (model.predict(X\_test) > 0.5).astype("int32")**

**# Calculate accuracy**

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

**print(f"Accuracy: {accuracy:.2f}")**

**# Print confusion matrix**

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

**print("Confusion Matrix:")**

**print(cm)**