```
import pandas as pd
# Load the dataset
df = pd.read_csv('path_to_your_downloaded/bank-customer-churn-modeling.csv')
# Display the first few rows of the dataset
df.head()
from sklearn.model_selection import train_test_split
# Features: Drop the target column and any other unnecessary columns like 'CustomerId',
'Surname'
X = df.drop(['Exited', 'CustomerId', 'Surname'], axis=1)
# Target: 'Exited' column represents customer churn (1 = will leave, 0 = won't leave)
y = df['Exited']
# Split the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
from sklearn.preprocessing import StandardScaler
# Initialize the scaler
scaler = StandardScaler()
# Normalize the training data
X_train_scaled = scaler.fit_transform(X_train)
# Normalize the testing data (using the same scaler from training data)
X_test_scaled = scaler.transform(X_test)
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
# Initialize the model
model = Sequential()
# Add the input layer (input_dim=number of features)
model.add(Dense(units=64, activation='relu', input_dim=X_train_scaled.shape[1]))
# Add one hidden layer
model.add(Dense(units=64, activation='relu'))
# Add the output layer (binary classification: sigmoid activation)
model.add(Dense(units=1, activation='sigmoid'))
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Print model summary
model.summary()
# Train the model
history = model.fit(X_train_scaled, y_train, epochs=10, batch_size=32,
validation_data=(X_test_scaled, y_test))
# Evaluate the model on the test data
loss, accuracy = model.evaluate(X_test_scaled, y_test)
print(f"Test accuracy: {accuracy}")
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Predict using the trained model
y_pred = (model.predict(X_test_scaled) > 0.5) # Threshold at 0.5 for binary classification
# Accuracy score
from sklearn.metrics import accuracy_score
print(f"Accuracy Score: {accuracy_score(y_test, y_pred)}")
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
# Plotting confusion matrix
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Not Churned", "Churned"],
yticklabels=["Not Churned", "Churned"])
plt.title("Confusion Matrix")
plt.show()
```