Customer Churn Prediction - Streamlit App

This document provides the Streamlit application code for predicting customer churn using a pre-trained Artificial Neural Network (ANN) model, along with encoders and scaler. The app allows users to input customer details and get churn probability predictions.

# Streamlit Application Code

python  
import streamlit as st  
import pandas as pd  
import tensorflow as tf  
import pickle  
  
# Load trained ANN model  
model = tf.keras.models.load\_model('model.h5')  
  
# Load encoders and scaler  
with open('label\_encoder\_gender.pkl', 'rb') as file:  
 label\_encoder\_gender = pickle.load(file)  
  
with open('one\_hot\_encode.pkl', 'rb') as file:  
 one\_hot\_encode = pickle.load(file)  
  
with open('Scalar.pkl', 'rb') as file:  
 scalar = pickle.load(file)  
  
# ---------------- Streamlit UI ----------------  
st.title("📊 Customer Churn Prediction App")  
st.write("Enter customer details to predict churn probability.")  
  
# User inputs  
geography = st.selectbox("🌍 Geography", one\_hot\_encode.categories\_[0])  
gender = st.selectbox("⚧ Gender", label\_encoder\_gender.classes\_)  
age = st.slider("🎂 Age", 18, 91, 30)  
credit\_score = st.number\_input("💳 Credit Score", min\_value=300, max\_value=900, value=600)  
balance = st.number\_input("🏦 Balance", min\_value=0.0, value=50000.0)  
estimated\_salary = st.number\_input("💰 Estimated Salary", min\_value=0.0, value=60000.0)  
tenure = st.slider("📅 Tenure (Years with bank)", 0, 10, 3)  
num\_of\_products = st.slider("📦 Number Of Products", 1, 4, 1)  
has\_cr\_card = st.selectbox("💳 Has Credit Card?", [0, 1])  
is\_active\_member = st.selectbox("✅ Is Active Member?", [0, 1])  
  
# ---------------- Preprocessing ----------------  
# Build input dataframe  
input\_data = pd.DataFrame({  
 'CreditScore': [credit\_score],  
 'Geography': [geography],  
 'Gender': [gender],  
 'Age': [age],  
 'Tenure': [tenure],  
 'Balance': [balance],  
 'NumOfProducts': [num\_of\_products],  
 'HasCrCard': [has\_cr\_card],  
 'IsActiveMember': [is\_active\_member],  
 'EstimatedSalary': [estimated\_salary]  
})  
  
# Encode Gender  
input\_data['Gender'] = label\_encoder\_gender.transform(input\_data['Gender'])  
  
# One-hot encode Geography  
geo\_encoded = one\_hot\_encode.transform(input\_data[['Geography']]).toarray()  
geo\_encoded\_df = pd.DataFrame(geo\_encoded, columns=one\_hot\_encode.get\_feature\_names\_out(['Geography']))  
  
# Drop Geography and merge  
input\_data = input\_data.drop(columns=['Geography'])  
input\_df = pd.concat([input\_data.reset\_index(drop=True), geo\_encoded\_df], axis=1)  
  
# Scale features  
input\_scaled = scalar.transform(input\_df)  
  
# ---------------- Prediction ----------------  
if st.button("🔮 Predict Churn"):  
 prediction = model.predict(input\_scaled)  
 prediction\_proba = prediction[0][0]  
  
 st.write(f"### Churn Probability: \*\*{prediction\_proba:.2f}\*\*")  
  
 if prediction\_proba > 0.5:  
 st.error("⚠️ The customer is likely to \*\*CHURN\*\*.")  
 else:  
 st.success("✅ The customer is likely to \*\*STAY\*\*.")