Customer Churn Prediction with ANN (TensorFlow + Scikit-Learn)

# 1. Dataset Preprocessing

We first load the dataset and perform preprocessing steps such as dropping irrelevant columns, encoding categorical features, and scaling numerical features.

python  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder  
import pickle  
  
# Load dataset  
data = pd.read\_csv("Churn\_Modelling.csv")  
  
# Drop irrelevant columns  
data = data.drop(['RowNumber','CustomerId','Surname'], axis=1)  
  
# Encode Gender using LabelEncoder  
label\_encoder\_gender = LabelEncoder()  
data['Gender'] = label\_encoder\_gender.fit\_transform(data['Gender'])  
  
# One-Hot Encode Geography  
one\_hot\_encode = OneHotEncoder()  
geo\_encode = one\_hot\_encode.fit\_transform(data[['Geography']])  
geo\_encoded\_df = pd.DataFrame(geo\_encode.toarray(), columns=one\_hot\_encode.get\_feature\_names\_out(['Geography']))  
  
# Merge with dataset  
data = pd.concat([data.drop('Geography', axis=1), geo\_encoded\_df], axis=1)  
  
# Save encoders  
with open('label\_encoder\_gender.pkl','wb') as file:  
 pickle.dump(label\_encoder\_gender, file)  
  
with open('one\_hot\_encode.pkl','wb') as file:  
 pickle.dump(one\_hot\_encode, file)  
  
# Split features and target  
X = data.drop('Exited', axis=1)  
Y = data['Exited']  
  
# Train-test split  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)  
  
# Scale features  
Scalar = StandardScaler()  
X\_train = Scalar.fit\_transform(X\_train)  
X\_test = Scalar.transform(X\_test)  
  
with open('Scalar.pkl','wb') as file:  
 pickle.dump(Scalar, file)

# 2. ANN Implementation

We build an Artificial Neural Network using TensorFlow/Keras with two hidden layers and a sigmoid output layer for binary classification (churn prediction).

python  
import tensorflow as tf  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense  
from tensorflow.keras.callbacks import EarlyStopping, TensorBoard  
import datetime  
  
# Build the model  
model = Sequential([  
 Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),  
 Dense(32, activation='relu'),  
 Dense(1, activation='sigmoid')  
])  
  
# Compile model  
opt = tf.keras.optimizers.Adam(learning\_rate=0.01)  
model.compile(optimizer=opt, loss='binary\_crossentropy', metrics=['accuracy'])  
  
# Setup callbacks  
log\_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")  
tensorboard\_callback = TensorBoard(log\_dir=log\_dir, histogram\_freq=1)  
early\_stopping\_callback = EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)  
  
# Train model  
history = model.fit(  
 X\_train, Y\_train,  
 validation\_data=(X\_test, Y\_test),  
 epochs=100,  
 callbacks=[tensorboard\_callback, early\_stopping\_callback]  
)  
  
# Save model  
model.save('model.h5')

# 3. Loading Encoders, Scaler, and Model

To reuse the model in production, load the saved `.pkl` encoders and scaler, along with the trained ANN model.

python  
from tensorflow.keras.models import load\_model  
  
# Load encoders and scaler  
with open('label\_encoder\_gender.pkl', 'rb') as file:  
 loaded\_label\_encoder\_gender = pickle.load(file)  
  
with open('one\_hot\_encode.pkl', 'rb') as file:  
 loaded\_one\_hot\_encode = pickle.load(file)  
  
with open('Scalar.pkl', 'rb') as file:  
 loaded\_scaler = pickle.load(file)  
  
# Load ANN model  
loaded\_model = load\_model('model.h5')

# 4. Making Predictions

We can now use the trained model to predict churn for new customer data by applying the same preprocessing steps.

python  
# Example new customer  
new\_data = pd.DataFrame({  
 'CreditScore': [600],  
 'Gender': ['Male'],  
 'Age': [40],  
 'Tenure': [3],  
 'Balance': [60000],  
 'NumOfProducts': [2],  
 'HasCrCard': [1],  
 'IsActiveMember': [1],  
 'EstimatedSalary': [50000],  
 'Geography': ['France']  
})  
  
# Encode Gender  
new\_data['Gender'] = loaded\_label\_encoder\_gender.transform(new\_data['Gender'])  
  
# OneHotEncode Geography  
geo\_new = loaded\_one\_hot\_encode.transform(new\_data[['Geography']]).toarray()  
geo\_new\_df = pd.DataFrame(geo\_new, columns=loaded\_one\_hot\_encode.get\_feature\_names\_out(['Geography']))  
  
# Merge into dataset  
new\_data = pd.concat([new\_data.drop('Geography', axis=1), geo\_new\_df], axis=1)  
  
# Scale features  
new\_data\_scaled = loaded\_scaler.transform(new\_data)  
  
# Prediction  
prediction = loaded\_model.predict(new\_data\_scaled)  
print("Churn Probability:", prediction[0][0])  
print("Prediction:", "Exit" if prediction[0][0] > 0.5 else "Stay")