Prediction

April 7, 2025

0.1 Prediction With Trained ANN Model

[31]:
```
import tensorflow as tf
from tensorflow.keras.models import load_model
import pickle
import pandas as pd
import numpy as np
```

0.1.1 Loading the models

[32]:
```
## Load the trained model, scaler pickle, onehot_encoder
model = load_model('model.h5')

## Load the encoder and scaler
with open('onehot_encoder_geo.pkl', 'rb') as file:
        label_encoder_geo = pickle.load(file)

with open('label_encoder_gender.pkl', 'rb') as file:
        label_encoder_gender = pickle.load(file)

with open('scaler.pkl', 'rb') as file:
        scaler = pickle.load(file)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be
built. `model.compile_metrics` will be empty until you train or evaluate the
model.
```

Taking the Input data for model

[33]:
```
## Example input data

input_data = {
        'CreditScore': 600,
        'Geography' : 'France',
        'Gender' : 'Male',
        'Age': 40,
        'Tenure': 3,
        'Balance': 60000,
```

```
        'NumOfProducts': 2,
        'HasCrCard': 1,
        'IsActiveMember': 1,
        'EstimatedSalary': 50000
}
```

[34]:
```
## converting the geography char data in int data using the one hot encoding
geo_encoded = label_encoder_geo.transform([[input_data['Geography']]]).toarray()
geo_encoded_df = pd.DataFrame(geo_encoded, columns=label_encoder_geo.
 ↪get_feature_names_out(['Geography']))
geo_encoded_df
```

```
c:\Users\Uditya\Desktop\End-to-End-Deep-Learning-Project-Using-
ANN\venv\Lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does
not have valid feature names, but OneHotEncoder was fitted with feature names
  warnings.warn(
```

[34]:
```
   Geography_France  Geography_Germany  Geography_Spain
0               1.0                0.0              0.0
```

[35]:
```
input_df = pd.DataFrame([input_data])
input_df
```

[35]:
```
   CreditScore Geography Gender  …  HasCrCard  IsActiveMember  EstimatedSalary
0          600    France   Male  …          1               1            50000

[1 rows x 10 columns]
```

[36]:
```
## Encode categorical variables
input_df['Gender']=label_encoder_gender.transform(input_df['Gender'])
input_df
```

[36]:
```
   CreditScore Geography  Gender  …  HasCrCard  IsActiveMember
EstimatedSalary
0          600    France       1  …          1               1
50000

[1 rows x 10 columns]
```

[37]:
```
## combine one-hot encoded columns with input data
input_df = pd.concat([input_df.drop("Geography", axis=1), geo_encoded_df],␣
 ↪axis=1)
input_df
```

[37]:
```
   CreditScore  Gender  …  Geography_Germany  Geography_Spain
0          600       1  …                0.0              0.0

[1 rows x 12 columns]
```

Scaling the data

[38]:
```
input_scaled= scaler.transform(input_df)
input_scaled
```

[38]:
```
array([[-0.53598516,  0.91324755,  0.10479359, -0.69539349, -0.25781119,
         0.80843615,  0.64920267,  0.97481699, -0.87683221,  1.00150113,
        -0.57946723, -0.57638802]])
```

Prediction churn

[39]:
```
## Prediction churn

prediction=model.predict(input_scaled)
prediction
```

```
1/1             0s 331ms/step
```

[39]:
```
array([[0.02760429]], dtype=float32)
```

[40]:
```
prediction_proba = prediction[0][0]
```

[41]:
```
prediction_proba
```

[41]:
```
np.float32(0.02760429)
```

[42]:
```
if prediction_proba > 0.5:
        print("The customer is likely to churn.")
else:
        print("The customer is not likely to churn.")
```

```
The customer is not likely to churn.
```

[ ]:
```

```