bankcustomer

September 27, 2024

[44]:

```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/bank-customer-churn-modeling/Churn_Modelling.csv
```

[45]:

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)

df = pd.read_csv("/kaggle/input/bank-customer-churn-modeling/Churn_Modelling.
 ↪csv")
```

[46]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

[47]:

```python
X = df.drop(columns=['RowNumber', 'CustomerId', 'Surname', 'Exited'])
y = df['Exited']
```

[48]:

```python
X = pd.get_dummies(X, columns=['Geography', 'Gender'], drop_first=True)
```

[49]:

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[50]:

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[56]:

```python
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout

model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
```

```python
model.fit(X_train, y_train, epochs=15, batch_size=32, validation_split=0.2)
```

```
Epoch 1/15
200/200             2s 3ms/step -
accuracy: 0.7843 - loss: 0.5043 - val_accuracy: 0.8344 - val_loss: 0.4030
Epoch 2/15
200/200             0s 2ms/step -
accuracy: 0.8265 - loss: 0.4150 - val_accuracy: 0.8462 - val_loss: 0.3739
Epoch 3/15
200/200             0s 2ms/step -
accuracy: 0.8348 - loss: 0.3993 - val_accuracy: 0.8512 - val_loss: 0.3608
Epoch 4/15
200/200             0s 2ms/step -
accuracy: 0.8524 - loss: 0.3683 - val_accuracy: 0.8525 - val_loss: 0.3527
Epoch 5/15
200/200             0s 2ms/step -
accuracy: 0.8498 - loss: 0.3601 - val_accuracy: 0.8562 - val_loss: 0.3498
Epoch 6/15
200/200             0s 2ms/step -
accuracy: 0.8607 - loss: 0.3453 - val_accuracy: 0.8581 - val_loss: 0.3460
Epoch 7/15
200/200             0s 2ms/step -
accuracy: 0.8578 - loss: 0.3479 - val_accuracy: 0.8519 - val_loss: 0.3454
Epoch 8/15
200/200             0s 2ms/step -
accuracy: 0.8580 - loss: 0.3525 - val_accuracy: 0.8537 - val_loss: 0.3433
Epoch 9/15
200/200             0s 2ms/step -
accuracy: 0.8571 - loss: 0.3451 - val_accuracy: 0.8569 - val_loss: 0.3440
Epoch 10/15
200/200             0s 2ms/step -
accuracy: 0.8543 - loss: 0.3531 - val_accuracy: 0.8575 - val_loss: 0.3391
Epoch 11/15
200/200             0s 2ms/step -
accuracy: 0.8588 - loss: 0.3438 - val_accuracy: 0.8594 - val_loss: 0.3381
Epoch 12/15
200/200             0s 2ms/step -
accuracy: 0.8685 - loss: 0.3304 - val_accuracy: 0.8537 - val_loss: 0.3430
Epoch 13/15
200/200             0s 2ms/step -
accuracy: 0.8618 - loss: 0.3418 - val_accuracy: 0.8581 - val_loss: 0.3398
Epoch 14/15
200/200             0s 2ms/step -
accuracy: 0.8507 - loss: 0.3522 - val_accuracy: 0.8581 - val_loss: 0.3363
Epoch 15/15
200/200             1s 2ms/step -
accuracy: 0.8630 - loss: 0.3360 - val_accuracy: 0.8562 - val_loss: 0.3425
```

```
[56]: <keras.src.callbacks.history.History at 0x77fcbc666e60>
```

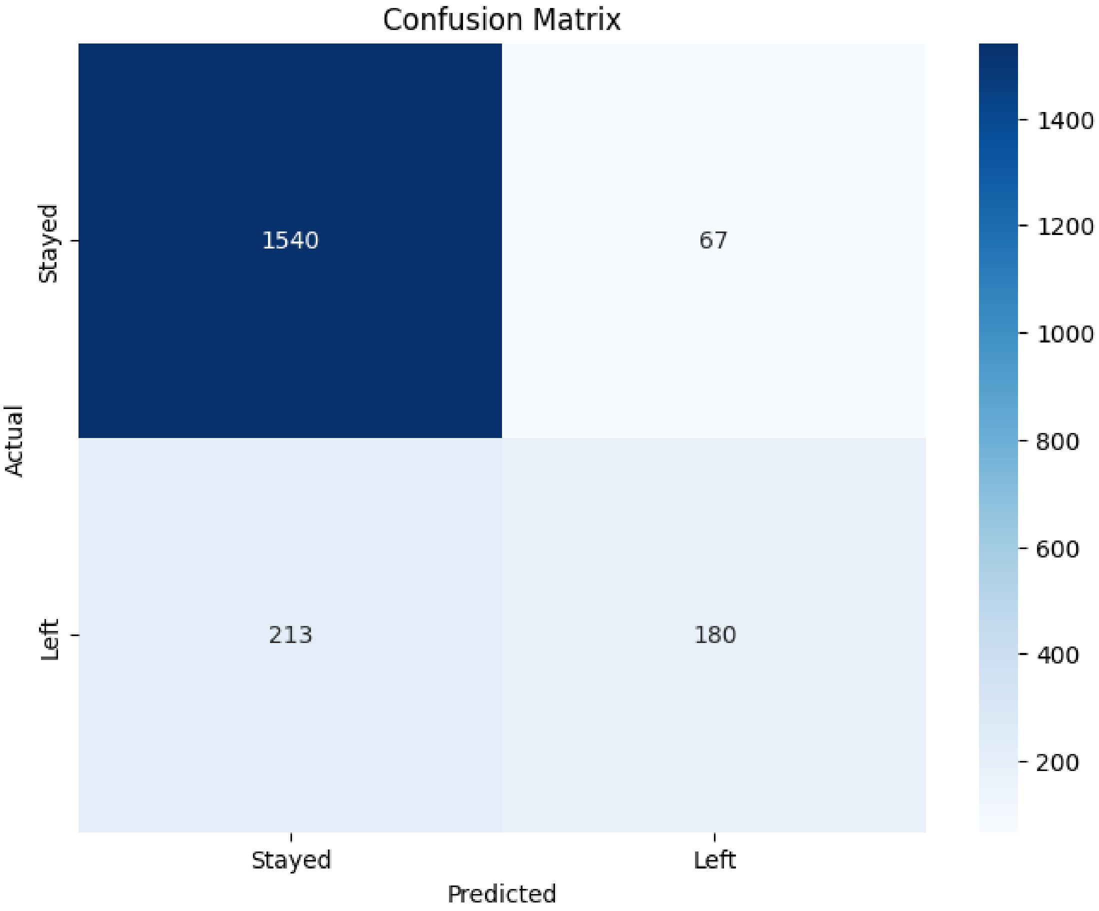
```
[58]: from sklearn.metrics import accuracy_score, confusion_matrix

      y_pred = (model.predict(X_test) > 0.5).astype(int)
      accuracy = accuracy_score(y_test, y_pred)
      cm = confusion_matrix(y_test, y_pred)
      print(f'Accuracy: {accuracy}')
```

```
      63/63                 0s 1ms/step
      Accuracy: 0.86
```

```
[59]: import seaborn as sns
      import matplotlib.pyplot as plt
      from sklearn.metrics import confusion_matrix

      plt.figure(figsize=(8,6))
      sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Stayed',␣
        ↪'Left'], yticklabels=['Stayed', 'Left'])
      plt.title('Confusion Matrix')
      plt.xlabel('Predicted')
      plt.ylabel('Actual')
      plt.show()
```


Confusion Matrix
Actual
Stayed
Left
Predicted
Stayed
Left
1540
67
213
180
1400
1200
1000
800
600
400
200