task-3

February 6, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df = pd.read_csv('Churn_Modelling.csv')
```

[3]:
```
df.head()
```

[3]:
```
   RowNumber  CustomerId   Surname  CreditScore Geography  Gender  Age  \
0          1    15634602  Hargrave          619    France  Female   42
1          2    15647311      Hill          608     Spain  Female   41
2          3    15619304      Onio          502    France  Female   42
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

[4]:
```
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
```

[5]:
```
df.isnull().sum()
```

[5]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

[6]:
```
df.hist(bins=5,figsize=(15,15), color='green')
plt.show()
```

[7]:
```
df.columns
```

[7]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

[8]:
```
df.head()
```

[8]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	\
0	1	15634602	Hargrave	619	France	Female	42	
1	2	15647311	Hill	608	Spain	Female	41	
2	3	15619304	Onio	502	France	Female	42	

```
3          4    15701354      Boni          699    France  Female   39
4          5    15737888  Mitchell          850     Spain  Female   43

   Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0       2       0.00              1          1               1
1       1   83807.86              1          0               1
2       8  159660.80              3          1               0
3       1       0.00              2          0               0
4       2  125510.82              1          1               1

   EstimatedSalary  Exited
0        101348.88       1
1        112542.58       0
2        113931.57       1
3         93826.63       0
4         79084.10       0
```

[9]:
```
df.drop(columns=['RowNumber', 'CustomerId', 'Surname'], axis=1, inplace=True)
```

[10]:
```
df.head()
```

[10]:
```
   CreditScore Geography  Gender  Age  Tenure    Balance  NumOfProducts  \
0          619    France  Female   42       2       0.00              1
1          608     Spain  Female   41       1   83807.86              1
2          502    France  Female   42       8  159660.80              3
3          699    France  Female   39       1       0.00              2
4          850     Spain  Female   43       2  125510.82              1

   HasCrCard  IsActiveMember  EstimatedSalary  Exited
0          1               1        101348.88       1
1          0               1        112542.58       0
2          1               0        113931.57       1
3          0               0         93826.63       0
4          1               1         79084.10       0
```

[11]:
```
from sklearn.preprocessing import LabelEncoder
lr=LabelEncoder()
df['Geography']=lr.fit_transform(df['Geography'])
df['Gender']=lr.fit_transform(df['Gender'])
```

[12]:
```
df.head()
```

[12]:
```
   CreditScore  Geography  Gender  Age  Tenure    Balance  NumOfProducts  \
0          619          0       0   42       2       0.00              1
1          608          2       0   41       1   83807.86              1
2          502          0       0   42       8  159660.80              3
3          699          0       0   39       1       0.00              2
```

```
4          850          2       0   43       2  125510.82              1

   HasCrCard  IsActiveMember  EstimatedSalary  Exited
0          1               1        101348.88       1
1          0               1        112542.58       0
2          1               0        113931.57       1
3          0               0         93826.63       0
4          1               1         79084.10       0
```

[13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   CreditScore      10000 non-null  int64
 1   Geography        10000 non-null  int32
 2   Gender           10000 non-null  int32
 3   Age              10000 non-null  int64
 4   Tenure           10000 non-null  int64
 5   Balance          10000 non-null  float64
 6   NumOfProducts    10000 non-null  int64
 7   HasCrCard        10000 non-null  int64
 8   IsActiveMember   10000 non-null  int64
 9   EstimatedSalary  10000 non-null  float64
 10  Exited           10000 non-null  int64
dtypes: float64(2), int32(2), int64(7)
memory usage: 781.4 KB
```

[14]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
```

[15]:
```
x = df.drop('Exited', axis=1)
y = df['Exited']
```

[16]:
```
x.head()
```

[16]:
```
   CreditScore  Geography  Gender  Age  Tenure    Balance  NumOfProducts  \
0          619          0       0   42       2       0.00              1
1          608          2       0   41       1   83807.86              1
2          502          0       0   42       8  159660.80              3
3          699          0       0   39       1       0.00              2
4          850          2       0   43       2  125510.82              1
```

```
   HasCrCard  IsActiveMember  EstimatedSalary
0          1               1        101348.88
1          0               1        112542.58
2          1               0        113931.57
3          0               0         93826.63
4          1               1         79084.10
```

[17]:
```
y.head()
```

[17]:
```
0    1
1    0
2    1
3    0
4    0
Name: Exited, dtype: int64
```

[18]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=123)
print('x_train',  x_train.shape)
print('x_test',  x_test.shape)
print('y_train',  y_train.shape)
print('y_test',  y_test.shape)
```

```
x_train (8000, 10)
x_test (2000, 10)
y_train (8000,)
y_test (2000,)
```

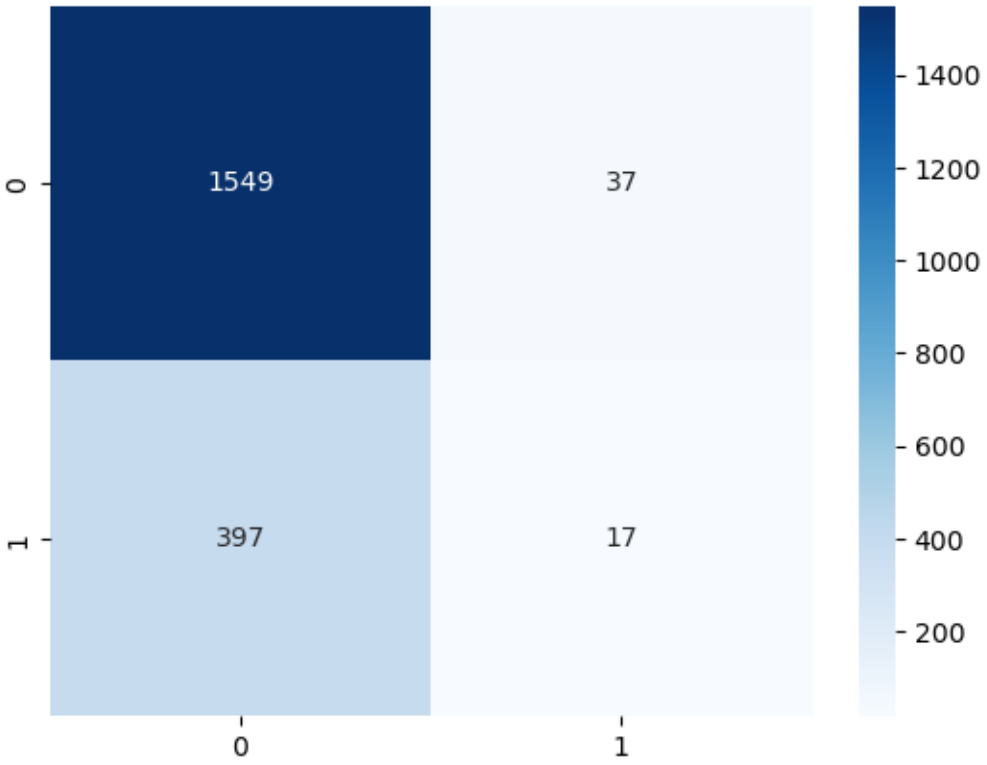
[19]:
```
lr= LogisticRegression()
lr.fit(x_train,y_train)
y_pred = lr.predict(x_test)
accuracy = accuracy_score(y_test,y_pred)
class_report = classification_report(y_test,y_pred)
conf_matrix = confusion_matrix(y_test,y_pred)
```

[20]:
```
print(f'The accuracy score of churn prediction is : {accuracy : .2f}%\n ')
print(f'The classification report of churn prediction is \n: {class_report}')
print(f'The confusion matrix of churn prediction is:\n{conf_matrix}')
```

```
The accuracy score of churn prediction is :  0.78%

The classification report of churn prediction is
:               precision    recall  f1-score   support

           0       0.80      0.98      0.88      1586
           1       0.31      0.04      0.07       414
```

```
    accuracy                           0.78      2000
   macro avg       0.56      0.51      0.47      2000
weighted avg       0.70      0.78      0.71      2000

The confusion matrix of churn prediction is:
[[1549   37]
 [ 397   17]]
```

[21]:
```
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')
plt.show()
```



[ ]:

[ ]:

[ ]: