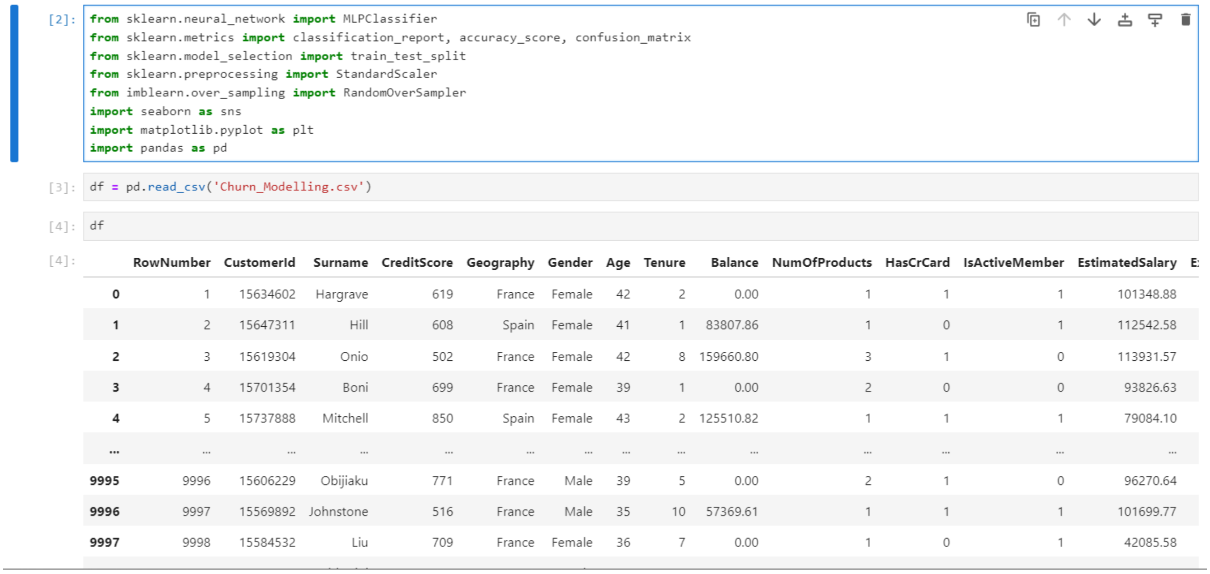


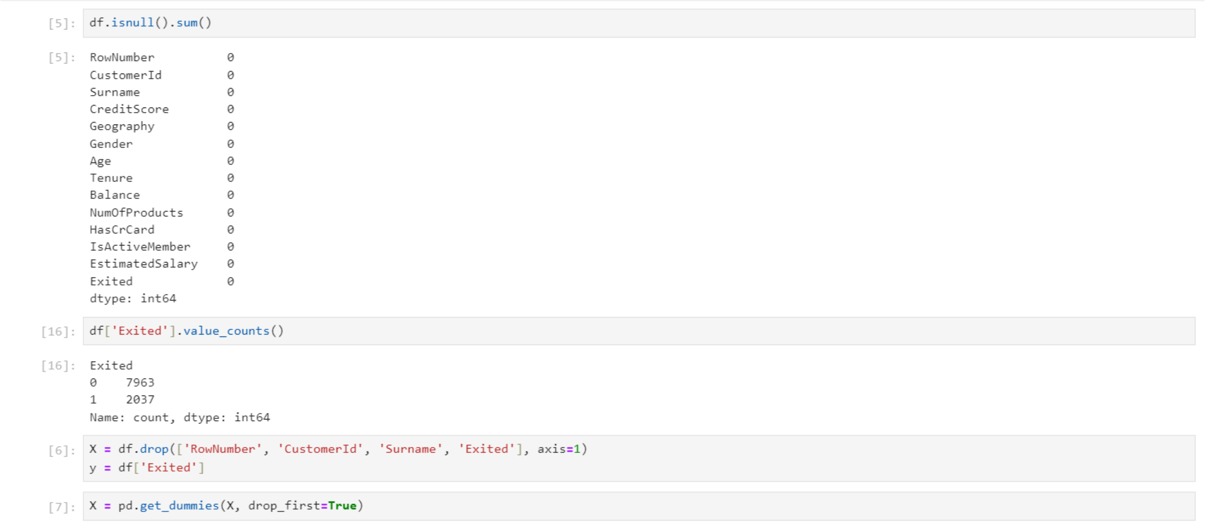
[2]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import RandomOverSampler
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

[3]:
```
df = pd.read_csv('Churn_Modelling.csv')
```

[4]:
```
df
```

[4]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	E
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	



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

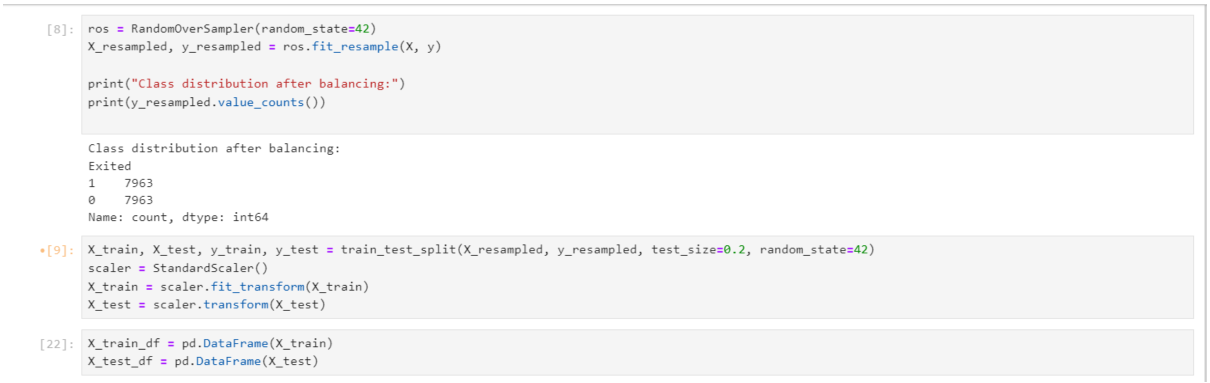
[5]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited            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

[16]:
```
df['Exited'].value_counts()
```

[16]:
```
Exited
0    7963
1    2037
Name: count, dtype: int64
```

[6]:
```
X = df.drop(['RowNumber', 'CustomerId', 'Surname', 'Exited'], axis=1)
y = df['Exited']
```

[7]:
```
X = pd.get_dummies(X, drop_first=True)
```

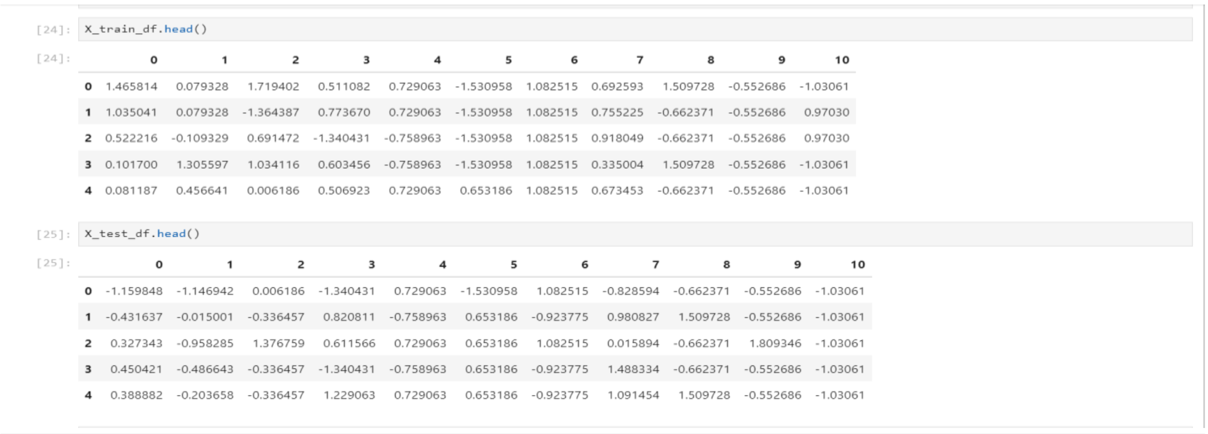


[8]:
```
ros = RandomOverSampler(random_state=42)
X_resampled, y_resampled = ros.fit_resample(X, y)

print("Class distribution after balancing:")
print(y_resampled.value_counts())
```

```
Class distribution after balancing:
Exited
1    7963
0    7963
Name: count, dtype: int64
```

•[9]:
```
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[22]:
```
X_train_df = pd.DataFrame(X_train)
X_test_df = pd.DataFrame(X_test)
```



```
[24]: X_train_df.head()
```

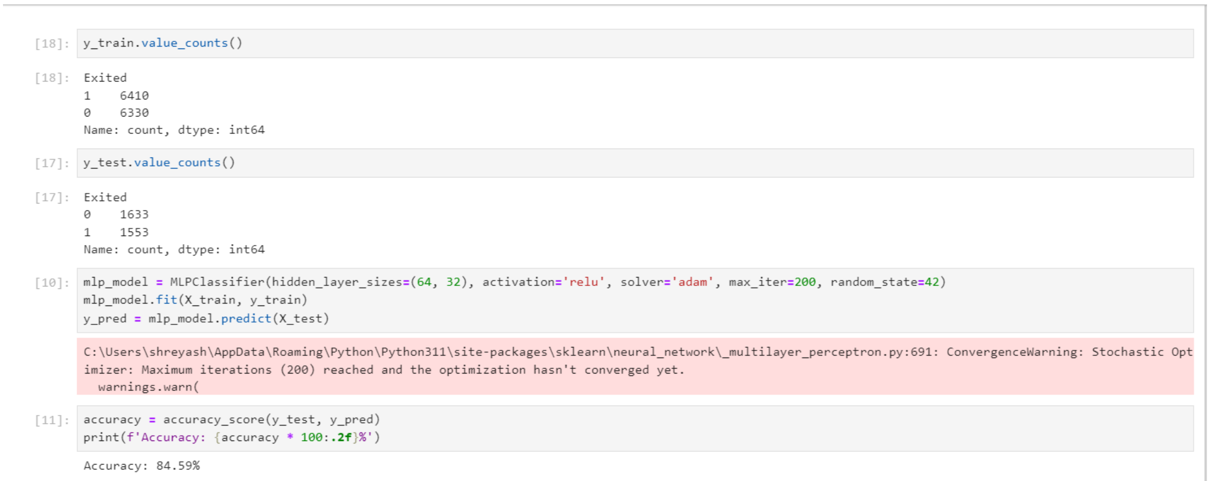
[24]:

	0	1	2	3	4	5	6	7	8	9	10
0	1.465814	0.079328	1.719402	0.511082	0.729063	-1.530958	1.082515	0.692593	1.509728	-0.552686	-1.03061
1	1.035041	0.079328	-1.364387	0.773670	0.729063	-1.530958	1.082515	0.755225	-0.662371	-0.552686	0.97030
2	0.522216	-0.109329	0.691472	-1.340431	-0.758963	-1.530958	1.082515	0.918049	-0.662371	-0.552686	0.97030
3	0.101700	1.305597	1.034116	0.603456	-0.758963	-1.530958	1.082515	0.335004	1.509728	-0.552686	-1.03061
4	0.081187	0.456641	0.006186	0.506923	0.729063	0.653186	1.082515	0.673453	-0.662371	-0.552686	-1.03061

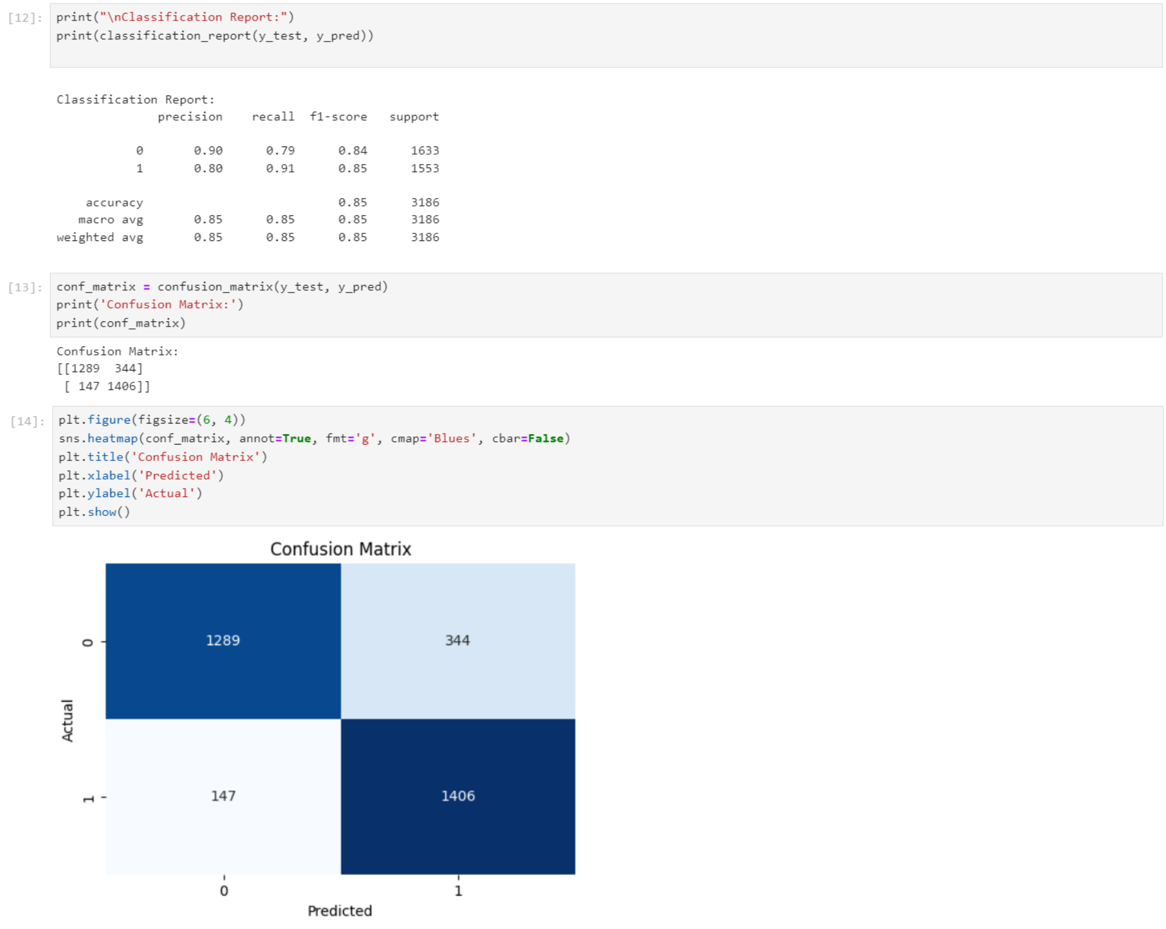
```
[25]: X_test_df.head()
```

[25]:

	0	1	2	3	4	5	6	7	8	9	10
0	-1.159848	-1.146942	0.006186	-1.340431	0.729063	-1.530958	1.082515	-0.828594	-0.662371	-0.552686	-1.03061
1	-0.431637	-0.015001	-0.336457	0.820811	-0.758963	0.653186	-0.923775	0.980827	1.509728	-0.552686	-1.03061
2	0.327343	-0.958285	1.376759	0.611566	0.729063	0.653186	1.082515	0.015894	-0.662371	1.809346	-1.03061
3	0.450421	-0.486643	-0.336457	-1.340431	-0.758963	0.653186	-0.923775	1.488334	-0.662371	-0.552686	-1.03061
4	0.388882	-0.203658	-0.336457	1.229063	0.729063	0.653186	-0.923775	1.091454	1.509728	-0.552686	-1.03061



```
[18]: y_train.value_counts()

[18]: Exited
      1    6410
      0    6330
      Name: count, dtype: int64

[17]: y_test.value_counts()

[17]: Exited
      0    1633
      1    1553
      Name: count, dtype: int64

[10]: mlp_model = MLPClassifier(hidden_layer_sizes=(64, 32), activation='relu', solver='adam', max_iter=200, random_state=42)
      mlp_model.fit(X_train, y_train)
      y_pred = mlp_model.predict(X_test)

      C:\Users\shreyash\AppData\Roaming\Python\Python311\site-packages\sklearn\neural_network\_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic Opt
      imizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
        warnings.warn(

[11]: accuracy = accuracy_score(y_test, y_pred)
      print(f'Accuracy: {accuracy * 100:.2f}%')

      Accuracy: 84.59%
```



```
[12]: print("\nClassification Report:")
      print(classification_report(y_test, y_pred))


      Classification Report:
                    precision    recall  f1-score   support

                 0       0.90      0.79      0.84      1633
                 1       0.80      0.91      0.85      1553

          accuracy                           0.85      3186
         macro avg       0.85      0.85      0.85      3186
      weighted avg       0.85      0.85      0.85      3186

[13]: conf_matrix = confusion_matrix(y_test, y_pred)
      print('Confusion Matrix:')
      print(conf_matrix)

      Confusion Matrix:
      [[1289  344]
       [ 147 1406]]

[14]: plt.figure(figsize=(6, 4))
      sns.heatmap(conf_matrix, annot=True, fmt='g', cmap='Blues', cbar=False)
      plt.title('Confusion Matrix')
      plt.xlabel('Predicted')
      plt.ylabel('Actual')
      plt.show()
```