Importing Library

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
sns.set_theme(color_codes=True)
```

Preparing Dataset

```
df = pd.read_csv('Bank Customer Churn Prediction.csv')
df.head()
```

	customer_id	credit_score	country	gender	age	tenure	balance	pr
0	15634602	619	France	Female	42	2	0.00	
1	15647311	608	Spain	Female	41	1	83807.86	
2	15619304	502	France	Female	42	8	159660.80	
3	15701354	699	France	Female	39	1	0.00	
4	15737888	850	Spain	Female	43	2	125510.82	

Exploratory Data Analysis

```
df.isnull().head()
```

	customer_id	credit_score	country	gender	age	tenure	balance	pr
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	

```
1 df.isnull().sum()
```

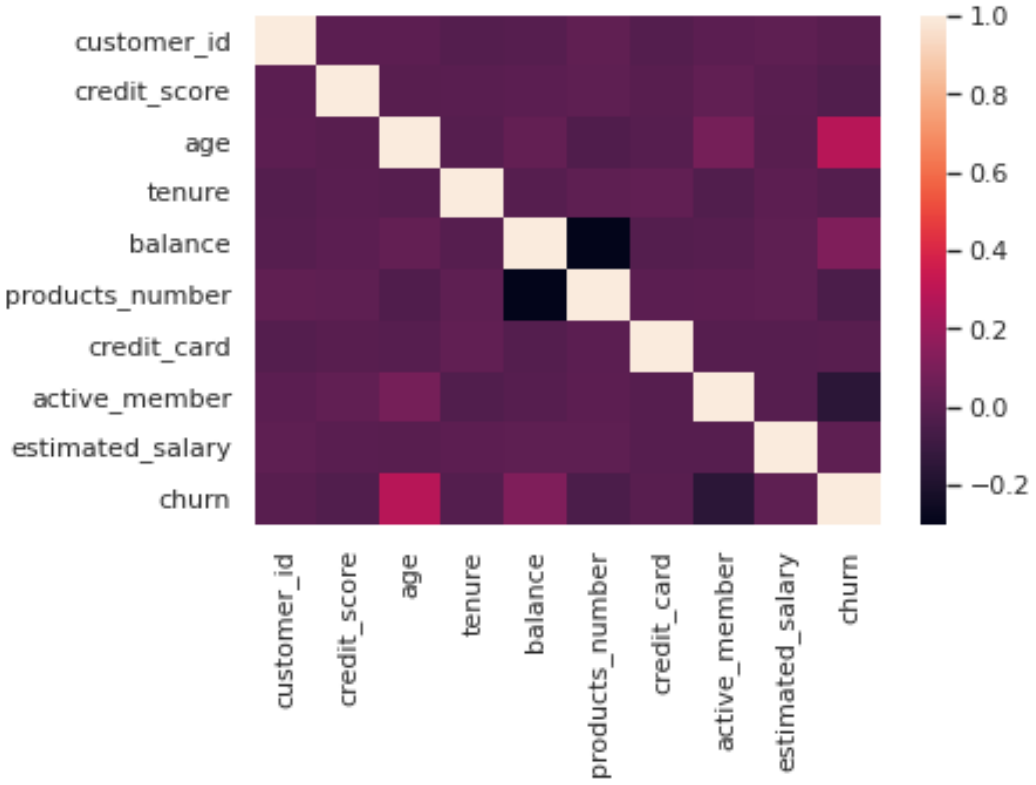
```
customer_id         0
credit_score        0
country             0
gender              0
age                 0
tenure              0
balance             0
products_number     0
credit_card         0
active_member       0
estimated_salary    0
churn               0
dtype: int64
```

```
1 df.dtypes
```

```
customer_id           int64
credit_score          int64
country              object
gender               object
age                   int64
tenure                int64
balance             float64
products_number       int64
credit_card           int64
active_member         int64
estimated_salary    float64
churn                 int64
dtype: object
```

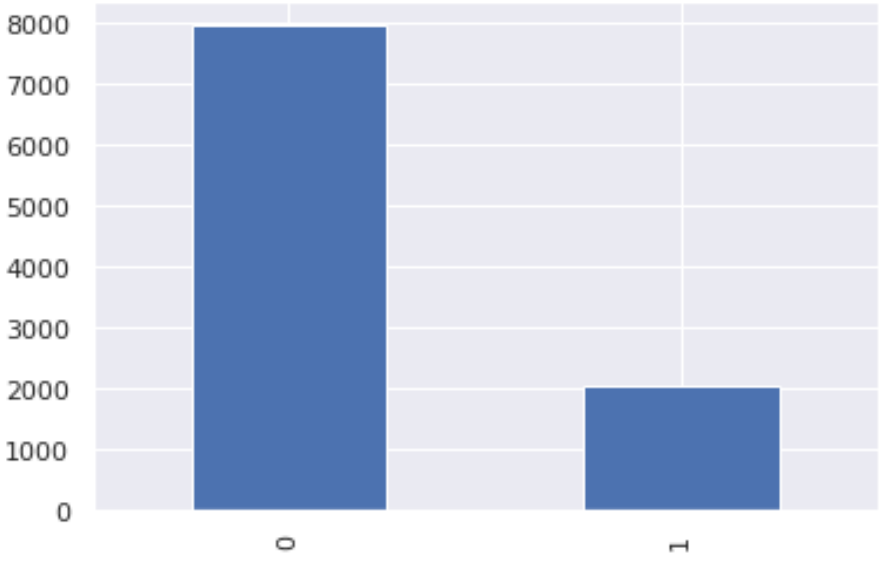
```
1 sns.heatmap(df.corr(), fmt='.2g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a8c45de50>
```




```
1 #Counting 1 and 0 Value in Churn column
2 color_wheel = {1: "#0392cf", 2: "#7bc043"}
3 colors = df["churn"].map(lambda x: color_wheel.get(x + 1))
4 print(df.churn.value_counts())
5 p=df.churn.value_counts().plot(kind="bar")
```

```
0    7963
1    2037
Name: churn, dtype: int64
```



```
#Change value in country column
df['country'] = df['country'].replace(['Germany'],'0')
df['country'] = df['country'].replace(['France'],'1')
df['country'] = df['country'].replace(['Spain'],'2')

#Change value in gender column
df['gender'] = df['gender'].replace(['Female'],'0')
df['gender'] = df['gender'].replace(['Male'],'1')

df.head()
```

	customer_id	credit_score	country	gender	age	tenure	balance	pr
0	15634602	619	1	0	42	2	0.00	
1	15647311	608	2	0	41	1	83807.86	
2	15619304	502	1	0	42	8	159660.80	
3	15701354	699	1	0	39	1	0.00	
4	15737888	850	2	0	43	2	125510.82	

```
#convert object data types column to integer
df['country'] = pd.to_numeric(df['country'])
df['gender'] = pd.to_numeric(df['gender'])
df.dtypes
```

```
customer_id          int64
credit_score         int64
country              int64
gender               int64
age                  int64
tenure               int64
balance            float64
products_number      int64
credit_card          int64
active_member        int64
estimated_salary   float64
```

```
churn                  int64
dtype: object
```

```
#Remove customer_id column
df2 = df.drop('customer_id', axis=1)
df2.head()
```

	credit_score	country	gender	age	tenure	balance	products_number
0	619	1	0	42	2	0.00	1
1	608	2	0	41	1	83807.86	1
2	502	1	0	42	8	159660.80	3
3	699	1	0	39	1	0.00	2
4	850	2	0	43	2	125510.82	1

Build Machine Learning model

```
X = df2.drop('churn', axis=1)
y = df2['churn']
```

```
#test size 20% and train size 80%
from sklearn.model_selection import train_test_split, cross_val_score, cross
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,randc
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
y_pred = dtree.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 79.2 %
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
y_pred = rfc.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 86.1 %
```

Support Vector Machine

```
from sklearn import svm

svm = svm.SVC()
svm.fit(X_train, y_train)
```

```
SVC()
```

```
y_pred = svm.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 79.45 %
```

XGBoost

```
from xgboost import XGBClassifier

xgb_model = XGBClassifier()
xgb_model.fit(X_train, y_train)
```

```
XGBClassifier()
```

```
y_pred = xgb_model.predict(X_test)
print("Accuracy Score :", accuracy_score(y_test, y_pred)*100, "%")
```

```
Accuracy Score : 86.45 %
```

Visualize Random Forest and XGBoost Algorithm because Random Forest and XGBoost Algorithm has the best accuracy

```
from sklearn.metrics import classification_report, confusion_matrix
```

Random Forest

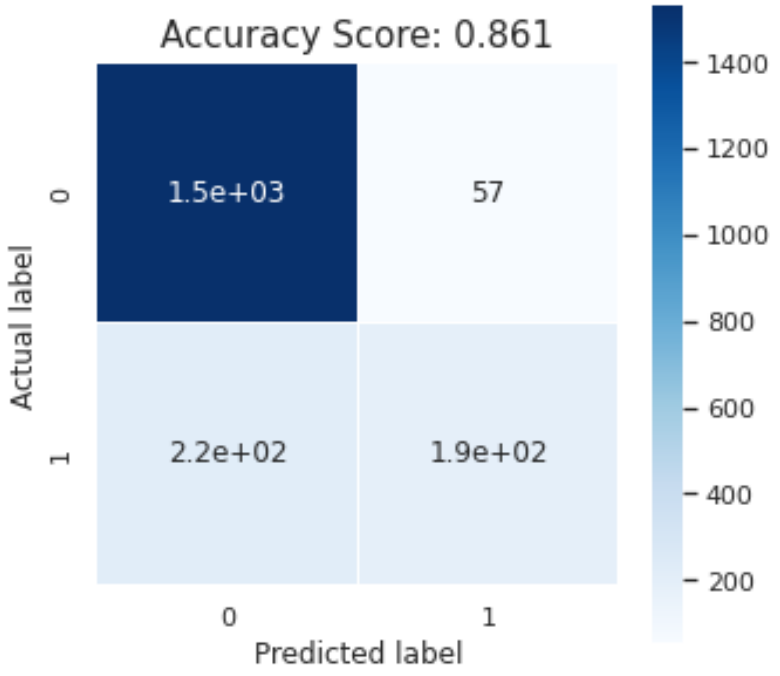
```
y_pred = rfc.predict(X_test)
print("Classification report - \n", classification_report(y_test,y_pred))
```

```
Classification report -
               precision    recall  f1-score   support

           0       0.87      0.96      0.92      1589
           1       0.77      0.46      0.58       411

    accuracy                           0.86      2000
```

```
       macro avg       0.82      0.71      0.75      2000
    weighted avg       0.85      0.86      0.85      2000
```

```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues'
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(rfc.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

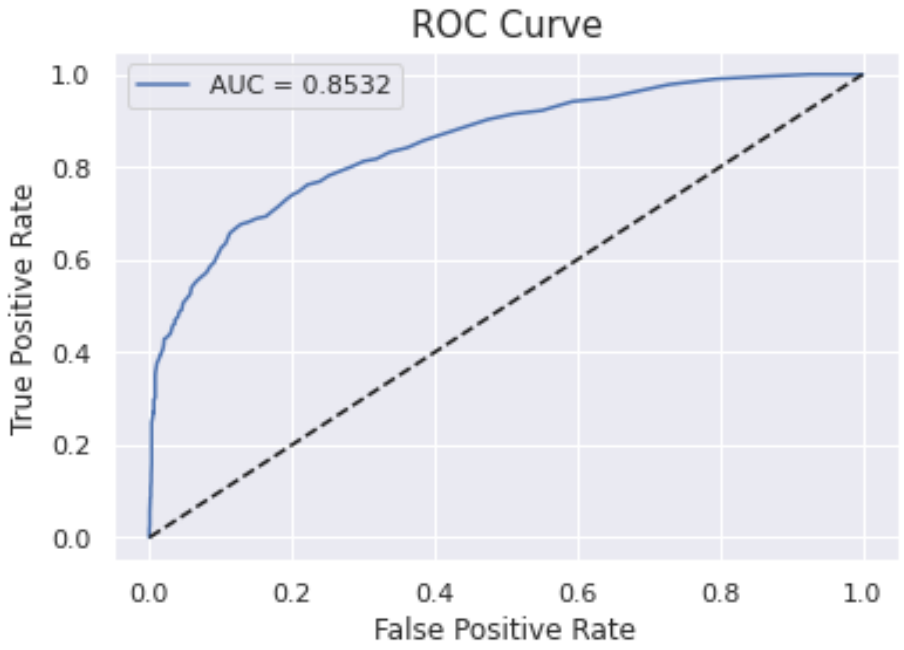
```
Text(0.5, 1.0, 'Accuracy Score: 0.861')
```




```
from sklearn.metrics import roc_curve, roc_auc_score
y_pred_proba = rfc.predict_proba(X_test)[:][:,1]

df_actual_predicted = pd.concat([pd.DataFrame(np.array(y_test), columns=['y_
df_actual_predicted.index = y_test.index

fpr, tpr, tr = roc_curve(df_actual_predicted['y_actual'], df_actual_predicte
auc = roc_auc_score(df_actual_predicted['y_actual'], df_actual_predicted['y_

plt.plot(fpr, tpr, label='AUC = %0.4f' %auc)
plt.plot(fpr, fpr, linestyle = '--', color='k')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
14 plt.title('ROC Curve', size = 15)
15 plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f2a7f3acd90>
```



XGBoost

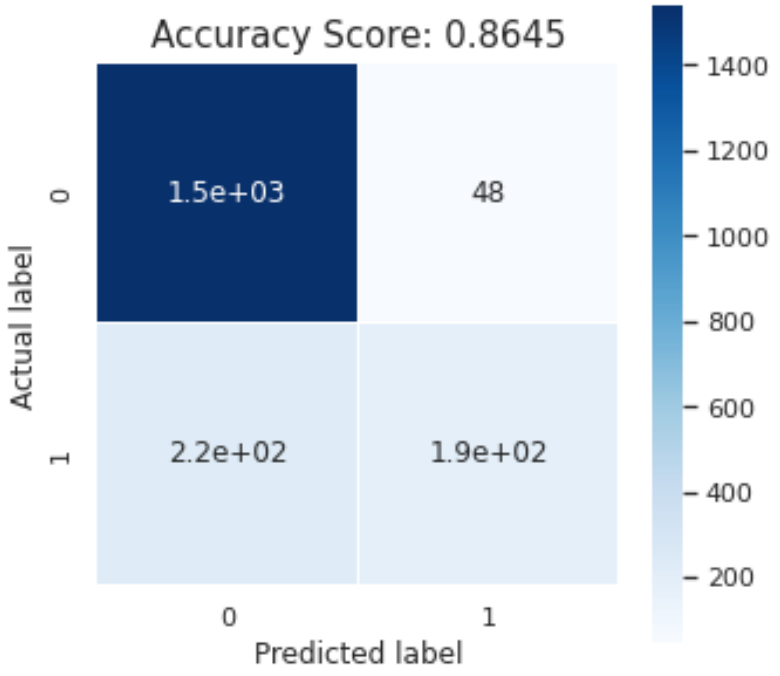
```
1 y_pred = xgb_model.predict(X_test)
2 print("Classification report - \n", classification_report(y_test,y_pred))
```

```
Classification report -
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      1589
           1       0.80      0.46      0.58       411

    accuracy                           0.86      2000
   macro avg       0.84      0.71      0.75      2000
weighted avg       0.86      0.86      0.85      2000
```

```
1 cm = confusion_matrix(y_test, y_pred)
2 plt.figure(figsize=(5,5))
3 sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues'
```

```
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(xgb_model.score(X_test, y_te
plt.title(all_sample_title, size = 15)
```

```
Text(0.5, 1.0, 'Accuracy Score: 0.8645')
```




```
from sklearn.metrics import roc_curve, roc_auc_score
y_pred_proba = xgb_model.predict_proba(X_test)[:][:,1]

df_actual_predicted = pd.concat([pd.DataFrame(np.array(y_test), columns=['y_
df_actual_predicted.index = y_test.index

fpr, tpr, tr = roc_curve(df_actual_predicted['y_actual'], df_actual_predicte
auc = roc_auc_score(df_actual_predicted['y_actual'], df_actual_predicted['y_

plt.plot(fpr, tpr, label='AUC = %0.4f' %auc)
plt.plot(fpr, fpr, linestyle = '--', color='k')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve', size = 15)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f2a7f2c7850>
```