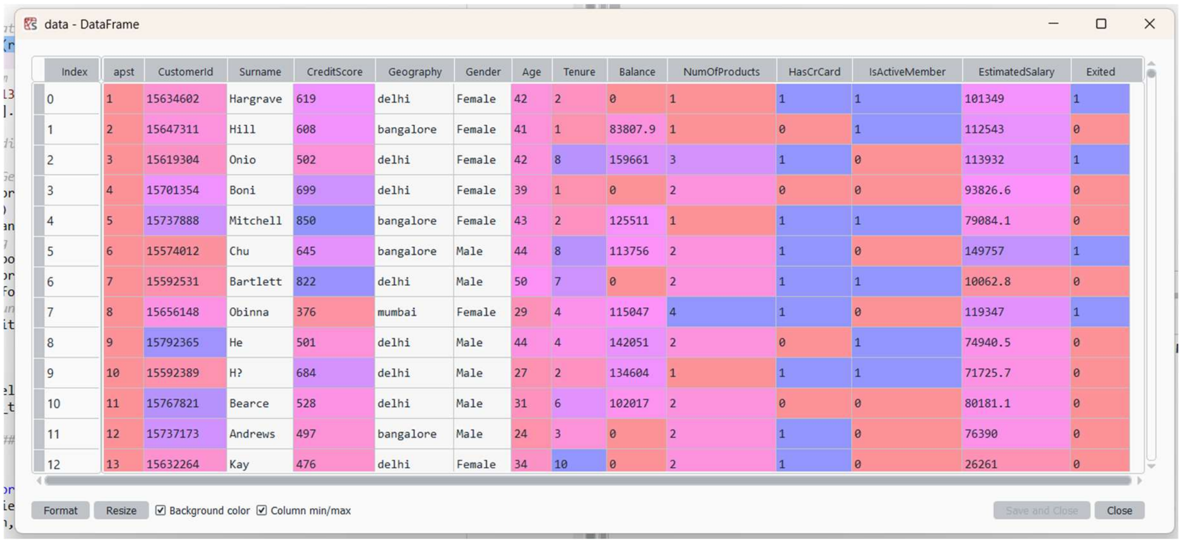
XGBoost-Classifier

#lets import libraries

import pandas as pd

#lets read the dataset

data=pd.read_csv(r"C:\Users\TharunMahendra\NIT\6.Algorithms\2.Classification\Churn_Modelling.csv")

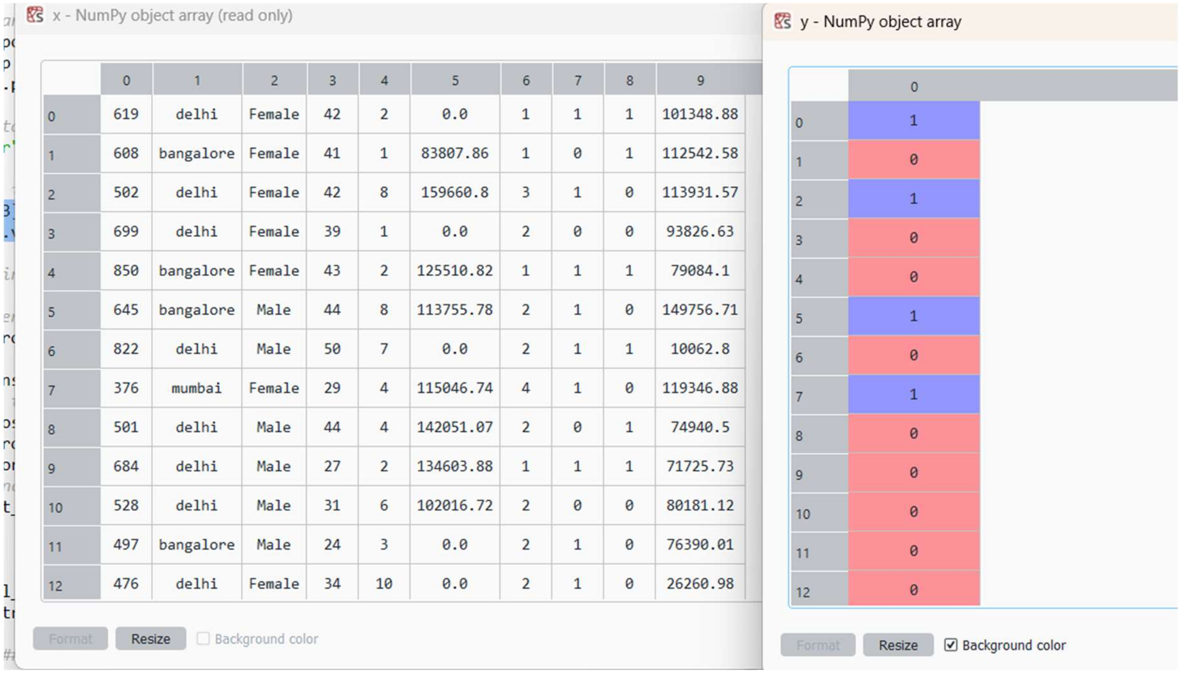


Index	apst	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	delhi	Female	42	2	0	1	1	1	101349	1
1	2	15647311	Hill	608	bangalore	Female	41	1	83807.9	1	0	1	112543	0
2	3	15619304	Onio	502	delhi	Female	42	8	159661	3	1	0	113932	1
3	4	15701354	Boni	699	delhi	Female	39	1	0	2	0	0	93826.6	0
4	5	15737888	Mitchell	850	bangalore	Female	43	2	125511	1	1	1	79084.1	0
5	6	15574012	Chu	645	bangalore	Male	44	8	113756	2	1	0	149757	1
6	7	15592531	Bartlett	822	delhi	Male	50	7	0	2	1	1	10062.8	0
7	8	15656148	Obinna	376	mumbai	Female	29	4	115047	4	1	0	119347	1
8	9	15792365	He	501	delhi	Male	44	4	142051	2	0	1	74940.5	0
9	10	15592389	H?	684	delhi	Male	27	2	134604	1	1	1	71725.7	0
10	11	15767821	Bearce	528	delhi	Male	31	6	102017	2	0	0	80181.1	0
11	12	15737173	Andrews	497	bangalore	Male	24	3	0	2	1	0	76390	0
12	13	15632264	Kay	476	delhi	Female	34	10	0	2	1	0	26261	0

#lets divide them into dependent & independent

x=data.iloc[:,3:13].values

y=data.iloc[:,-1].values


x - NumPy object array (read only)

	0	1	2	3	4	5	6	7	8	9
0	619	delhi	Female	42	2	0.0	1	1	1	101348.88
1	608	bangalore	Female	41	1	83807.86	1	0	1	112542.58
2	502	delhi	Female	42	8	159660.8	3	1	0	113931.57
3	699	delhi	Female	39	1	0.0	2	0	0	93826.63
4	850	bangalore	Female	43	2	125510.82	1	1	1	79084.1
5	645	bangalore	Male	44	8	113755.78	2	1	0	149756.71
6	822	delhi	Male	50	7	0.0	2	1	1	10062.8
7	376	mumbai	Female	29	4	115046.74	4	1	0	119346.88
8	501	delhi	Male	44	4	142051.07	2	0	1	74940.5
9	684	delhi	Male	27	2	134603.88	1	1	1	71725.73
10	528	delhi	Male	31	6	102016.72	2	0	0	80181.12
11	497	bangalore	Male	24	3	0.0	2	1	0	76390.01
12	476	delhi	Female	34	10	0.0	2	1	0	26260.98

y - NumPy object array

	0
0	1
1	0
2	1
3	0
4	0
5	1
6	0
7	1
8	0
9	0
10	0
11	0
12	0

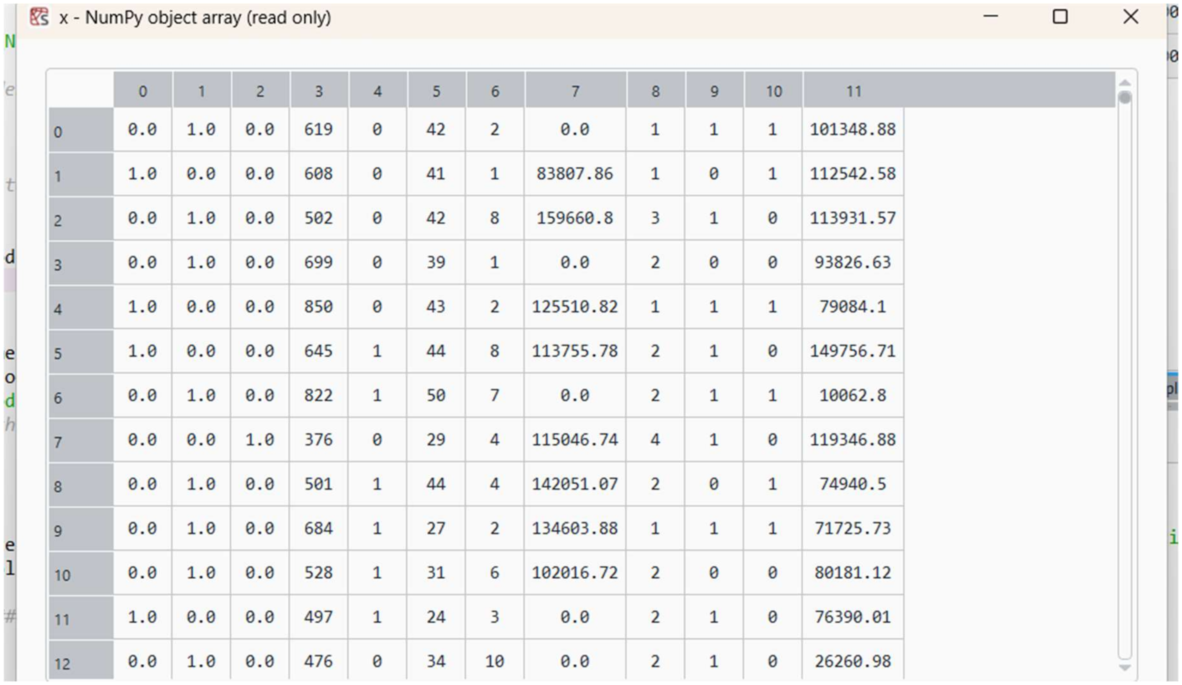
#Converting/Encoding categorical variables to numerical

# LabelEncoding Gender column

```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

x[:,2]=le.fit_transform(x[:,2])
```

# OneHot Encoding the Geography

```
from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')# other columns unchanged (remainder='passthrough').

x= np.array(ct.fit_transform(x))
```

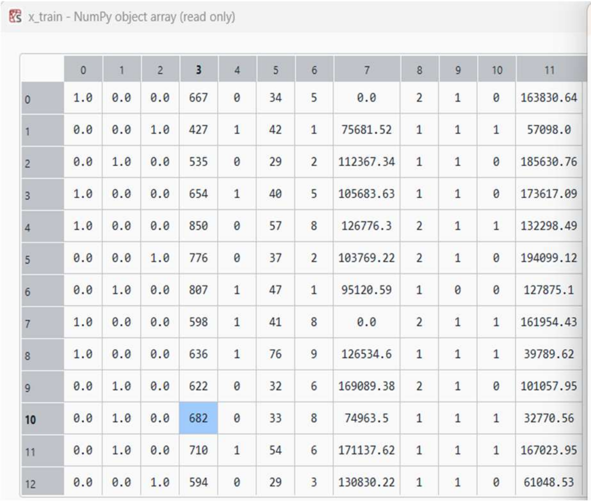


x - NumPy object array (read only)

	0	1	2	3	4	5	6	7	8	9	10	11
0	0.0	1.0	0.0	619	0	42	2	0.0	1	1	1	101348.88
1	1.0	0.0	0.0	608	0	41	1	83807.86	1	0	1	112542.58
2	0.0	1.0	0.0	502	0	42	8	159660.8	3	1	0	113931.57
3	0.0	1.0	0.0	699	0	39	1	0.0	2	0	0	93826.63
4	1.0	0.0	0.0	850	0	43	2	125510.82	1	1	1	79084.1
5	1.0	0.0	0.0	645	1	44	8	113755.78	2	1	0	149756.71
6	0.0	1.0	0.0	822	1	50	7	0.0	2	1	1	10062.8
7	0.0	0.0	1.0	376	0	29	4	115046.74	4	1	0	119346.88
8	0.0	1.0	0.0	501	1	44	4	142051.07	2	0	1	74940.5
9	0.0	1.0	0.0	684	1	27	2	134603.88	1	1	1	71725.73
10	0.0	1.0	0.0	528	1	31	6	102016.72	2	0	0	80181.12
11	1.0	0.0	0.0	497	1	24	3	0.0	2	1	0	76390.01
12	0.0	1.0	0.0	476	0	34	10	0.0	2	1	0	26260.98

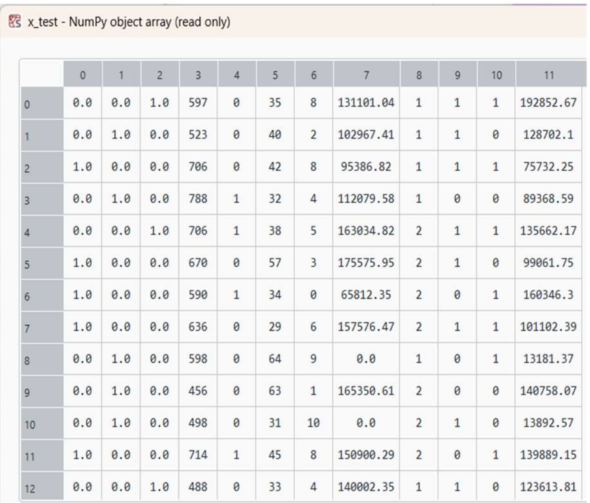
#splitting data

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20, random_state=0)
```



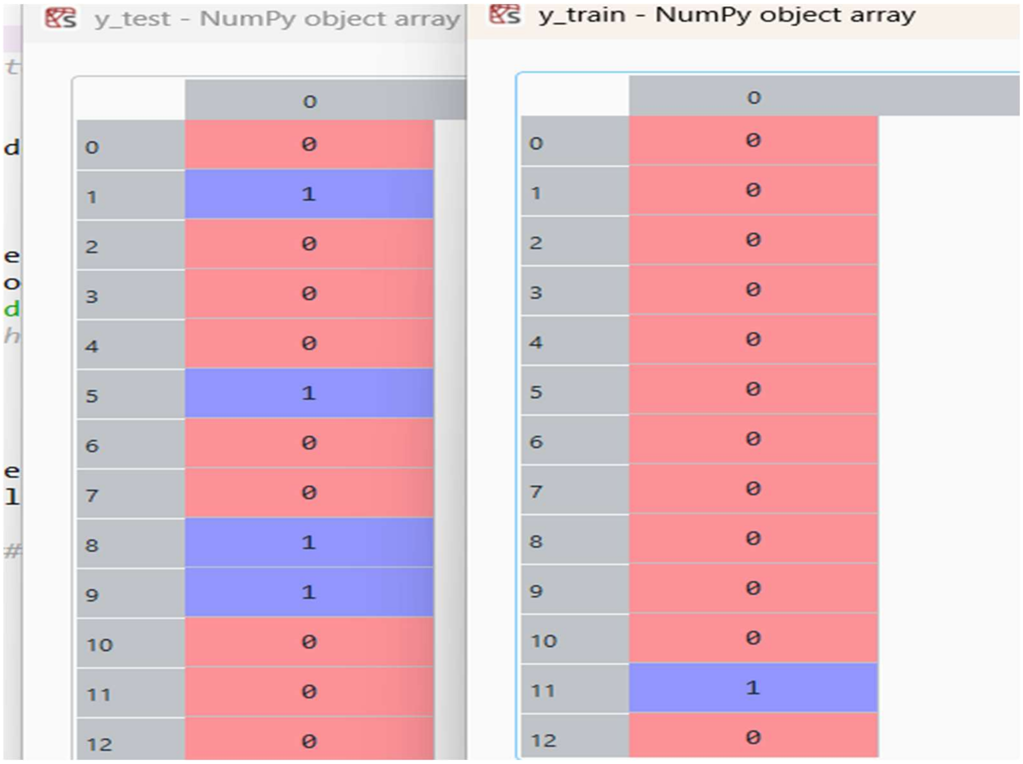
x_train - NumPy object array (read only)

	0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	0.0	0.0	667	0	34	5	0.0	2	1	0	163830.64
1	0.0	0.0	1.0	427	1	42	1	75681.52	1	1	1	57098.0
2	0.0	1.0	0.0	535	0	29	2	112367.34	1	1	0	185630.76
3	1.0	0.0	0.0	654	1	40	5	105683.63	1	1	0	173617.09
4	1.0	0.0	0.0	850	0	57	8	126776.3	2	1	1	132298.49
5	0.0	0.0	1.0	776	0	37	2	103769.22	2	1	0	194099.12
6	0.0	1.0	0.0	807	1	47	1	95120.59	1	0	0	127875.1
7	1.0	0.0	0.0	598	1	41	8	0.0	2	1	1	161954.43
8	1.0	0.0	0.0	636	1	76	9	126534.6	1	1	1	39789.62
9	0.0	1.0	0.0	622	0	32	6	169089.38	2	1	0	101057.95
10	0.0	1.0	0.0	682	0	33	8	74963.5	1	1	1	32770.56
11	0.0	1.0	0.0	710	1	54	6	171137.62	1	1	1	167023.95
12	0.0	0.0	1.0	594	0	29	3	130830.22	1	1	0	61048.53



x_test - NumPy object array (read only)

	0	1	2	3	4	5	6	7	8	9	10	11
0	0.0	0.0	1.0	597	0	35	8	131101.04	1	1	1	192852.67
1	0.0	1.0	0.0	523	0	40	2	102967.41	1	1	0	128702.1
2	1.0	0.0	0.0	706	0	42	8	95386.82	1	1	1	75732.25
3	0.0	1.0	0.0	788	1	32	4	112079.58	1	0	0	89368.59
4	0.0	0.0	1.0	706	1	38	5	163034.82	2	1	1	135662.17
5	1.0	0.0	0.0	670	0	57	3	175575.95	2	1	0	99061.75
6	1.0	0.0	0.0	590	1	34	0	65812.35	2	0	1	160346.3
7	1.0	0.0	0.0	636	0	29	6	157576.47	2	1	1	101102.39
8	0.0	1.0	0.0	598	0	64	9	0.0	1	0	1	13181.37
9	0.0	1.0	0.0	456	0	63	1	165350.61	2	0	0	140758.07
10	0.0	1.0	0.0	498	0	31	10	0.0	2	1	0	13892.57
11	1.0	0.0	0.0	714	1	45	8	150900.29	2	0	1	139889.15
12	0.0	0.0	1.0	488	0	33	4	140002.35	1	1	0	123613.81



y_test - NumPy object array

	0
0	0
1	1
2	0
3	0
4	0
5	1
6	0
7	0
8	1
9	1
10	0
11	0
12	0

y_train - NumPy object array

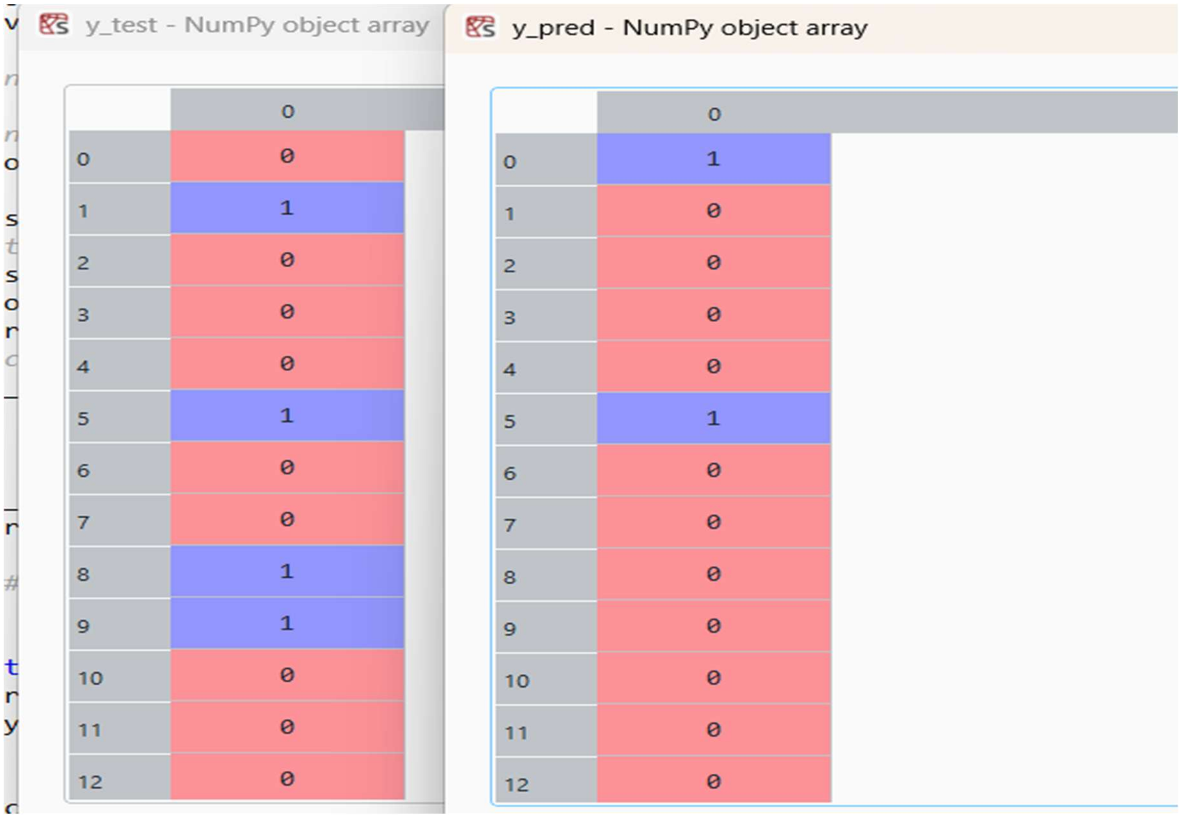
	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	1
12	0

#model building

```
  from xgboost import XGBClassifier
  model=XGBClassifier()
  model.fit(x_train,y_train)
```

#prediction

```
y_pred=model.predict(x_test)
```



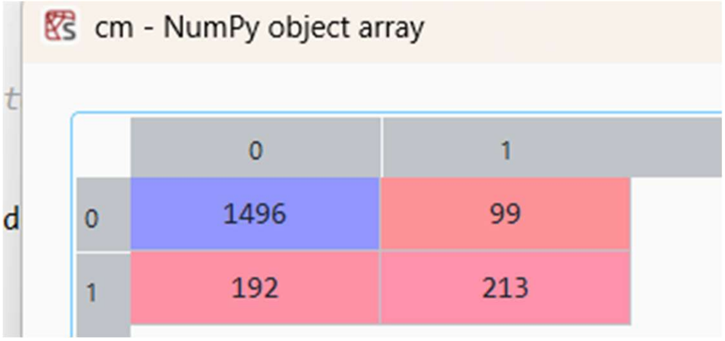
	0
0	0
1	1
2	0
3	0
4	0
5	1
6	0
7	0
8	1
9	1
10	0
11	0
12	0

	0
0	1
1	0
2	0
3	0
4	0
5	1
6	0
7	0
8	0
9	0
10	0
11	0
12	0

#confusion Matrix

from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test, y_pred)



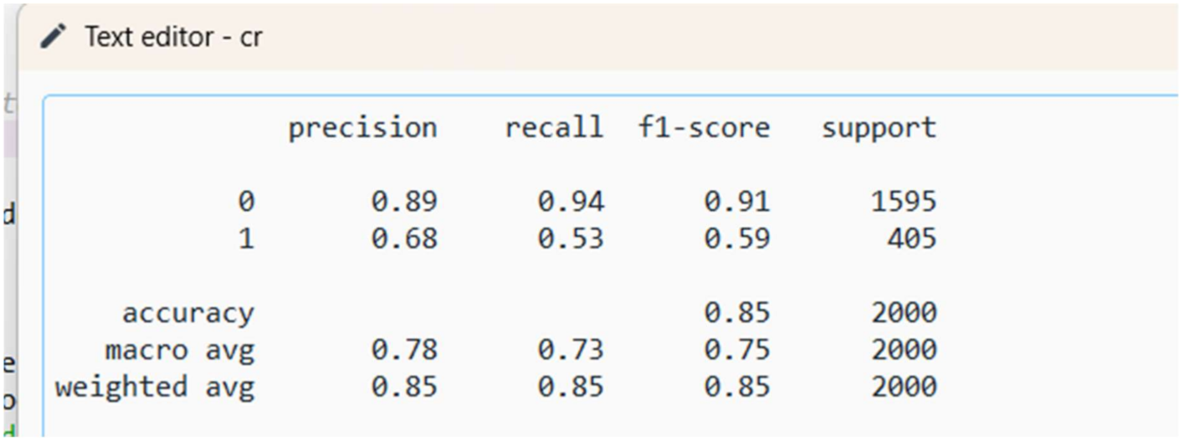
	0	1
0	1496	99
1	192	213

from sklearn.metrics import accuracy_score

ac=**accuracy_score**(y_test,y_pred) **->0.8545** from sklearn.metrics import

classification_report

cr=classification_report(y_test,y_pred)



```
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      1595
           1       0.68      0.53      0.59       405

    accuracy                           0.85      2000
   macro avg       0.78      0.73      0.75      2000
weighted avg       0.85      0.85      0.85      2000
```

bias=model.score(x_train,y_train) **-> 0.953875**

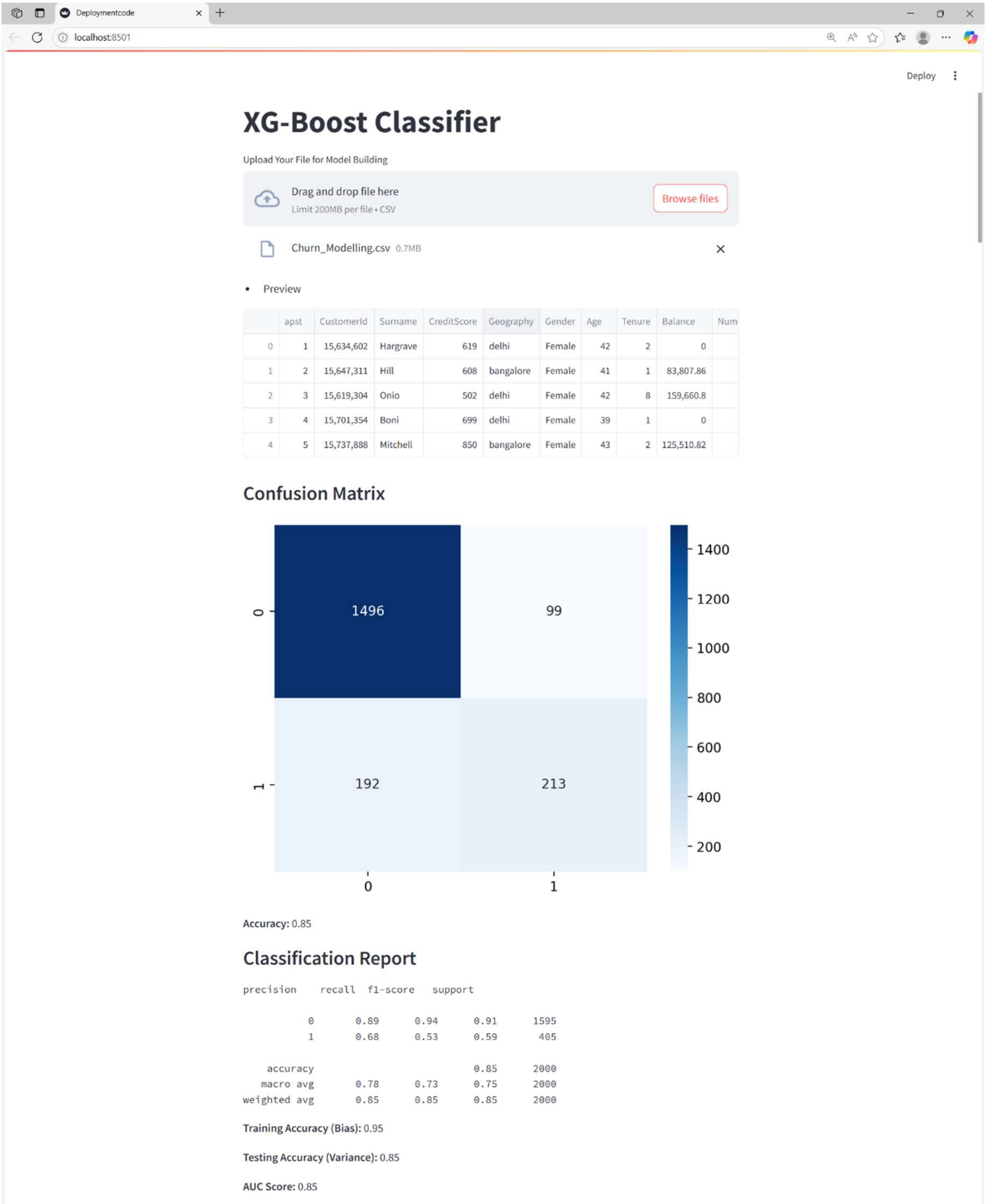
variance=model.score(x_test,y_test)-**> 0.8545**

Deployment Code

```
# importing libraries
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.preprocessing import OneHotEncoder,LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import(confusion_matrix,accuracy_score,classification_report,roc_curve,roc_auc_score)
import seaborn as sns

st.title("XG-Boost Classifier")

# Uploading File
file=st.file_uploader('Upload Your File for Model Building',type=['csv'])
if file is not None:
    # Load
    data=pd.read_csv(file)
    st.write('- Preview')
    st.dataframe(data.head())
```

```
# FeatureSelection
x=data.iloc[:,3:13].values
y=data.iloc[:,-1].values

# Converting Categorical to Numerical
le=LabelEncoder()
x[:,2]=le.fit_transform(x[:,2])

ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[1])],remainder='passthrough')
x=np.array(ct.fit_transform(x))

# SplittingData
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)

# model building
model=XGBClassifier()
model.fit(x_train,y_train)

# prediction
y_pred=model.predict(x_test)
y_prob=model.predict_proba(x_test)[:,1]

# Metrics
st.subheader("Confusion Matrix")
cm = confusion_matrix(y_test, y_pred)
fig_cm, ax = plt.subplots()
```

```
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", ax=ax)
    st.pyplot(fig_cm)

    ac = accuracy_score(y_test, y_pred)
    st.write(f"**Accuracy:** {ac:.2f}")

    st.subheader("Classification Report")
    st.text(classification_report(y_test, y_pred))

    st.write(f"**Training Accuracy (Bias):** {model.score(x_train, y_train):.2f}")
    st.write(f"**Testing Accuracy (Variance):** {model.score(x_test, y_test):.2f}")

    # ROC Curve and AUC
    fpr, tpr, _ = roc_curve(y_test, y_prob)
    auc_score = roc_auc_score(y_test, y_prob)
    st.write(f"**AUC Score:** {auc_score:.2f}")

    st.subheader("ROC Curve")
    fig_roc, ax = plt.subplots()
    ax.plot(fpr, tpr, color="blue", label=f"ROC curve (AUC = {auc_score:.2f})")
    ax.plot([0, 1], [0, 1], color="gray", linestyle="--")
    ax.set_xlabel("False Positive Rate")
    ax.set_ylabel("True Positive Rate")
    ax.set_title("ROC Curve")
    ax.legend(loc="lower right")
    st.pyplot(fig_roc)
```



XG-Boost Classifier

Upload Your File for Model Building

- Preview

	apst	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	Num
0	1	15,634,602	Hargrave	619	delhi	Female	42	2	0	
1	2	15,647,311	Hill	608	bangalore	Female	41	1	83,807.86	
2	3	15,619,304	Onio	502	delhi	Female	42	8	159,660.8	
3	4	15,701,354	Boni	699	delhi	Female	39	1	0	
4	5	15,737,888	Mitchell	850	bangalore	Female	43	2	125,510.82	

Confusion Matrix

Accuracy: 0.85

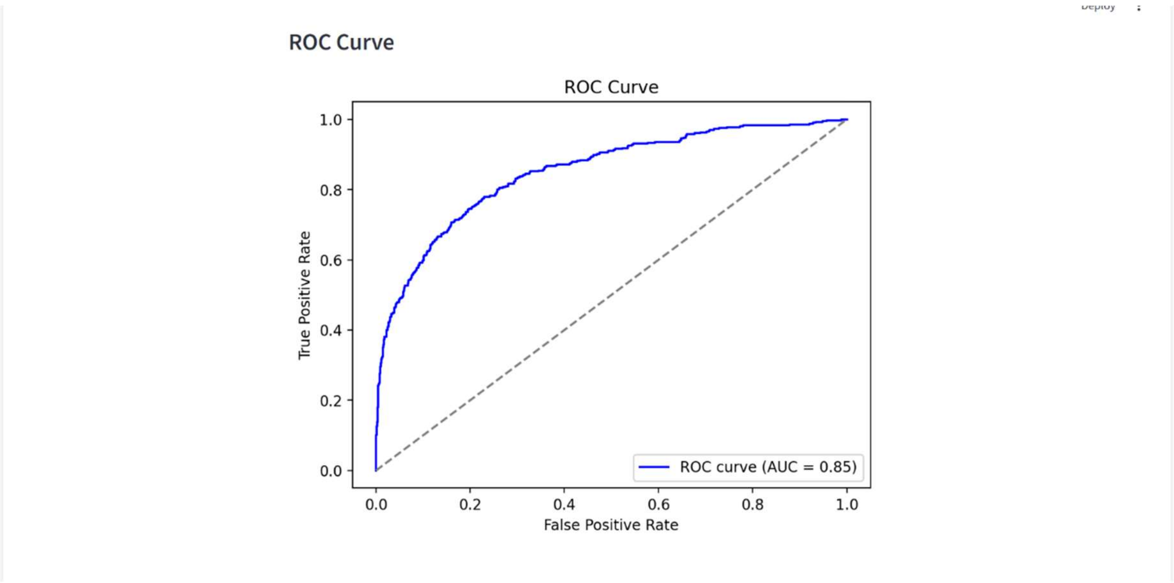
Classification Report

```
precision    recall  f1-score   support

           0       0.89      0.94      0.91      1595
           1       0.68      0.53      0.59       405

    accuracy                           0.85      2000
   macro avg       0.78      0.73      0.75      2000
weighted avg       0.85      0.85      0.85      2000
```

Training Accuracy (Bias): 0.95

Testing Accuracy (Variance): 0.85

AUC Score: 0.85



Deploy
ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.85)

ROC Curve

Importing the libraries

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [2]:

```
dataset = pd.read_csv(r"C:\Users\TharunMahendra\NIT\6.Algorithms\2.Classification\Churn_Mod
X = dataset.iloc[:, 3:-1].values
y = dataset.iloc[:, -1].values
```

In [3]:

```
print(X)
```

```
[[619 'delhi' 'Female' ... 1 1 101348.88]
 [608 'bangalore' 'Female' ... 0 1 112542.58]
 [502 'delhi' 'Female' ... 1 0 113931.57]
 ...
 [709 'delhi' 'Female' ... 0 1 42085.58]
 [772 'mumbai' 'Male' ... 1 0 92888.52]
 [792 'delhi' 'Female' ... 1 0 38190.78]]
```

In [4]:

```
print(y)
```

```
[1 0 1 ... 1 1 0]
```

Encoding categorical data

Label Encoding the "Gender" column

In [5]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
```

In [6]:

```
print(X)
```

```
[[619 'delhi' 0 ... 1 1 101348.88]
 [608 'bangalore' 0 ... 0 1 112542.58]
 [502 'delhi' 0 ... 1 0 113931.57]
 ...
 [709 'delhi' 0 ... 0 1 42085.58]
 [772 'mumbai' 1 ... 1 0 92888.52]
 [792 'delhi' 0 ... 1 0 38190.78]]
```

One Hot Encoding the "Geography" column

In [7]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthr
X = np.array(ct.fit_transform(X))
```

In [8]:

```
print(X)
```

```
[[0.0 1.0 0.0 ... 1 1 101348.88]
 [1.0 0.0 0.0 ... 0 1 112542.58]
 [0.0 1.0 0.0 ... 1 0 113931.57]
 ...
 [0.0 1.0 0.0 ... 0 1 42085.58]
 [0.0 0.0 1.0 ... 1 0 92888.52]
 [0.0 1.0 0.0 ... 1 0 38190.78]]
```

Splitting the dataset into the Training set and Test set

In [9]:

```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 = 0
```

Training XGBoost on the Training set

In [10]:

```python
from xgboost import XGBClassifier
classifier = XGBClassifier()
classifier.fit(X_train, y_train)
```

Out[10]:

▾ XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              feature_weights=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
              max_leaves=None, min_child_weight=None, missing=nan,
```

Predicting the Test set results

In [11]:

```python
y_pred = classifier.predict(X_test)
```

Making the Confusion Matrix

In [12]:

```python
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[1496   99]
 [ 192  213]]
```