# modelpart1

April 24, 2025

### 0.0.1 Objective:

Build different models and evaluate each one of them using different metrics.

[1]:

```
## import essential libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from functions_model import *
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

[2]:

```
## Reading and printing csv file
df = pd.read_csv("FileModel.csv")
df.head()
```

[2]:

```
   Gender  Age  Academic Pressure  CGPA  Study Satisfaction   Degree  \
0    Male   33                  5  8.97                   2  B.Pharm
1  Female   24                  2  5.90                   5      BSc
2    Male   31                  3  7.03                   5       BA
3  Female   28                  3  5.59                   2      BCA
4  Female   25                  4  8.13                   3   M.Tech

  Have you ever had suicidal thoughts ?  Work/Study Hours  Financial Stress  \
0                                   Yes                 3                 1
1                                    No                 3                 2
2                                    No                 9                 1
3                                   Yes                 4                 5
4                                   Yes                 1                 1

  Family History of Mental Illness               State  Sleep Duration_encoded  \
0                               No      Andhra Pradesh                       1
1                              Yes           Karnataka                       1
2                              Yes   Jammu and Kashmir                       0
3                              Yes       Uttar Pradesh                       3
4                               No           Rajasthan                       1
```

```
   Dietary Habits_encoded  Depression
0                       2           1
1                       1           0
2                       2           0
3                       1           1
4                       1           0
```

[3]:

```
print(df.dtypes)
```

```
Gender                                     object
Age                                         int64
Academic Pressure                           int64
CGPA                                      float64
Study Satisfaction                          int64
Degree                                     object
Have you ever had suicidal thoughts ?      object
Work/Study Hours                            int64
Financial Stress                            int64
Family History of Mental Illness           object
State                                      object
Sleep Duration_encoded                      int64
Dietary Habits_encoded                      int64
Depression                                  int64
dtype: object
```

### 0.0.2 One Hot Encoding Columns

Columns need to be one hot encoded.

- Gender
- Degree
- Have you ever had suicidal thoughts
- Family History of Mental Illness
- State

[4]:

```
df = one_hot_encode_column(df, 'Gender')
df = one_hot_encode_column(df, 'Degree')
df = one_hot_encode_column(df, 'Have you ever had suicidal thoughts ?')
df = one_hot_encode_column(df, 'Family History of Mental Illness')
df = one_hot_encode_column(df, 'State')

df.head()
```

[4]:

```
   Age  Academic Pressure  CGPA  Study Satisfaction  Work/Study Hours  \
0   33                  5  8.97                   2                 3
1   24                  2  5.90                   5                 3
2   31                  3  7.03                   5                 9
3   28                  3  5.59                   2                 4
```

```
4   25                  4  8.13                   3                 1

   Financial Stress  Sleep Duration_encoded  Dietary Habits_encoded  \
0                 1                       1                       2
1                 2                       1                       1
2                 1                       0                       2
3                 5                       3                       1
4                 1                       1                       1

   Depression  Gender_Female  …  State_Jammu and Kashmir  State_Karnataka  \
0           1              0  …                        0                0
1           0              1  …                        0                1
2           0              0  …                        1                0
3           1              1  …                        0                0
4           0              1  …                        0                0

   State_Madhya Pradesh  State_Maharashtra  State_Punjab  State_Rajasthan  \
0                     0                  0             0                0
1                     0                  0             0                0
2                     0                  0             0                0
3                     0                  0             0                0
4                     0                  0             0                1

   State_Tamil Nadu  State_Telangana  State_Uttar Pradesh  State_West Bengal
0                 0                0                    0                  0
1                 0                0                    0                  0
2                 0                0                    0                  0
3                 0                0                    1                  0
4                 0                0                    0                  0

[5 rows x 57 columns]
```

```
[ ]: ## checking all the datatypes after one hot encoding
     print(df.dtypes)
```

```
Age                                                int64
Academic Pressure                                  int64
CGPA                                             float64
Study Satisfaction                                 int64
Work/Study Hours                                   int64
Financial Stress                                   int64
Sleep Duration_encoded                             int64
Dietary Habits_encoded                             int64
Depression                                         int64
Gender_Female                                      int64
Gender_Male                                        int64
Degree_'Class 12'                                  int64
Degree_B.Arch                                      int64
Degree_B.Com                                   int64
Degree_B.Ed                                    int64
Degree_B.Pharm                                 int64
Degree_B.Tech                                  int64
Degree_BA                                      int64
Degree_BBA                                     int64
Degree_BCA                                     int64
Degree_BE                                      int64
Degree_BHM                                     int64
Degree_BSc                                     int64
Degree_LLB                                     int64
Degree_LLM                                     int64
Degree_M.Com                                   int64
Degree_M.Ed                                    int64
Degree_M.Pharm                                 int64
Degree_M.Tech                                  int64
Degree_MA                                      int64
Degree_MBA                                     int64
Degree_MBBS                                    int64
Degree_MCA                                     int64
Degree_MD                                      int64
Degree_ME                                      int64
Degree_MHM                                     int64
Degree_MSc                                     int64
Degree_PhD                                     int64
Have you ever had suicidal thoughts ?_No       int64
Have you ever had suicidal thoughts ?_Yes      int64
Family History of Mental Illness_No            int64
Family History of Mental Illness_Yes           int64
State_Andhra Pradesh                           int64
State_Bihar                                    int64
State_Delhi                                    int64
State_Gujarat                                  int64
State_Haryana                                  int64
State_Jammu and Kashmir                        int64
State_Karnataka                                int64
State_Madhya Pradesh                           int64
State_Maharashtra                              int64
State_Punjab                                   int64
State_Rajasthan                                int64
State_Tamil Nadu                               int64
State_Telangana                                int64
State_Uttar Pradesh                            int64
State_West Bengal                              int64
dtype: object
```

### 0.0.3 Models to be used:

- Logistic Regression
- Decision Tree
- Random Forest
- SVM

### 0.0.4 Baseline Model

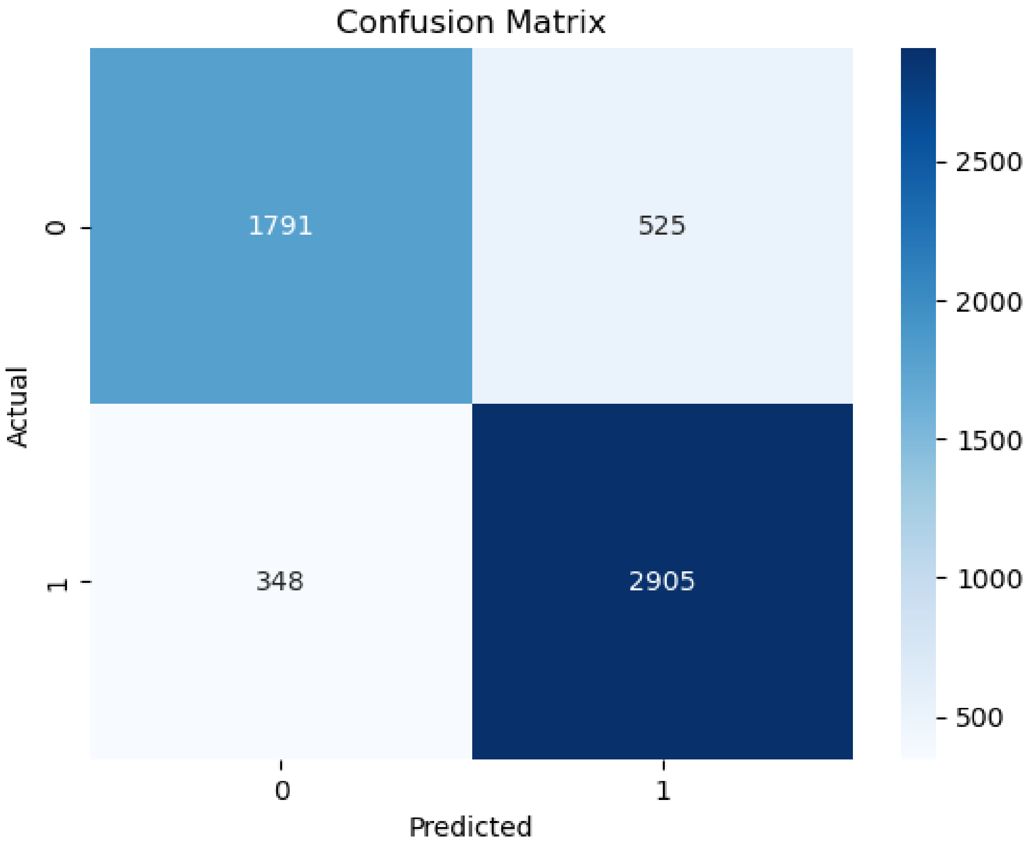
Constructing a baseline model, using Logistic Regression because we have binary target variable.

### 0.0.5 Logistic Regression

[6]:

```
X = df.drop("Depression", axis=1)
y = df["Depression"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state= 42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model1 = LogisticRegression()

model_evaulate(model1, X_train_scaled, y_train, X_test_scaled, y_test)
```

```
<IPython.core.display.HTML object>

              precision    recall  f1-score   support

           0       0.84      0.77      0.80      2316
           1       0.85      0.89      0.87      3253

    accuracy                           0.84      5569
   macro avg       0.84      0.83      0.84      5569
weighted avg       0.84      0.84      0.84      5569
```

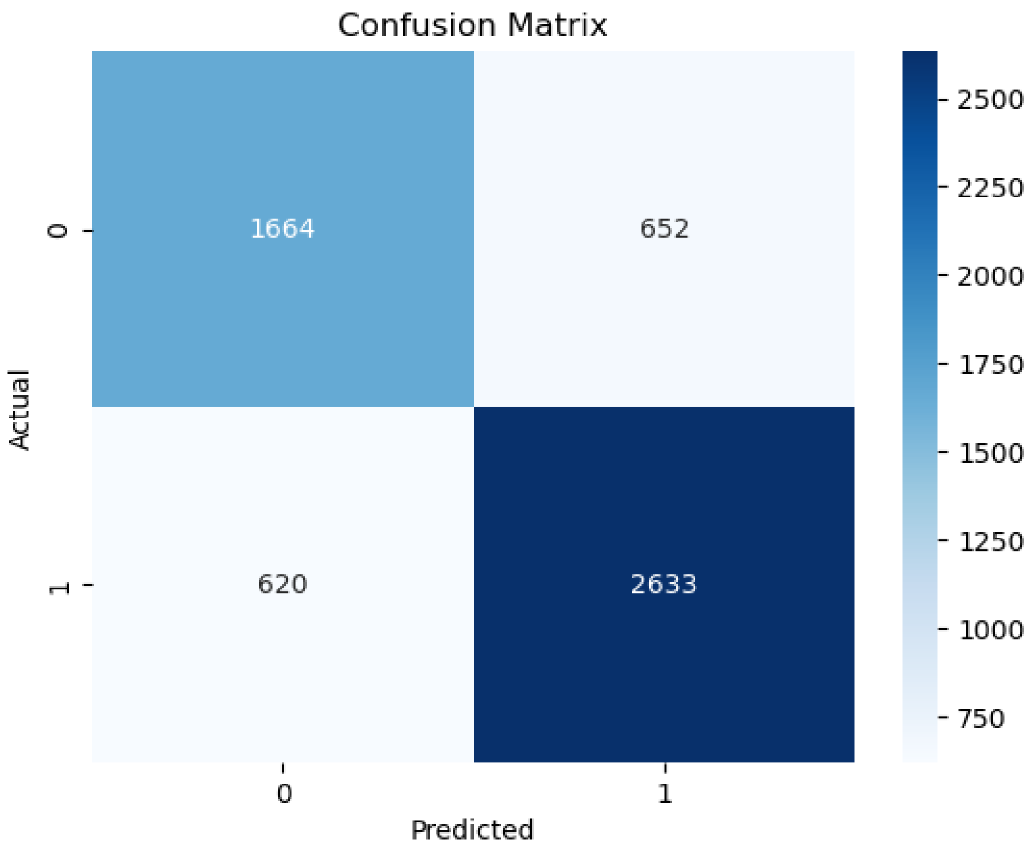


### 0.0.6 Decision Tree

```
[7]: from sklearn.tree import DecisionTreeClassifier
```

```
[8]: model2 = DecisionTreeClassifier(criterion= 'gini', random_state=42)
     model_evaulate(model2, X_train, y_train, X_test, y_test)
```

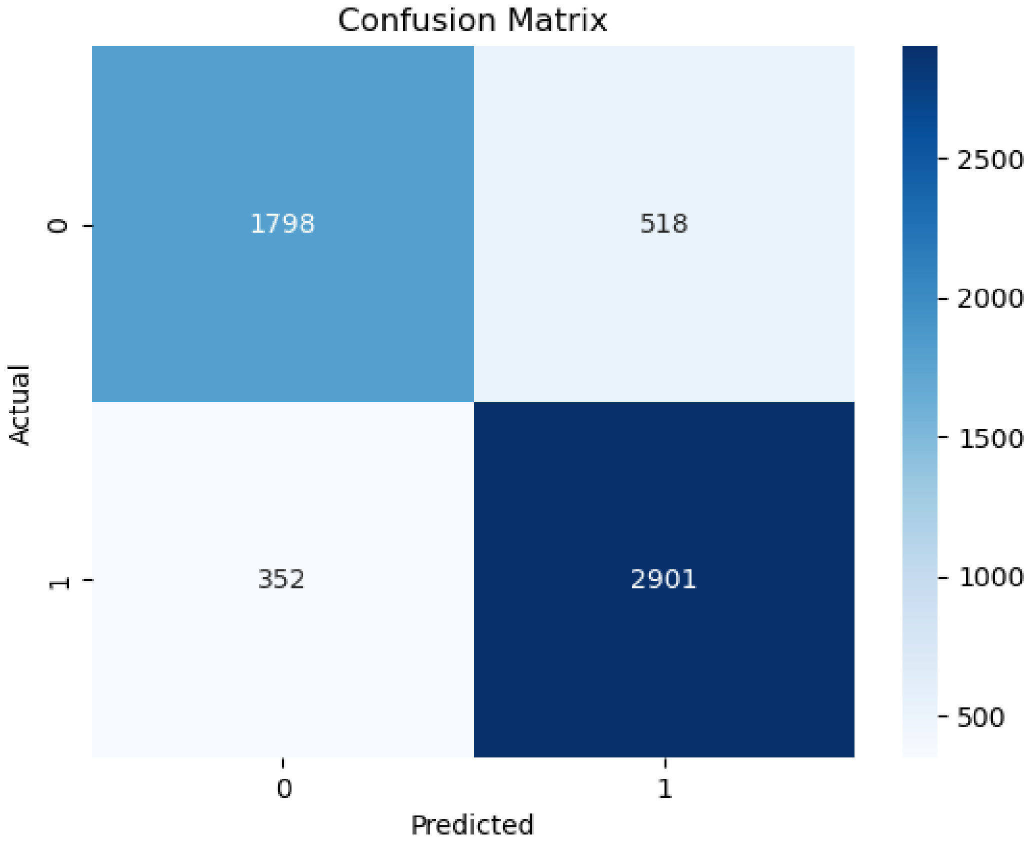
```
<IPython.core.display.HTML object>

              precision    recall  f1-score   support

           0       0.73      0.72      0.72      2316
           1       0.80      0.81      0.81      3253

    accuracy                           0.77      5569
   macro avg       0.77      0.76      0.76      5569
weighted avg       0.77      0.77      0.77      5569
```



### 0.0.7 Random Forest

[9]:
```
from sklearn.ensemble import RandomForestClassifier
model3 = RandomForestClassifier(random_state=42)
model_evaulate(model3, X_train, y_train, X_test, y_test)
```

```
<IPython.core.display.HTML object>

              precision    recall  f1-score   support

           0       0.84      0.78      0.81      2316
           1       0.85      0.89      0.87      3253

    accuracy                           0.84      5569
   macro avg       0.84      0.83      0.84      5569
weighted avg       0.84      0.84      0.84      5569
```



### 0.0.8 Hyperparameter Tuning for Random Forest

[ ]:

```
RFC = RandomForestClassifier()

params = {
    'n_estimators': [100, 300],
    'max_depth': [None, 20, 30],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 4],
    'max_features': ['sqrt', 0.8],
    'bootstrap': [True],
    'class_weight': ['balanced'],
    'criterion': ['gini', 'entropy']
}

 ## Using GridSearch
grid_search = GridSearchCV(
    estimator=RFC,
    param_grid=params,
    cv=5,
    scoring='recall',
    n_jobs=-1,
    verbose=2
)
```

Finding the best model using best.estimator_ function

- Note: Uncomment the line below to see the best hyperparameters

[12]:
```
##model_evaulate(grid_search, X_train, y_train, X_test, y_test)
```

### 0.0.9 Cross Validation:

Cross Validating the data. Trying to improve recall.

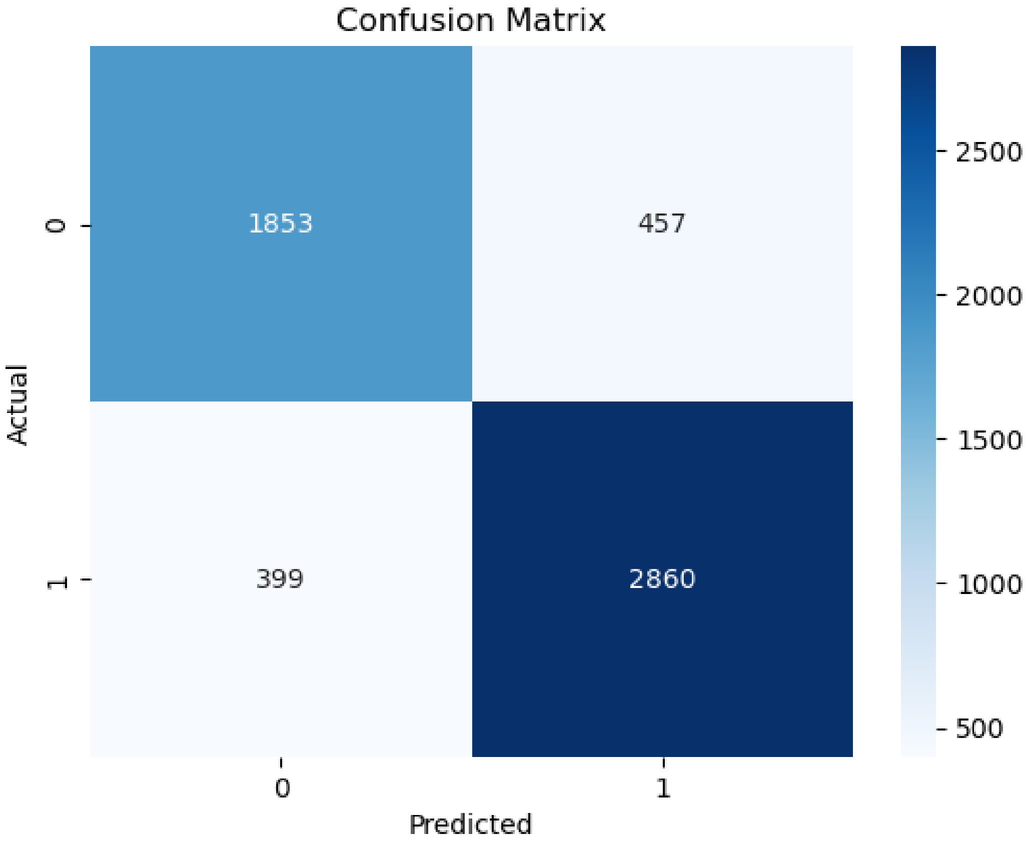
[ ]:
```
from sklearn.model_selection import StratifiedKFold, cross_val_score
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
RF = RandomForestClassifier(class_weight='balanced', max_depth=30,␣
 ↪n_estimators=300)
best_RF_cv = cross_val_score(RF, X_train, y_train, cv=cv, scoring='recall')    ␣
 ↪         ## using recall
print(f"Fold Metrics:\n{best_RF_cv}\n")
print(f"Average Score:\n{best_RF_cv.mean()}\n")                                 ␣
 ↪         ## Average Score of Recall
```

```
Fold Metrics:
[0.88995399 0.88113497 0.89033742 0.87423313 0.88190184]

Average Score:
0.8835122699386504
```

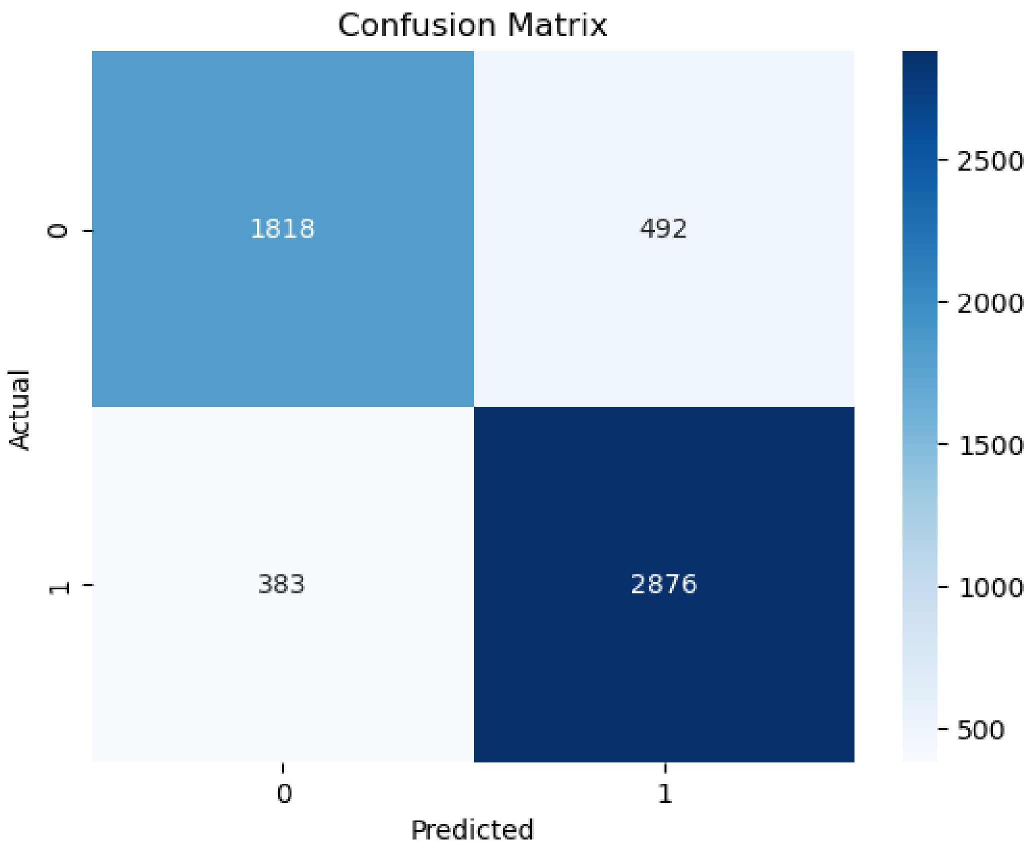
[ ]:
```
## Printing each iteration, since the dataset was split into 5 parts
crossval_evaluate(RF, X, y)
```

```
 Fold 1 Recall: 0.8776
Confusion Matrix:
[[1853  457]
 [ 399 2860]]

<IPython.core.display.HTML object>

              precision    recall  f1-score   support

           0       0.82      0.80      0.81      2310
           1       0.86      0.88      0.87      3259

    accuracy                           0.85      5569
   macro avg       0.84      0.84      0.84      5569
```

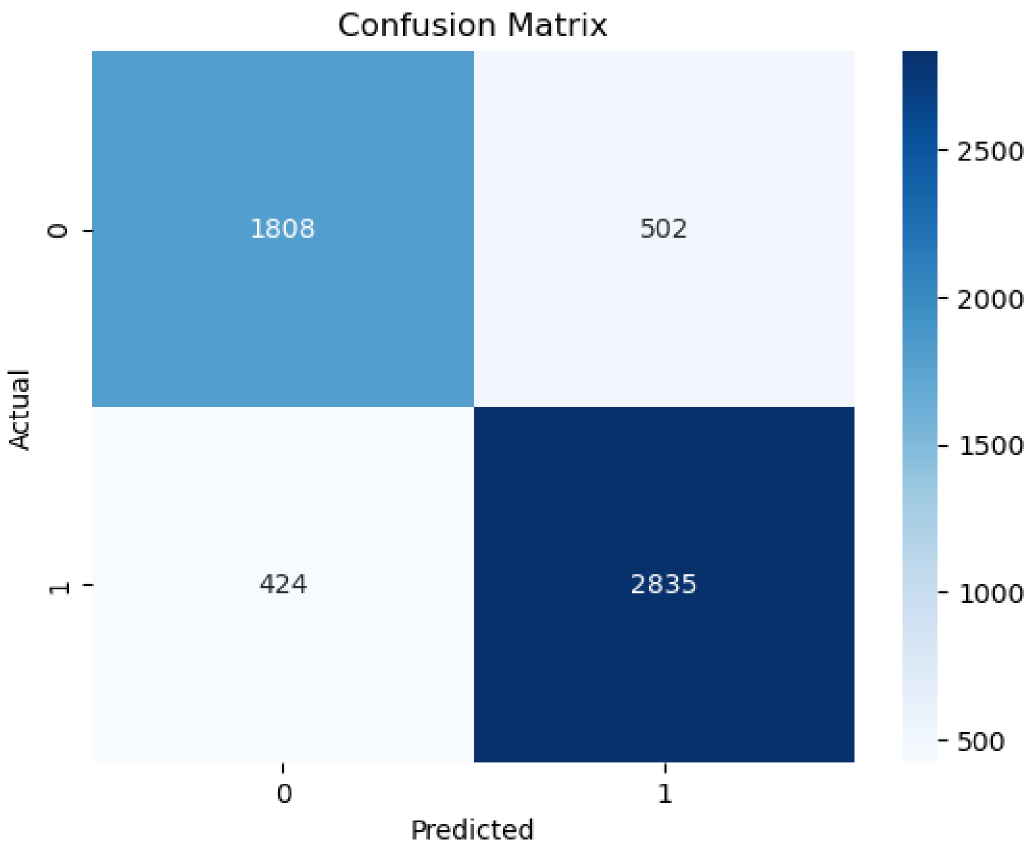
```
weighted avg       0.85      0.85      0.85      5569
```



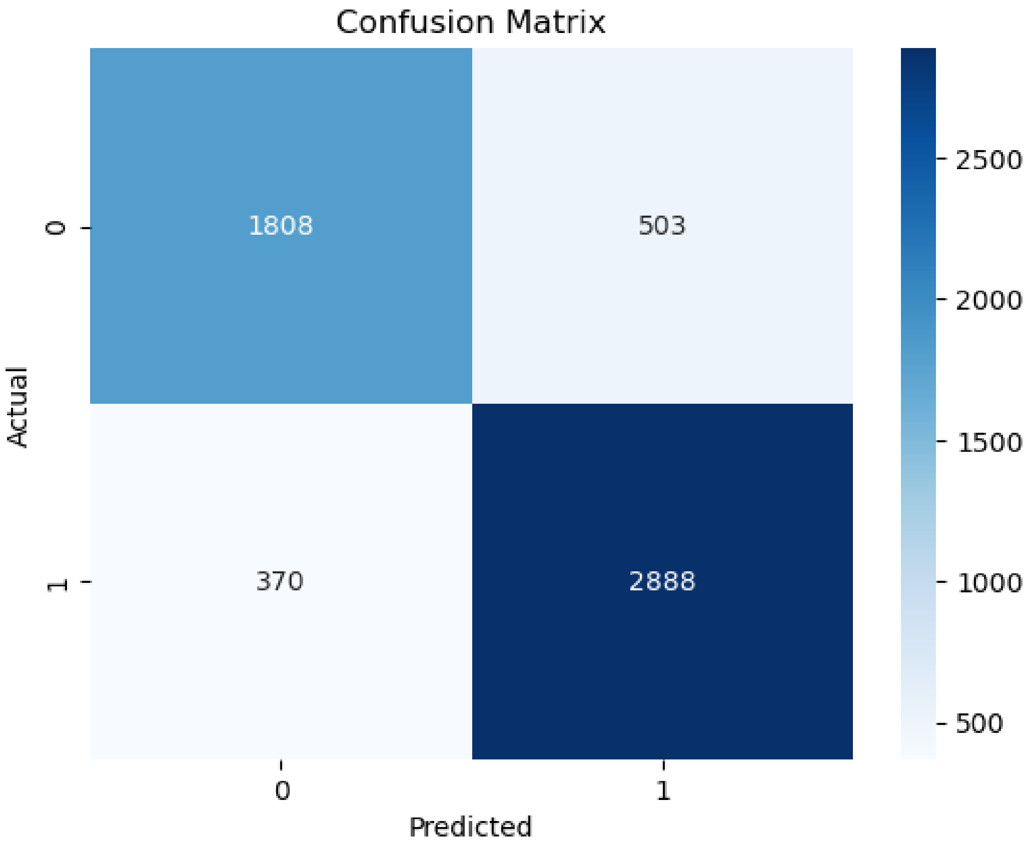
```
 Fold 2 Recall: 0.8825
Confusion Matrix:
[[1818  492]
 [ 383 2876]]

<IPython.core.display.HTML object>

              precision    recall  f1-score   support

           0       0.83      0.79      0.81      2310
           1       0.85      0.88      0.87      3259

    accuracy                           0.84      5569
   macro avg       0.84      0.83      0.84      5569
weighted avg       0.84      0.84      0.84      5569
```



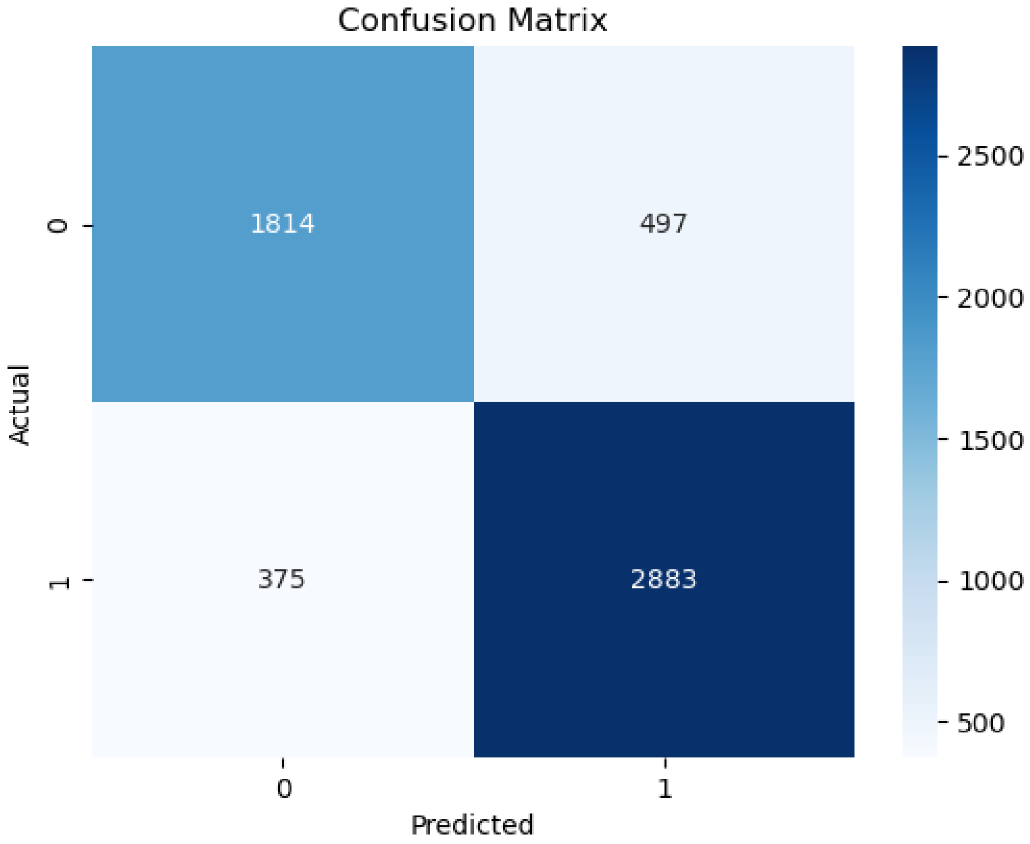
```
 Fold 3 Recall: 0.8699
Confusion Matrix:
[[1808  502]
 [ 424 2835]]

<IPython.core.display.HTML object>

              precision    recall  f1-score   support

           0       0.81      0.78      0.80      2310
           1       0.85      0.87      0.86      3259

    accuracy                           0.83      5569
   macro avg       0.83      0.83      0.83      5569
weighted avg       0.83      0.83      0.83      5569
```



```
 Fold 4 Recall: 0.8864
Confusion Matrix:
[[1808  503]
 [ 370 2888]]

<IPython.core.display.HTML object>

              precision    recall  f1-score   support

           0       0.83      0.78      0.81      2311
           1       0.85      0.89      0.87      3258

    accuracy                           0.84      5569
   macro avg       0.84      0.83      0.84      5569
weighted avg       0.84      0.84      0.84      5569
```



```
 Fold 5 Recall: 0.8849
Confusion Matrix:
[[1814  497]
 [ 375 2883]]

<IPython.core.display.HTML object>

              precision    recall  f1-score   support

           0       0.83      0.78      0.81      2311
           1       0.85      0.88      0.87      3258

    accuracy                           0.84      5569
   macro avg       0.84      0.83      0.84      5569
weighted avg       0.84      0.84      0.84      5569
```



```
 Cross-Validation Summary
All Fold Recalls: [0.8775698066891685, 0.8824792881251918, 0.869898741945382,
0.8864333947206875, 0.8848987108655617]
Mean Recall: 0.8803
Best Fold Recall: 0.8864
Best Confusion Matrix:
[[1808  503]
 [ 370 2888]]
```