Fateh Ahmad

BUAN 6341

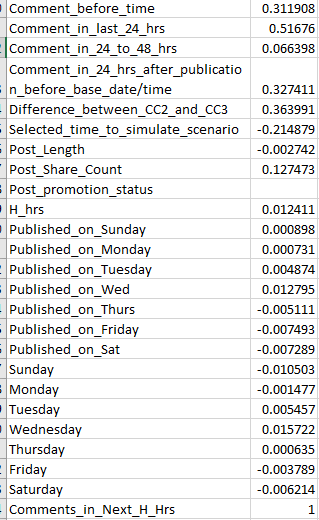
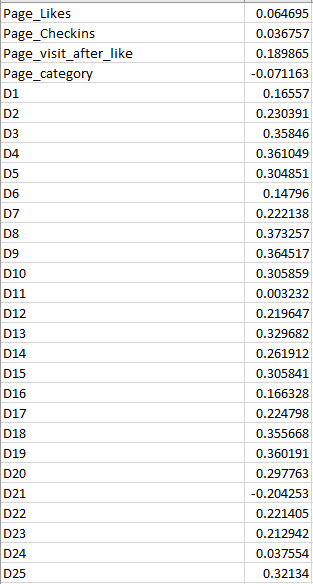
APPLIED MACHINE LEARNING

ASSIGNMENT 2

Due date: March 17, 11:59 pm

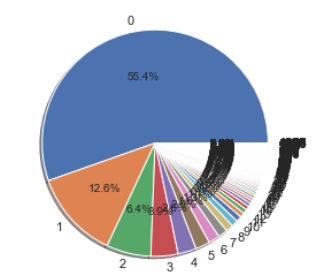
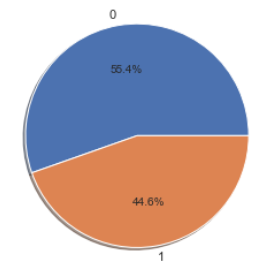
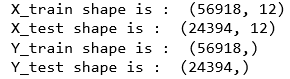
We have been tasked to predict the total number of comments that a facebook post will receive in the next H hours after it is published. In order to do our analysis, the dataset that we used was obtained from the machine learning repository from the Center of Machine Learning and Intelligent Systems. Our data set comprises of 54 features and the target variable is the number of comments in next H hrs. The datatypes of our features are either continuous variables or binary variables with the target variable being the former. Please note that in our code, we are importing the file called ‘data’ and this is the first variant of the dataset.

Although we would’ve liked to use all the features to conduct our analysis, it was noted that this was very expensive computationally and for this reason, we decided to simplify our models by reducing the number of features in the dataset. The method that was chosen to drop the variables that was to determine the correlation of each independent variable with the dependent variable. A summary of the correlation is as follows:



As we can see, neither of our features was very highly correlated with the independent variable so our method to reduce the number of features was to firstly remove all the derived variables as it would be hard to interpret them in our final analysis, from there, we removed the features that had a correlation close to 0 with the independent variable. To further simplify out model, we decided to remove the features that mentioned what day the post was published. Our model consisted of 12 independent variables for our analysis. The final variables that we were left with were Page\_Likes, Page\_Checkins, Page\_visit\_after\_like, Page\_category, Comment\_before\_time, Comment\_in\_last\_24\_hrs, Comment\_in\_24\_to\_48\_hrs, Comment\_in\_24\_hrs\_after\_publication\_before\_base\_date/time, Difference\_between\_CC2\_and\_CC3, Selected\_time\_to\_simulate\_scenario, Post\_Share\_Count and H\_hrs and these features were used as our dependent variables with the independent variable being Comment\_in\_Next\_H\_Hrs.

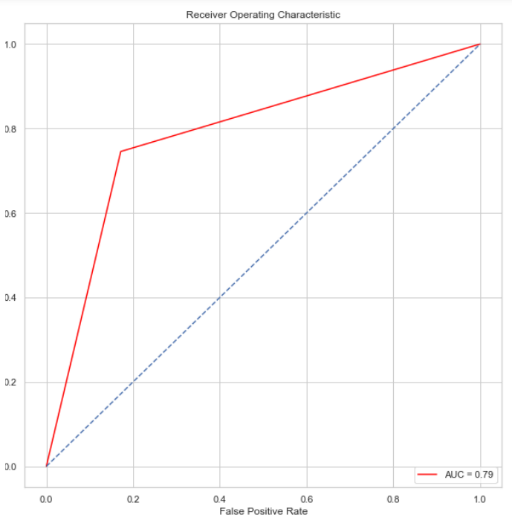
Upon looking at the target variable, we noticed that it was a continuous variable with the values ranging from 0 to 1966 and we needed to come up with a way to break this down into a binary variable. We decided to check our threshold based on the frequency distribution of each value and it was noticed that 55.4% of the values were coded as 0, therefore, we decided to leave any value that is 0 as 0 and any value greater than 0 as 1. This modified our hypothesis a little which can now be interpreted as whether a post received a comment in the next H hours or not. A bar chart of our target variable before and after assigning a threshold is as follows along with our data specification after we split it into training and testing set to reduce overfitting:

The last step to prepare our data prior to modelling it was to scale our dependent variables. This would help in increasing the accuracy of our models and also to reduce processing time when running our models.

Models on facebook dataset:

For each model, a prediction accuracy score was predicted to check for the quality of fit on the data. The higher this score, the better the model is at predicting on unseen data. Furthermore, confusion matrices were computed and ROC curves were drawn to analyze the error rates.

1. Support Vector Machines

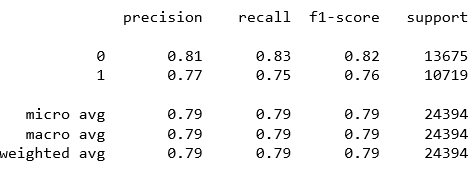
We used 3 different kernels for our analysis and they were linear, polynomial, and radial(RBF) kernels. A comparison was then carried out for each kernel.

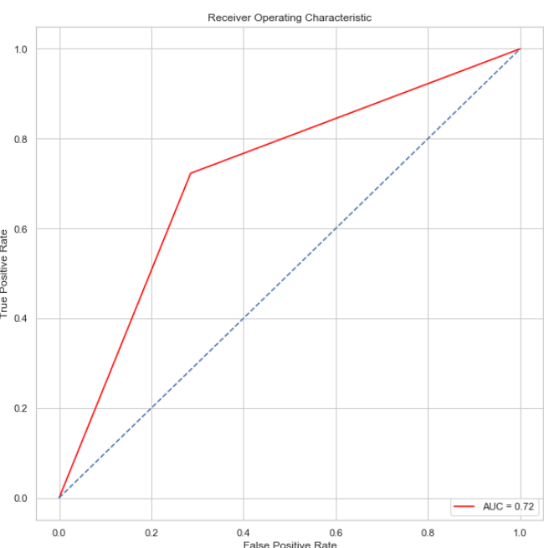
1. Linear Kernel:

Prediction Accuracy: 79.2

Confusion Matrix: [[11328 2347]

[ 2728 7991]]

Misclassification rate: 20.8 %

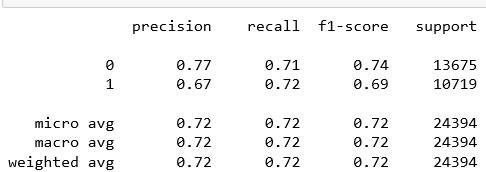
1. Polynomial Kernel

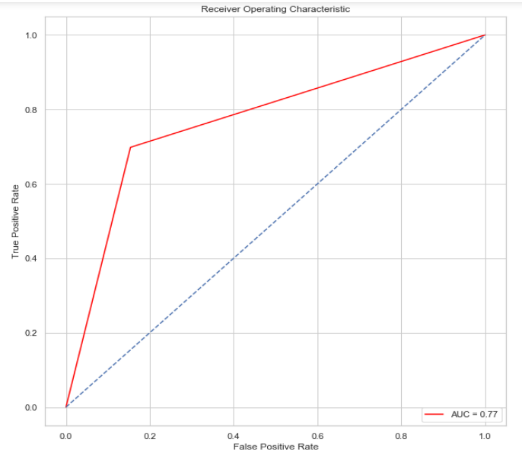
Prediction Accuracy: 71.83

Confusion Matrix: [[9775 3900]

[2972 7747]]

Misclassification Rate: 28.17%





1. RBF Kernel

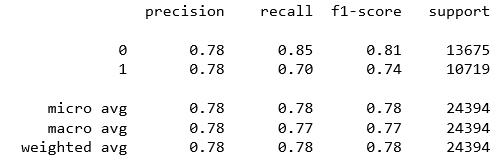
Prediction Accuracy: 78.1

Confusion Matrix:

[[11570 2105]

[ 3237 7482]]

Misclassification Rate: 21.9%



*Conclusion:* The SVM model using the linear kernel yields provides us with the highest prediction accuracy score, therefore, this is the ideal kernel when using SVM on our model. Furthermore, looking at the ROC curve, we can see that the linear kernel displays the greatest number of True Positives against False Positives.

1. Decision Tree

The Decision Tree Classifier help us perform multi-class classification on our dataset. Pruning is carried out by experimenting with the maximum allowable depth feature of the Decision Tree Classifier. Various maximum depths are set and the resulting prediction accuracy and misclassification rates are observed. For our purposes, we decided to use the gini index for our analysis rather than using entropy primarily because we wanted our models to be computationally inexpensive and to minimize misclassification.

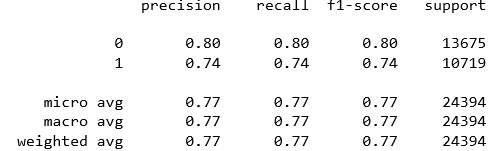
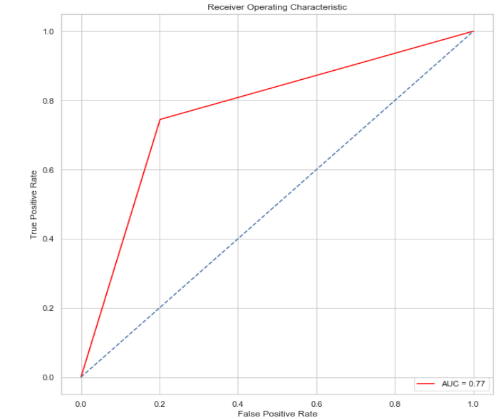
1. Unrestrained decision tree

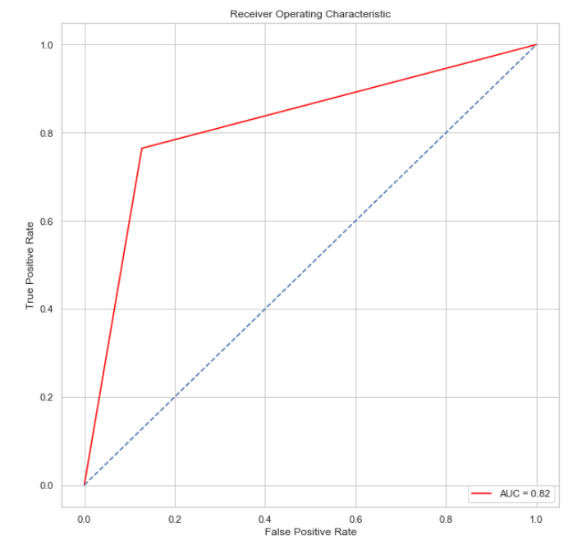
Prediction Accuracy: 77.49

Confusion Matrix: [[10920 2755]

[ 2735 7984]]

Misclassification Rate: 22.51



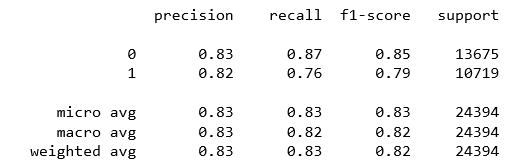
1. Decision tree with max depth = 3

Prediction Accuracy: 82.52

Confusion Matrix: [[11934 1741]

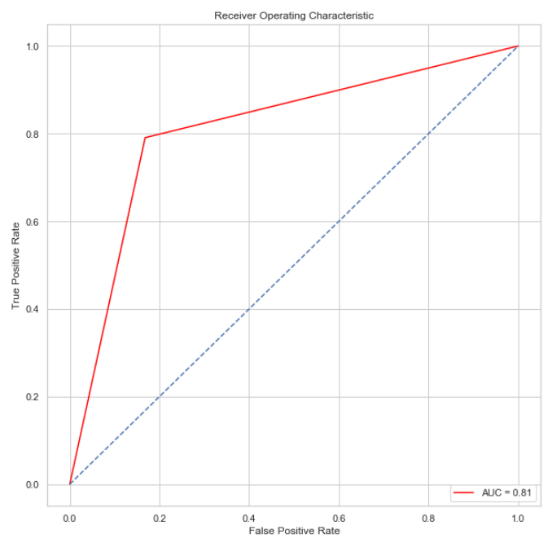
[ 2524 8195]]

Misclassification Rate: 17.48%



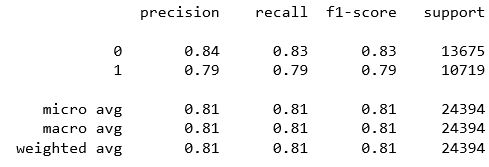
1. Decision tree with max depth = 2

Prediction Accuracy: 81.38

Confusion Matrix: [[11377 2298]

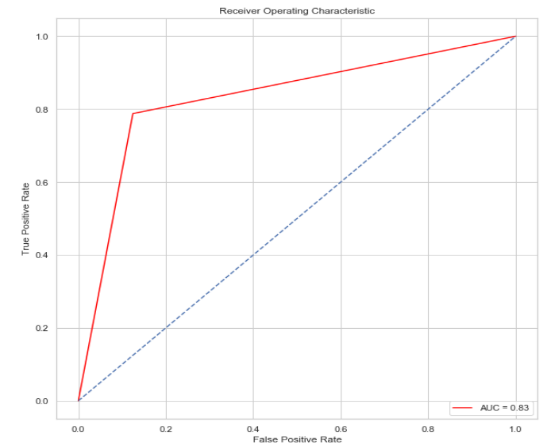
[ 2243 8476]]

Misclassification Rate: 18.62%



*Conclusion:* The tree with maximum allowable depth of 3 provides us with the highest prediction accuracy, therefore, it is the best model when using decision trees.

1. Boosting Decision Trees using AdaBoost

We would like to note that we wanted to use xgboost for our analysis but when trying to implement it, we found that the implementation was not as easy as AdaBoost since a special installation of xgboost is needed.

1. Adaboost with max depth = none

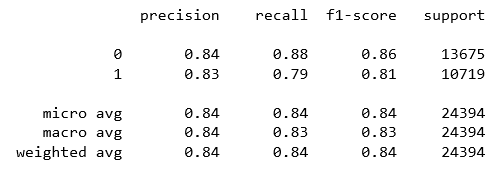
Prediction accuracy: 83.68

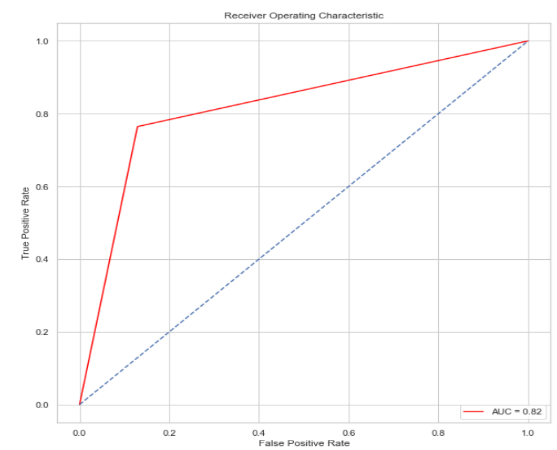
Confustion matrix:

[[11970 1705]

[ 2275 8444]]

Misclassification Rate: 16.32%





1. Adaboost with max depth = 3

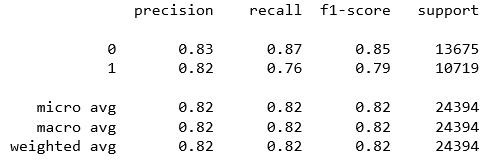
Prediction accuracy: 82.41

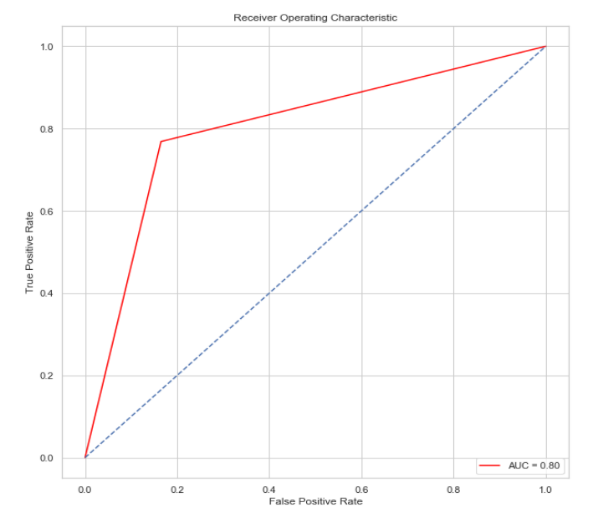
Confusion matrix:

[[11909 1766]

[ 2526 8193]]

Misclassification Rate: 17.59%



1. Adaboost with max depth = 5

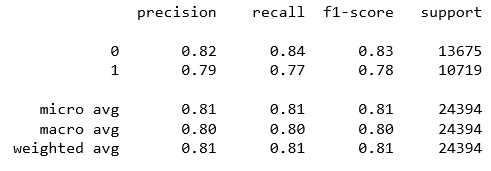
Prediction accuracy: 80.58

Confusion matrix:

[[11421 2254]

[ 2483 8236]]

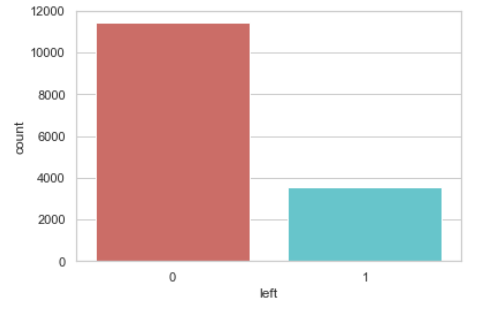
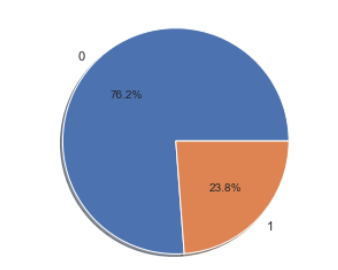
Misclassification Rate: 19.42%



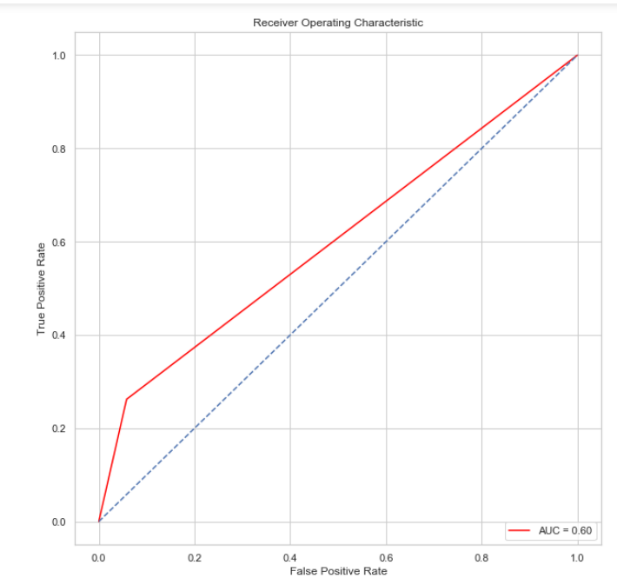
***Conclusion: Prediction accuracy with max depth = none is highest, and the ROC curve illustrates this, hence, when using boosting, our first model is the best predictor. When considering all the different algorithms that were used, boosting with max depth=none seems to be the best model when training our dataset and calculating the error rates on the test data.***

Turnover dataset:

Arguably, the biggest problem facing the world of Human Resources nowadays is employee turnover and several studies have been done to try to reduce this issue. Because this issue has major consequences on businesses, we decided to use an employee turnover dataset for our analysis. The dataset was obtained from Kaggle(<https://www.kaggle.com/lnvardanyan/prediction-of-employee-turnover/data>) and it consists of 10 features out of which the target variable is left which is a binary variable coded as 0 or 1. For this dataset, we didn’t need to reduce the number of features that we had, however, we did go ahead and scale the dependent variables after splitting the data into training and testing set. Upon reviewing the frequency distribution of the dataset, it appears that 76.2% employees did not leave while 23.8% did.



**Models:**

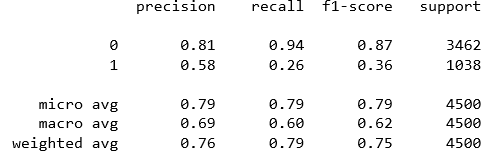
1. Support Vector Machines
2. Prediction Accuracy: 78.51

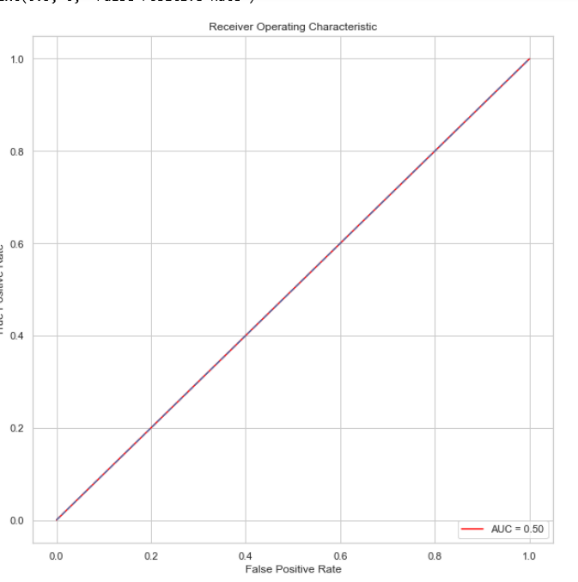
Confusion Matrix:

[[3261 201]

[ 766 272]]

Misclassification Rate: 21.49%



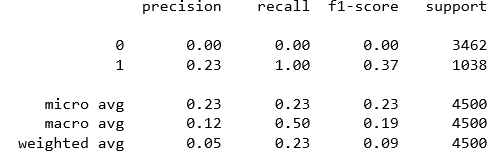


1. Polynomial Kernel

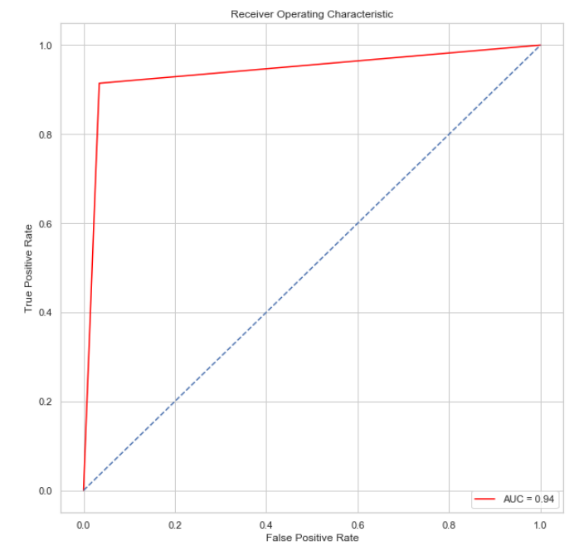
Prediction Accuracy: 82.08

Confustion Matrix:

[[ 0 3462]

 [ 0 1038]]

Misclassification Rate: 17.98%



1. RBF Kernel

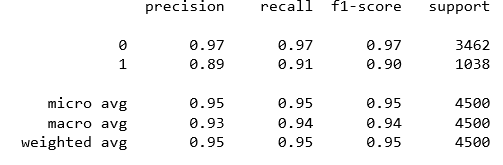
Prediction Accuracy: 95.85

Confusion Matrix:

[[3343 119]

[ 89 949]]

Misclassification Rate: 4.15%



Conclusion: We can easily infer that the RBF kernel is the most suitable algorithm when using SVM given the high prediction accuracy and when looking at the ROC curve.

1. Decision Trees

For this particular dataset as well, we decided to use the gini index primarily to reduce processing time and to decrease misclassification rate.

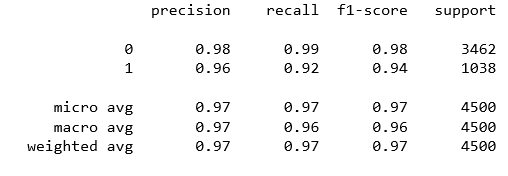
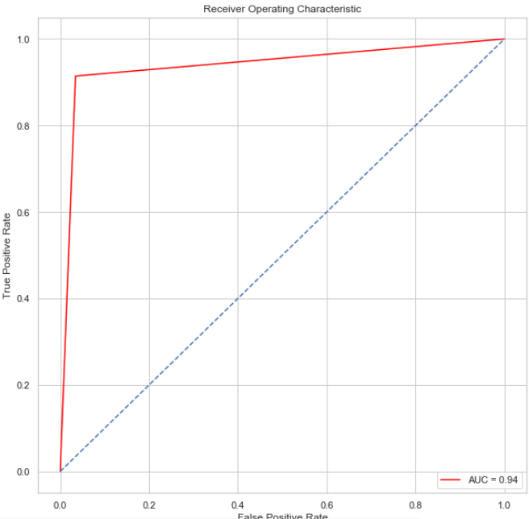
1. Decision tree with max depth = 5

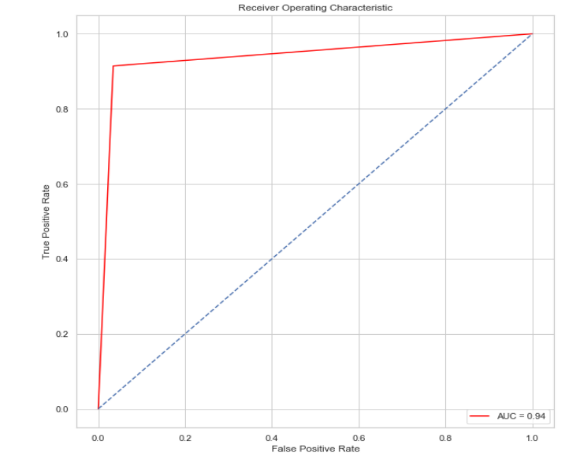
Prediction Accuracy: 97.29

Confusion Matrix: [[3418 44]

[ 78 960]]

Misclassification Rate: 2.71%



1. Decision tree with max depth = 10

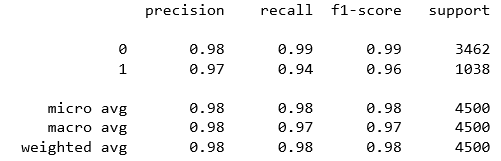
Prediction Accuracy = 98.04

Confusion Matrix:

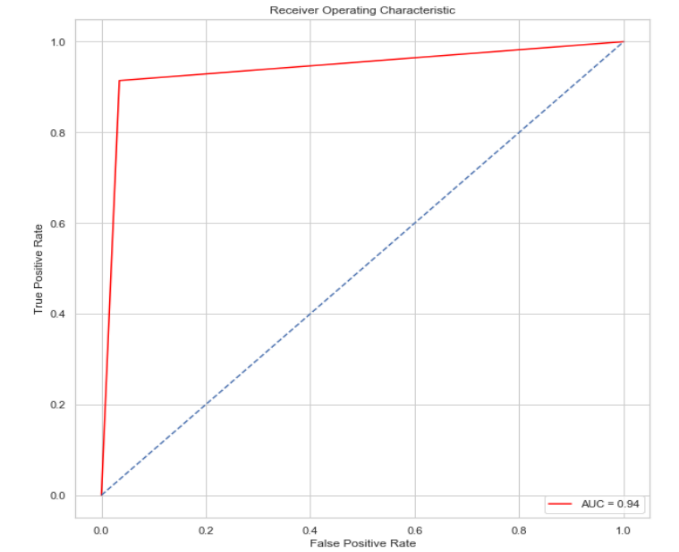
[[3433 29]

[ 59 979]]

Misclassification Rate: 1.96



1. Decision tree with max depth = None

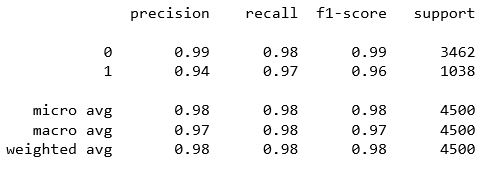
Prediction Accuracy: 97.96

Confusion Matrix:

[[3400 62]

[ 30 1008]]

Misclassification Rate: 2.04%



1. Boosting with Adaboost
2. Boosting using decision tree max depth = 5

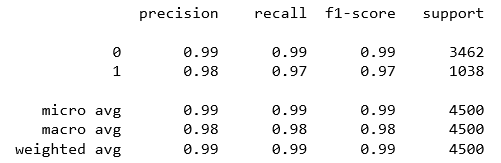
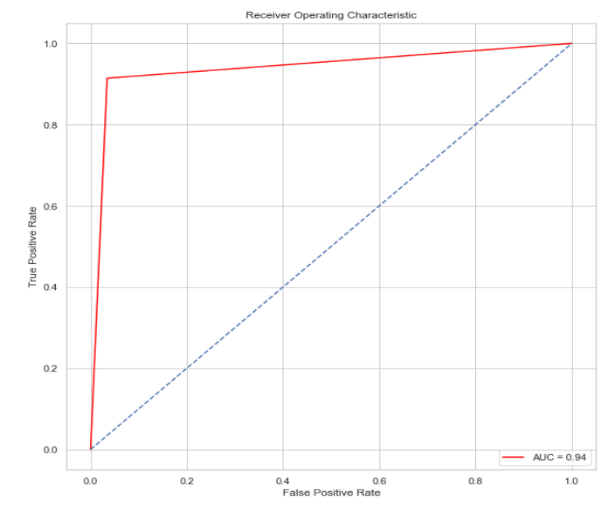
Prediction Accuracy: 98.8

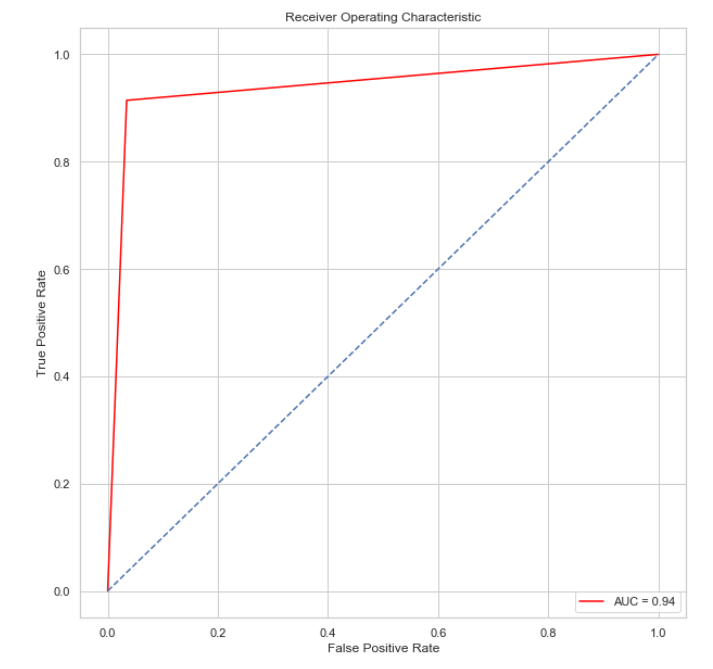
Confusion Matrix:

[[3439 23]

[ 31 1007]]

Misclassification Rate: 1.92%





1. Boosting using decision trees with max depth= 10

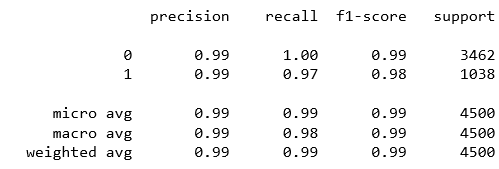
Prediction accuracy: 98.8

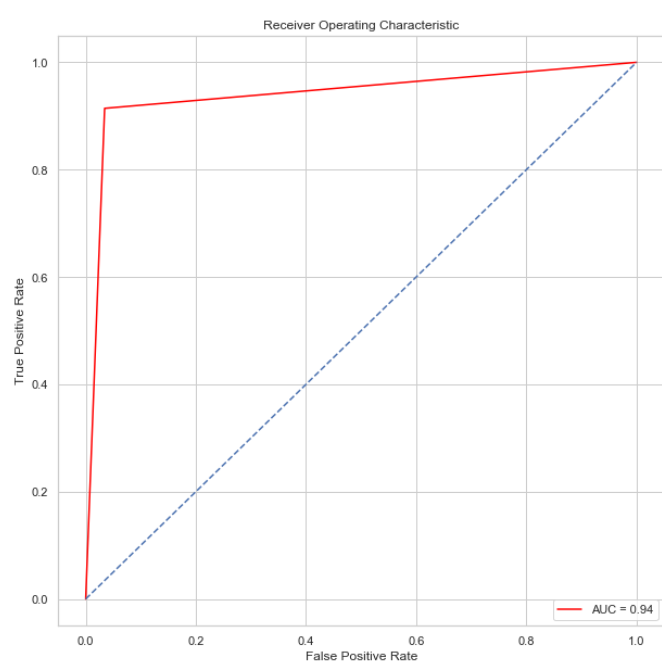
Confusion Matrix:

[[3450 12]

[ 34 1004]]

Misclassification Rate: 1.92%





1. Boosting using decision trees with max depth = 3

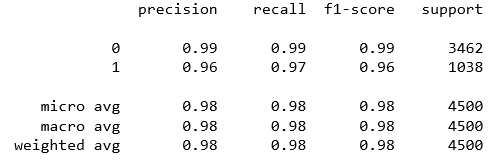
Prediction accuracy: 98.36

Confusion Matrix:

[[3419 43]

[ 31 1007]]

Misclassification Rate: 1.64%



***Conclusion: Given all the different variants of SVM, decision trees and boosting that were used, the best results were obtained using boosting where the max depth of the decision was 3 since the prediction accuracy score was highest and misclassification rate was lowest.***

Overall, the models ran on the second dataset provided more consistent results for each model as opposed to the models that were ran on the first dataset with the exception of SVM with the polynomial kernel. In our analysis, we wanted to plot learning curves for each model that was ran but the processing time was too long when running the code on the first dataset, therefore, we had to resort to using ROF curves instead. Had we been able to plot learning curves, they would’ve provided us with insight as to how the error on the training set compares with the error on the test set.