Shanmukha Bodapati - ssb180006
Raghav Sriram - rxs180123
CS 4372.001

Tree Classification Results Analysis

Introduction

This report will analyze a tree classification analysis on the Bank Customer Churn Dataset, which we found on Kaggle (https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset?fbclid=IwAR39t4yf1Kxp4W9ZUSz0sQ9N7mzdPpa9BkoIVM9tfKrko6vBjOizaIamWqo) . The dataset is a collection of data points collected from customers in ABC Multistate bank and it contains 10000 data points. Each datapoint contains 12 features:
customer_id - an identification number for the customer
credit_score - the customer's credit score
country - the location the customer's account originated from (France, Germany, or Spain)
gender - the customer's gender (Male or Female)
age - the customer's age
tenure - how long the customer's account has been active for
balance - the current balance of the customer's account
products_number - number of products the customer has bought from the bank
credit_card - whether the customer owns a credit card from the bank (1=yes, 0=no)
active_member - whether the customer is an active member (1=yes, 0=no)
estimated_salary - the estimated salary of the customer
churn - whether the customer has left the bank during some period (1=yes, 0=no)

Below are the first 5 rows of data:

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

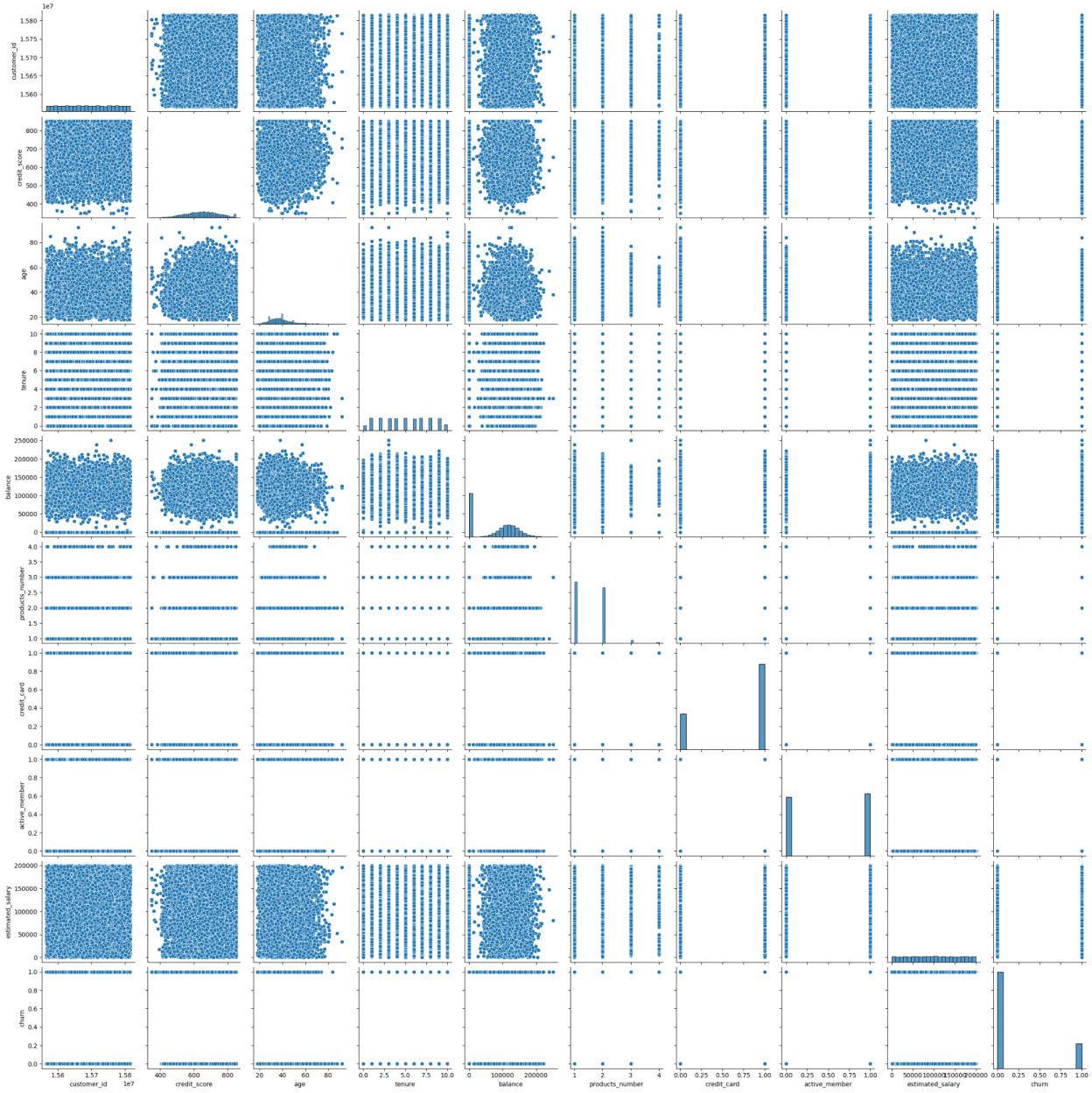
Data Cleaning

Before beginning the process of building the classification models, the data was checked for empty/null values and cleaned appropriately. For this specific dataset, there were no empty/null values.

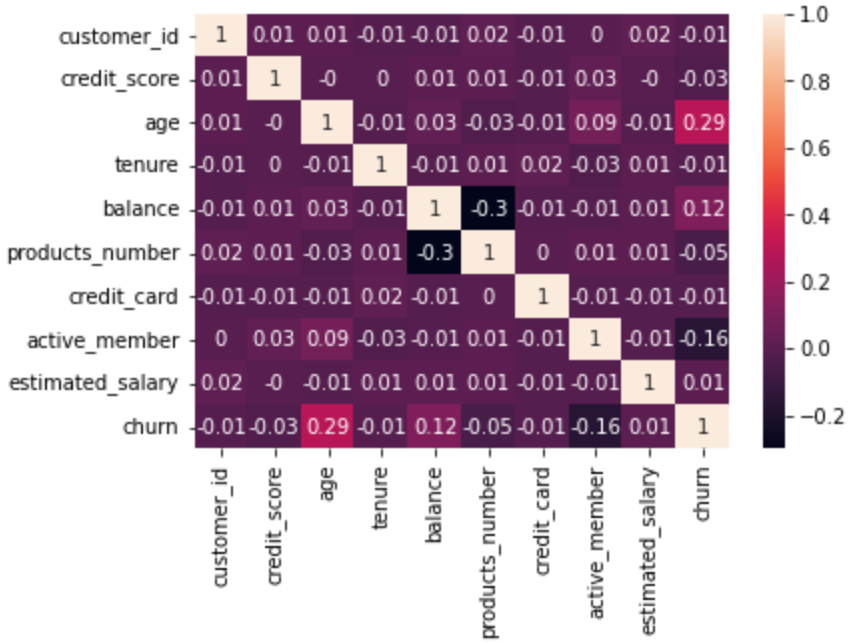
Tree Classification Attributes

Prior to testing different attributes, the pairplot was created for all 12 data fields. Here, a pairplot offers little to no information about the strength of input variables because the data is generally

encoded and/or because this is a classification task. However, we can visually see that the data for the credit score and balance seem to be normally distributed while the data for Age seems to be bell shaped but skewed to the right.



However, a correlation matrix was constructed and applied to a heatmap to gain a higher level understanding of how different elements of the data relate to one another.



Through this correlation matrix and heatmap, it is still not clear as to which attributes should be used because, again, this is a classification task. However, we already know the variable to predict (Churn). We definitely know not to use the customer_id attribute because it will not offer us any significant material for classification. We will use the remaining attributes ('credit_score', 'country', 'gender', 'age', 'tenure', 'balance', 'products_number', 'credit_card', 'active_member', 'estimated_salary'). Another thing we needed to do was to encode the categorical variables in the dataset. After encoding, the values of “country” were mapped as such – France: 0, Germany: 1, Spain: 2. The values of “gender” were mapped as such – Female: 0, Male: 1.

Standardization and Outliers

We don't want to standardize or normalize the data here for 3 reasons.

1) Standardization/Normalization will destroy important information because the each respective scale is necessary to understand the data (ie. age has a different scale than balance but the scales are necessary as clearly, 52 in age is different than 52 in balance as 52000 in age is different than 52000 in balance)
2) Many of the input variables are encoded into numeric representations of categorical/boolean variables. Standardization/normalization will offer no better model in these situations
3) Decision Trees, Random Forests, adaboost, and xgboost are generally impervious to standardization

We also do not remove outliers for 2 reasons:

1) For this specific dataset, removing outliers might change the classification drastically as it is often in the extremes that the Churn results in a value of 1
2) Decision Trees, Random Forests, adaboost, and xgboost are generally impervious to removing outliers

*****Please note the values might be slightly different after running on your PC. However, the differences are extremely small and do not alter our conclusions and analysis. The differences are because the training and testing splits are randomized.*****

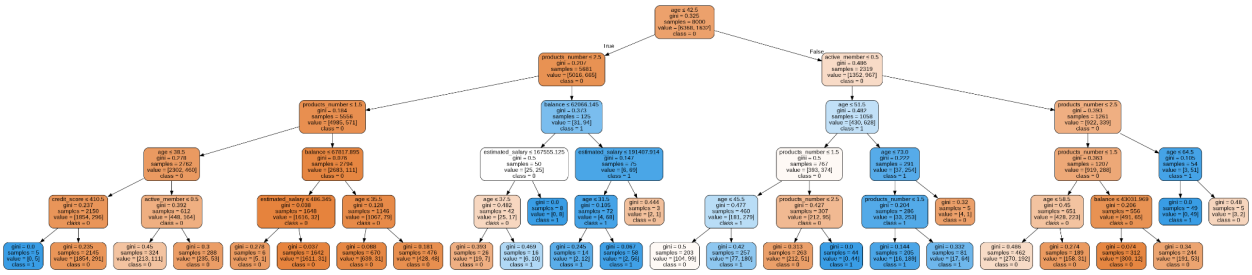
Decision Tree - DecisionTreeClassifier Model

We created a pipeline with the MinMaxScaler() and DecisionTreeClassifier() and passed it to GridSearchCV along with the following parameters: [{'dt__max_depth': [3, 5, 7, 9], 'dt__min_samples_leaf': [2, 3, 5]}] to find the best hyperparameters.

We utilized the GridSearchCV library to tune our hyperparameters and it resulted in the following output: {'dt__max_depth': 5, 'dt__min_samples_leaf': 3}

The training accuracy of the resulting model was: 0.858125

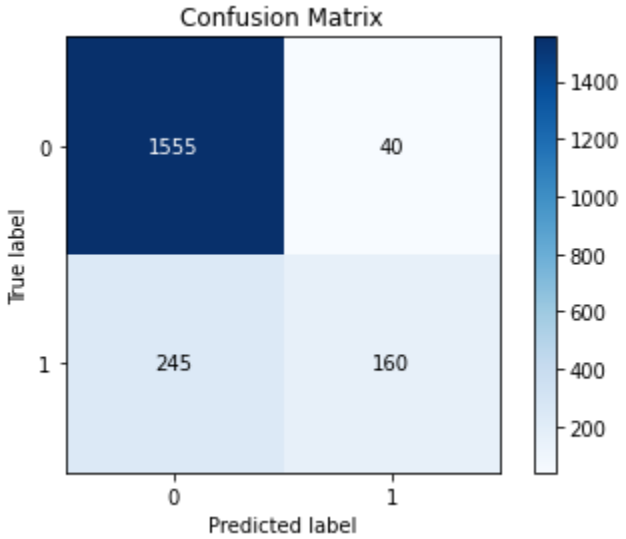
* This was the best we were able to show the picture. Please zoom in or look at the results in the code to see the tree more clearly. *
Tree:



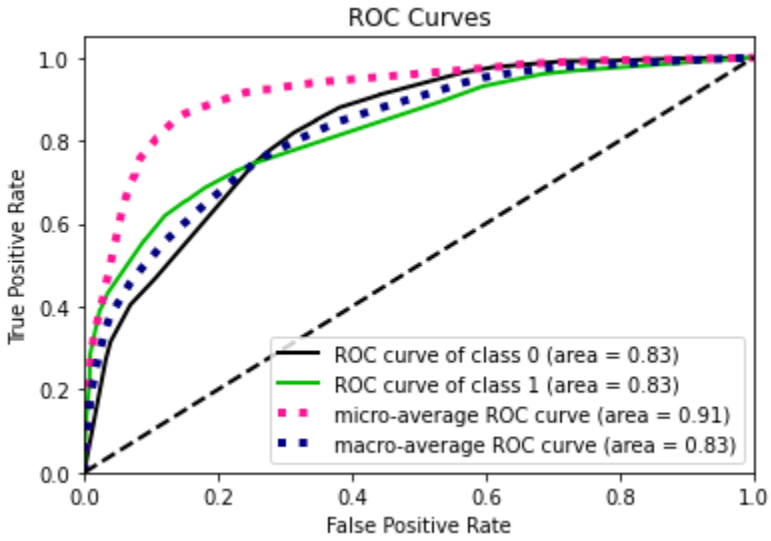
Metrics (on testing data):

```
DecisionTreeClassifier Report
              precision    recall  f1-score   support

           0       0.86      0.97      0.92      1595
           1       0.80      0.40      0.53       405

    accuracy                           0.86      2000
   macro avg       0.83      0.68      0.72      2000
weighted avg       0.85      0.86      0.84      2000

DecisionTreeClassifier Accuracy:  0.8575
```

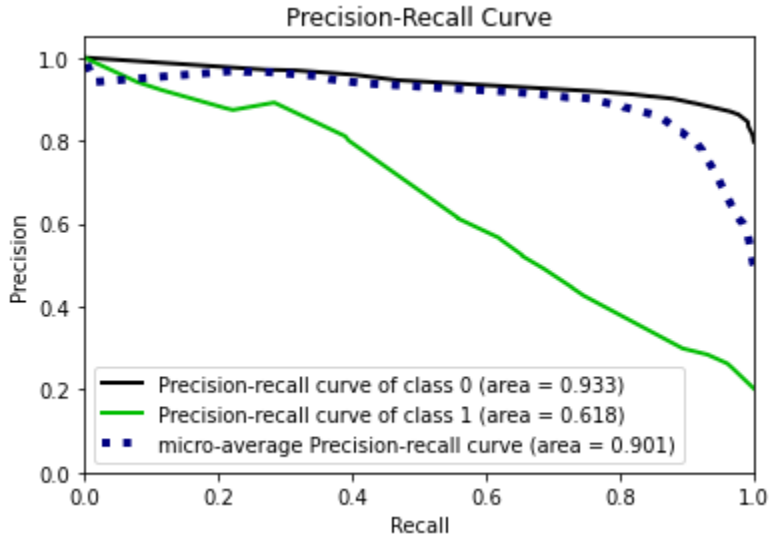
The classification report on the testing data gives us a testing accuracy of 0.8575, which is very marginally lower than the training accuracy. The precision for both variables is also fairly high, indicating a low false positive rate. The recall, or sensitivity, is high for the class label 0 but 0.40 for the class label 1, meaning that there is way more data with a class label of 0 compared to a class label of 0. The f1-score is high for the class label 0 but 0.53 for the class label 1, again indicating that there is a class imbalance (and again evidenced by the support).



For the above confusion matrix, we can see that the majority of the entries were true positives (churn was not present), at a count of 1555. The 235 false positives (FPs) were relatively low when compared to the true positives, showing that the model is effective at determining the true positive case. The 160 true negatives (churn was present) are also greater than the 40 false negatives, but at a smaller multiple than the positive case. This makes sense however, as there are many more entries that did not exhibit churn when compared to the opposite.



The above ROC curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) has an area of 0.83 for both classes. The ideal area for an ROC curve is 1, which shows that our model does a good job of predicting the data.



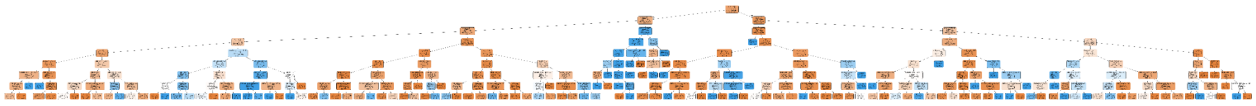
The Precision-Recall Curve is plotting Precision on the y-axis and Recall on the x-axis. In other words, this is plotting the positive predictive value (PPV) vs the True Positive Rate (TPR). The ideal area under the curve would be 1. We get a very high AUC for class 0 (0.933) but a fairly lower AUC for class 1 (0.618), which makes sense because of the class imbalance and the precision-recall curve and the precision and recall measures can be used to detect class imbalances.

RandomForestClassifier Model

We created a RandomForestClassifier() model and passed it to GridSearchCV along with the following parameters: [{'n_estimators': [200, 500], 'max_features': ['sqrt', 'log2'], 'max_depth' : [4,5,6,7,8]}] to find the best hyperparameters.

The training accuracy of the resulting model was: 0.877375

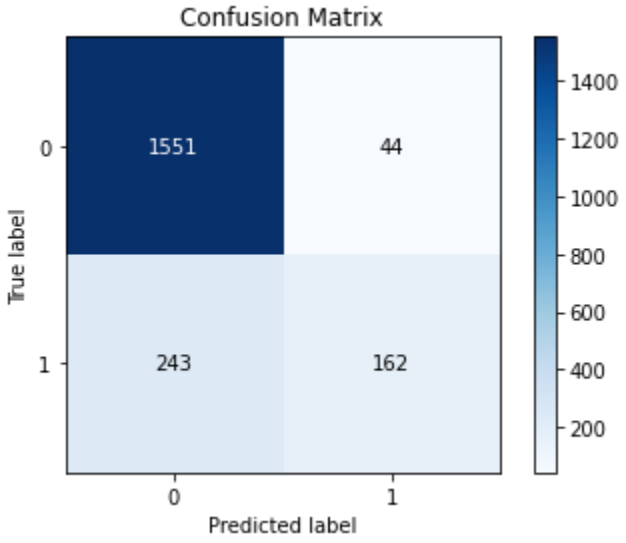
We displayed a random tree within the RandomForestClassifier Model. Since all the trees in random forests are meant to be overfitted (and we basically take the average of these trees), the tree below looks to be clearly overfitted.
* This was the best we were able to show the picture. Please zoom in or look at the results in the code to see the tree more clearly. *
Tree:



Metrics (on testing data):

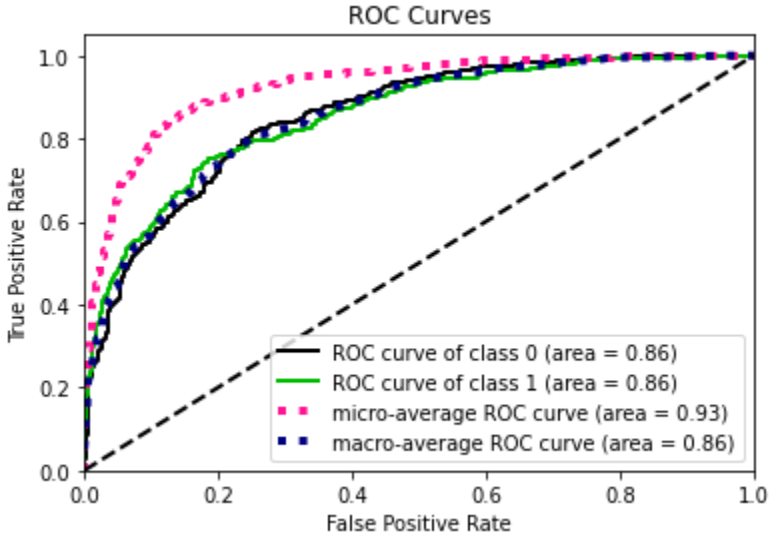
```
RandomForest Report
              precision    recall  f1-score   support

           0       0.86      0.97      0.92      1595
           1       0.79      0.40      0.53       405

    accuracy                           0.86      2000
   macro avg       0.83      0.69      0.72      2000
weighted avg       0.85      0.86      0.84      2000

RandomForest Accuracy:  0.8565
```

The classification report on the testing data gives us a testing accuracy of 0.8565, which is only slightly lower than the training accuracy. The precision for both variables is also fairly high, indicating a low false positive rate. The recall, or sensitivity, is high for the class label 0 but 0.40 for the class label 1, meaning that there is way more data with a class label of 0 compared to a class label of 0. The f1-score is high for the class label 0 but 0.53 for the class label 1, again indicating that there is a class imbalance (and again evidenced by the support).

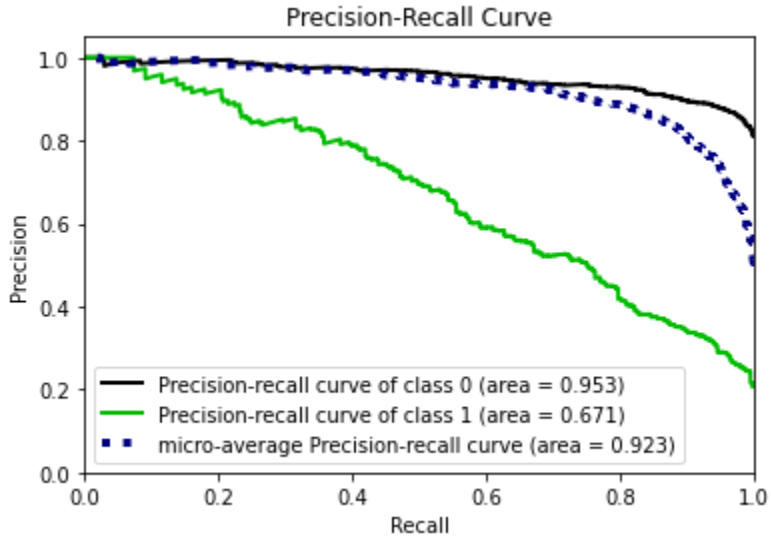


For the above confusion matrix, we can see that the majority of the entries were true positives (churn was not present), at a count of 1551. The 243 false positives (FPs) were relatively low when compared to the true positives, showing that the model is effective at determining the true positive case. The 162 true negatives (churn was present) are also greater than the 44 false

negatives, but at a smaller multiple than the positive case. This makes sense however, as there are many more entries that did not exhibit churn when compared to the opposite.



The above ROC curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) has an area of 0.86 for both classes. The ideal area for an ROC curve is 1, which shows that our model does a good job of predicting the data.



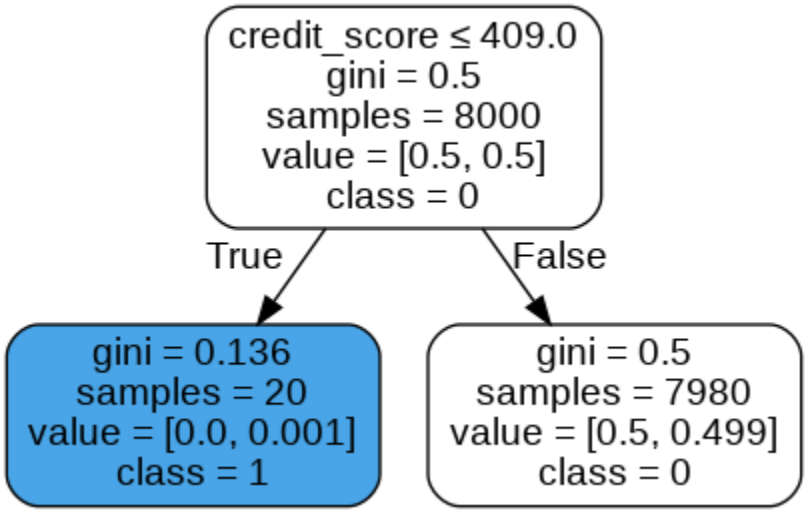
The Precision-Recall Curve is plotting Precision on the y-axis and Recall on the x-axis. In other words, this is plotting the positive predictive value (PPV) vs the True Positive Rate (TPR). The ideal area under the curve would be 1. We get a very high AUC for class 0 (0.953) but a fairly lower AUC for class 1 (0.671), which makes sense because of the class imbalance and the precision-recall curve and the precision and recall measures can be used to detect class imbalances.

AdaBoost Model

We created a AdaBoostClassifier() model and passed it to GridSearchCV along with the following parameters: [{'n_estimators':[10, 50, 100, 250, 500], 'learning_rate':[0.0001, 0.001, 0.01, 0.1]} to find the best hyperparameters.

The training accuracy of the resulting model was: 0.861125

We displayed the tree with the best estimator in the AdaBoost Model.
Tree:

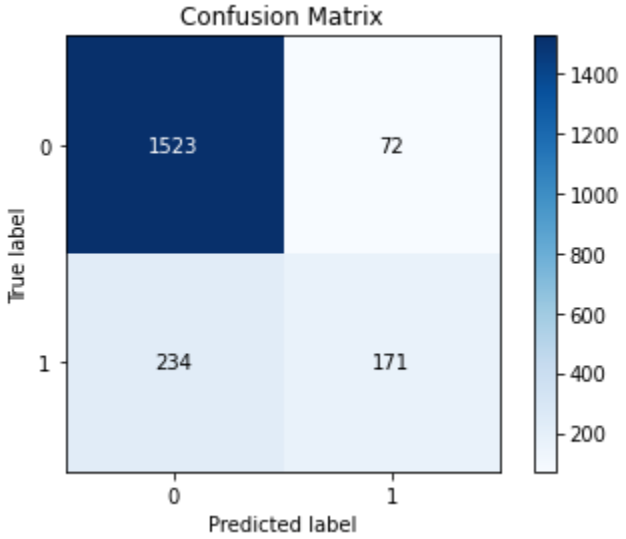


Metrics (on testing data):

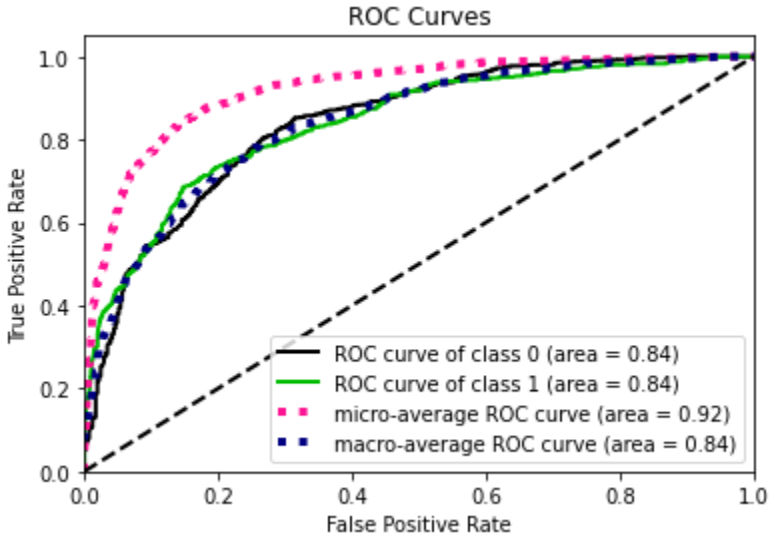
```
AdaBoost Classication Report
              precision    recall  f1-score   support

           0       0.87      0.95      0.91      1595
           1       0.70      0.42      0.53       405

    accuracy                           0.85      2000
   macro avg       0.79      0.69      0.72      2000
weighted avg       0.83      0.85      0.83      2000

AdaBoost Accuracy:  0.847
```

The classification report on the testing data gives us a testing accuracy of 0.847, which is only slightly lower than the training accuracy. The precision for both variables is also fairly high, indicating a low false positive rate. The recall, or sensitivity, is high for the class label 0 but 0.42 for the class label 1, meaning that there is way more data with a

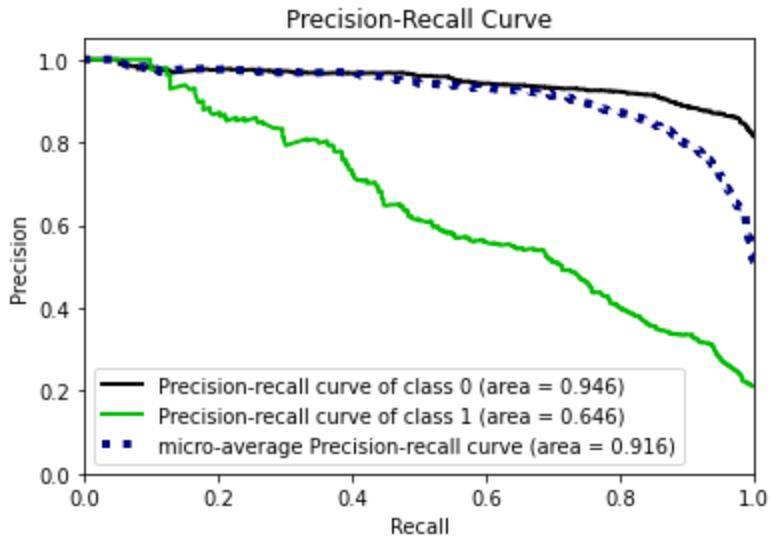
class label of 0 compared to a class label of 0. The f1-score is high for the class label 0 but 0.53 for the class label 1, again indicating that there is a class imbalance (and again evidenced by the support).



For the above confusion matrix, we can see that the majority of the entries were true positives (churn was not present), at a count of 1523. The 234 false positives (FPs) were relatively low when compared to the true positives, showing that the model is effective at determining the true positive case. The 171 true negatives (churn was present) are also greater than the 72 false negatives, but at a smaller multiple than the positive case. This makes sense however, as there are many more entries that did not exhibit churn when compared to the opposite.



The above ROC curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) has an area of 0.84 for both classes The ideal area for an ROC curve is 1, which shows that our model does a good job of predicting the data.



The Precision-Recall Curve is plotting Precision on the y-axis and Recall on the x-axis. In other words, this is plotting the positive predictive value (PPV) vs the True Positive Rate (TPR). The ideal area under the curve would be 1. We get a very high AUC for class 0 (0.946) but a fairly lower AUC for class 1 (0.646), which makes sense because of the class imbalance and the precision-recall curve and the precision and recall measures can be used to detect class imbalances.

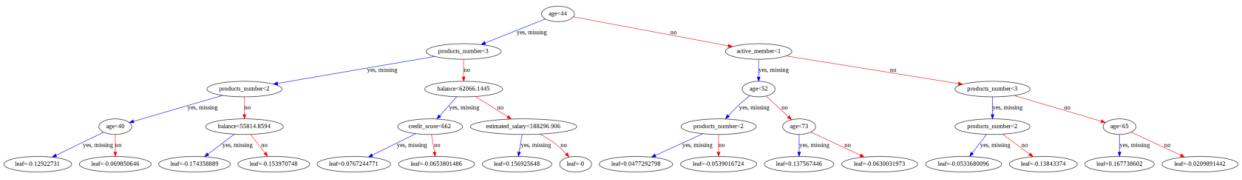
XGBoost Model

We created a XGBClassifier() model and passed it to GridSearchCV along with the following parameters: ['max_depth': [2,4,6], 'n_estimators':[10, 50, 100, 250, 500], 'learning_rate':[0.0001, 0.001, 0.01, 0.1]] to find the best hyperparameters.

{'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 100}
0.877875

The training accuracy of the resulting model was: 0.877875

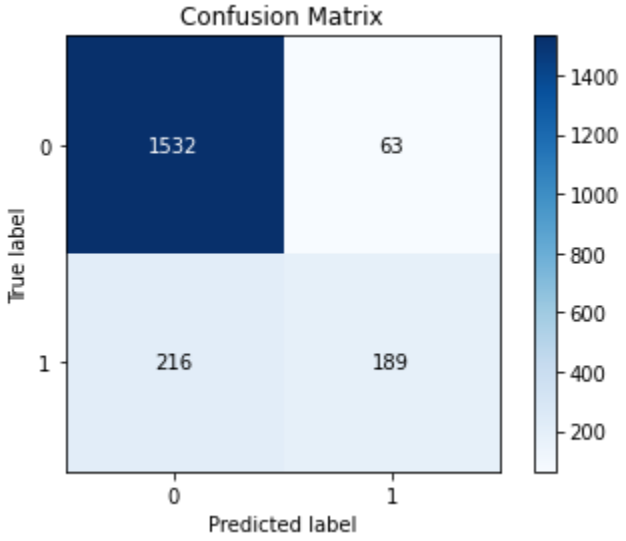
Tree:



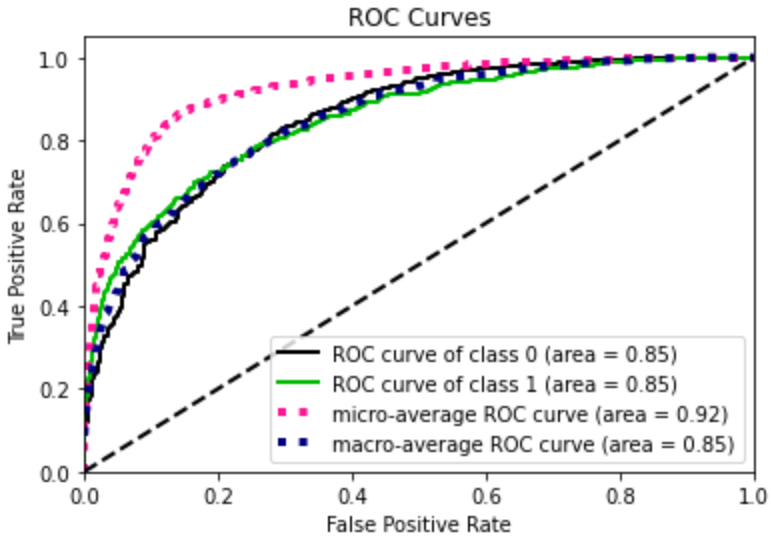
Metrics (on testing data):

```
XGBoost Classification Report
              precision    recall  f1-score   support

           0       0.88      0.96      0.92      1595
           1       0.75      0.47      0.58       405

    accuracy                           0.86      2000
   macro avg       0.81      0.71      0.75      2000
weighted avg       0.85      0.86      0.85      2000

XGBoost Accuracy:  0.8605
```

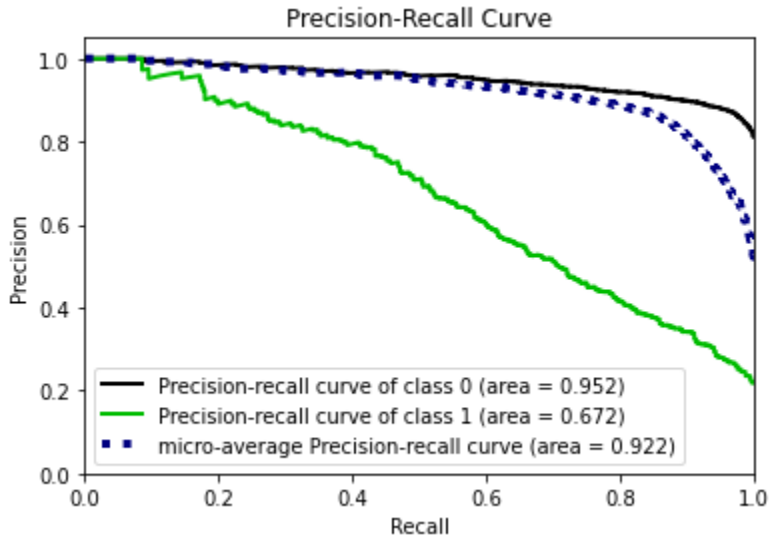
The classification report on the testing data gives us a testing accuracy of 0.8605, which is only slightly lower than the training accuracy. The precision for both variables is also fairly high, indicating a low false positive rate. The recall, or sensitivity, is high for the class label 0 but 0.47 for the class label 1, meaning that there is way more data with a class label of 0 compared to a class label of 0. The f1-score is high for the class label 0 but 0.58 for the class label 1, again indicating that there is a class imbalance (and again evidenced by the support).



For the above confusion matrix, we can see that the majority of the entries were true positives (churn was not present), at a count of 1532. The 216 false positives (FPs) were relatively low when compared to the true positives, showing that the model is effective at determining the true positive case. The 189 true negatives (churn was present) are also greater than the 63 false negatives, but at a smaller multiple than the positive case. This makes sense however, as there are many more entries that did not exhibit churn when compared to the opposite.



The above ROC curve, which plots the True Positive Rate (TPR) against the False Positive Rate (FPR) has an area of 0.85 for both classes. The ideal area for an ROC curve is 1, which shows that our model does a good job of predicting the data.



The Precision-Recall Curve is plotting Precision on the y-axis and Recall on the x-axis. In other words, this is plotting the positive predictive value (PPV) vs the True Positive Rate (TPR). The ideal area under the curve would be 1. We get a very high AUC for class 0 (0.952) but a fairly lower AUC for class 1 (0.672), which makes sense because of the class imbalance and the precision-recall curve and the precision and recall measures can be used to detect class imbalances.

Comparing Models

If we compare the training accuracy of all of our models, the ranking is as follows (highest to lowest):

1. XGBoost Accuracy: 0.877875
2. Random Forest Accuracy: 0.87725
3. AdaBoost Accuracy: 0.861125
4. Decision Tree Accuracy: 0.858125

Among all of the models, XGBoost had the highest training accuracy.

Next, if we compare the testing accuracy, we observe a similar trend:

1. XGBoost Accuracy: 0.8605
2. RandomForest Accuracy: 0.8585
3. AdaBoost Accuracy: 0.847
4. DecisionTreeClassifier Accuracy: 0.8575

Based on the training and testing accuracies, XGBoost was the best model for the dataset we chose because it had both the best training and testing accuracies. It also is expected to perform the best because it is the latest version (out of the four models) and utilizes sequential learning (instead of parallel like random forest).

One thing we did notice is that there is a class imbalance with more data for class label 0 and far less data for class label 1. In the future, the dataset should be better and have a higher class balance.