# CS 4372
# ASSIGNMENT 2

Names of students in your group: Natalie Pedigo (nbp220000), Kylie Quinney (krq210000)

Number of free late days used: 0

Note: You are allowed a **total** of 4 free late days for the **entire semester**. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

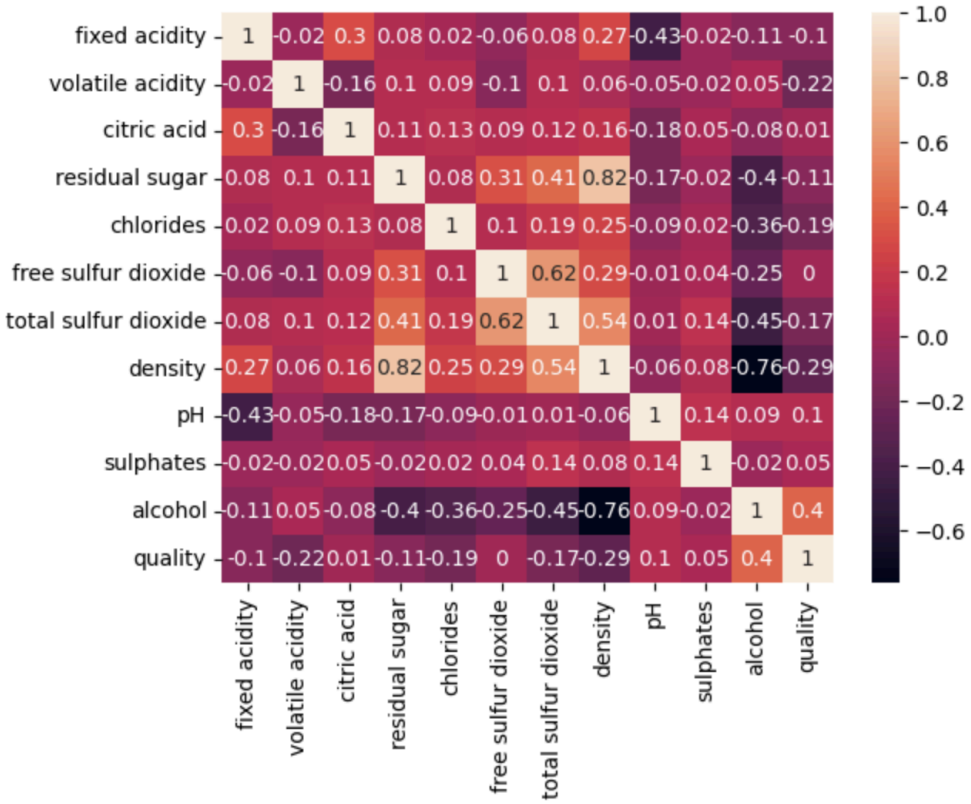
## Overview

The data set we analyze includes physicochemical test measurements of white wine. The goal is to measure the wine's quality. In our models, we aim to classify wine as either “good” quality, or “bad” quality. Some of the attributes include density, chlorides, alcohol, and fixed acidity.

## Pre-Processing

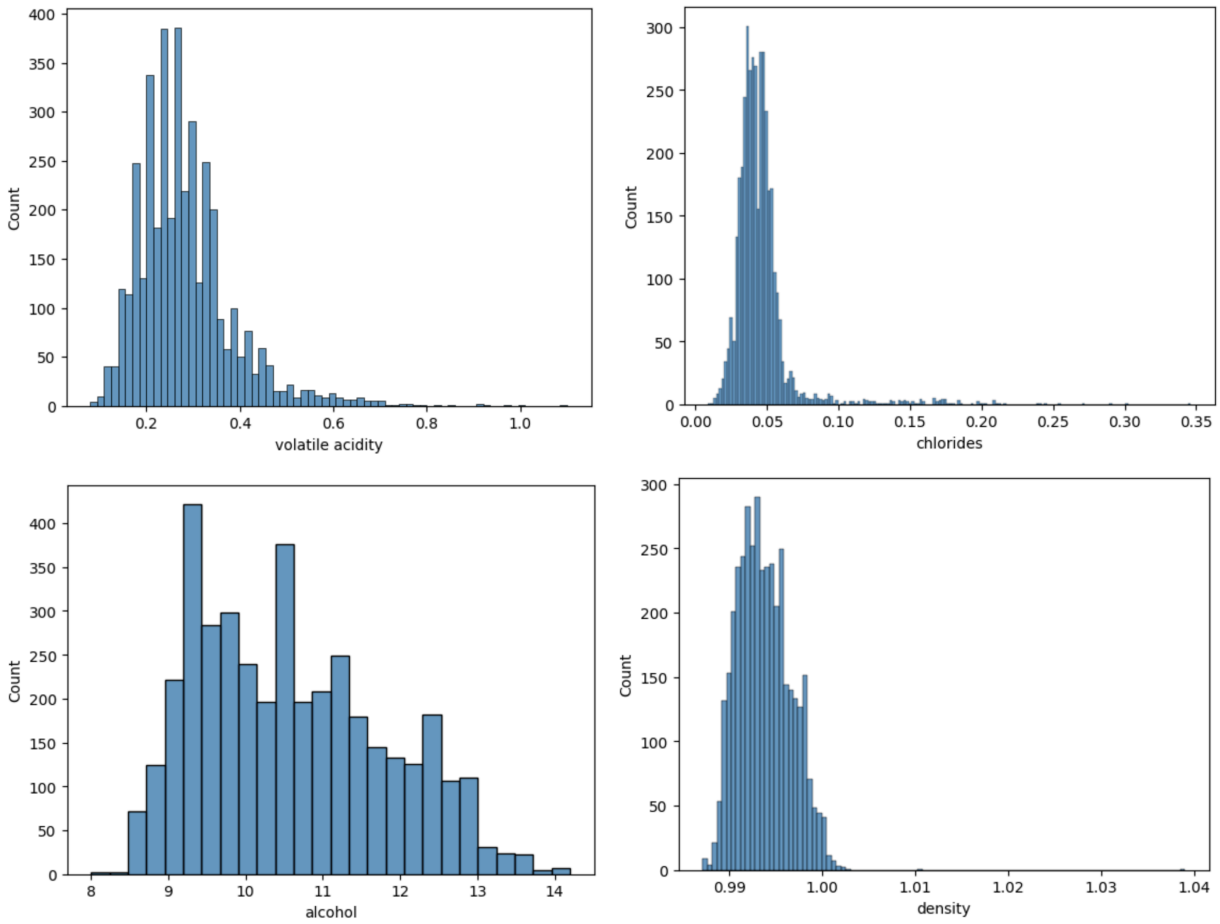
Our dataset has no null values, missing data, or duplicates. The main change to our dataset that came from pre-processing is that we decided to change the way we score the “quality”. The dataset originally scored the quality of the wine on a scale of 3 to 9, but after running both the decision tree classifier and the random forest classifier, we found that the model was unable to predict the extremes of the wine quality (scores of 3, 4, 8, 9). Because of this, we decided to reclassify the quality as two classes, good and bad. Good quality being anything that was originally classified as a 6 or higher, and bad quality being anything that was originally classified as a 5 or below. After making this change, we were able to see much higher accuracy scores for our models, which were originally scoring around 0.5 even after hyper-parameter tuning.

## Feature Selection



Using the above correlation matrix, we chose features with the highest correlation to our target variable “quality”. The features we use in our models are density, alcohol, chlorides, and volatile acidity, as they are the most correlated to “quality”.

The features we have chosen are normally distributed, as shown by the following histogram plots:




# Plain Decision Tree Classifier

## Hyper-parameter tuning/ cross validation

| Regular DT | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Attempt # | max_depth | min_samples_le | min_samples_sp | max_features | max_leaf_nodes | max_impurity_de | Accuracy score | Test Accuracy |
| 1 | 7 | 1 | 2 | None | None | 0 | 0.7389659521 | 0.74 |

For hyper-parameter tuning, our initial run of GridSearch gave us an accuracy score of 0.739. We tried using other hyper-parameters not listed on this chart as well as changing the possible values in the GridSearch, but we were unable to get an accuracy score higher than 0.739, so we decided to stick with our initial results.

For cross-validation, after plugging in several different values to cv, we continued to get the same accuracy score. This means that our accuracy is not biased and should perform similarly on unseen/test data.

**Classification Report**

For our plain decision tree classifier model, we received an accuracy score of 0.74. This measures the percentage of correct predictions (both true negatives and true positives) out of all predictions. This score is not excellent, but it is acceptable. One possible reason for this accuracy score is the afore mentioned issue of the model not being able to predict the quality accurately when the quality is extremely low or extremely high. This failure to predict the extreme cases of quality is likely due to the lack of data for these extreme cases, which is outlines below.

# of data with quality = 3: 20

# of data with quality = 4: 153

# of data with quality = 5: 1175

# of data with quality = 6: 1788

# of data with quality = 7: 689

# of data with quality = 8: 131

# of data with quality = 9: 5

As you can see, there is a severe lack of data for wine quality of 3, 4, 8, and 9. This makes it much more difficult for out model to classify them.

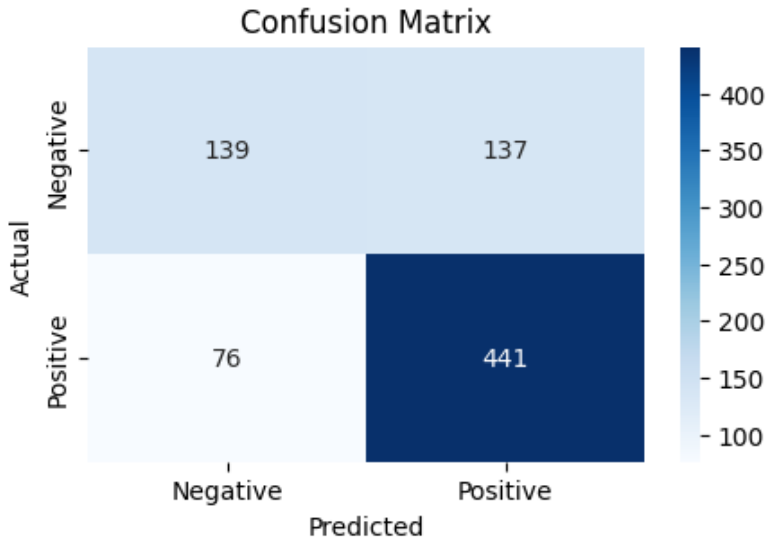
As for precision, we received percentages of 0.65 and 0.78 for our “bad quality” and “good quality” classes, respectively. Precision measures the amount of true positives out of all positives classified (true and false positives). In other words, how many of the predicted positives were truly positive. This shows that for our “bad quality” class, precision was quite low, and our model often incorrectly classified data as “bad quality”. However, our model was better at correctly classifying “good quality”. Our model likely did poorly with the “bad quality” data due to the lack of data mentioned previously. “Bad quality” is considered anything with a score of 3, 4, or 5, and as mentioned before, we have a severe lack of data for qualities equaling 3 and 4.

In regards to recall, we received percentages of 0.55 and 0.84 for our “bad quality” and “good quality” classes, respectively. Recall measures the proportion of actual positives that were correctly identified by the model (out of true positives and false negatives). For our “bad quality” class, recall was very low, indicating that our model could not correctly identify “bad quality” well at all. In fact, it’s only slightly better than random guessing. However, for “good quality”, our

model was much more efficient in predicting positive instances. With a recall percentage of 0.84, our model is able to identify most of the relevant results, at least for “good quality”.

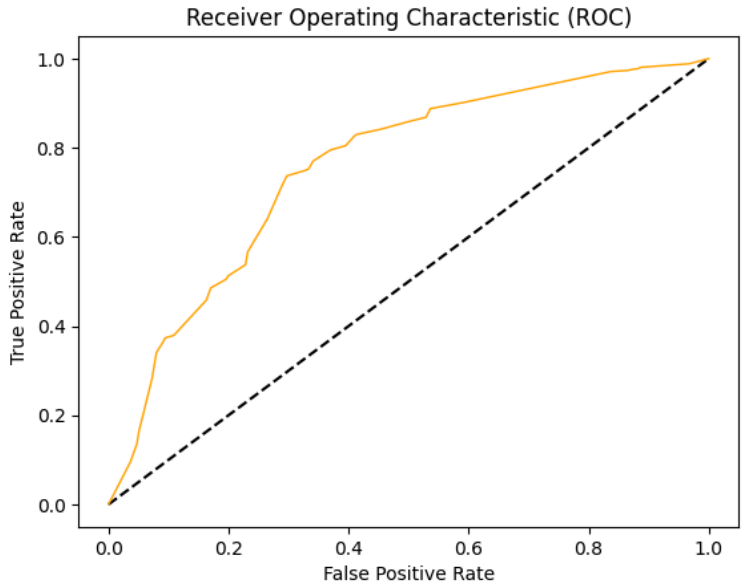
Finally, our F1 score, (harmonic mean of precision and recall), came out to be 0.59 and 0.81 for our “bad quality” and “good quality” classes, respectively. Our “bad quality” class is being predicted with lower effectiveness. This low score shows that the model struggles to identify positive instances as well as avoid false positives for this class. Our “good quality” class is being predicted with a much higher accuracy. It’s able to correctly classify a good portion of our data.

**Confusion Matrix**



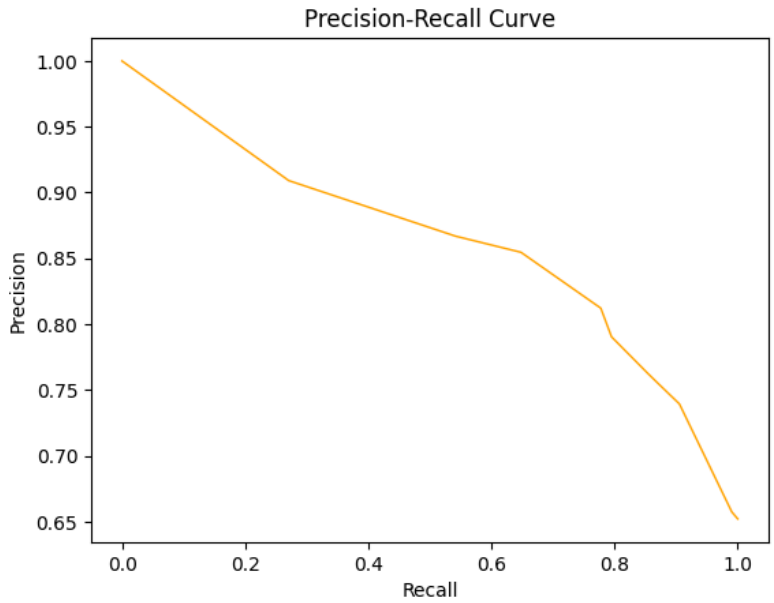

In the confusion matrix above, we can see that the plain decision tree classifier model was best at classifying true positives. However, it was not so good at predicting true negatives.

**ROC Curve**




AUC: 0.7558

**Precision-Recall Curve**




Average precision-recall score: 0.82

Looking at the curves above, we can see that both graphs show the lack of accuracy in our model. Analyzing the area under the curves, we see that there could be major improvements to this model. Overall, this model did moderately well with predicting one class over the other. However, in order to improve our curves and score higher for our “bad quality” class, we may

have to keep tuning or preprocess our data once more. Although we believe in order to make a sizeable improvement, we would require more data.

# Random Forest Classifier

**Hyper-parameter tuning/ cross validation**

| Random Forest DT | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Attempt # | max_depth | min_samples_le | min_samples_sp | max_features | n_estimators | max_leaf_nodes | Accuracy score | Test Accuracy |
| 1 | 5 | 3 | 2 | None | 150 | 20 | 0.7351828499 | |
| 2 | 7 | 5 | 2 | None | 50 | 25 | 0.7351828499 | |
| 3 | 9 | 5 | 2 | None | 25 | 30 | 0.7427490542 | 0.74 |

For the hyper-parameter tuning of our random forest classifier, we were able to make a subtle improvement on our accuracy score, as shown above. We made these improvements by changing the possible inputs for each hyper-parameter in our GridSearch. After the third recorded attempt, further changes did not improve our accuracy score, so we chose to stick with the hyper-parameters given in our third attempt.

For our random forest classifier model, we received an accuracy score of 0.74. This measures the percentage of correct predictions (both true negatives and true positives) out of all predictions. This score is not excellent, but it is decent. This model scored about the same as our plain decision tree classifier in regards to accuracy. There was not a significant improvement.

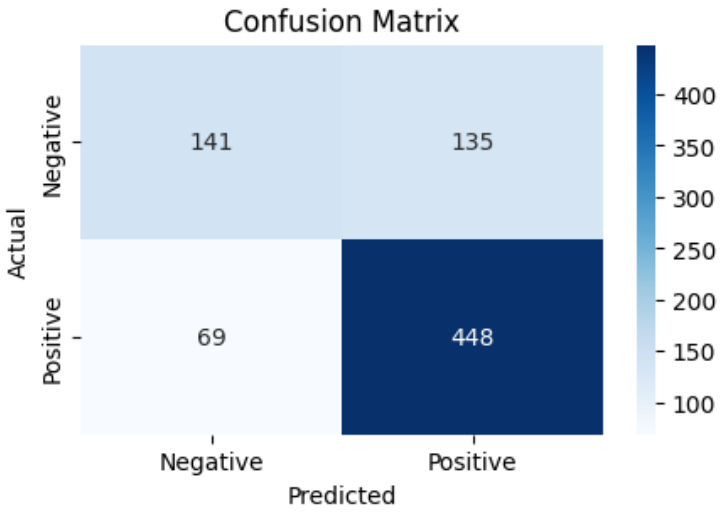
As for precision, we received percentages of 0.67 and 0.77 for our “bad quality” and “good quality” classes, respectively. Precision measures the amount of true positives out of all positives classified (true and false positives). In other words, how many of the predicted positives were truly positive. This shows that for our “bad quality” class, precision was quite low, and our model often incorrectly classified data as “bad quality”. However, our model was better at correctly classifying “good quality”. A possible reason for these scores could be a data imbalance. Though, something important to note is that the random forest classifier did have a higher precision score for our “bad quality” class as opposed to our first model.

In regards to recall, we received percentages of 0.51 and 0.87 for our “bad quality” and “good quality” classes, respectively. Recall measures the proportion of actual positives that were correctly identified by the model (out of true positives and false negatives). For our “bad quality” class, recall was very low, indicating that our model could not correctly identify “bad quality” well at all. In fact, it’s about on par with random guessing, and slightly worse than our first model. However, for “good quality”, our model was much more efficient in predicting positive instances. With a recall percentage of 0.87, our model is able to identify most of the relevant results, at least for “good quality”. It’s slightly better than our first classifier model.

Finally, our F1 score, (harmonic mean of precision and recall), came out to be 0.58 and 0.81 for our “bad quality” and “good quality” classes, respectively. Our “bad quality” class is being predicted with very low effectiveness. This low score shows that the model struggles to identify

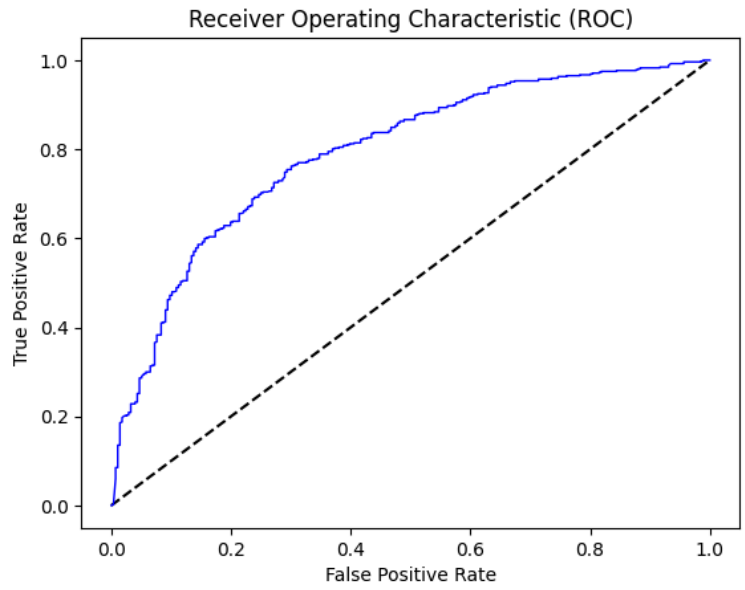
positive instances as well as avoid false positives for this class. Our “good quality” class is being predicted with a much higher accuracy. It’s able to correctly classify a decent portion of our data.

**Confusion Matrix**



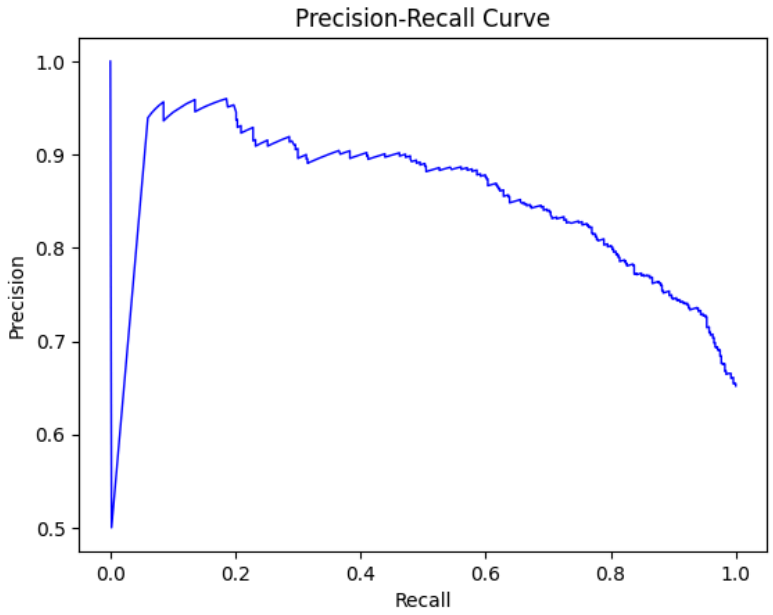

In the confusion matrix above, we can see that the random forest model was best at classifying true positives. However, it was not so good at identifying true negatives, similar to our previous model. This could be another indicator of a data imbalance.

**ROC Curve**




AUC: 0.7918

### Precision-Recall Curve




Average precision-recall score: 0.86

Looking at the curves above, we can see that both graphs show the lack of accuracy in our model. Analyzing the area under the curves, we see that there could be major improvements to this model. Overall, this model did moderately well with predicting one class over the other. These curves do however show a slight improvement from our previous model, as they have a greater area under both of their curves.

## AdaBoost Classifier

### Hyper-parameter tuning

| AdaBoost DT | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Attempt # | n_estim | learning_rate | | | | | Accuracy score | Test Accuarcy |
| 1 | 250 | 0.1 | | | | | 0.7313997478 | |
| 2 | 300 | 0.1 | | | | | 0.7326607818 | 0.73 |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |

For the AdaBoost Classifier, n_estimates and learning_rate were the only parameters that had a positive impact on our accuracy score. For these two hyper-parameters, after our first attempt, we were only able to slightly improve the accuracy score by adjusting inputs in our GridSearch.

For our AdaBoost classifier model, we received an accuracy score of 0.73. This measures the percentage of correct predictions (both true negatives and true positives) out of all predictions.

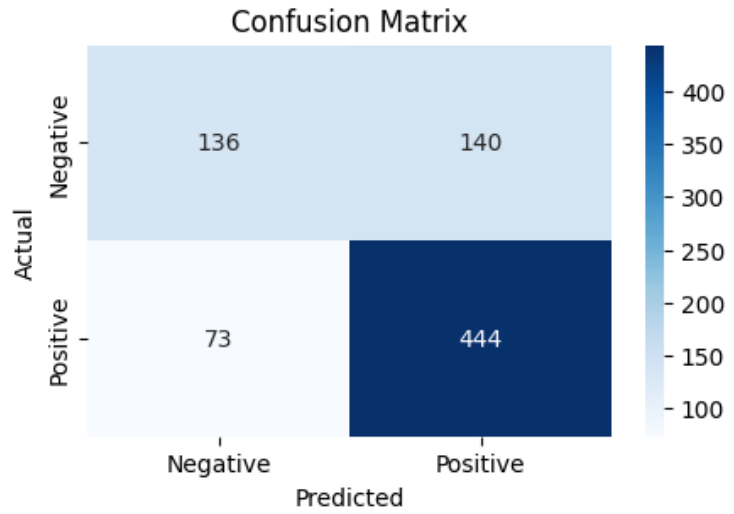
This score is not excellent, but it is decent. This model scored about the same as our previous models in regards to accuracy, only slightly worse.

As for precision, we received percentages of 0.65 and 0.78 for our “bad quality” and “good quality” classes, respectively. Precision measures the amount of true positives out of all positives classified (true and false positives). In other words, how many of the predicted positives were truly positive. This shows that for our “bad quality” class, precision was quite low, and our model often incorrectly classified data as “bad quality”. However, our model was better at correctly classifying “good quality”. A possible reason for these scores could be a data imbalance. This Model scored very similarly to our random forest model.

In regards to recall, we received percentages of 0.55 and 0.84 for our “bad quality” and “good quality” classes, respectively. Recall measures the proportion of actual positives that were correctly identified by the model (out of true positives and false negatives). For our “bad quality” class, recall was very low, indicating that our model could not correctly identify “bad quality” well at all. In fact, it’s only slightly better than random guessing, and slightly better than our random forest model. However, for “good quality”, our model was much more efficient in predicting positive instances. With a recall percentage of 0.84, our model is able to identify most of the relevant results, at least for “good quality”. It’s slightly worse than our random forest model.

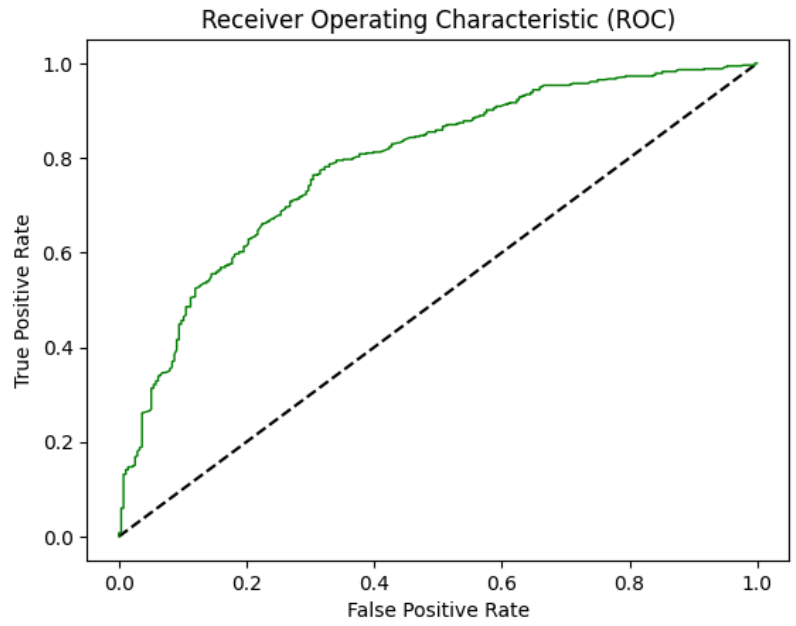
Finally, our F1 score, (harmonic mean of precision and recall), came out to be 0.59 and 0.81 for our “bad quality” and “good quality” classes, respectively. Our “bad quality” class is being predicted with very low effectiveness. This low score shows that the model struggles to identify positive instances as well as avoid false positives for this class. Our “good quality” class is being predicted with a much higher accuracy. It’s able to correctly classify a decent portion of our data.This is very similar to our previous model.

**Confusion Matrix**



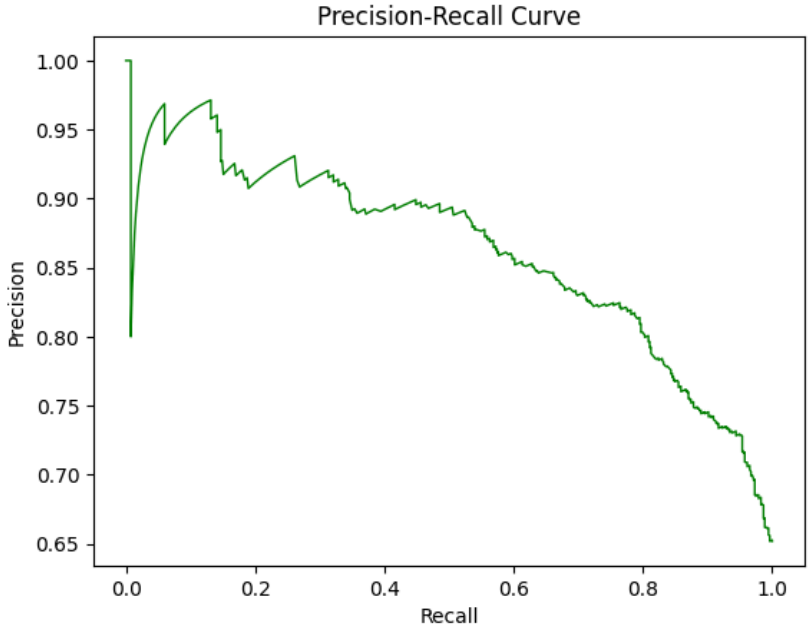
In the confusion matrix above, we can see that the AdaBoost model was best at classifying true positives. This model performed slightly worse at identifying true positives as well as true negatives, though not by much. This could be another indicator of a data imbalance.

**ROC Curve**




AUC: 0.7879

**Precision-Recall Curve**



Average precision-recall score: 0.86

Looking at the curves above, we can see that both graphs show the lack of accuracy in our model. Analyzing the area under the curves, we see that there could be major improvements to this model. Overall, this model did moderately well with predicting one class over the other. These graphs indicate that the AdaBoost showed no improvement from our random forest model, but instead performed slightly worse. Again, this shows an imbalance in our data.

# XGBoost Classifier

### Hyper-parameter tuning

| XGBoost DT | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Attempt # | n_estimator | learning_rate | max_depth | max_leaves | | | Accuracy score | Test Accuarcy |
| 1 | 200 | 0.01 | 5 | None | | | 0.751893492 | |
| 2 | 200 | 0.01 | 6 | 25 | | | 0.7531568167 | 0.74 |

For XGBoost, there were four hyper-parameters that made an impact in our accuracy score. By tuning these hyper-parameters, we were able to make a subtle increase in our accuracy score.

For our XGBoost classifier model, we received an accuracy score of 0.74. This measures the percentage of correct predictions (both true negatives and true positives) out of all predictions. This score is not excellent, but it is decent. This model scored about the same as our previous models in regards to accuracy.

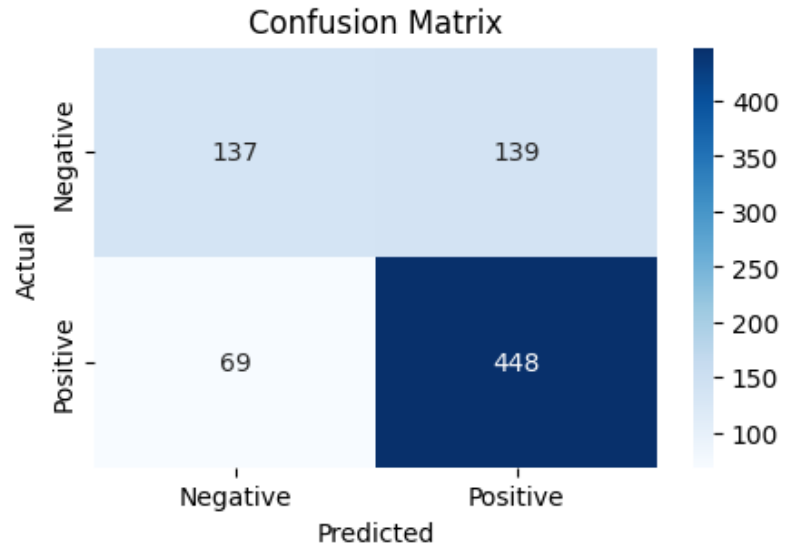
As for precision, we received percentages of 0.67 and 0.76 for our “bad quality” and “good quality” classes, respectively. Precision measures the amount of true positives out of all positives classified (true and false positives). In other words, how many of the predicted positives were truly positive. This shows that for our “bad quality” class, precision was quite low, and our model often incorrectly classified data as “bad quality”. However, our model was better at correctly classifying “good quality”. A possible reason for these scores could be a data imbalance. This Model scored very similarly to our AdaBoost model.

In regards to recall, we received percentages of 0.50 and 0.87 for our “bad quality” and “good quality” classes, respectively. Recall measures the proportion of actual positives that were correctly identified by the model (out of true positives and false negatives). For our “bad quality” class, recall was very low, indicating that our model could not correctly identify “bad quality” well at all. In fact, it’s essentially on par with random guessing, and the lowest score out of all of our models. However, for “good quality”, our model was much more efficient in predicting positive instances. With a recall percentage of 0.87, our model is able to identify most of the relevant results, at least for “good quality”. It’s slightly better than our AdaBoost model.

Finally, our F1 score, (harmonic mean of precision and recall), came out to be 0.57 and 0.81 for our “bad quality” and “good quality” classes, respectively. Our “bad quality” class is being predicted with very low effectiveness. This low score shows that the model struggles to identify positive instances as well as avoid false positives for this class. Our “good quality” class is being

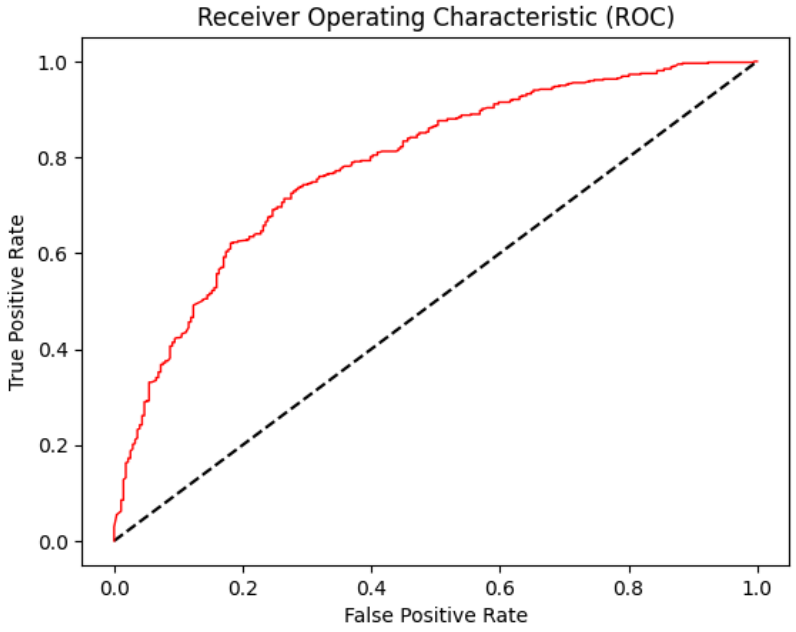
predicted with a much higher accuracy, though it's only decent. This is very similar to our previous models.

**Confusion Matrix**



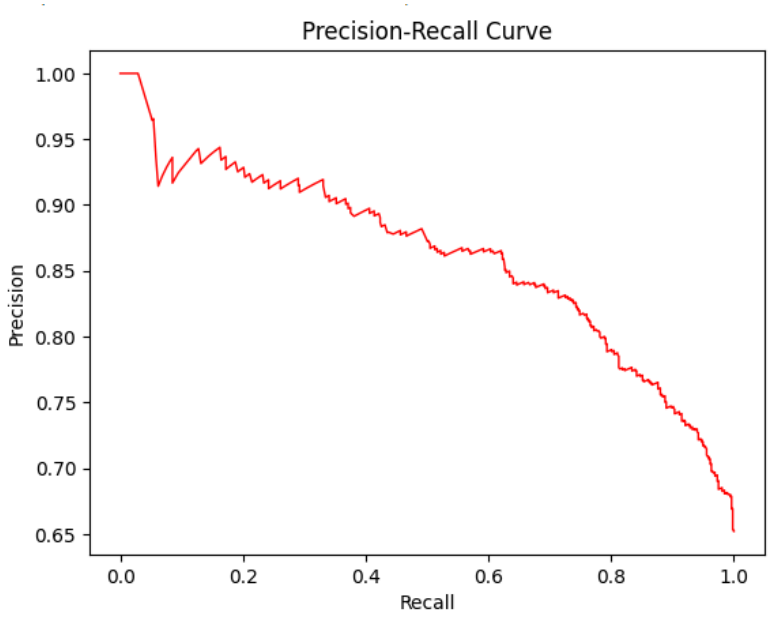

In the confusion matrix above, we can see that the AdaBoost model was best at classifying true positives. This model performed about the same in terms of identifying true positives and negatives in regards to our previous model. This could be another indicator of a data imbalance.

**ROC Curve**



AUC: 0.7846

**Precision-Recall Curve**




Average precision-recall score: 0.86

Looking at the curves above, we can see that both graphs show the lack of accuracy in our model. Analyzing the area under the curves, we see that there could be major improvements to this model. Overall, this model did moderately well with predicting one class over the other. These graphs indicate that the XGBoost showed little to no improvement from our AdaBoost model. Again, this shows an imbalance in our data.

## Model Comparison

Overall, we noticed little improvement from one model to the next. We suspect there could be an imbalance in our data, as noted extensively in the report above. Because of this, it is difficult to conclude which model performed the “best”.

However, with the evaluation metrics we were able to gather and analyze, we conclude that our random forest model performed the best out of the four constructed. We did not come to this conclusion based on one or two metrics, but all of them combined. The primary reasons why we believe our random forest model performed the best is based on the confusion matrix and curves. For our confusion matrix, the random forest model was able to identify the most true positives and most true negatives out of the four models. Additionally, the random forest model had the best scores for both ROC and Recall-Precision curves.

## References

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://pandas.pydata.org/docs/getting_started/intro_tutorials/05_add_columns.html

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

## Assumptions

We assume only two classes in our data: good quality, or bad quality (represented as 1 and 0, respectively).