Predicting Flight Delays with Machine Learning

Sammy Rosofsky

1 Introduction

Flight delays are a major annoyance of travelling. For passengers, a delay can ruin carefully scheduled plans, and for airlines they can cause country-wide disruptions resulting in loss of revenue or a higher operation cost. This project looks at flights in the United States from January of 2019 and attempts to predict when flights will be delayed based on various factors. The variables involved can broadly be divided into two categories: those that airlines can not control, like temperature, average wind speed, or total flights leaving from an airport in a given time period, and those that airlines can control, like the age of the plane and the length of the flight. My goal is to develop a model that airlines would be able to use to plan around the uncontrollable variables by limiting any causes of flight delays that they can.

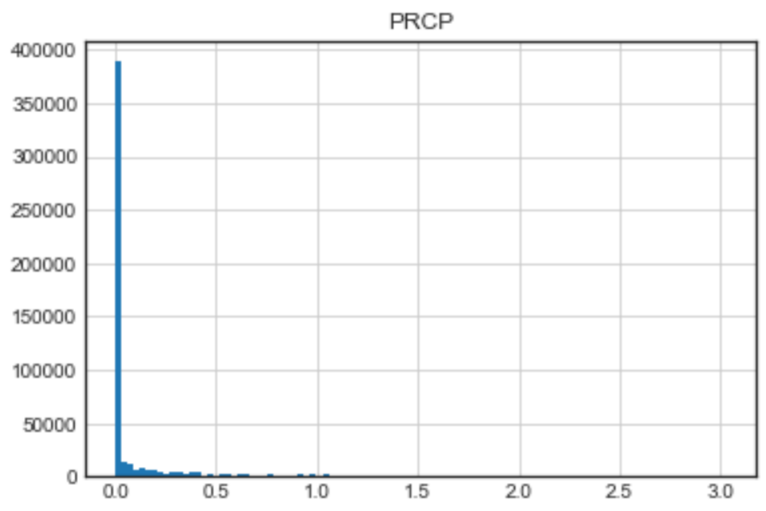
2 Data

The data I am using was collected by user Jen Wadkins on kaggle and was sourced from the US Bureau of Transportation and National Centers of Environmental Information. The full dataset consists of flights for the entire year of 2019, but I will just be using flights from January, which total just under 500,000, for the sake of having more reasonable run-times for my models. The variables in the data are as follows:

- DEP_DEL15: Binary categorical data for whether flight was delayed, 0 for no and 1 for yes. This will be the predicted variable.
- DAY_OF_WEEK: Day of the week
- DISTANCE_GROUP: General distance of a flight, ordered into large groups
- DEP_BLOCK: Block of time in the day
- SEGMENT_NUMBER: Segment of flight path that plane is on
- CONCURRENT_FLIGHTS: Number of flights leaving airport in the same department block
- NUMBER_OF_SEATS: Number of seats on plane
- CARRIER_NAME: Carrier of flight
- AIRPORT_FLIGHTS_MONTH: Average flights per month for the departure airport
- AIRLINE_FLIGHTS_MONTH: Average flights per month for airline
- AIRLINE_AIRPORT_FLIGHTS_MONTH: Average flights per month for both the airline and airport
- AVG_MONTHLY_PASS_AIRPORT: Average Passengers per month for the airport
- AVG_MONTHLY_PASS_AIRLINE: Average Passengers per month for the airline

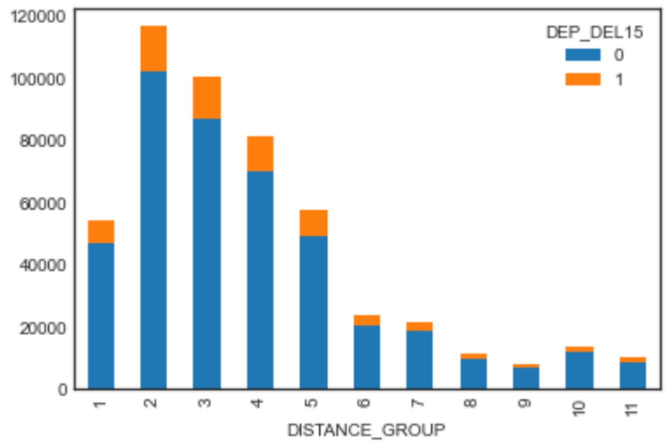
- FLT_ATTENDANTS_PER_PASS: Ground employees per passenger for airline
- PLANE_AGE: Age of plane
- DEPARTING_AIRPORT: Airport of departure
- LATITUDE: Latitude of departing airport
- LONGITUDE: Longitude of departing airport
- PREVIOUS_AIRPORT: Previous airport plane departed from
- PRCP: Inches of precipitation for the day of departure
- SNOW: Inches of snowfall
- SNWD: Inches of snow on the ground
- TMAX: Max temperature for the day
- AWND: Max wind speed for the day

The first thing I took a look at was the continuous data to see if anything stood out. Unsurprisingly, these variables were not normally distributed. For example, here is the histogram for precipitation:

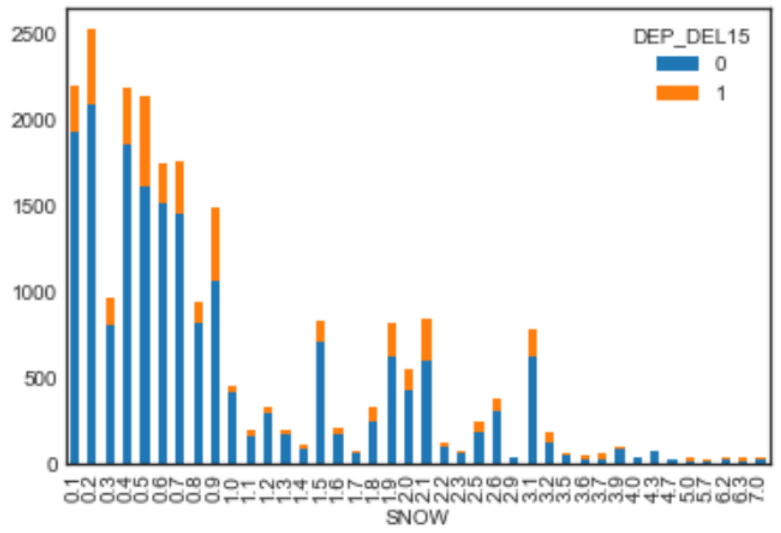


It makes sense that precipitation, and likewise snow, will be distributed in this way, after all most days don't have any rain or snow. This will be something to keep in mind when developing models.

My next step was to look at the distribution of delays throughout each variable to see if any trends stood out. A bit more surprisingly, it became hard to really notice any trends at all. For an example of a categorical variable, here is delays based on distance group:



This seems fairly proportional for each distance group. Likewise, the following graph shows delays based on snowfall (filtered for SNOW>0, so the graph isn't just visually dominated by the days where there is no snow):



This one was quite surprising. It seems fairly intuitive that more snow would lead to more delays, but this didn't exactly seem to be the case. Seeing a lack of significant correlation among the variables was a trend throughout the dataset. It's possible that there do exist strong correlations that simply aren't obvious from visualization, but will influence the models through hundreds of thousands of observations.

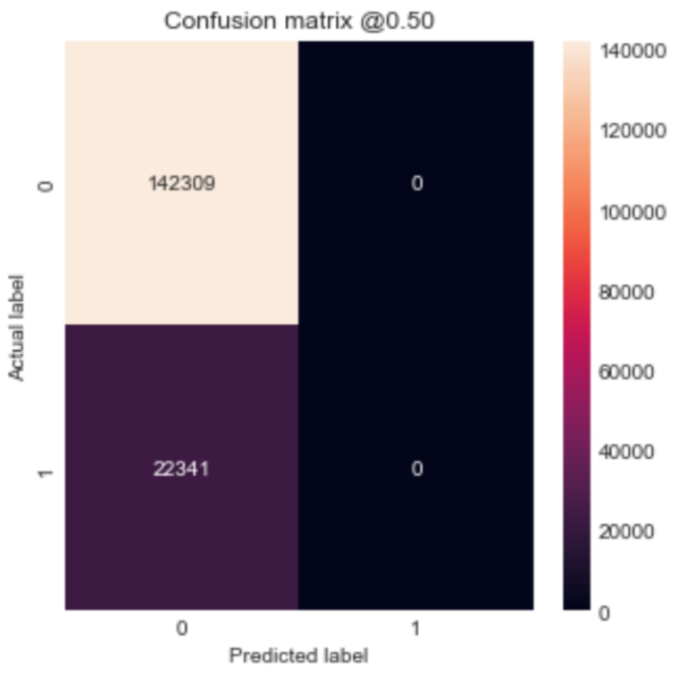
The data as created was already very clean, with most categorical data already in numerical form and no missing values, so minimal data cleaning was needed. Due to the lack of obvious correlations, along with the massive size of the data and large amount of explanatory variables, I removed the more complicated categorical variables like airline and airport. It would simply take too much time and be outside of the time scope of this project to run models with these variables included. To set up the data for training, I randomly split $\frac{2}{3}$ of the data into a training set and the remainder into a testing set

3 Analysis

Logistic Regression

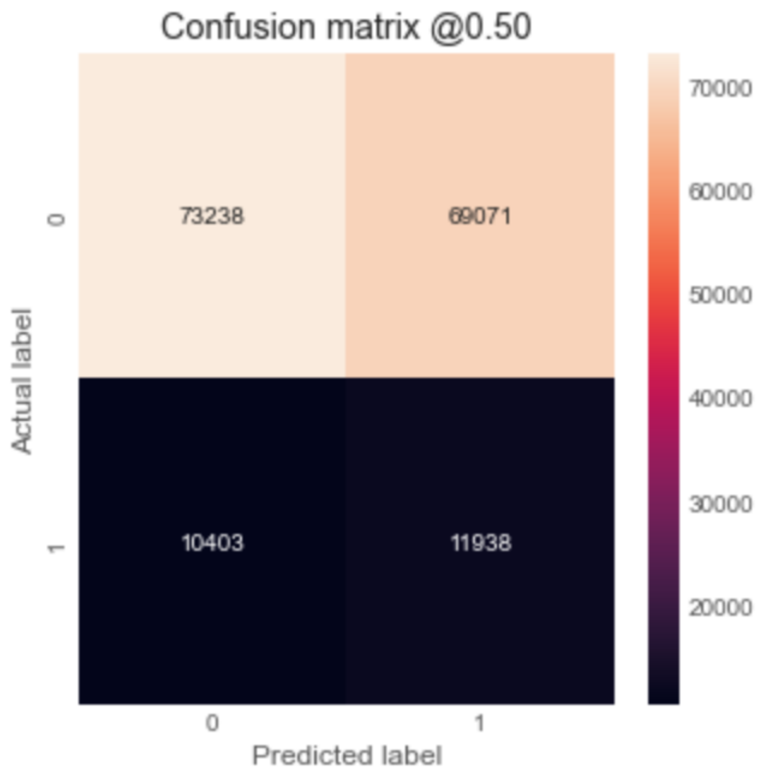
The first model I made was a simple logistic regression, which maps variable inputs to a percentage. The main advantage of logistic regression is that in addition to assigning observations a predicted class, it gives us an overall probability for an observation being in a specific class, and it also gives us coefficients for predictors so we can see which are the most important features of a model. The disadvantage of logistic regression is that it assumes linearity for its predictors, so would be a poor fit for non-linear data.

After fitting the regression, it was able to predict 86.4% of flight delays correctly. This seemed quite good, until I looked at the following confusion matrix:



The model simply predicted there would never be a delay, which happened exactly 86.4% of the time. This is clearly a useless model, which is also reflected by its ROC-score of .5. My guess for what happened is that the correlations of the explanatory variables with flight delays didn't actually end up being that significant, so without much to differentiate the target classes, the model simply predicts the much more likely event overall.

Because it is much more likely for a flight to not be delayed, I decided to next try fitting a logistic regression with the classes of flight delays weighted. This would penalize the model more for incorrectly predicting an actual flight delay than predicting a flight that wasn't delayed. This returned the following confusion matrix:



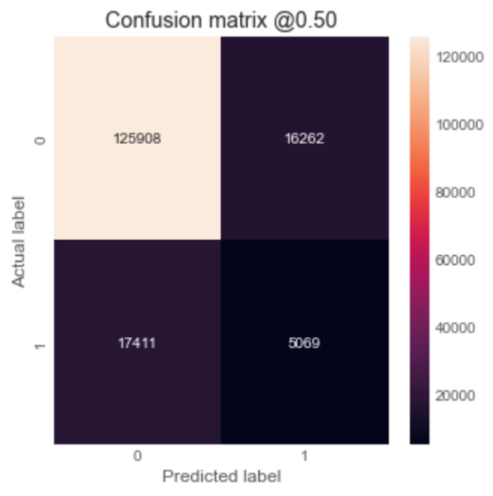
This model resulted in the opposite problem. Instead of just predicting that a flight delay would never happened, it essentially flipped a coin for whether a flight would be delayed. Technically this resulted in a marginally better ROC-score of .524, but this model was still effectively useless. It seems like it would be very difficult to salvage a usable model that is strictly a logistic regression, so rather than try to optimize this I will look into alternative models, and hopefully one of those will be of use.

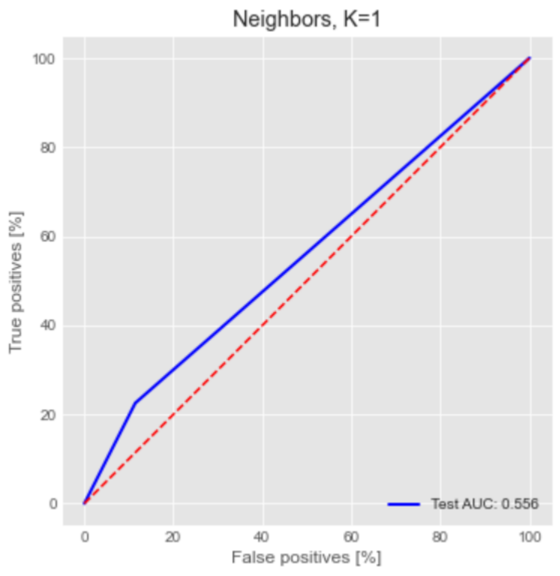
KNN

As stated previously, one of the problems with using logistic regression is that the model is inherently linear even when the underlying data isn't, which is the case for the data I am using. Because of this, it would be a good idea to look into more flexible models. An example of one such model is K-nearest neighbors (KNN). This model uses an algorithm that takes in a point, makes a group with the k points that are closest to it, and then calculates the overall probability of the points in the group being in either target class, assigning the initial point to the class that has the higher probability in the group. The benefit of KNN is that it is completely non-parametric, meaning it makes no assumptions of the underlying relations between variables. This is helpful if the data at hand is not linear.

Different models using KNN can be fit using different values for k. In general, lower values of k are more flexible but in turn can lead to an over-fitting of the model, while higher high values of k approach a linear decision boundary. The latter can be observed when fitting a KNN model to my data with $k = 100$. The resulting decision matrix is virtually identical to that of the logistic regression, where nearly every observation is predicted to be no delay.

On the contrary fitting with $k = 1$ leads to a slight improvement in the model's ROC-score, as demonstrated with the following confusion matrix and AUC graph:





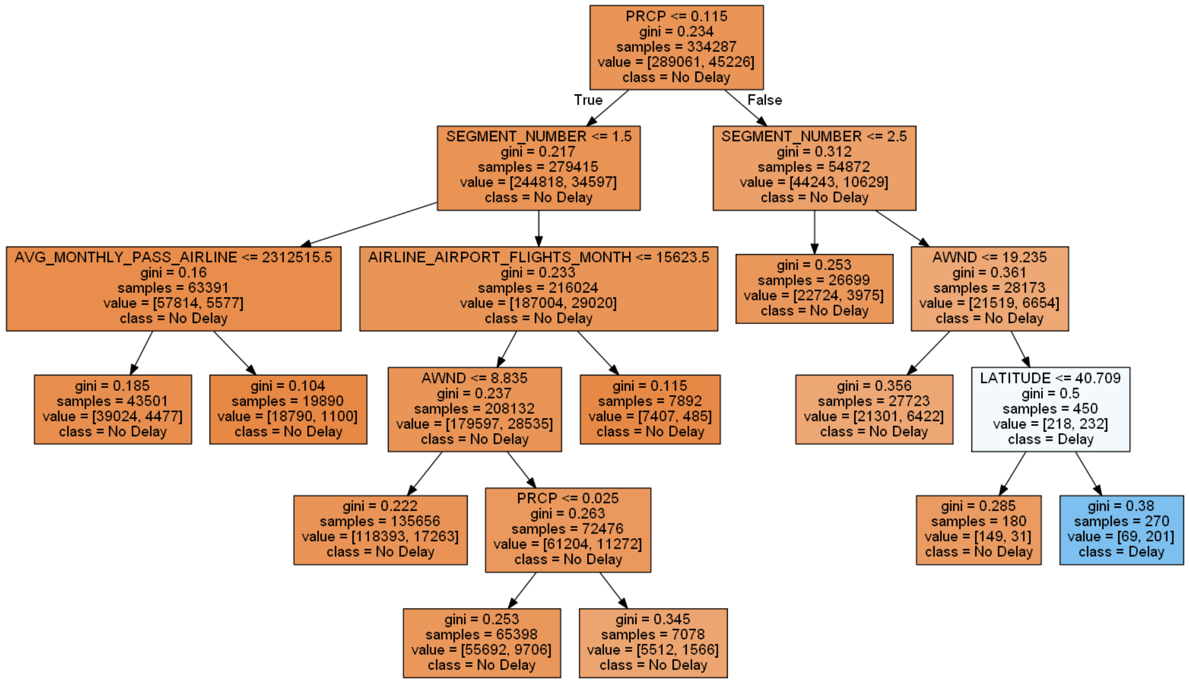
This is still not a very useful model, given that most real flight delays are predicted as not delayed, and most predicted flight delays are actually not delayed. However, this does demonstrate a model that isn't simply guessing all for one class or coin flipping between classes.

Other values of k can be implemented, and typically values of k around 5-10 produce the best results. In this case, fitting a model with $k = 5$ underperformed against the $k = 1$ model, with still having high false negative and false positive rates and a lower ROC-score of .538.

In practice, as it turns out, a major downside of implementing KNN models is that they are computationally very heavy. Each model took about half an hour to run, so it became unreasonable to spend hours trying to fine tune k, especially given that it was unlikely to result in a significantly improved model.

Decision Tree

The last group of models I will implement are tree-based, starting with the basic decision tree. This model works by splitting up the explanatory variables into regions, and then assigning each region a class based on which level is more common in the resulting area. The benefit of decision trees is that they are very easy to interpret. The following is a decision tree with 10 leaves:



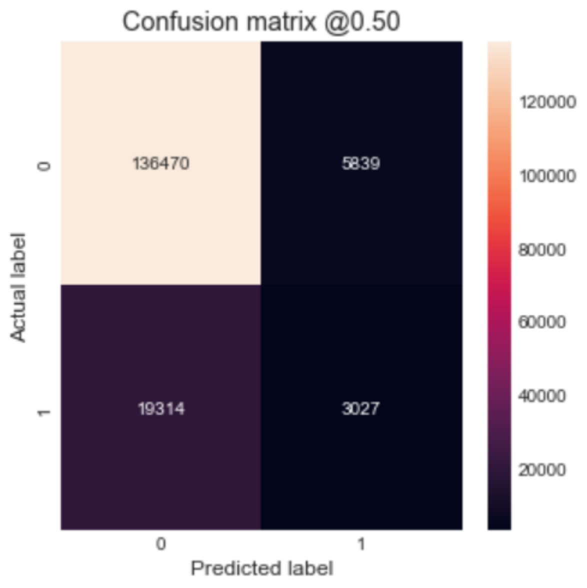
PRCP <= 0.115 gini = 0.234 samples = 334287 value = [289061, 45226] class = No Delay
True
False
SEGMENT_NUMBER <= 1.5 gini = 0.217 samples = 279415 value = [244818, 34597] class = No Delay
SEGMENT_NUMBER <= 2.5 gini = 0.312 samples = 54872 value = [44243, 10629] class = No Delay
AVG_MONTHLY_PASS_AIRLINE <= 2312515.5 gini = 0.16 samples = 63391 value = [57814, 5577] class = No Delay
AIRLINE_AIRPORT_FLIGHTS_MONTH <= 15623.5 gini = 0.233 samples = 216024 value = [187004, 29020] class = No Delay
gini = 0.253 samples = 26699 value = [22724, 3975] class = No Delay
AWND <= 19.235 gini = 0.361 samples = 28173 value = [21519, 6654] class = No Delay
gini = 0.185 samples = 43501 value = [39024, 4477] class = No Delay
gini = 0.104 samples = 19890 value = [18790, 1100] class = No Delay
AWND <= 8.835 gini = 0.237 samples = 208132 value = [179597, 28535] class = No Delay
gini = 0.115 samples = 7892 value = [7407, 485] class = No Delay
gini = 0.356 samples = 27723 value = [21301, 6422] class = No Delay
LATITUDE <= 40.709 gini = 0.5 samples = 450 value = [218, 232] class = Delay
gini = 0.222 samples = 135656 value = [118393, 17263] class = No Delay
PRCP <= 0.025 gini = 0.263 samples = 72476 value = [61204, 11272] class = No Delay
gini = 0.285 samples = 180 value = [149, 31] class = No Delay
gini = 0.38 samples = 270 value = [69, 201] class = Delay
gini = 0.253 samples = 65398 value = [55692, 9706] class = No Delay
gini = 0.345 samples = 7078 value = [5512, 1566] class = No Delay

Decision trees are useful for drawing direct relations between different values and predicted classes. This resulting tree performed quite poorly, however, with it having the same problem with logistic regression where it simply predicted almost every flight as being not delayed.

Tree Ensembles (Bagging and Random Forest)

One of the biggest disadvantages of decision trees is that they tend to have high variance, meaning that take the same dataset and splitting it into random samples can result in wildly different trees due to even small differences between the samples, which is not typically the case for other models. This happens because a different split at the first level will have a cascading effect where different splits will then be made in the resulting sub-trees. One way of addressing this is to make an ensemble model, where multiple different trees are modeled and then averaged out. This will reduce the overall variance, although the resulting model will lose the simplicity the ability to easily interpret of a single decision tree model.

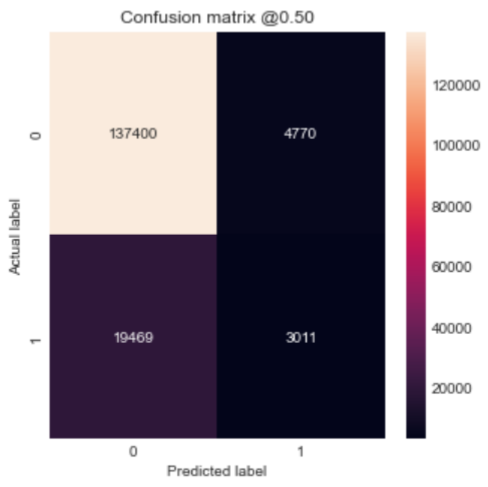
The first approach is called bootstrap aggregation, or bagging. This method takes random samples of the data and makes a tree for each sample, and in the end averages the trees. The resulting model returned an ROC-score of .547 with the following confusion matrix:

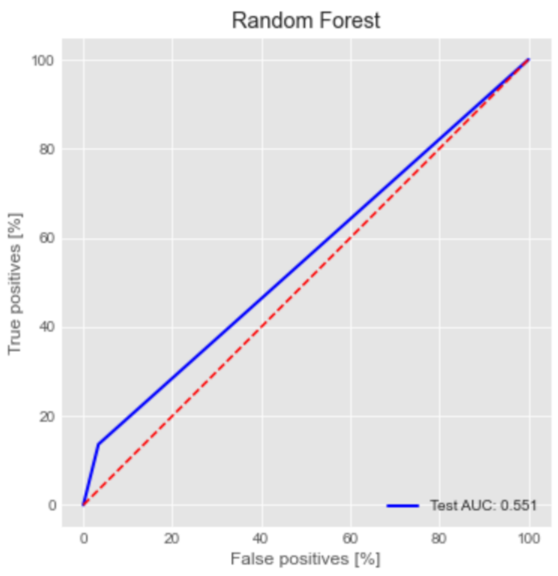


This model performs similarly to KNN model with $k = 1$, with more false negatives, fewer false positives, and fewer true positives correctly predicted.

Another approach to tree ensembles is random forest. Whereas bagging splits the data by observation, random forest splits the data on predictors, with each model having a random sample of predictors. The idea behind random forest is that bagged trees can be highly correlated if there is a single predictor that dominates the model relative to other predictor, which results in more similar bagged trees that don't end up reducing variance as much as they could. With random forest, sample models can be made without the most tightly correlated variables being the primary splits of the top level branches, which improves the robustness of the model when faced with new data.

The random forest model resulted in the following confusion matrix and AUC graph with an ROC-score of .55:





The very slight improvement over just using bagging implies that there isn't a problem with any single predictor dominating the tree models. Overall, these tree ensemble models perform similarly to the previous best model, KNN with $k = 1$, but take a fraction of the time to compute.

4 Conclusion

For this project, several models for prediction were developed implementing logistic regression, KNN, and tree-based models. Overall, the best performing models were KNN with $k = 1$ and random forest. Their metrics are displayed below:

Model	Accuracy	ROC
KNN	.795	.556
Random Forest	.853	.551

Ultimately, even these ”best” models aren’t of much real use. While accuracy was high, this was mostly due to how there were simply many more flights they did not get delayed rather than flights that were, and the models tended to just predict more flights not being delayed overall. The models had extremely high false negative and false positive rates, so using them to actually predict whether a flight delay would happen is impractical.

The primary cause for these problems seems to be that there simply wasn’t a strong correlation between the predictors used and the likelihood of a flight being delayed, resulting in little differentiation between the target classes. If I were to explore the data more, I would try to see if any transformations on the underlying data improved the performance of the models, although given that both parametric and non-parametric models performed poorly this likely wouldn’t result in large improvements. Another approach would be to see if there is any additional data that can be added to the model, and whether stronger predictors can be found.