Omar Farooq
Sharbesh Adhikari
CS 484 Project

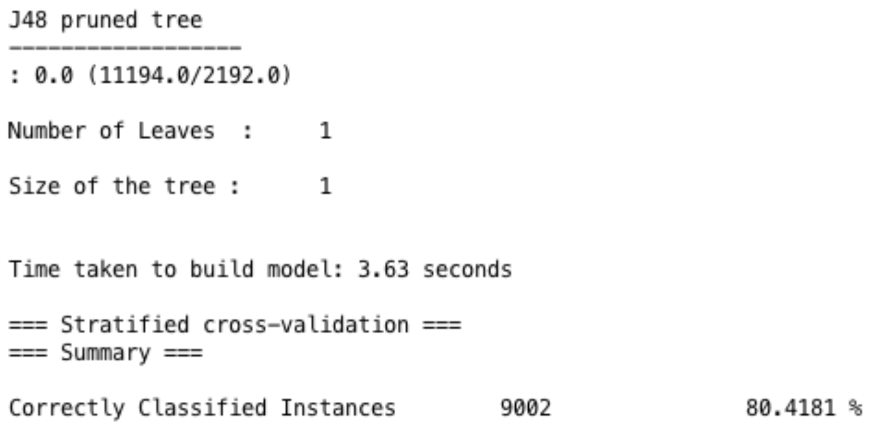
Predicting Flight Delays Using C4.5 Decision Trees and Naïve Bayes

Abstract: The Department of Transportation (DOT) gives out flight data to anyone who wants find information about it. With this data, my partner and I wanted to predict flight delays. From this project, we were able to find that predicting flight delays can be accurately predicted with minimal information about a flight. Using data mining techniques, we can find ways to make improvements on flight scheduling so that flights can be on time, and make it so that people can make it to their destination with as little delay as possible. Making flights on time as accurately as possible would reduce the costs of flight rescheduling, and save airliners money and resources.

Introduction: We wanted to predict flight delays as accurately as we could. We thought this would be a good way to help people get to where they are going with as much ease and enjoyment as possible. We wanted to predict flight delays using information about a flight that is given to passengers before their flight. Such attributes we would include would be pre flight data like Departure time, arrival time, origin, airline, etc. We also wanted to compare our data with different algorithms that we would implement from scratch, and compare our algorithms with other algorithms using Weka.

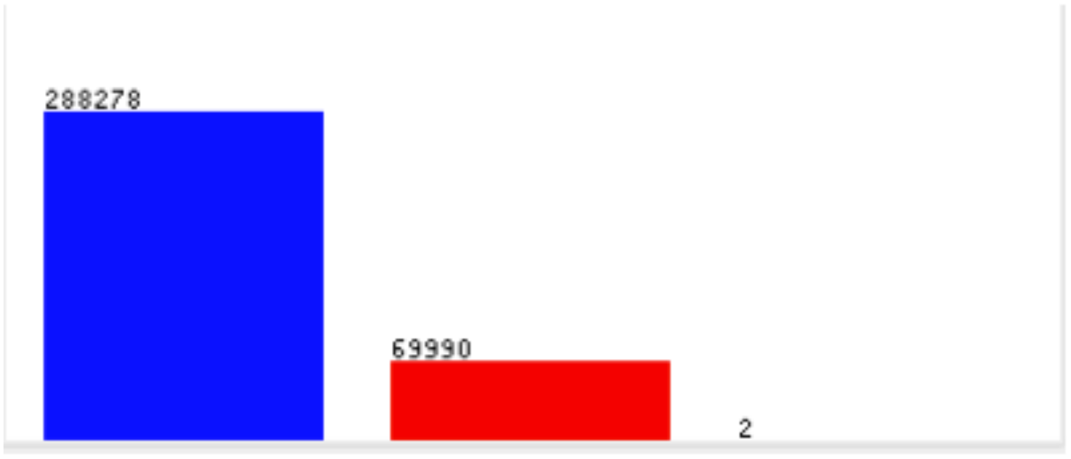
Originally, we predicted our flights would be delayed if the distance of the flight was a large amount. However, using Weka, we found that distance was not a good attribute use to predict flight delays. We found that just the airline, original airport a passenger flew out of, and the departure and arrival times are our most important attributes we would need to predict flights with a relatively high accuracy. Any additional attributes would not change the percentage of the accuracy by a significant amount. This paper, we will describe our findings, and what our best approach would be to accurately predict flight delays.

Related work: We were surprised that we even found a relatively high amount of accuracy of flight delays. We were wondering why other organizations have not been implementing this approach to reduce flight delays. So we did some research. After running our dataset on Weka, we found that based on no rules, and no decision tress at all, Weka automatically assigns each flight to test on as ‘On time’. Using this approach, about 80% of all flights were on time. Which meant that any set of flight data we would download from the Department of Transportation would have 80% of its flights on as time. Even if the flight is delayed, almost half of the flights are delayed because of weather related reasons. The figure below shows this finding.



```
J48 pruned tree
------------------
: 0.0 (11194.0/2192.0)

Number of Leaves  :     1

Size of the tree :      1


Time taken to build model: 3.63 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        9002               80.4181 %
```

Weka using no rules to predict flight delays



Weka showing 80% of all of the flights as ‘on time’ (in blue)

We found that many people have also seen similar results as ours. KnowDaily.com also predicts flight delays, however they use attributes such as weather (they ask their users to submit their flight within 3 days before their flight to give an accurate result). Our approach will give people accurate result weeks/months before your flight. Their approach would give an even higher accuracy if their flight would be delayed or not if they submitted their flight within 3 days before their flight.

Airbnb does a lot of their data analysis using RandomForest for their customers to predict information for passengers, which is what we found to build the most accurate model for predicting flight delays. We could not find out more on what Airbnb was doing with RandomForest because they don't give out a lot of their own information, but we found it interesting that they also found what we were using as well.

Solution: We were able to get up to 88.57% accuracy just using where the original flight departed from, the airline, the departure time, and arrival time to determine if a flight would be delayed or not. We did a lot of data preprocessing which we describe in the Experimental section below, implemented Naïve Bayes and the J48 algorithm in python, and ran our data set on Weka's algorithms. We found that Random Forest, which is an Ensemble Method with J48 decision tree classifier, gave us the 88% accuracy below. We describe more on these results in the experimental section.

Experimental:

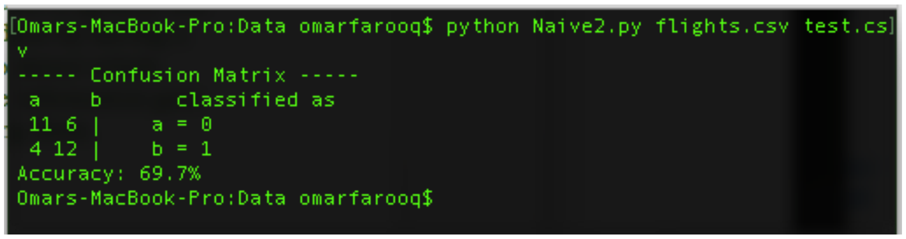
Data: We got our data from the Department of Transportation (DOT). They record every all domestic flight from the past decade. We decided to get the past 5 years' worth of flights. We downloaded them one month at a time, saved it as a csv file, and used python scripting to combine all of the months into one document named 'data.csv'. From there, we wrote another python script to just print out every 100th line, so that our data would be small enough to fit into our computer's memory. This outputted file was named 'flights.csv' and had about 30,000 flights, and was a size of 300kb.

Experimental setup: We used a lot of software to get an accurate and readable amount of data. We used Excel to format our data into csv. With our formatted data, we got it to fit into Weka. In Weka, we found that a lot of our data had to be normalized and modified. We used a combination of Weka, and python scripting to preprocess our data.

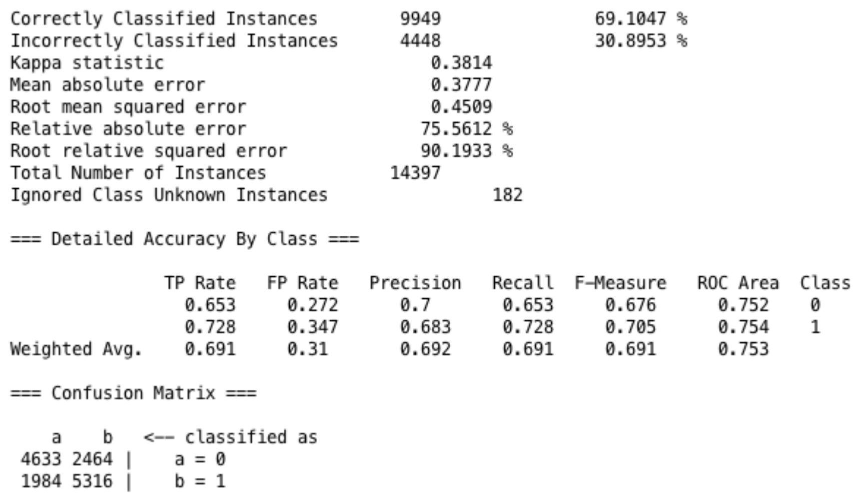
On Weka, we used SMOTE to balance the total number of delayed flights, and on time flights to be equal, so that we didn't have one class dominating another. We also originally had up to 109 attributes that we had to reduce. We decided which attributes to reduce by having a few factors, such as any post-flight data, Weka's "Select Attribute" feature to detect which attributes to remove using BestFirst CfsSubsetEval algorithm, and calculating the information gain with our J48 algorithm to find the attributes with the lowest gain and removing them.

After that we resampled our data, just to make sure our data was consistent. We had to remove any floating point numbers Weka automatically calculated for us that we had to covert into integers. After all of this, we had our data to be accurate as we possibly could.

Experimental results: To solve our problem, we wrote our own algorithms and compared them with Weka's. We used Naïve Bayes and tested it with Weka's own implementation of Naïve Bayes. For our version of Naïve Bayes, we tested our data using training data, and got consistent results with Weka's Naïve Bayes approach. Our results are displayed below:



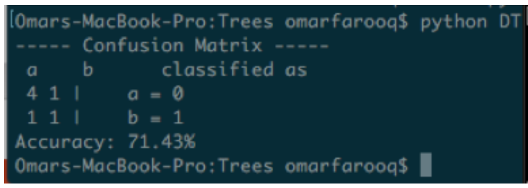
```
[Omars-MacBook-Pro:Data omarfarooq$ python Naive2.py flights.csv test.cs]
v
----- Confusion Matrix -----
 a    b      classified as
 11 6 |     a = 0
 4 12 |     b = 1
Accuracy: 69.7%
Omars-MacBook-Pro:Data omarfarooq$
```



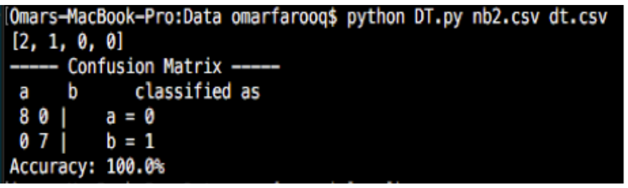
```
Correctly Classified Instances        9949               69.1047 %
Incorrectly Classified Instances      4448               30.8953 %
Kappa statistic                          0.3814
Mean absolute error                      0.3777
Root mean squared error                  0.4509
Relative absolute error                 75.5612 %
Root relative squared error             90.1933 %
Total Number of Instances            14397
Ignored Class Unknown Instances                182

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                 0.653    0.272    0.7       0.653    0.676     0.752     0
                 0.728    0.347    0.683     0.728    0.705     0.754     1
Weighted Avg.    0.691    0.31     0.692     0.691    0.691     0.753

=== Confusion Matrix ===

    a    b   <-- classified as
 4633 2464 |    a = 0
 1984 5316 |    b = 1
```

Weka's Implementation of Naïve Bayes using training data for testing options

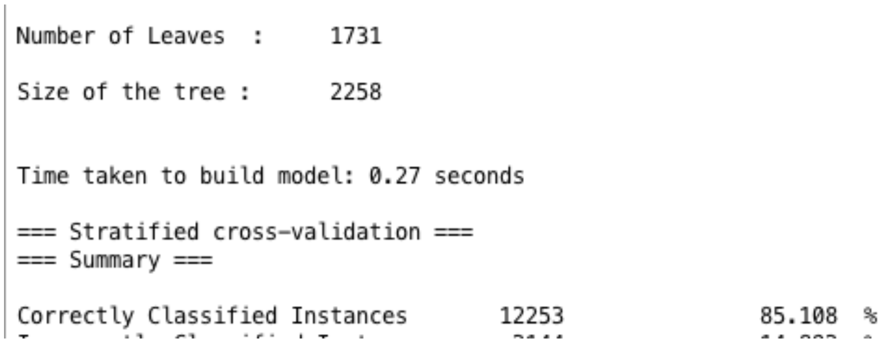
We also got similar results using cross validation



```
Omars-MacBook-Pro:Trees omarfarooq$ python DT
----- Confusion Matrix -----
 a    b      classified as
 4 1 |     a = 0
 1 1 |     b = 1
Accuracy: 71.43%
Omars-MacBook-Pro:Trees omarfarooq$
```

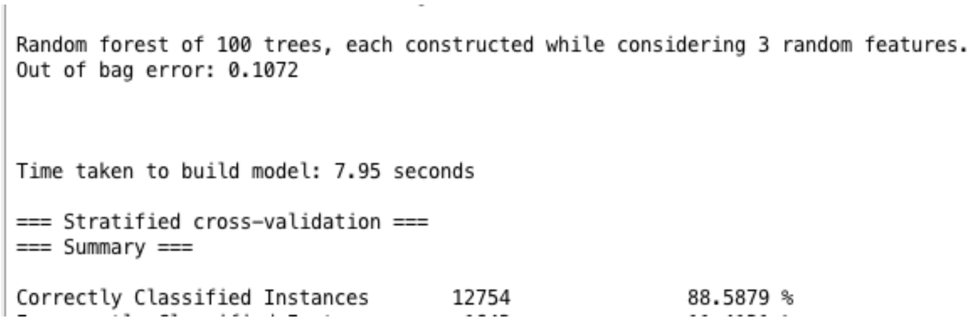


```
Omars-MacBook-Pro:Data omarfarooq$ python DT.py nb2.csv dt.csv
[2, 1, 0, 0]
----- Confusion Matrix -----
 a    b      classified as
 8 0 |     a = 0
 0 7 |     b = 1
Accuracy: 100.0%
```



```
Number of Leaves  :     1731

Size of the tree :      2258


Time taken to build model: 0.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       12253               85.108  %
```

Weka’s J48 Algorithm



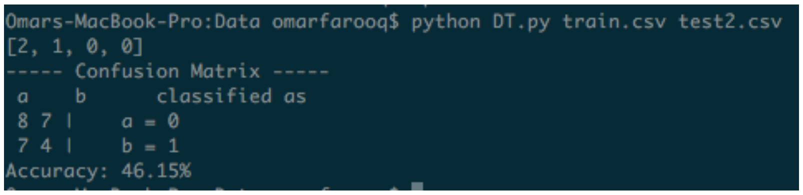
```
Random forest of 100 trees, each constructed while considering 3 random features.
Out of bag error: 0.1072



Time taken to build model: 7.95 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       12754               88.5879 %
```

Weka’s Random Forest

*The J48 may have a more complicated model, which would have a high variance of findings

We think that Random Forest is the best classifier, because J48 gave us a more accurate model than Naïve Bayes. We realized that Naïve bayes looks at probability, but J48 uses information gain to determine which is the best choice, which we thought was smarter to do. It finds the highest gain, decides (splits), and repeats over and over again with the next highest information gain. We also ran an ensemble method for our J48 algorithm using Random Forest in Weka.

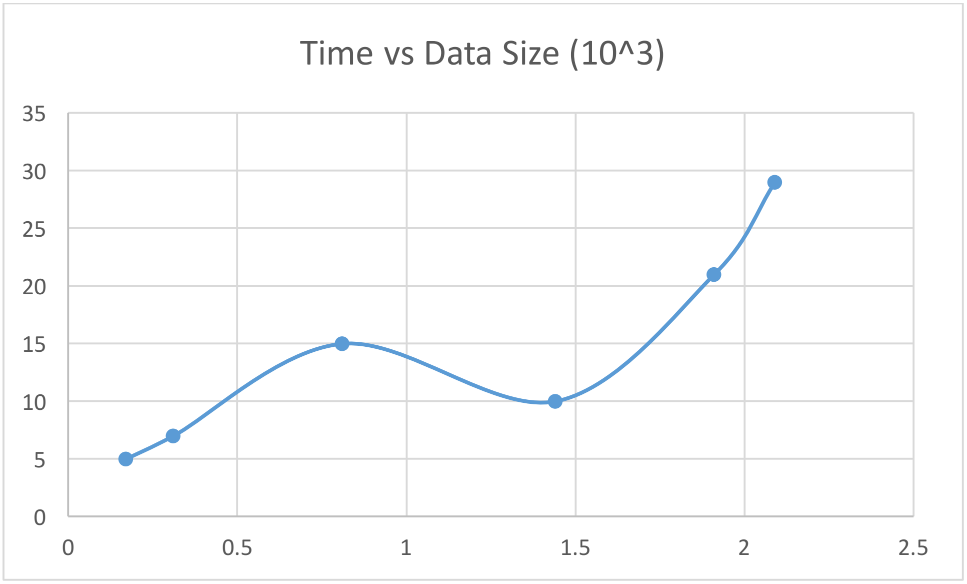
We found that our implementation of the J48 algorithm is very inefficient, and has issues of overfitting. This gave us a great appreciation of what Weka does to build models, as their implementation of J48 does not have an issue with overfitting, and their algorithm is a lot more efficient than ours. Our J48 has a very low training error, but a very high testing error. Also, the larger our data set, the longer it takes and the less accurate it is as well. The smaller our data set is, the more accurate our model would be. With Naïve Bayes, it was a lot faster, more efficient, and quicker to build than with our J48 algorithm. We also got more consistent results that matched Weka's Naïve bayes results.

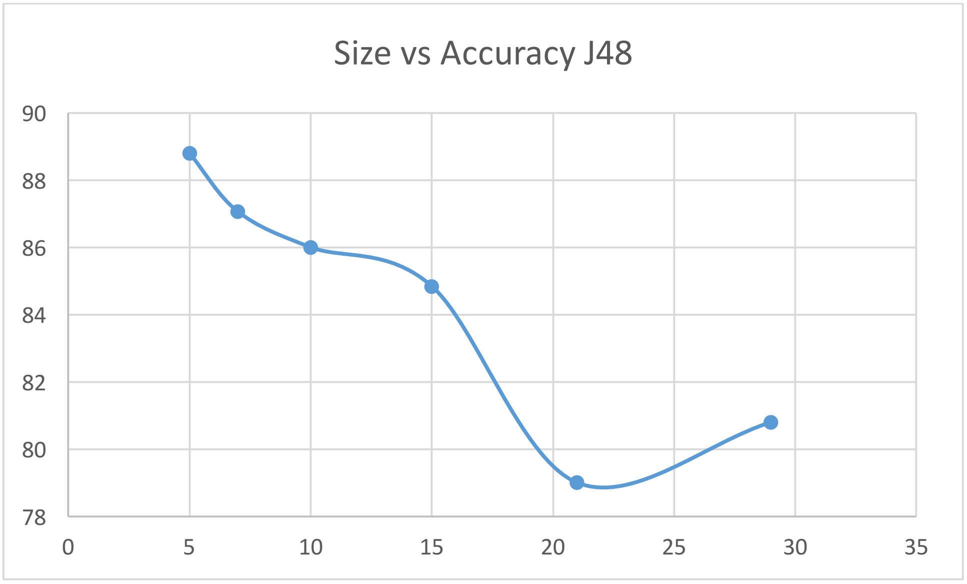


```
Omars-MacBook-Pro:Data omarfarooq$ python DT.py train.csv test2.csv
[2, 1, 0, 0]
----- Confusion Matrix -----
 a    b      classified as
 8 7 |    a = 0
 7 4 |    b = 1
Accuracy: 46.15%
```

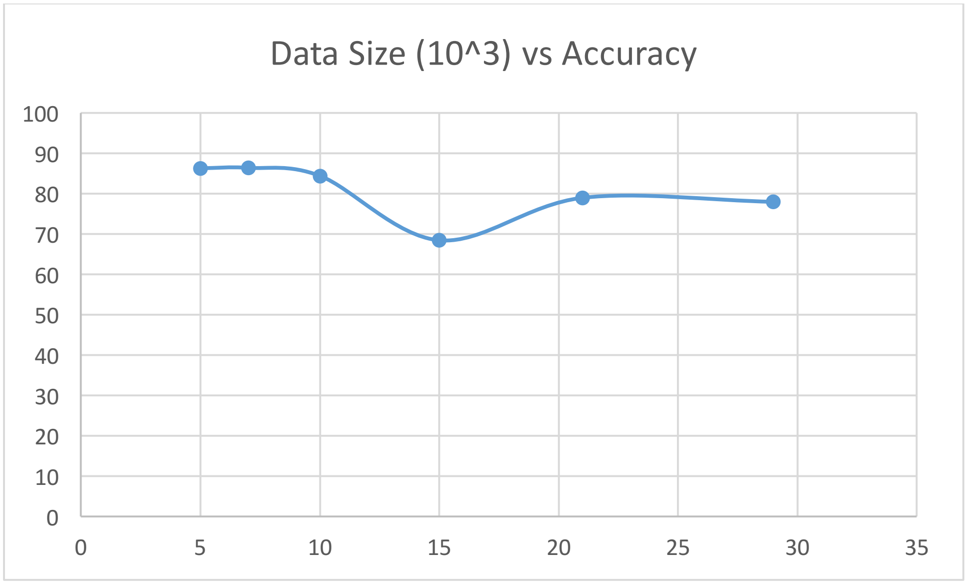
An example of model overfitting for testing J48

The size of the data also affected our results. We found that the size of the data affected our accuracy, and the time it took to build a model. Below are graphs of Time vs Data Size, and Data Size vs Accuracy for Weka's J48 classifier.





We also graphed the same data for Naïve Bayes classifier. We found that the time it took to build a Naïve Bayes model would be very low, and would change from 0-to 0.06 randomly. So we decided it would not be relevant to graph it. However, we thought the size vs accuracy graph would be interesting. Below is a graph showing this:



We see that, for Naïve Bayes, the statistics seemed pretty consistent with time and accuracy. However, for J48, the statistics seemed to be a positive quadratic relationship between the two attributes. This made sense, because Naïve Bayes just does a probabilistic equation to see which class to classify on, and is a very simple approach to making classification, compard to J48. J48 seems to get less accurate with large data. This could be for overfitting, since our model gets more complex with larger data, therefore our accuracy would go down. We can tell our model gets more complex because the time increases with data size, therefore it takes more time to build the model and this makes our model more complex. This also tells us that, if we test out our data with a bigger size, we might even get less accuracy. However, due to computation resource limitations, we could not generate models with bigger than 30,000 flights.

Brief conclusion

Using data that is available online, we were able to find accurate findings for predicting flight delays. We found that the biggest indicators of flight delays are where the origin of the destination is, the arrival time, and the airline. According to our data, it suggests that if you want to reduce flight delays, you would need to work on the airports that have the most delays, the airlines with the most delays, and would need to work on scheduling the ETA arriving time better for passengers. This can not only help the airline industry, but also reduce time, energy, and resources to help move the airline industry towards a better future to better passenger experiences, improvements, and overall technology.

Partner contribution

Omar Farooq: did data preprocessing, wrote report, implemented Naïve Bayes algorithm, implmented c4.5 algorithm, compared algorithms with Weka's algorithms, wrote proposal, parsed data to be accurate/consistent for Excel and Weka, prepared presentation

Sharbesh Adhikari: Helped review and modify the proposal, found flight data, help test for consistencies in the data, implemented some of c4.5 algorithm, prepared presentation