IST 707

Group Project – Flight Data

Jose Reyes, Christopher Romero, Jeffrey Ronay

# **Airline Delays and Cancellations - Project Overview**

Annual flight delays cost an estimated 28.9 billion dollars due to lost time, opportunity costs, cancellations, and missed connections. Passengers bear more than half of the total cost. ​In 2019, passengers experienced delays of more than 95 million minutes. Approximately 20% of the total flights in the United States arrived late by more than 15 minutes.

Clearly, flight delays and cancellations have numerous time and cost implications. In the sections that follow, flight data obtained from the United States Bureau of Transportation Statistics underwent examination using several data modelling techniques. The goal of the modelling is to predict on-time flights accurately, given multiple flight attributes such as weather, airport location, route, month, air traffic control re-routes, and several others.

The data mining/modelling techniques employed follow.

Data Mining Techniques:

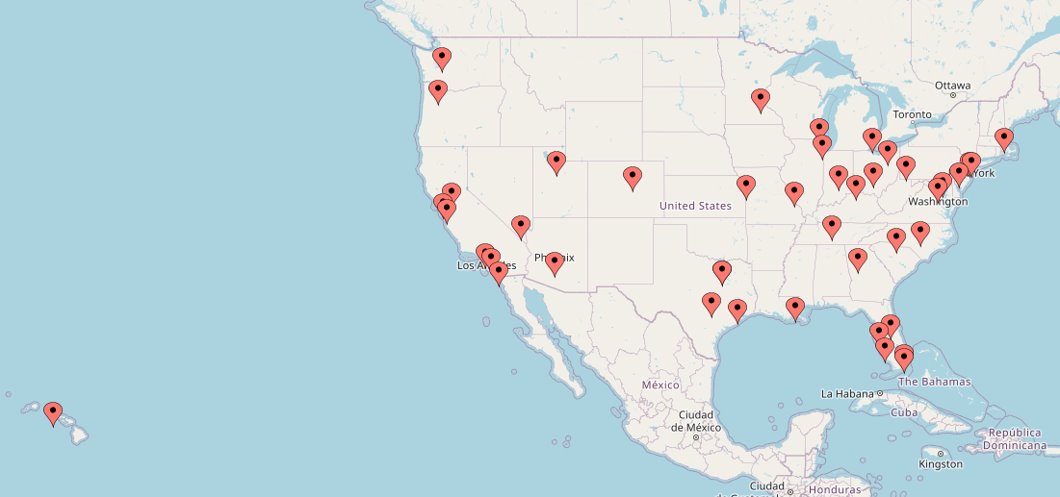
* Association Rule Mining
* Clustering
* Support Vector Machine
* Decision Tree
* Naïve Bayes

# **Data**

The original dataset was downloaded from the Bureau of Transportation Statistics website (<https://www.transtats.bts.gov/OT_Delay/ot_delaycause1.asp?display=data&pn=1>) for the years 2017 through 2019. 2020 data was not included in this analysis due to the ongoing coronavirus pandemic which was considered to represent a large outlier and would likely result in inaccurate modelling.. However, for the purposes of this project, the time range considered is appropriate and scalable (i.e., could be updated as more data resulting from 2020 forward is collected).

The original dataset of 360 airports and 19 airline carriers (between 2017 and 2019) was reduced to the top 50 Busiest Airports in the United States and resulted in 21,196 rows and 22 columns of data.

Figure 1 below illustrates a plot of the data, showing the locations of the 50 busiest U.S. airports.



### Figure 1. A plot of 50 busiest airports used in the analysis.

Each row of the dataset includes the month, airport, airline carrier, and several flight data statistics used as features for the analysis. Note that cleansing the dataset occurred through several different methods, depending on the applied data mining technique. Contained in each modelling section are the precise cleansing algorithms.

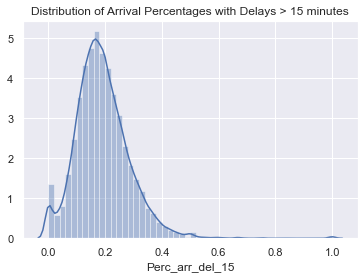
Shown below in Table 1 is a data dictionary (Description of Features) that defines the attribute names and meaning for each data row.

|  |  |
| --- | --- |
| **arr\_flights** | # of flights that arrived at the airport |
| **arr\_del15** | # of flights that arrived >= 15 minutes late |
| **carrier\_ct** | # of flights delayed due to the carrier |
| **weather\_ct** | # of flights delayed due to weather |
| **nas\_ct** | # of flights delayed due to national air system |
| **security\_ct** | # of flights delayed due to security |
| **late\_aircraft\_ct** | # flights delayed because a previous flight using the same aircraft was late |
| **arr\_cancelled** | # of canceled arrivals |
| **arr\_diverted** | # of diverted scheduled arrivals |
| **arr\_delay** | Sum of the delay minutes |
| **carrier\_delay** | Total minutes of delays due to carriers |
| **weather\_delay** | Total minutes of delays due to weather |
| **nas\_delay** | Total minutes of delays due to natl. air service |
| **security\_delay** | Total minutes of delays due to security |
| **late\_aircraft\_delay** | The total minutes of delay due to a previous flight using the same aircraft arriving late. |

### Table 1. Description of Features

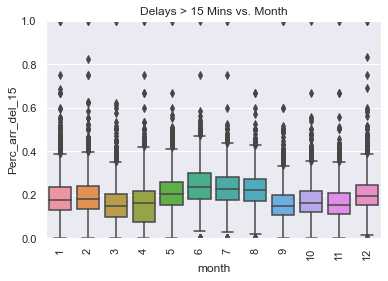
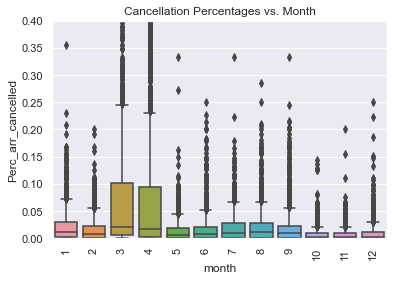
# **Exploratory Data Analysis**

Several columns were divided by the total flights (arr\_flights) to calculate percentages of overall flights delayed over 15 minutes (arr\_del15), delayed due to carrier (carrier\_ct), weather (weather\_ct), national air system (nas\_delay), security (security\_delay), late aircraft (late\_aircraft\_delay), and cancelled flights (arr\_cancelled).



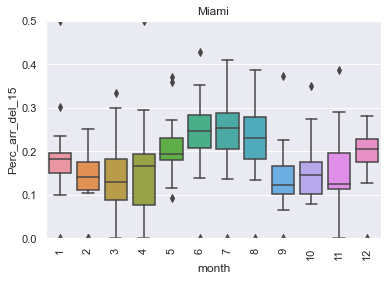
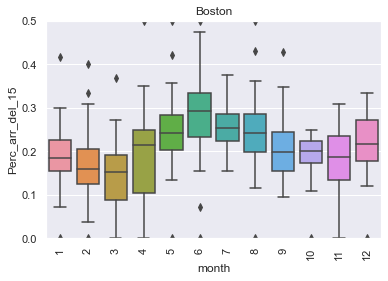
### Figure 2. Distribution of total delays

The distribution of late flights appears to be quite close to a normal distribution with of about 20% of flights having delays over 15 minutes. However, separating this data by month reveals the variability.

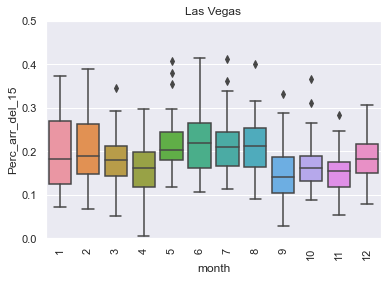
### Figures 3 and 4. Boxplots of delays and cancellations per month

There are more delays in the summer months of June, July, and August, followed by the holiday months of December and January. During the months listed, airports experience the highest traffic, which can lead to more delays. For cancellations, there is considerable variability during March and April likely due to weather.

### Figures 5 and 6. Boxplots of delays in Miami and Boston per month

When looking at delays for each month in a specific city, the variability becomes even more visible. Flights during the summer months of June, July, and August still appear to experience the most delays even in a snowy city such as Boston, where one may expect to see more delays and cancellations in the winter months.



### Figure 7. Boxplot of delays in Las Vegas per month

There are, of course, outliers and exceptions such as Las Vegas where the winter months have similar or higher chances of delays than the summer. Possible explanations include weather and certain dates such as Valentine's Day leads to increased airport traffic.

## **Association Rule Mining**

Before Association Rule Mining (ARM) with R, the percentage features were discretized into four equal bins using Python to allow for proper analysis.

Discretization method on Python

1. #Create 4 equal bins
2. bins\_Perc\_arr\_del\_15 = [-1, .13, .182, .25, 1.1]
3. bins\_Perc\_carrier\_ct  = [-1, .034, .051, .073, 1.1]
4. bins\_Perc\_weather\_ct = [-1, 0.000001, .003, .008, 1.1]
5. bins\_Perc\_nas\_ct = [-1, .030, .052, .083, 1.1]
6. bins\_Perc\_late\_aircraft\_ct  = [-1, .032, .056, .084, 1]
7. bins\_Perc\_arr\_cancelled = [-1, .000001, .008, .024, 1.1]
8. #Create Define Labels
9. labels\_Perc\_arr\_del\_15 = ['LOWEST\_Delay\_Over\_15','LOW\_Delay\_Over\_15','HIGH\_Delay\_Over\_15','HIGHEST\_Delay\_Over\_15']
10. labels\_Perc\_carrier\_ct  = ['LOWEST\_Carrier\_Delay','LOW\_Carrier\_Delay','HIGH\_Carrier\_Delay','HIGHEST\_Carrier\_Delay']
11. labels\_Perc\_weather\_ct = ['LOWEST\_Weather\_Delay','LOW\_Weather\_Delay','HIGH\_Weather\_Delay','HIGHEST\_Weather\_Delay']
12. labels\_Perc\_nas\_ct = ['LOWEST\_NAS\_Delay','LOW\_NAS\_Delay','HIGH\_NAS\_Delay','HIGHEST\_NAS\_Delay']
13. labels\_Perc\_late\_aircraft\_ct  = ['LOWEST\_LateAircraft\_Delay','LOW\_LateAircraft\_Delay',
14. 'HIGH\_LateAircraft\_Delay','HIGHEST\_LateAircraft\_Delay]
15. labels\_Perc\_arr\_cancelled = ['LOWEST\_Cancellation','LOW\_Cancellation','HIGH\_Cancellation','HIGHEST\_Cancellation']
16. #Discretize
17. ARM['Perc\_arr\_del\_15'] = pd.cut(ARM['Perc\_arr\_del\_15'],bins = bins\_Perc\_arr\_del\_15, labels = labels\_Perc\_arr\_del\_15)
18. ARM['Perc\_carrier\_ct'] = pd.cut(ARM['Perc\_carrier\_ct'],bins = bins\_Perc\_carrier\_ct, labels = labels\_Perc\_carrier\_ct)
19. ARM['Perc\_weather\_ct'] = pd.cut(ARM['Perc\_weather\_ct'],bins = bins\_Perc\_weather\_ct, labels = labels\_Perc\_weather\_ct)
20. ARM['Perc\_nas\_ct'] = pd.cut(ARM['Perc\_nas\_ct'],bins = bins\_Perc\_nas\_ct, labels = labels\_Perc\_nas\_ct)
21. ARM['Perc\_late\_aircraft\_ct'] = pd.cut(ARM['Perc\_late\_aircraft\_ct'],bins = bins\_Perc\_late\_aircraft\_ct, labels = labels\_Perc\_late\_aircraft\_ct)
22. ARM['Perc\_arr\_cancelled'] = pd.cut(ARM['Perc\_arr\_cancelled'],bins = bins\_Perc\_arr\_cancelled, labels = labels\_Perc\_arr\_cancelled)

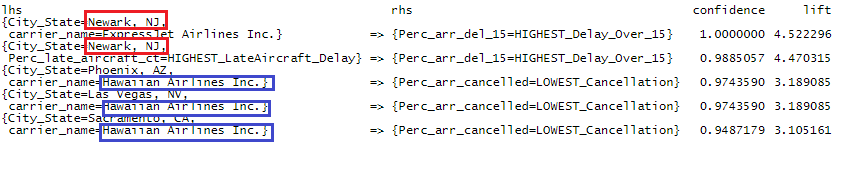
The Apriori principle will be applied to reduce the overall computational complexity. This is based on the principle that if an itemset is frequent, then all subsets must also be frequent. This algorithm will produce a list of rules which each have a set of items that commonly occur together.

Rule effectiveness comprises Support, Confidence, and Lift. Combining these factors enables a proper assessment of the overall effectiveness of the rule.

* Support - How often a rule applies to a given data set
* Confidence - How frequently certain items appear with other things in a transaction
* Lift - The ratio of the confidence of the rule and the expected confidence of the rule

Due to the large variety of airports, carriers, and features, the support for the vast majority of rules is low. The analysis focuses on rules with high confidence and lift.

When mining for rules with the highest percentages of delays over 15 minutes (bin name: HIGHEST\_Delay\_Over\_15) or most elevated rate of cancellations (bin name: Perc\_arr\_cancelled=LOWEST\_Cancellation), the following five rules produced the highest lift and confidence values.

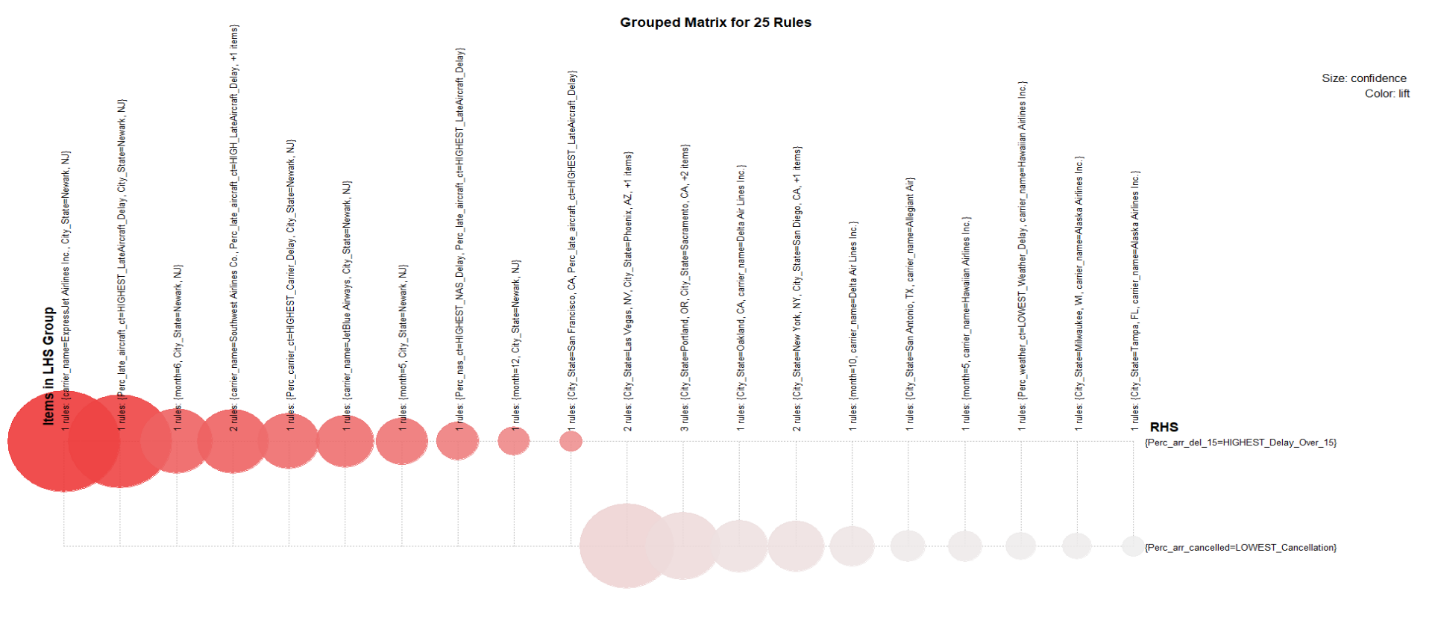


### Figure 8: ARM results for highest delays and cancellations

The top rule includes flights arriving in Newark, NJ, Express Jet Airlines, and having the highest delays. This rule also has the maximum confidence score meaning that 100% of Express Jet Airlines arriving in Newark have some of the highest percentages of flight delays.

Further, all of the top rules associated with the highest percentages of cancellations include Hawaiian Airlines, including arriving flights through this carrier in Phoenix AZ, Las Vegas, NV, and Sacramento, CA.

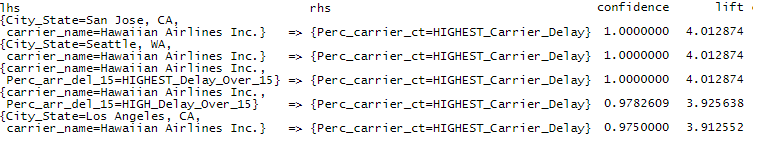
The reasons for these strong association rules will require additional information. Flight distance, airline carrier competency, airport operations, and weather can all play a role.



### Figure 9: Graphed ARM results for highest delays and cancellations

When graphing these results as a matrix where the size of the plot is confidence and color is lift, the top rules associated with Newark, NJ, and high delays indicate the highest confidence and lift values.

When limiting the right-hand side (RHS) to the highest percentages of carrier-related delays, Hawaiian Airlines makes an appearance in all 5 of the top rules when sorted by confidence and lift. The flights that arrived in San Jose, CA, and Seattle, WA, all produced maximum confidence scores.

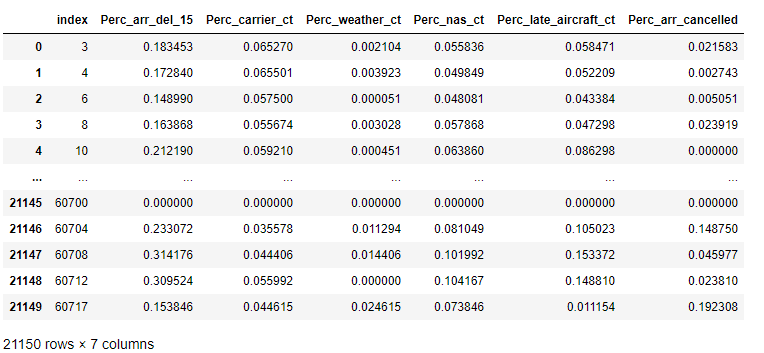


### Figure 10: ARM results for highest carrier-related delays

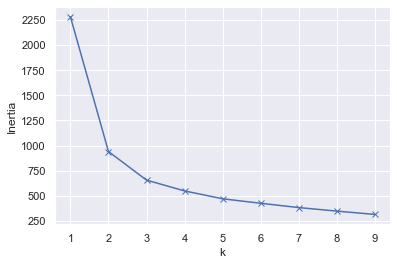
### **Clustering**

The data were clustered using K-means as part of the scikit-learn package on Python. With this method, data points are sorted into clusters based on their proximity to the nearest 'centroid', or cluster centers. The practice is useful in identifying groups and patterns in a dataset that are hidden or not explicitly apparent.

The number of centroids and their initial placement must first be defined before the process. Proper analysis requires the normalization of the percentage features used in the association rule mining process so that each feature receives equal weight.



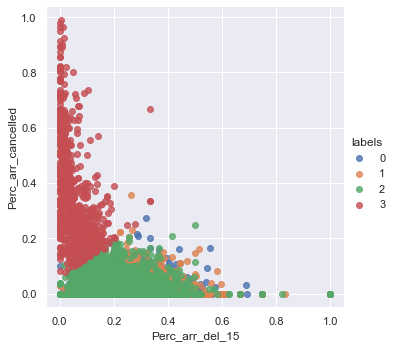
### Figure 11: Discretization data for clustering analysis (before normalization)



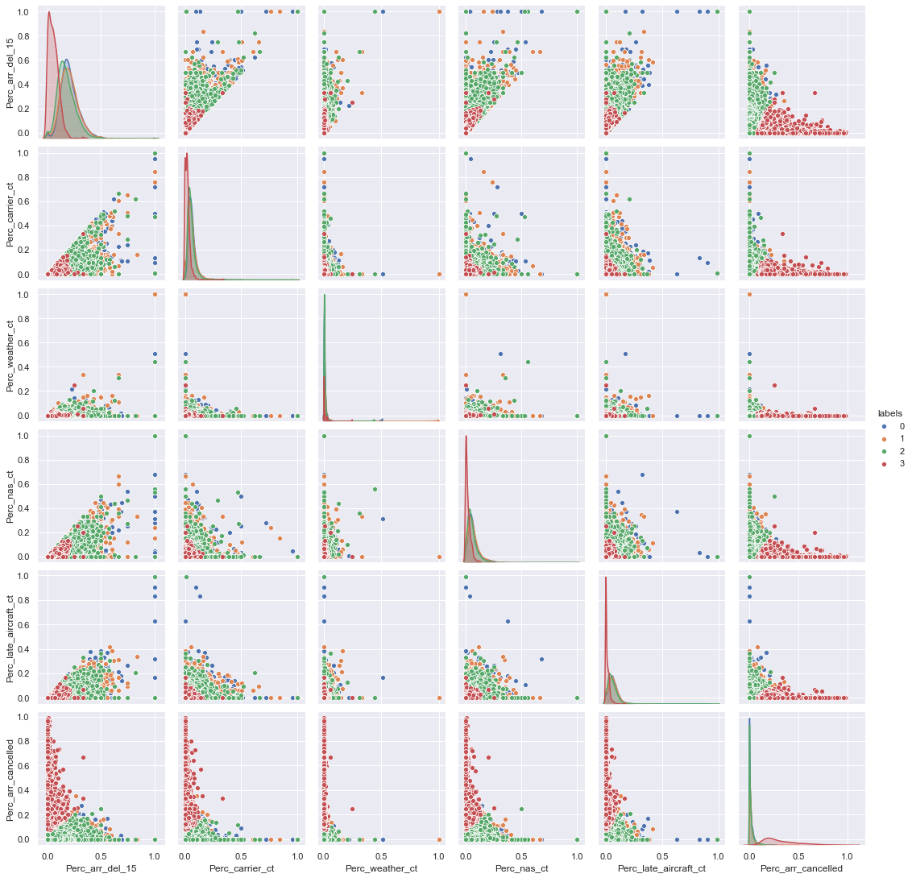
### Figure 12: Elbow method to determine optimal k clusters

The elbow method revealed the optimal number of clusters (defined by k) -- 4 groups resulted in the most visually separated clusters.

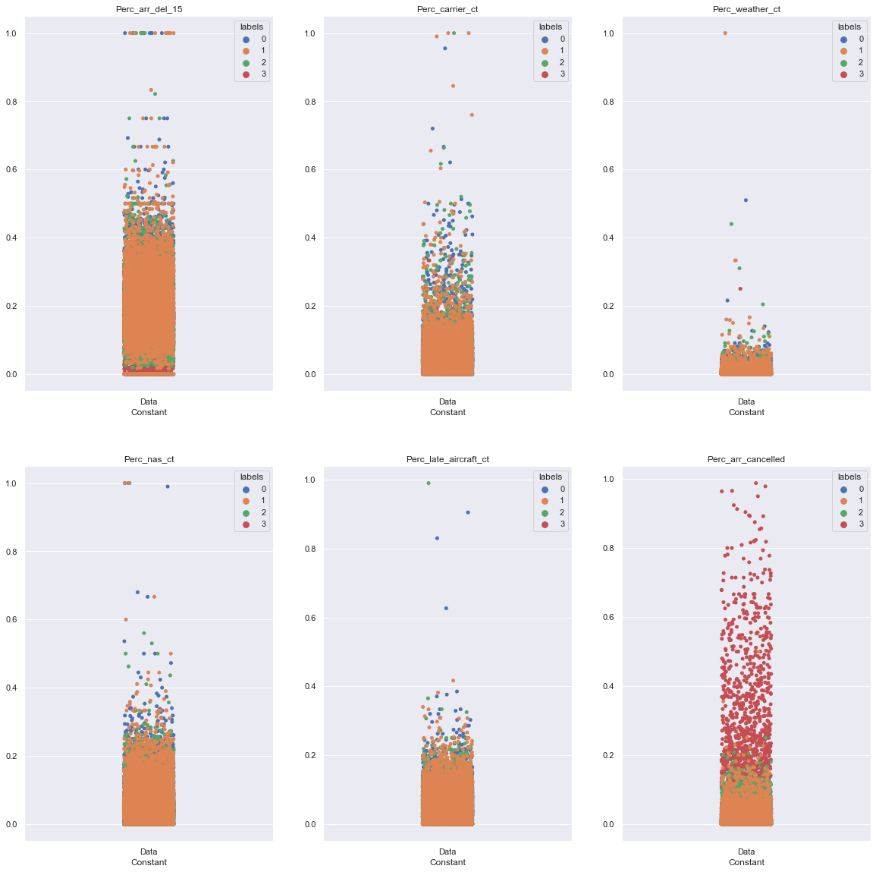
Because this is multi-dimensional clustering, there are several different ways to visualize the clustering results. In the following plots, the clustered data is visualized on various two-dimensional axes, focusing on individual features.



### Figure 13: Two-dimensional clustering results



### Figure 14: Matrix of clustering results of all six features in varying x and y axes



### Figure 15: Matrix of clustering results focusing on each feature

The clustered groups are most visually apparent when graphing on two dimensions, comparing the percentage of cancellations vs. total delays over 15 minutes. When looking at the mean feature values for each cluster, it is clear that the clustering algorithm focused on separating by overall delay and cancellation percentages.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Cluster** | **del\_15** | **carrier\_ct** | **weather\_ct** | **nas\_ct** | **late\_aircraft\_ct** | **arr\_cancelled** |
| **0** | 0.205826 | 0.063389 | 0.069411 | 0.069411 | 0.065621 | 0.015077 |
| **1** | 0.207904 | 0.065134 | 0.069647 | 0.069647 | 0.066426 | 0.014901 |
| **2** | 0.17646 | 0.056682 | 0.057737 | 0.057737 | 0.056197 | 0.017512 |
| **3** | 0.055762 | 0.022382 | 0.019563 | 0.019563 | 0.012344 | 0.326246 |

### Table 2: Mean feature values for each cluster

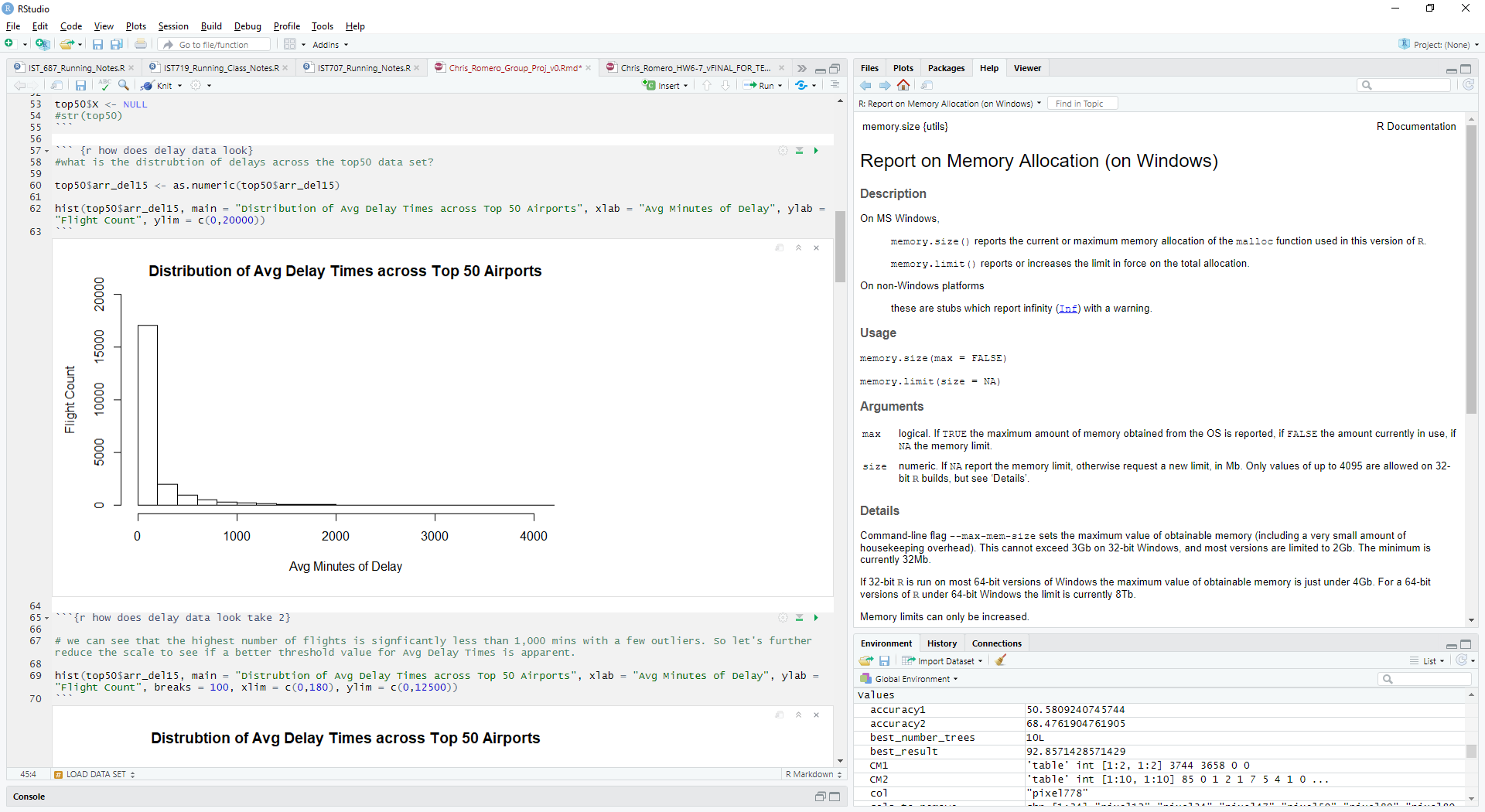
## **Support Vector Machines**

Support Vector Machine (SVM) was the next model applied to predict the on-time arrival of a given flight.

Each flight from the data was first classified as "on time" or "delayed" to predict the probability of on-time arrival.

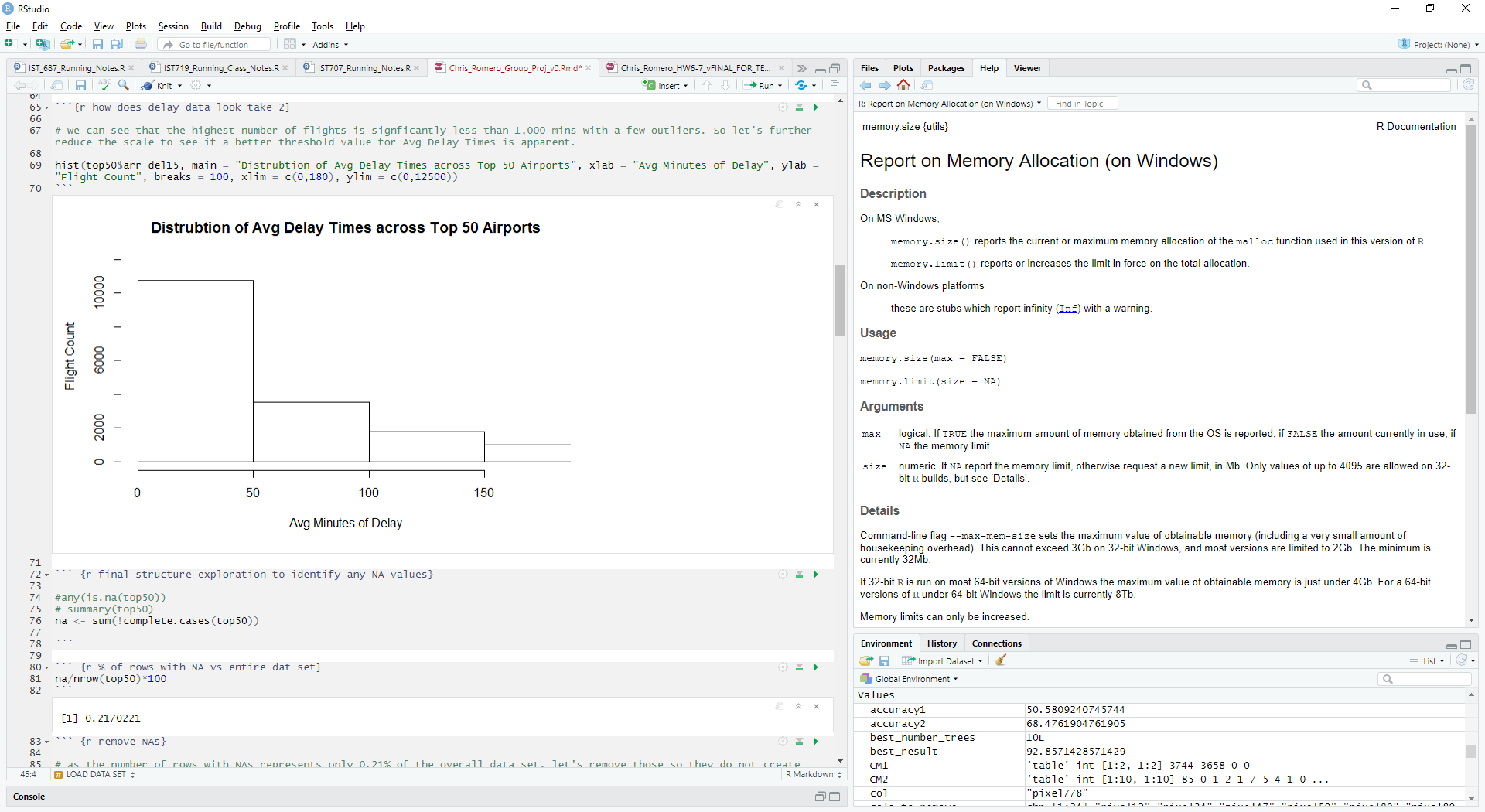
On-time classification requires the definition of an on-time threshold (i.e., number of minutes beyond which a flight is considered a delay). Although any flight arriving after its scheduled time of arrival might classify 'delayed,' such a characterization is not a practical solution or threshold.

For this project, the definition of an on-time threshold considered both a statistical and pragmatic approach: an evaluation of the distribution of delayed flights combined with an application of personal travel experience to determine what that threshold should be.



### Figure 16: Distribution of Flights vs. Average Minutes of Delay

Initially, the entire dataset was plotted as a histogram to understand the various lengths of delays observed. This analysis provided a quick snapshot revealing that aside from a few outliers, most of the flight delays were less than 200 minutes (as shown in the first and largest bar of Figure 16).



### Figure 17: Distribution of Flights vs. Average Minutes of Delay (reduced scale)

Replotting the data to look at flights with less than 150 minutes (2.5 hours) of delay resulted in the graph shown above.

In Figure 17, note the clear break at 50 minutes, indicating that more than 10,000 (of the approximately 21,196 flights) arrived within 50 minutes of their scheduled arrival time.

From a practical point of view, this break of 50 minutes as an on-time threshold can be considered realistic: passengers with connecting flights still have a reasonable amount of time to reach their connection, and those passengers arriving at their final destination also may allow up to about 50 minutes before needing to change or cancel subsequent plans (e.g., meetings, dinner appointments).

After evaluating the data to define an on-time threshold, further cleansing removed NAs (i.e., missing values) inherent to the raw data.

This was necessary since the presence of NA's would be passed in the model and lead to a lesser number of modeling results versus actual results. Ultimately this would inhibit model comparison and prevent any understanding of the model accuracy. The number of NA's present was determined to be less than 1% and so deemed negligible if removed from the data set. 

### Figure 18: Removal of NAs from Data Set

Finally, before setting up and running the model, attributes were evaluated to understand whether they added value to the model or may be considered a hindrance to the model (or perhaps covered by another feature). Evaluation removed the following:

|  |  |
| --- | --- |
| **Attribute** | **Rationale for Removal** |
| Airport (airport) | Features of the airport that may contribute to delays, likely covered by other attributes such as security\_delay |
| Airport Name (airport\_name) | The name of the airport should not impact delays |
| City (City) | The main city attribute leading to delays is its geographic location is likely covered by other features such as the month, weather\_ct, weather\_delay |
| State (City\_State) | The state should not impact delays |

After cleaning the data of missing values, defining an on-time threshold as a factor, and removing unnecessary attributes, training and testing data sets were then created. Applying a seed value ensured result repeatability.

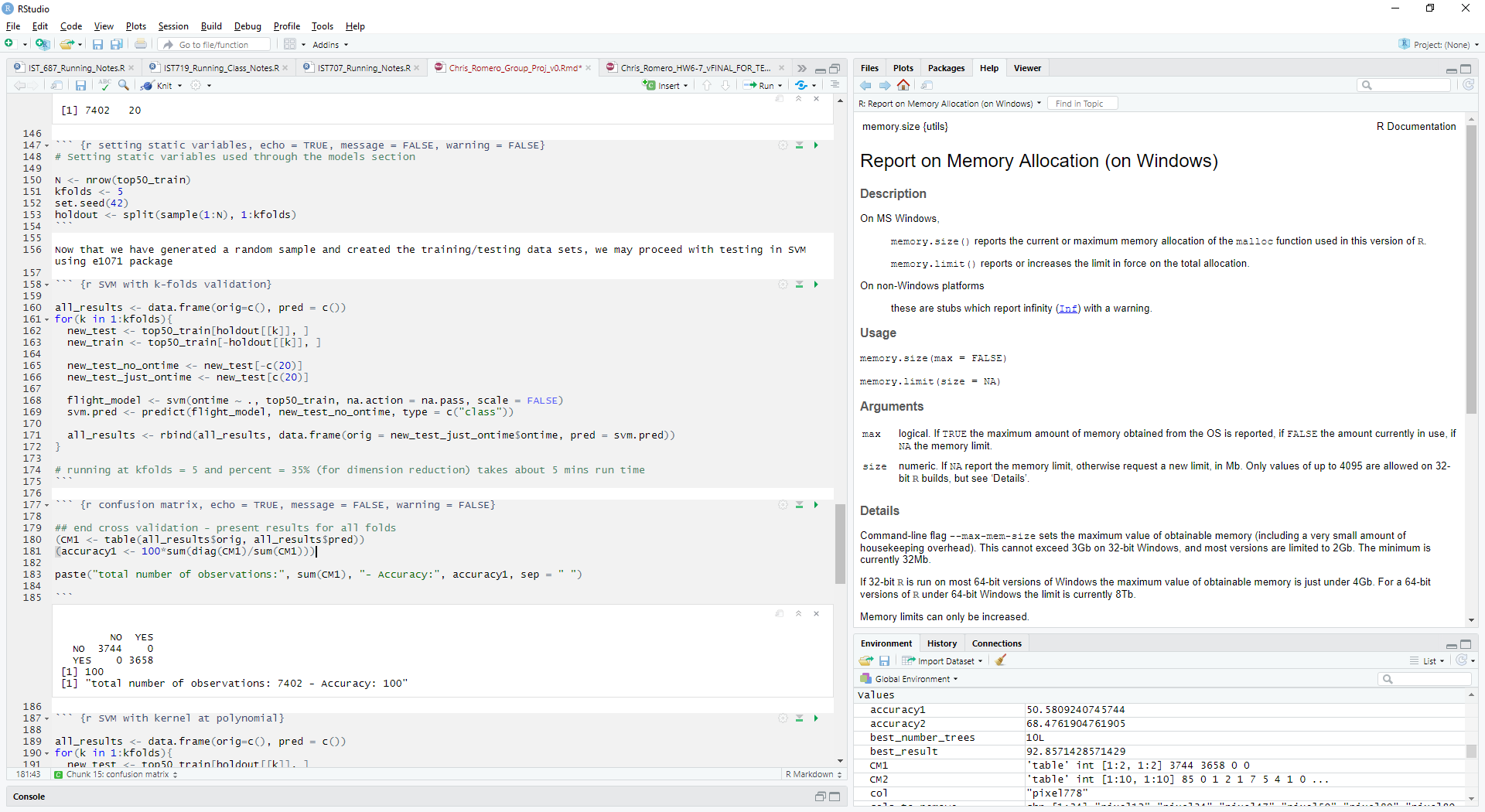
During the initial model run, the data set appeared too large for SVM to process in a reasonable amount of time. Reducing the data set using just 5% of the original size confirmed that the SVM model would process within a fair amount of time.

The percent of the data set was then increased in 5 to 10% increments to understand the optimal data set size. Ultimately, a data set size of 35% resulted in computing time of approximately 5 to 7 minutes.

Similarly, k-fold was adjusted in a step-wise fashion, starting with two folds and increasing up to ten. After multiple attempts, setting k-fold to 5 appeared to be optimal in allowing a reasonable amount of computing time but not such a low k-fold value that would introduce uncertainty to cross-validation.

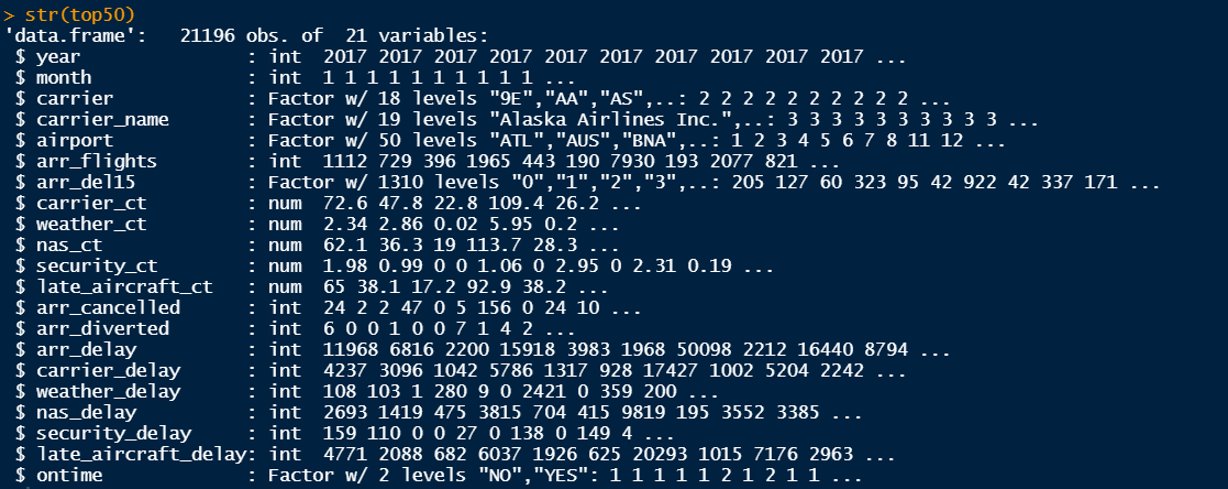
Finally, after tuning the data set size and number of k-folds, the kernel was adjusted between model runs from radial to polynomial to sigmoid.

Setting the kernel to polynomial resulted in errors indicating "reaching the max number of iterations." Choosing a sigmoid kernel reduced the accuracy to 50.58%. When leaving the kernel as radial (the default kernel in SVM), the resulting accuracy was 100%.



## **Decision Tree**

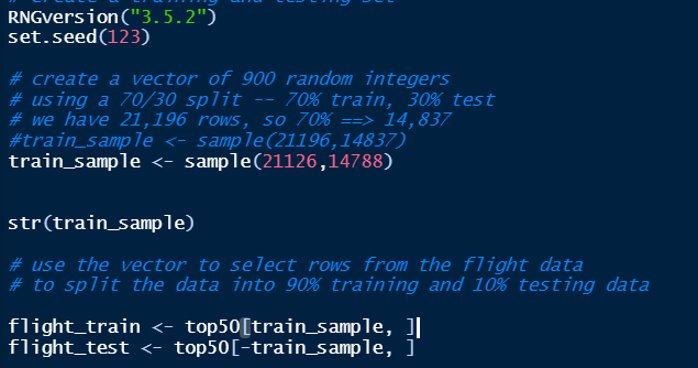
Importing the data revealed the following structure, indicating 21,196 observations of 21 variables. As in prior models, the focus in on delays > 15 minutes, and the creation of an on-time factor.



### Figure 19 Decision Tree Data Structure

Following data importation, an on-time factor was created and used as in SVM.



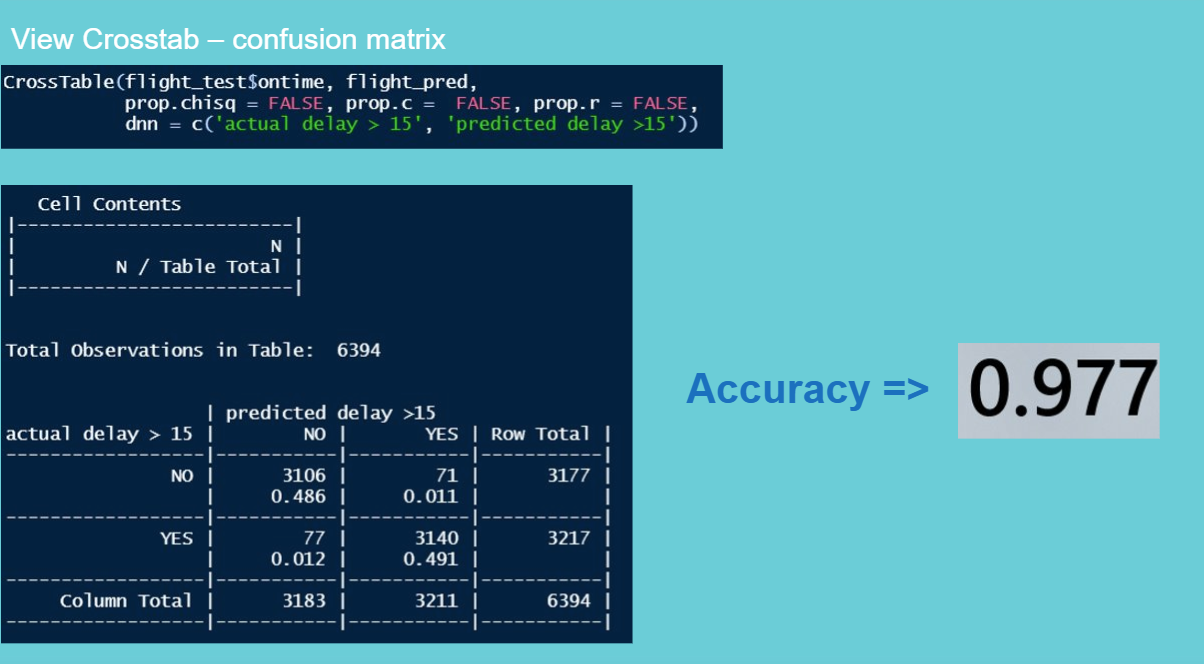


### 

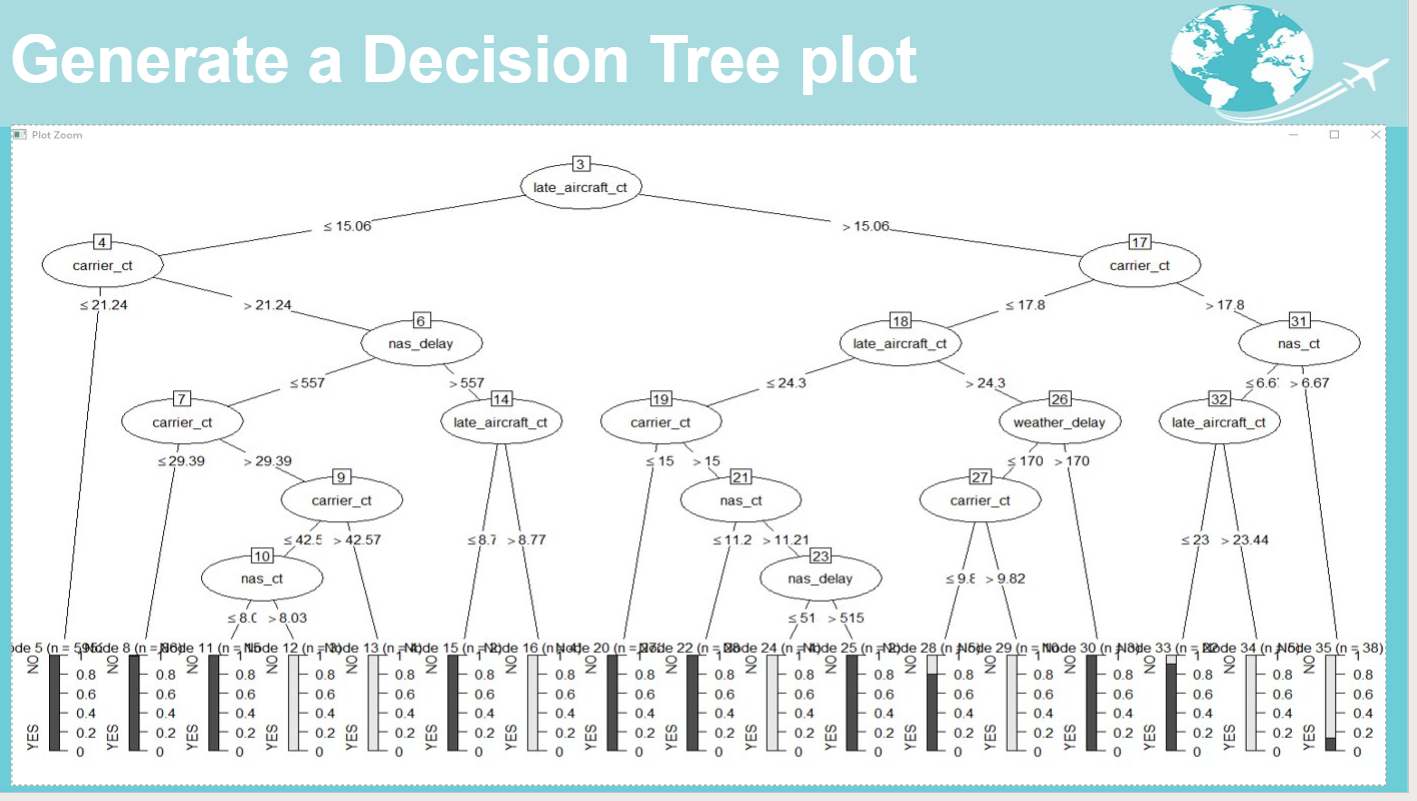
Next the C50 Model was loaded, referenced, and run.



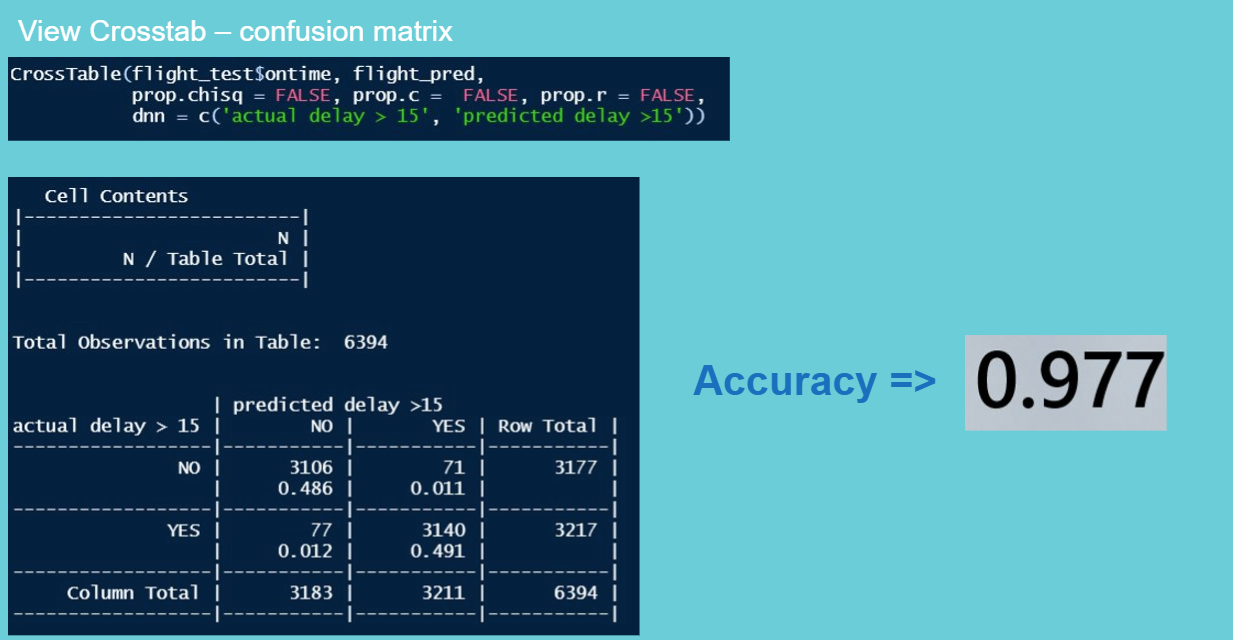
Applying the cross-table function reveals the following decision tree results.



Plotting the results reveals the following decision tree.



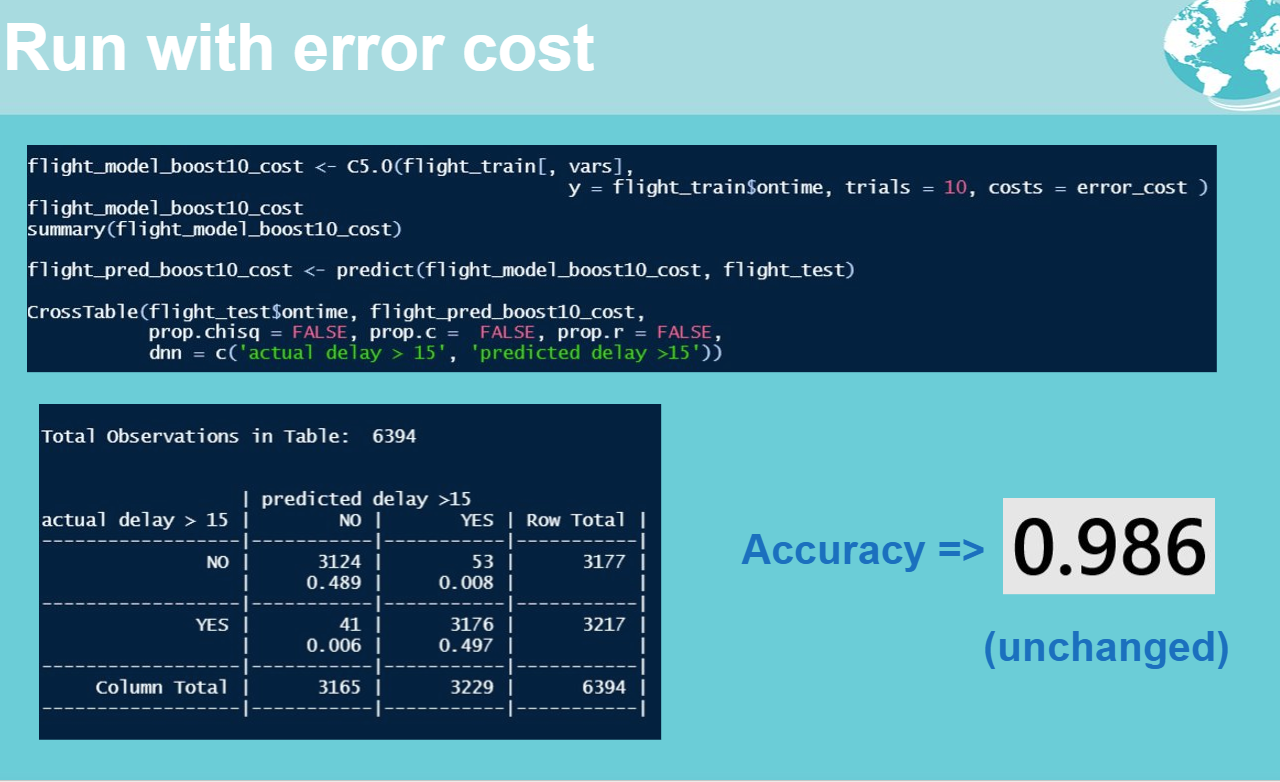
### Figure 20 Decision Tree Data Plot



### Figure 21 Decision Tree Accuracy = 97.7%

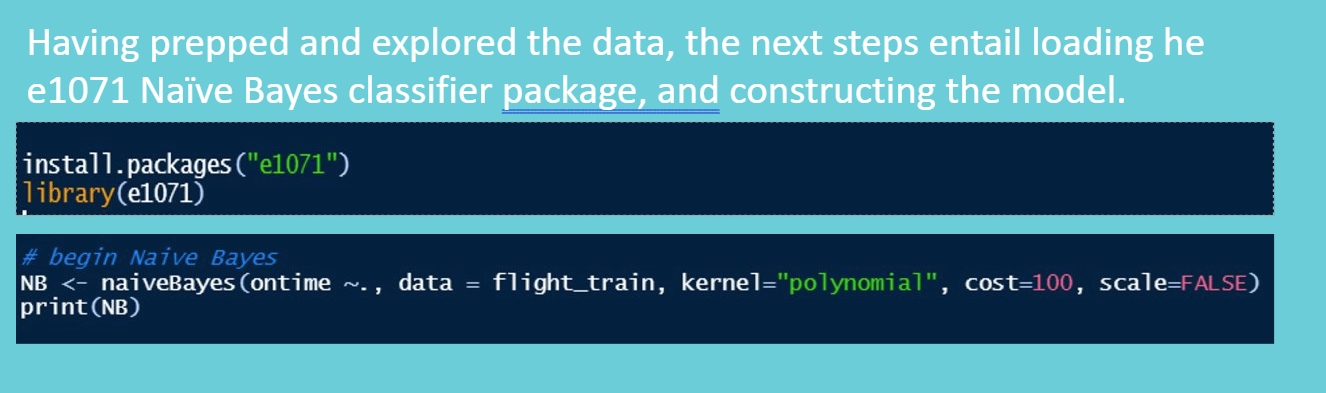
## Running the prediction revealed 97.7% accuracy rate.

Next, error cost code was generated and applied to the model. This has a slight improvement in model accuracy, although a larger improvement was expected.

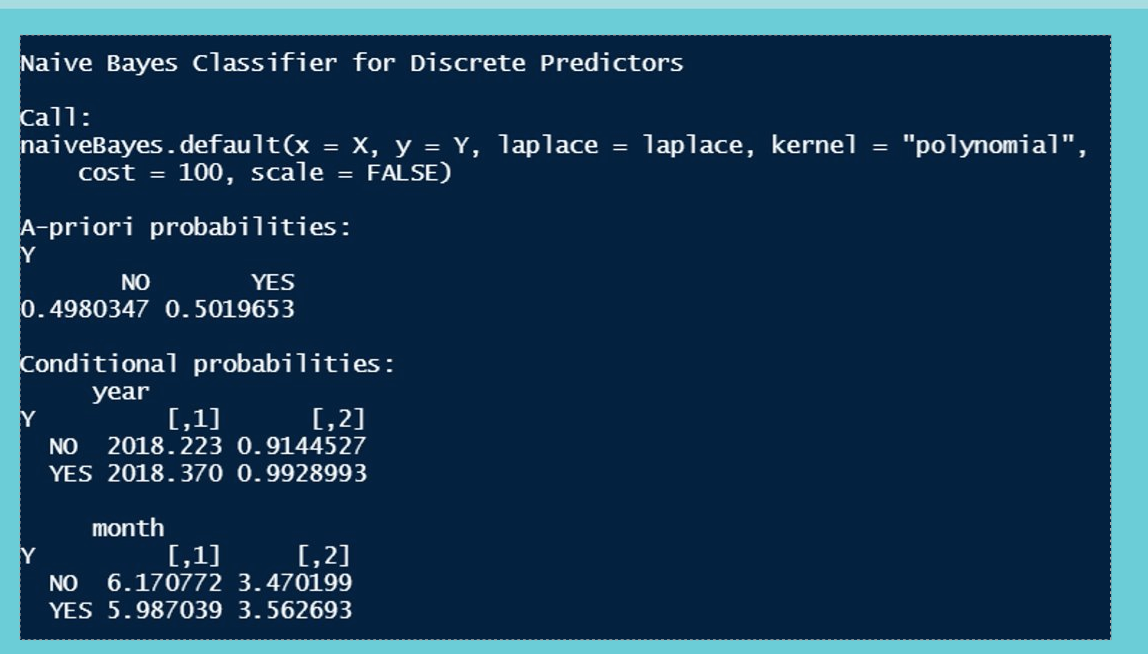


### Figure 22 Decision Tree Accuracy with error cost = 98.6%

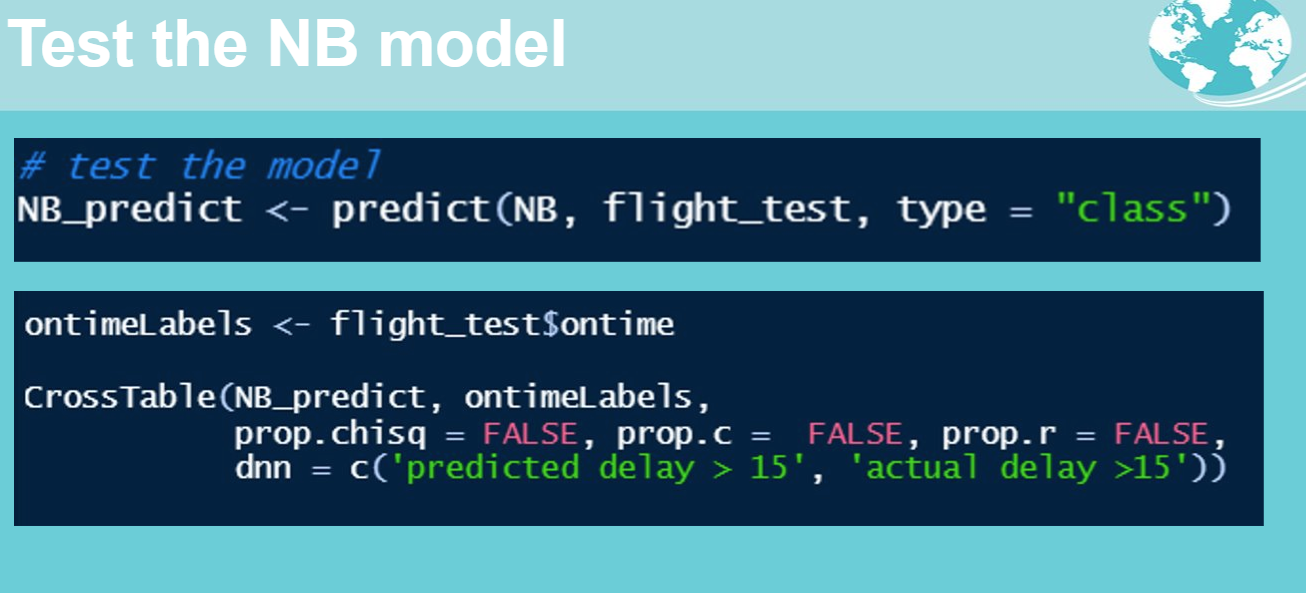
## **Naïve Bayes – Jeffrey Ronay**



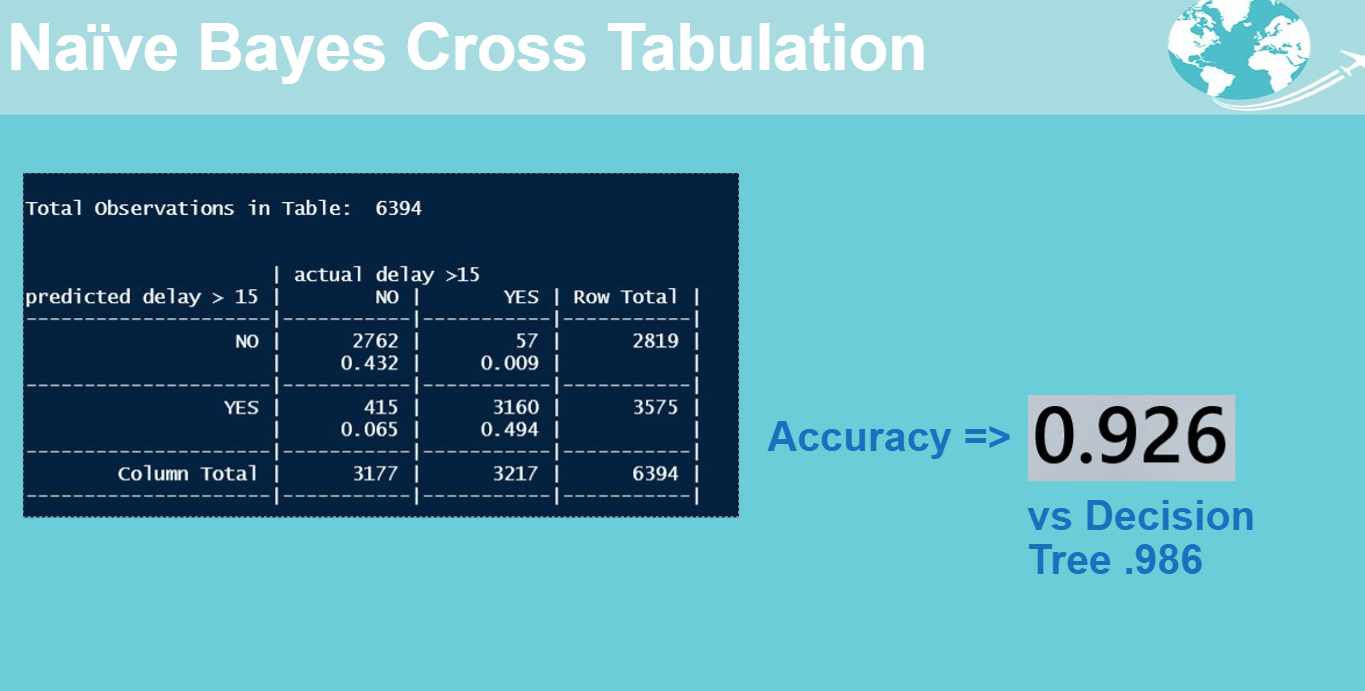
Next, the classifier is called.



## Testing the Naïve Bayes model



### Inspecting the Naïve Bayes Cross Tabulation



Note the accuracy is .926 which is somewhat lower than the Decision Tree accuracy of .986. Both models performed relatively quickly, processing the data set in less than 2 seconds. Finally, the model is changed to include the addionon of the the Laplace factor with a setting = 1. This erasulted in negligible lower false positive and false negatives results.

## **Conclusion**

The final project provided the ablitity to examine the same data set using a variety of data modelling algorithms, models, and techniques. This is valuable insight as real world problems are often best analyzed using a variety of methods to arrive at sound and meaningful conclusions. Put another way, it is good to have different perspectives on the same data set.

In this case, using the Decision Tree and SVM models proved to be the most accurate predictors, with accuracty rates approaching 95-100%.

These models could be incorporated into a variety of front-ends or interfaces, such as a web page, where a user might pick a an airport destination and month, and receive an “on-time” prediction.