ST2195 Programming for Data Science

Ochitha Bandara
200630447

Table of Contents

Introduction

To produce this report, I utilized the years 2006 and 2007, which had around 7,100,000 and 7,400,000 records, respectively, and the final merge of the two data sets gave a grand total of over 14,500,000 records. The questions ask about which flights had the least delays as its premise, therefore I made use of Arrival delays and Departure delays for answering most of the questions, moreover, removed all cancelled and delayed flights as they would not be counted and be useless as variables. Question 1 asks about which was the best time of day, week and year for the flights with the least number of delays, question 2 asks if older planes were likely to suffer delays, question 3 asks how the number of passengers to and from locations have fluctuated over time, question 4 asks to find if cascading failures in a specific airport would cause delays in another and question 5 asks to create a model for predicting delays. Some columns have name descriptions to better understand them, for example, CRSDepTime is the scheduled departure time in hours and minutes form and ArrTime is the actual arrival time in hours and minutes for. For further explanations on the questions, a brief answer and conclusion was made under each one.

Question 1

For question 1, I first tried to find the total number of non-delayed flights. All the non-available values were checked, which led to all cancelled and diverted flights not being considered and equated to 0, as mentioned in the introduction. To find the least number of delays in each situation a different variable was made and for the best time of day, the hours were separated from the minutes as it would be easier to talk about the results in hours. In the end it was found out that 2007 had both the highest percentage for no delays and delays when both years were considered, as seen in Table 1 below.

Year	No Delays	Delays
2007	75.74%	75.02%
2008	24.26%	24.98%

Table 1: Yearly delays and no delays

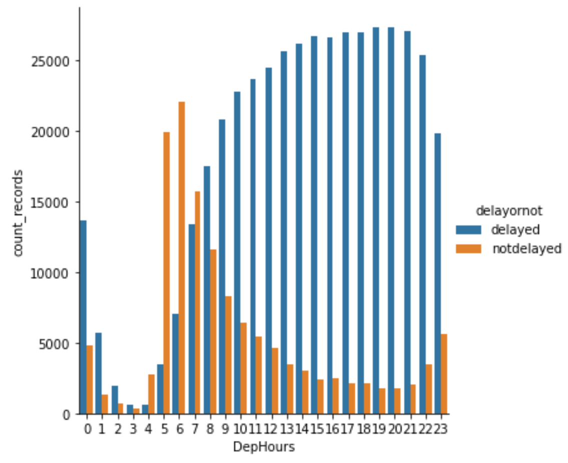
For the best time of day, the hours were separated from the minutes as it would be easier to talk about the results in hours. In order to get a better perspective, the Departure Hours or in this case, DepHours was taken into consideration and shown below,

Dep Hours	Percentages
6	16.56%
5	14.77%
7	11.75%
8	8.65%
9	6.19%
10	4.77%

Table 2: Hourly no delays

Conclusion

To further elaborate on this, Figures 1 and 2 can be used, and it was found out with reinforcement, that the best time of day was 5am to 7am or in the morning hours, for the least number of delays.



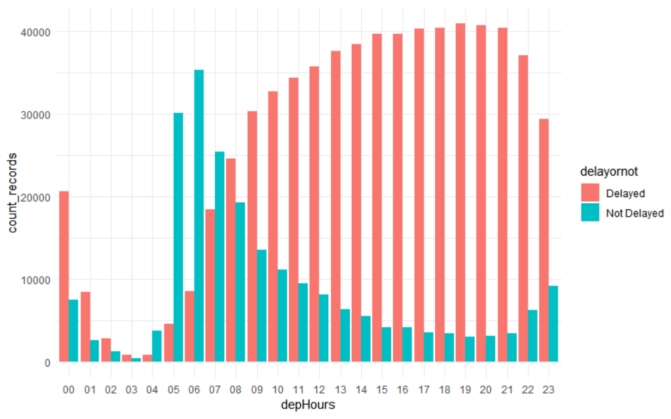
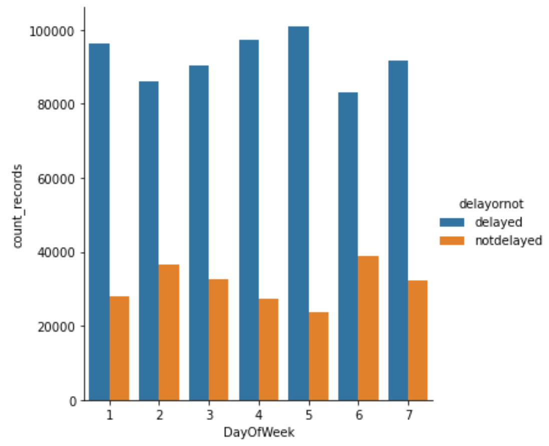


Figure 1: Python Dep Hours

Figure 2: R Dep Hours

The best days of the week with the least number of delays was found out to be on Tuesday and Saturday as shown below, but it can also be assumed that the weekends had a lower proportionate number of delays when compared to the weekdays. Both plots using R and Python have produced similar results.



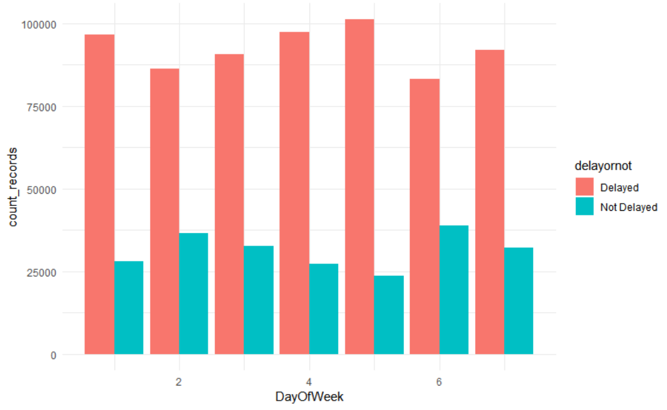


Figure 3: Python Day of Week

Figure 4: R Day of Week

Further, the time of the year with the least number of delays was found out to be September 2007 with October and November also being ideal, so 2007 had the least likely chances of having delays when considering the time of year, with both R and Python producing similar plots.

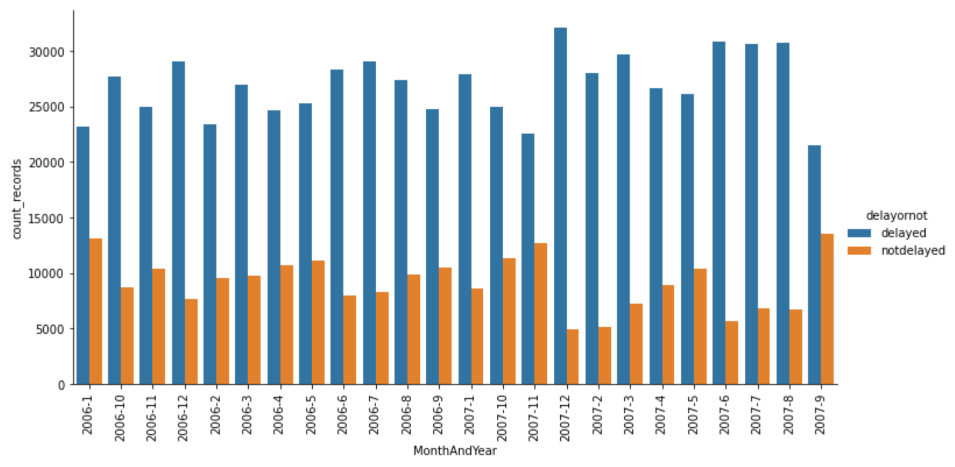


Figure 5: Python Time of Year

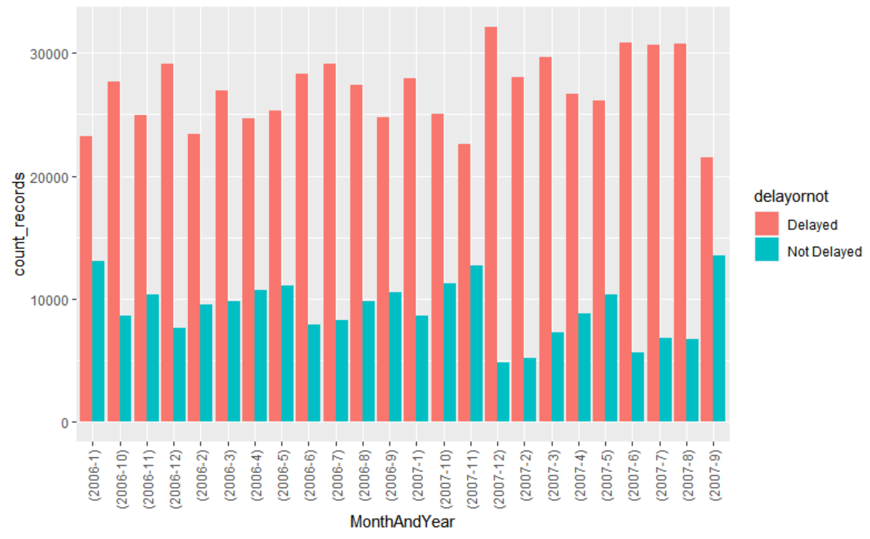


Figure 6: R Time of Year

Question 2

In order to find out the total number of delays a variable was made combining Arrival Delays and Departure Delays. Before plotting the graph, some planes were registered as having a year of “0000”, which I considered to be part of the “none” variable for further reference. By comparing the variable that consists of both delays, to the year the plane was made, planes from the year 1985 onwards seemed to have suffered an increasing number of delays so there was a distinct pattern showing that the newer planes seemed to suffer more delays in comparison to the older planes. And it can be seen by both R and python, the exact same results were found.

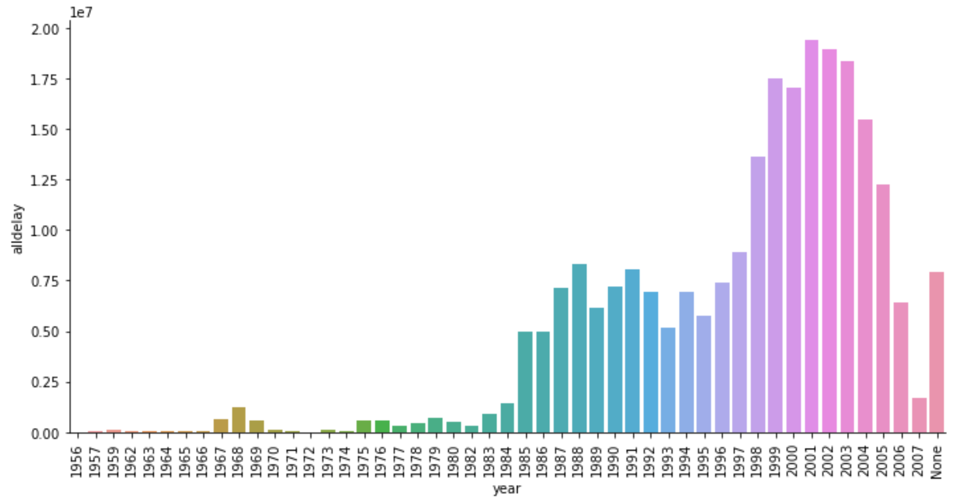


Figure 7: Python Plane Delays against their manufacture year

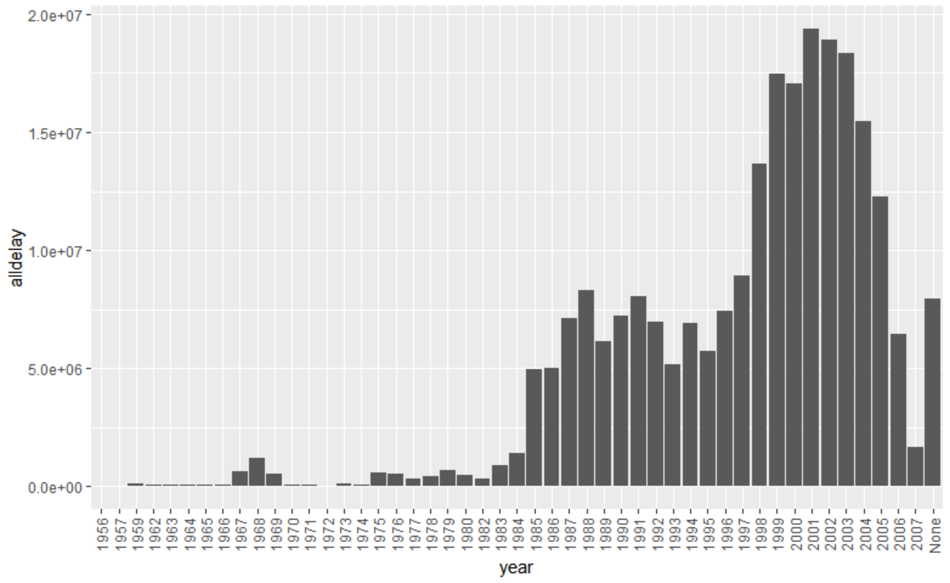


Figure 8: R Plane Delays against their manufacture year

Question 3

In order to check how the number of people flying between different locations changed over time, the top 10 locations which had the greatest number of flights to-and-fro were taken into consideration. As the below figure demonstrates, there isn't a significant pattern to be seen so it is hard to give a proper insight. Since there was no way to check the total number of passengers per flight, I assumed that each flight has around 100 people, as flights should have around that much in order to take off. We could say that the flights from Kahului to Honolulu have shown a 5-10% increase or around a 1000 more passengers from 2006 to 2007, and the same can be said for New York to Arlington as well which shows off around a 2% increase.

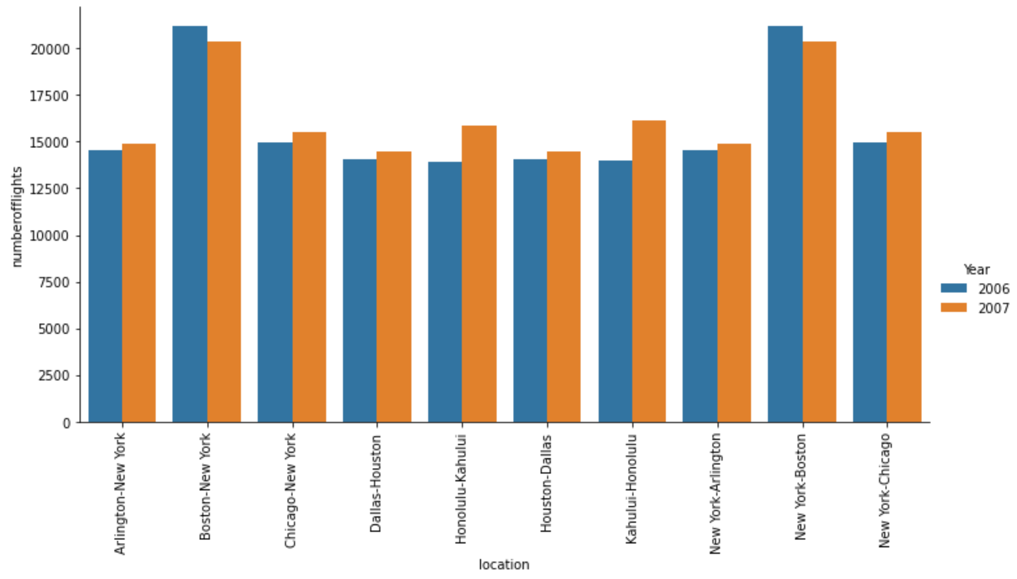


Figure 9: Python Top 10 flights

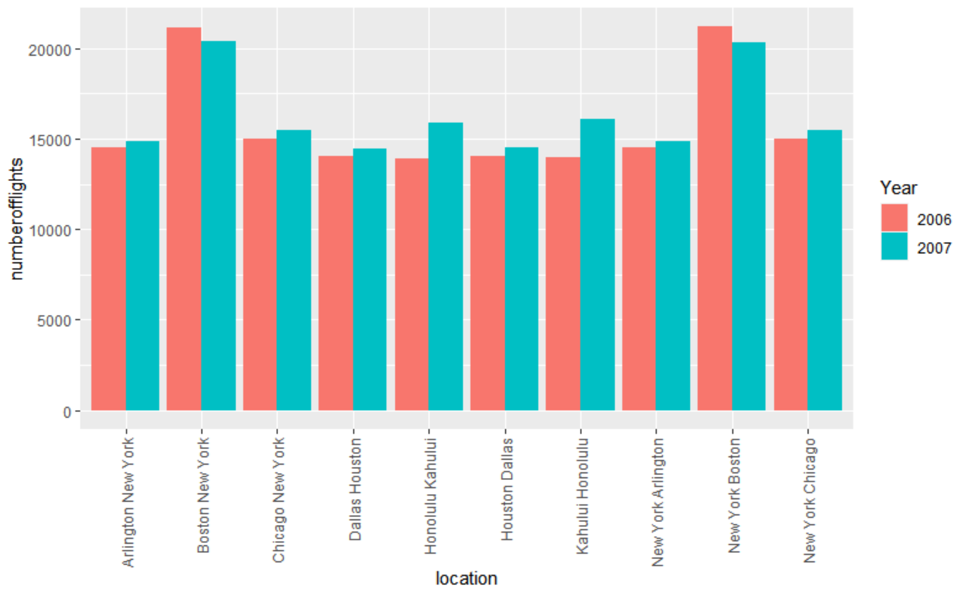


Figure 10: R Top 10 flights

Question 4

By extracting and creating a new variable for previous delays, I charted it against the variable that considers all delays using a scatterplot. There were certain outliers, such as there being almost 1600 total minutes in current delays while barely 100 minutes in previous delays and the same goes towards the scatter point which had almost 0 minutes in previous delays but had around -270 minutes, meaning the planes had taken off early regardless of the slight previous delay. But when considering most of the scatter points, there is no visible relationship between previous delays and current delays after looking at figures 11 and 12, which confirms that R and Python produce the same result.

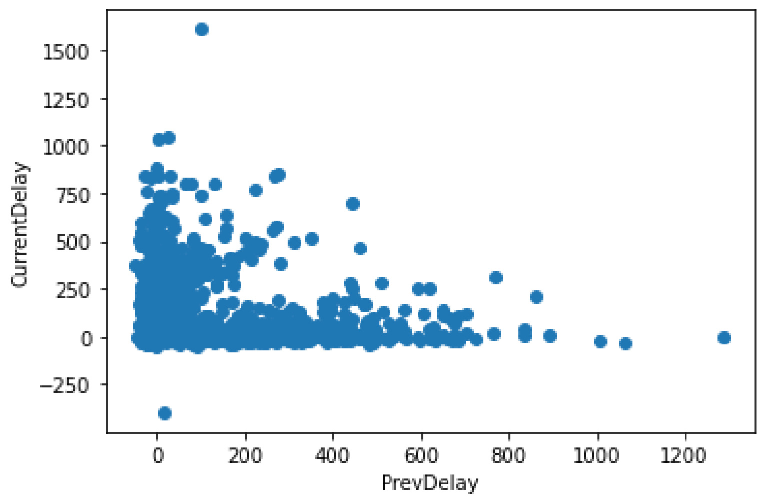


Figure 11: Python Scatter plot

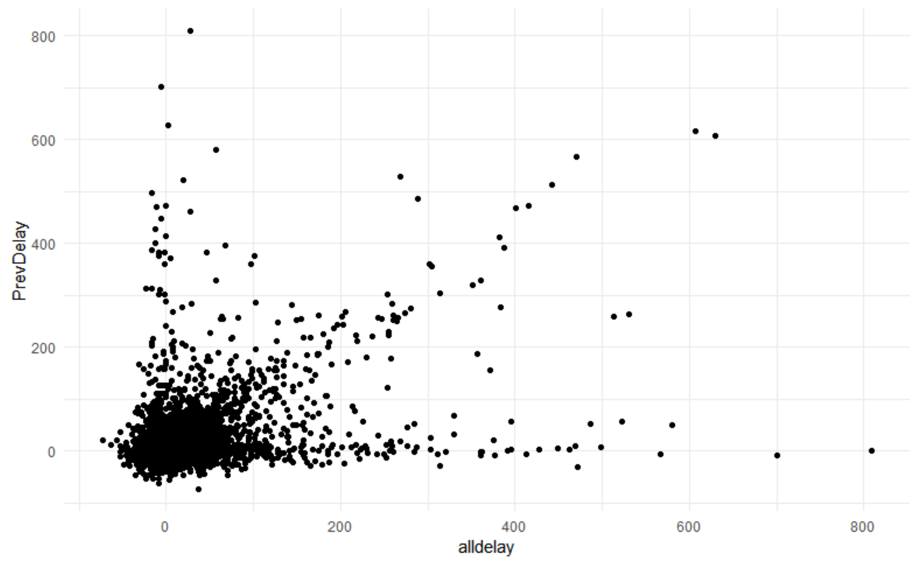


Figure 12: R scatter plot

Current Delay		0	1
Previous Delay	0	7930487	6663095
	1	796	759

Table 3: Comparing Current Delays to Previous Delays

To further elaborate on this, Table 3 considers 0 as no delay and 1 as a delay, and it shows that there is a high number of flights which had no previous delay nor a current delay and a high number of flights which had a current delay but no previous delays, so this concludes that previous delays did not have much of an impact on current delays of flights.

Question 5

In order to start off the model, I identified my target variable as isdelayed for Python, which consists of arrival and departure delays and then allocated the columns into their respective numerical and categorical features. I removed the other delays mentioned such as CarrierDelay and WeatherDelay as they would not assist in predicting future data. To find the highest correlation between isdelayed and all the other features, I utilized a heatmap as well as a bar chart shown as Figure 13, which helped identify the features with the highest correlation, with TaxiOut having more than 25% correlation and Departure Time having almost 20%, moreover, we dropped the features which had no correlation.

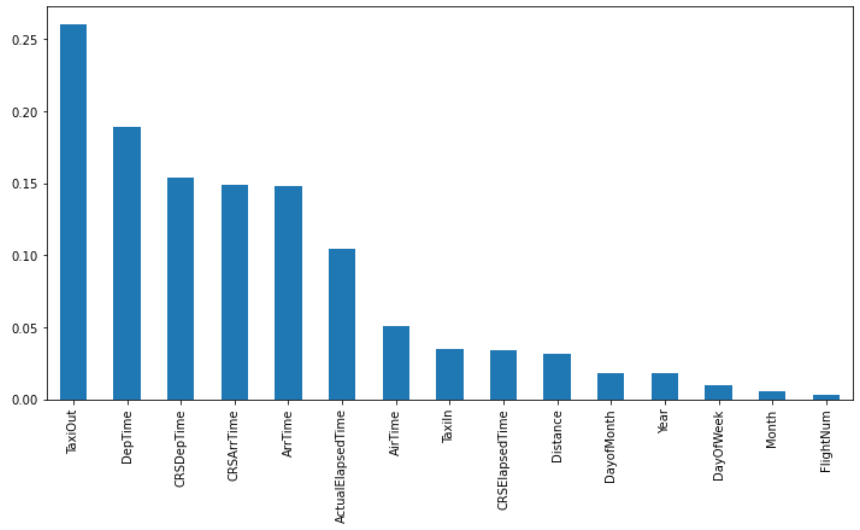


Figure 13: Correlation with isdelayed

For encoding, I used LabelEncoder despite having other methods such as integer encoding and onehotencoding. I did not use onehotencoder because some categories had too many unique values. I split the data set into a 4:1 ratio (80 to 20) between training and testing models, and applied Standardscaler as my scaling method. The confusion matrix created using Logistic Regression had a high value of 1,312,628 for True positive and 932,814 for false negative, as seen in Table 3, whilst Table 4 shows that the confusion matrix using DecisionTreeClassifier had a True positive value of 1412482 and a false negative figure of 1217734. To test the model for accuracy at first, I used Logistic Regression, getting an accuracy of 0.79 or 79% and a f-1 score for class 0, meaning no delays, of 0.81 and for class 1, meaning delays, of 0.75, as seen in Figure 1, therefore, the model is very good for prediction, however I found out that using DecisionTreeClassifier, I got a higher accuracy of 0.92, as seen in Figure 2, which is what I used to prove my final model accuracy with. I later plotted an ROC curve, which shows how the classification model has performed at all its classification thresholds, illustrated in Table 16, which gave an AUC value of around 0.91, and the threshold can be stated as around 0.1, which is very good as it would count every value upwards of only 0.1 as a score of 1 and anything below that as a score of 0, so predictions would be 91% accurate, so it is a good model for predicting.

```
              precision    recall  f1-score   support

           0       0.77      0.86      0.81   1523580
           1       0.82      0.70      0.75   1332238

    accuracy                           0.79   2855818
   macro avg       0.79      0.78      0.78   2855818
weighted avg       0.79      0.79      0.78   2855818
```

1314333	209247
401534	930704

Table 4: Confusion matrix for Logistic Regression

Figure 14: Logisitic Regression Classification Report

```
              precision    recall  f1-score   support

           0       0.92      0.93      0.93   1523580
           1       0.92      0.91      0.91   1332238

    accuracy                           0.92   2855818
   macro avg       0.92      0.92      0.92   2855818
weighted avg       0.92      0.92      0.92   2855818
```

1411693	111887
116936	1215302

Table 5: Confusion matrix for DecisionTreeClassifier

Figure 15: Decision Tree Classification Report

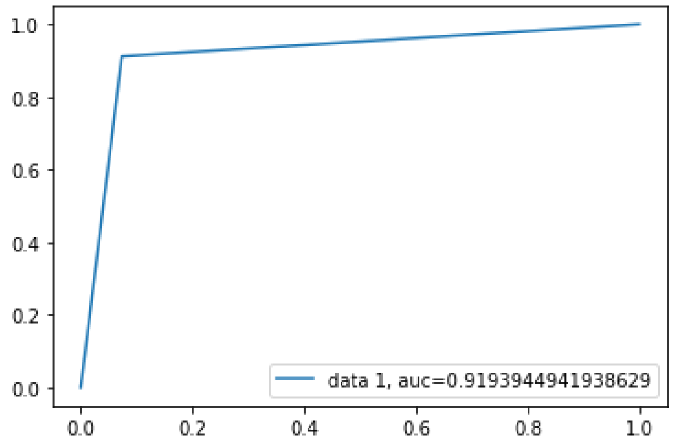


Figure 16: ROC curve in Python

When doing question 5 for R, I had to take a sample of around 200,000 because of the time that it would take to process and label each row, therefore, the accuracy and AUC score might not be as accurate as what I found using Python, however it is acceptable. As shown in Figure 17, using a correlation graph in R, TaxiOut and DepTime still had the highest correlations against the target variable I identified, in this case, isdelayed, when compared to Python. After splitting the data set into a 3:1 ratio this time for Training and Testing data sets and by using RandomForest Classifier for the model, a false negative value of 29115 was found with a true positive value of 24976 for the confusion matrix and got a high accuracy of 0.92, which is similar to the accuracy level when I used the Decision Tree Classifier in Python.

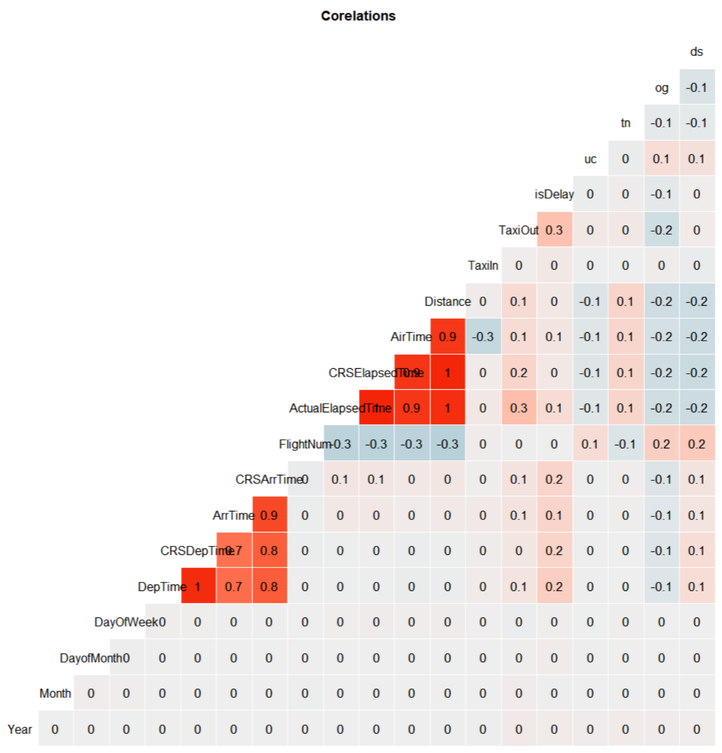
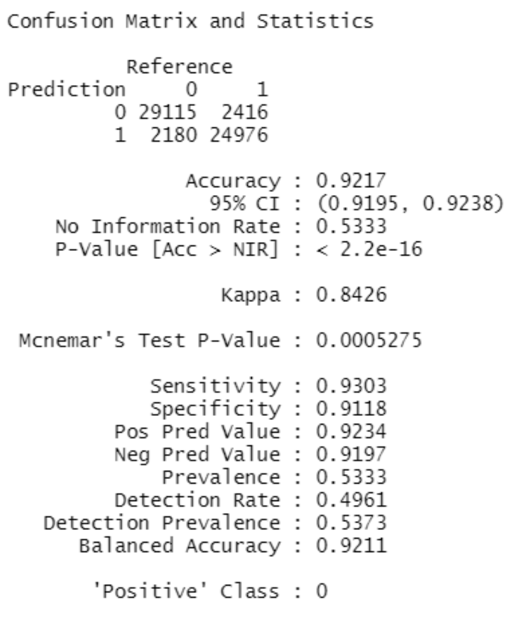


Figure 17: R correlation against isdelayed



```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 29115  2416
         1  2180 24976

               Accuracy : 0.9217
                 95% CI : (0.9195, 0.9238)
    No Information Rate : 0.5333
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8426

 Mcnemar's Test P-Value : 0.0005275

            Sensitivity : 0.9303
            Specificity : 0.9118
         Pos Pred Value : 0.9234
         Neg Pred Value : 0.9197
             Prevalence : 0.5333
         Detection Rate : 0.4961
   Detection Prevalence : 0.5373
      Balanced Accuracy : 0.9211

       'Positive' Class : 0
```

Figure 18:Classification Report RandomForest Classifier for R

The ROC curve plotted in R with 200,000 records, shown in Figure 19, gave an AUC score of 94.1% or 0.941, therefore the model would have correct predictions more than 94% of the time, hence making it very accurate.

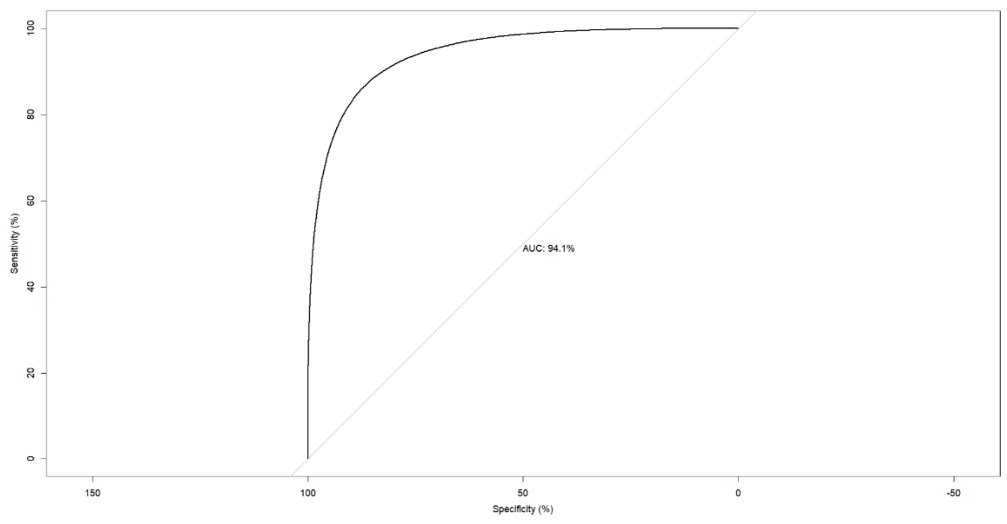


Figure 19: R ROC curve