ST2195 Programming for Data Science Coursework

DATASET ANALYSIS

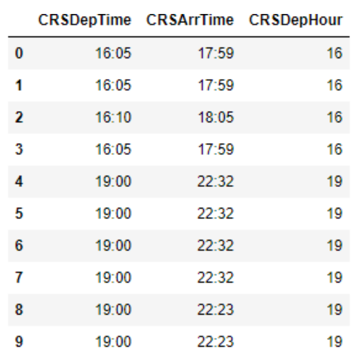
Name: Dillon Yew
STUDENT ID: 10196936 | DATE: 1/3/2022

Table of Contents

Q1. When is the best time of day, day of the week, and time of year to fly to minimise delays?

1.1. Python Analysis for Q1

Firstly, we subset our main dataframe, df, to only display time-related variables and delays. Next, we create a function to convert integers from ‘CRSDepTime’ and ‘CRSArrTime’ into a time-based format. An integer 1605 would be converted to 16:05 to better represent time. Then, we create a new column called ‘CRSDepHour’ whereby we slice the first 2 characters of a variable from ‘CRSDepTime’ to obtain only the hour of departure. We would be using the hour of departure to represent the scheduled departure time for our analysis later on. The figure below shows the output from our code:



	CRSDepTime	CRSArrTime	CRSDepHour
0	16:05	17:59	16
1	16:05	17:59	16
2	16:10	18:05	16
3	16:05	17:59	16
4	19:00	22:32	19
5	19:00	22:32	19
6	19:00	22:32	19
7	19:00	22:32	19
8	19:00	22:23	19
9	19:00	22:23	19

While there is a column ‘DepTime’ which represents the actual departure time of the flight, we would only be using data from the Scheduled Departure Time (‘CRSDepTime’) for our analysis in Q1. Moving on, we plot 3 different bar plots to display the best time of the day, day of the week and time of the year to fly in order to minimise delays.

Figure 1.1a Distribution of Departure & Arrival Delays vs Scheduled Departure Time

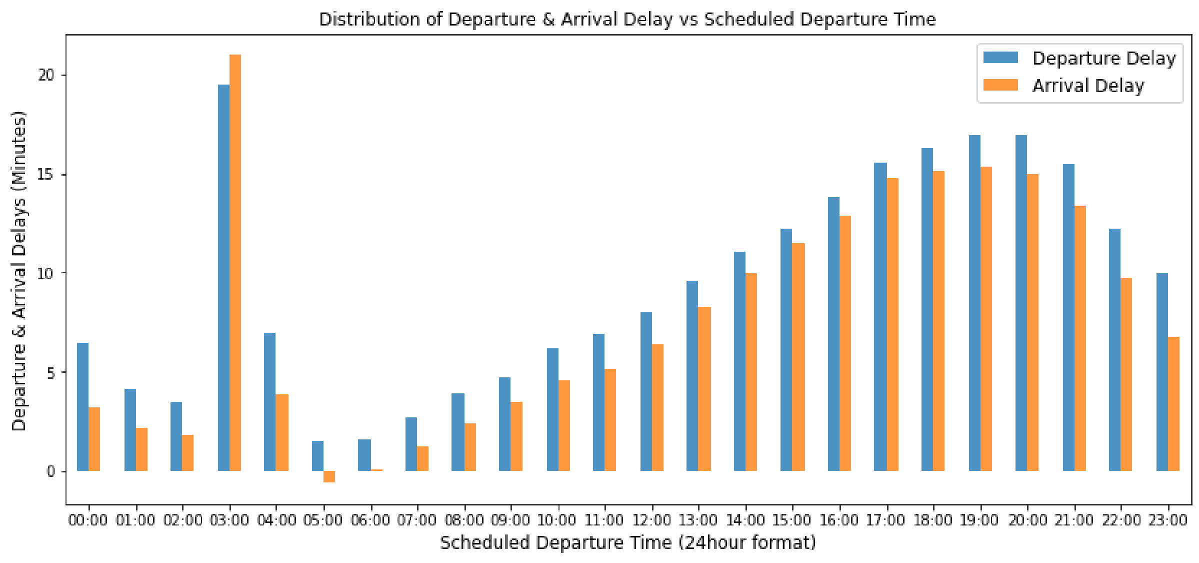


Figure 1.1a shows that the best time of the day to fly is anytime from 5am to 6am as there is minimal departure and arrival delay as illustrated in the bar plot.

Figure 1.1b Distribution of Departure & Arrival Delays vs Day of Week

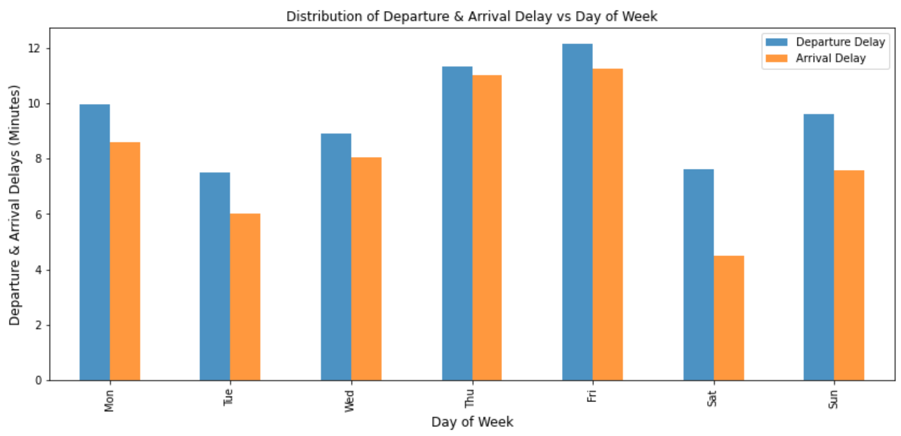


Figure 1.1b shows that the best day of the week to fly is on a Saturday as illustrated with the shortest departure and arrival delays.

Figure 1.1c Distribution of Departure & Arrival Delays vs Month

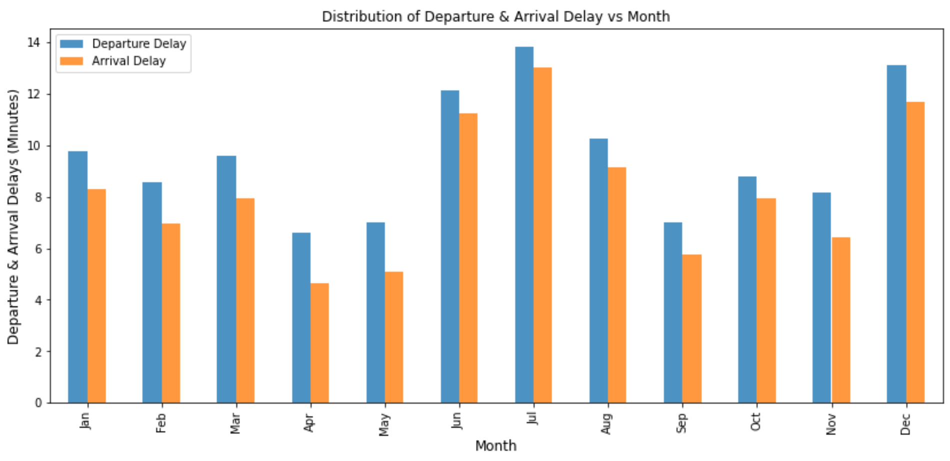
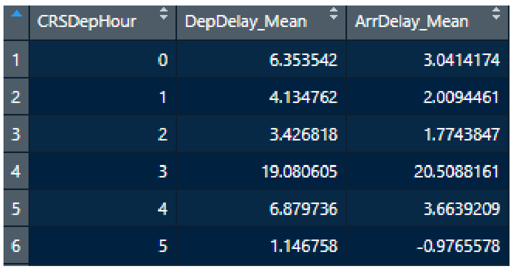


Figure 1.1c shows that the best time of the year to fly is in April or May as illustrated with the shortest departure and arrival delays.

1.2. R Analysis for Q1

Since we began our analysis in Python, we try to replicate the results into R also. Similar to what we did in Python, we convert scheduled departure time into hours format:



	CRSDepHour	DepDelay_Mean	ArrDelay_Mean
1	0	6.353542	3.0414174
2	1	4.134762	2.0094461
3	2	3.426818	1.7743847
4	3	19.080605	20.5088161
5	4	6.879736	3.6639209
6	5	1.146758	-0.9765578

Next, we plot 3 different bar plots to display the best time of the day, day of the week and time of the year to fly in order to minimise delays.

Figure 1.2a Distribution of Departure & Arrival Delays vs Scheduled Departure Time

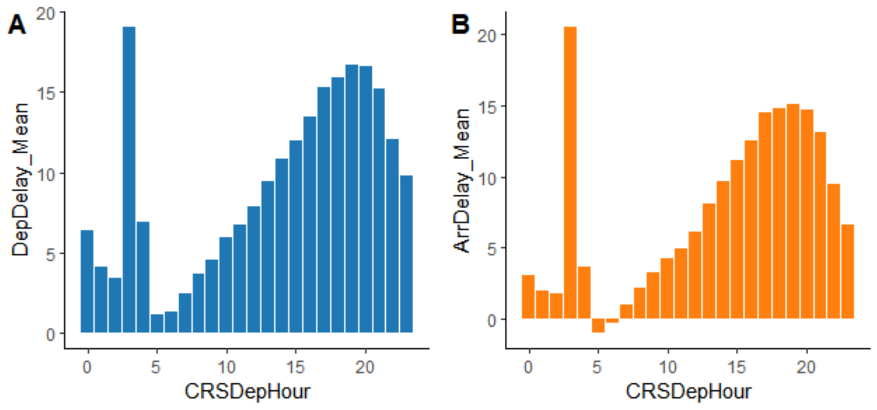


Figure 1.2a shows that the best time of the day to fly is anytime from 5am to 6am as there is minimal departure and arrival delay as illustrated in the bar plot.

Figure 1.2b Distribution of Departure & Arrival Delays vs Scheduled Departure Time

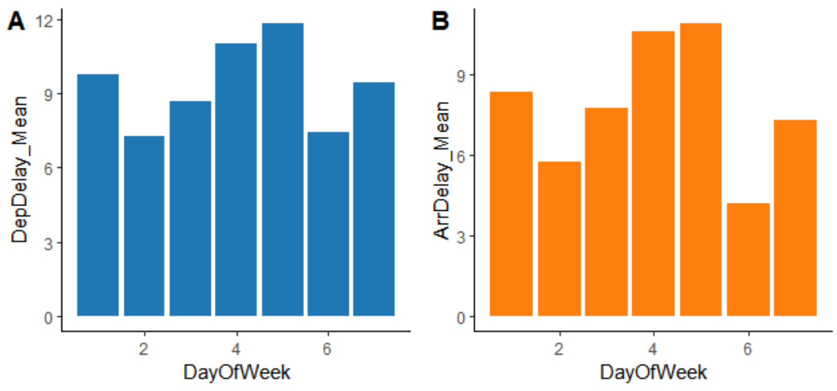


Figure 1.2b shows that the best day of the week to fly is on a Saturday as illustrated with the shortest departure and arrival delays.

Figure 1.2c Distribution of Departure & Arrival Delays vs Scheduled Departure Time

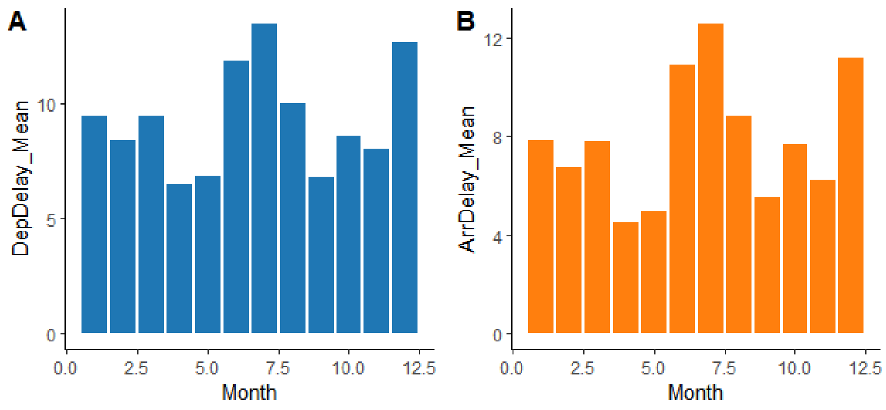
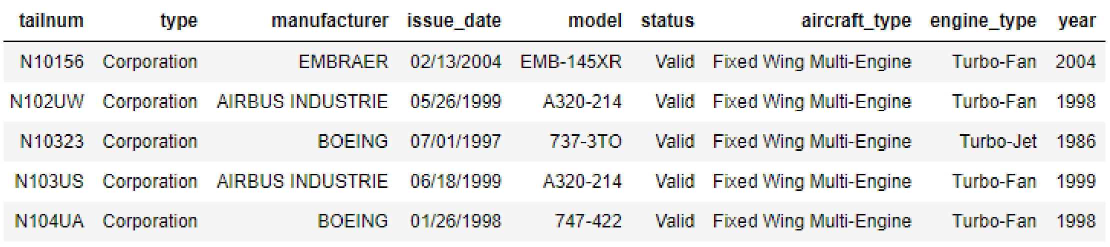


Figure 1.2c shows that the best time of the year to fly is in April or May as illustrated with the shortest departure and arrival delays.

Q2. Do older planes suffer more delays?

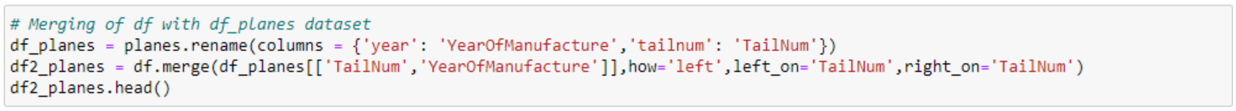
2.1. Python Analysis for Q2

We begin by viewing the planes dataset we imported previously from ‘plane-data.csv’. Then, we check for any null values in the dataset and use the **dropna()** function to remove them.

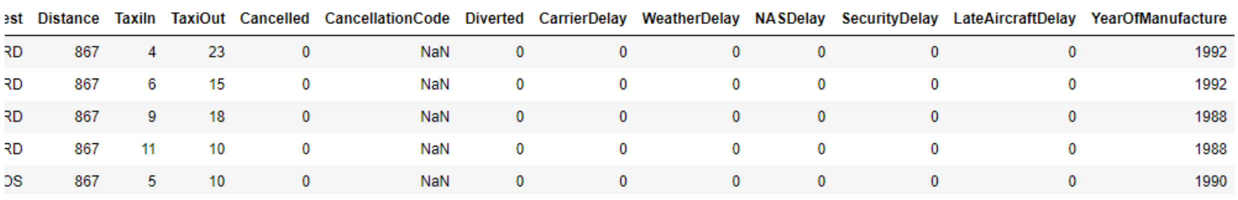


tailnum	type	manufacturer	issue_date	model	status	aircraft_type	engine_type	year
N10156	Corporation	EMBRAER	02/13/2004	EMB-145XR	Valid	Fixed Wing Multi-Engine	Turbo-Fan	2004
N102UW	Corporation	AIRBUS INDUSTRIE	05/26/1999	A320-214	Valid	Fixed Wing Multi-Engine	Turbo-Fan	1998
N10323	Corporation	BOEING	07/01/1997	737-3TO	Valid	Fixed Wing Multi-Engine	Turbo-Jet	1986
N103US	Corporation	AIRBUS INDUSTRIE	06/18/1999	A320-214	Valid	Fixed Wing Multi-Engine	Turbo-Fan	1999
N104UA	Corporation	BOEING	01/26/1998	747-422	Valid	Fixed Wing Multi-Engine	Turbo-Fan	1998

For our analysis in Q2, we would be using the **merge()** function to join our main dataframe, df with the planes dataset, df_planes. We begin by renaming the column “talinum” in df_planes to “Talinum” to match the “Tailnum” column in df. Next, we rename the “year” column in df_planes to “YearOfManufacture” .Then, we merge df and df_planes via ‘Left Join’ to combine both datasets into a single one, df2_planes. This is shown in the figure below:



```
# Merging of df with df_planes dataset
df_planes = planes.rename(columns = {'year': 'YearOfManufacture','tailnum': 'TailNum'})
df2_planes = df.merge(df_planes[['TailNum','YearOfManufacture']],how='left',left_on='TailNum',right_on='TailNum')
df2_planes.head()
```

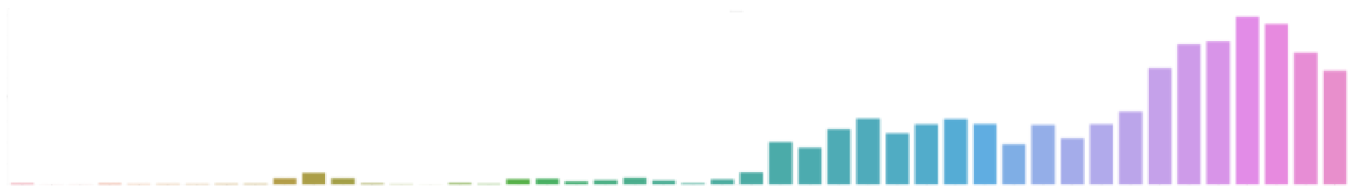


est	Distance	TaxiIn	TaxiOut	Cancelled	CancellationCode	Diverted	CarrierDelay	WeatherDelay	NASDelay	SecurityDelay	LateAircraftDelay	YearOfManufacture
RD	867	4	23	0	NaN	0	0	0	0	0	0	1992
RD	867	6	15	0	NaN	0	0	0	0	0	0	1992
RD	867	9	18	0	NaN	0	0	0	0	0	0	1988
RD	867	11	10	0	NaN	0	0	0	0	0	0	1988
DS	867	5	10	0	NaN	0	0	0	0	0	0	1990

It can be seen that the ‘YearOfManufacture’ column has also been added to our main dataframe and we can start properly analysing the dataset. We begin by finding the sum of departure and arrival delay respectively. Then we use the **sns.catplot()** function to plot the distribution of Arrival/Departure Delays vs Year of Manufacture.

Figure 2.1a Distribution of Arrival Delays vs Year of Manufacture

```
Text(0.5, 0.98, 'Distribution of Arrival Delays vs Year of Manufacture')
```

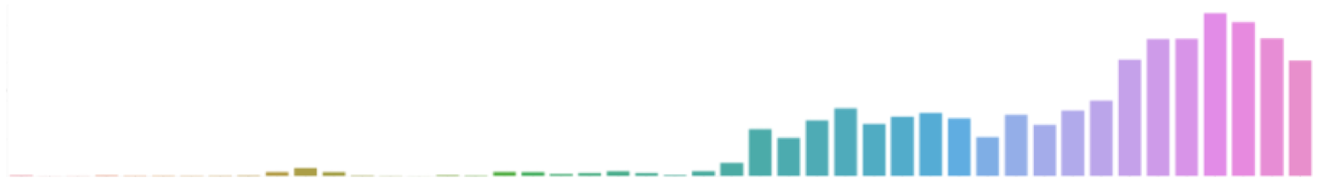


This figure shows the distribution of Arrival Delays vs Year of Manufacture. The x-axis consists of values from the manufacturing year of the planes while the y-axis consists of values from the arrival delay in minutes.

From this bar chart, we can see that majority of arrival delays is found in newer models of planes. As the manufacturing year increase, there is a huge increase in arrival delay.

Figure 2.1b Distribution of Departure Delays vs Year of Manufacture

```
Text(0.5, 0.98, 'Distribution of Departure Delays vs Year of Manufacture')
```



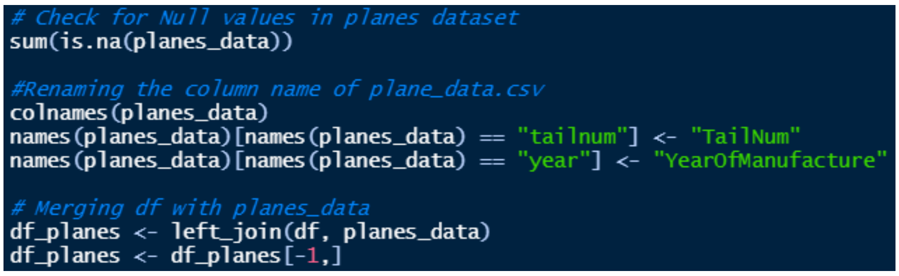
This figure shows the distribution of Departure Delays vs Year of Manufacture. The x-axis consists of values from the manufacturing year of the planes while the y-axis consists of values from the departure delay in minutes.

From this bar chart, we can see that majority of departure delays is found in newer models of planes. As the manufacturing year increase, there is a huge increase in departure delay.

All in all, results in both plots suggest that newer models of planes are responsible for majority of delays. Hence, we conclude that older planes do not suffer more delays.

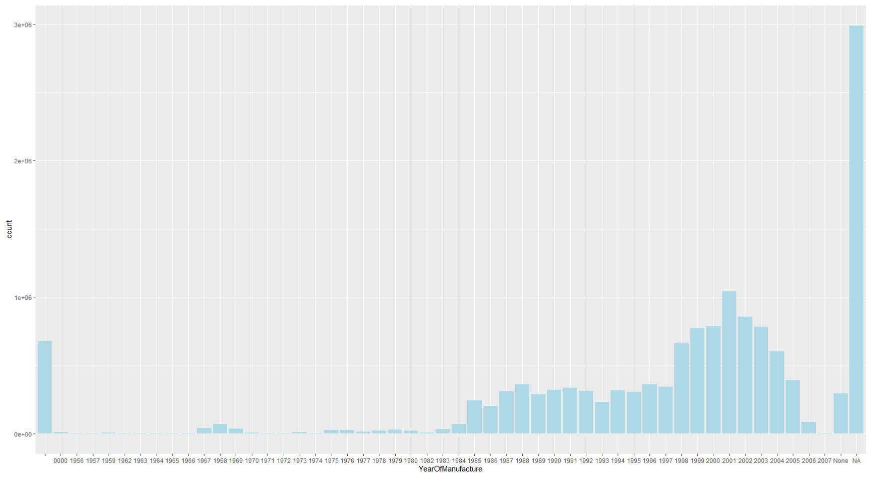
2.2. R Analysis for Q2

Similarly, to our steps in Python for Q2, we begin by checking for any null values in our planes dataframe, planes_data. Next, we rename the column names in planes_data. “talinum” was renamed to “TailNum” while “year” was renamed to “YearOfManufacture”. Then, we merge the main dataframe, df, with planes_data with the **left_join()** function from the dplyr library. The following steps are shown in the figure below:



```
# Check for Null values in planes dataset
sum(is.na(planes_data))

#Renaming the column name of plane_data.csv
colnames(planes_data)
names(planes_data)[names(planes_data) == "tailnum"] <- "TailNum"
names(planes_data)[names(planes_data) == "year"] <- "YearOfManufacture"

# Merging df with planes_data
df_planes <- left_join(df, planes_data)
df_planes <- df_planes[-1,]
```

Moving on, we use the **ggplot()** function and **geom_histogram()** function to plot the distribution of Departure Delays & Arrival Delays over the Year of Manufacture for the planes. The plot is shown in the figure below:

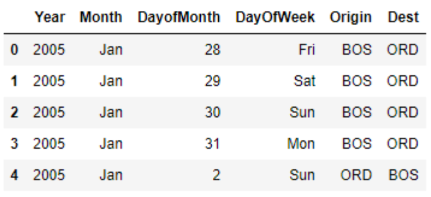


This figure shows that majority of flight delays seems to occur in planes that were manufactured between the years of 1998 and 2004. Flight delays from planes manufactured between the years of 1956 and 1982 were significantly lower as compared to newer planes.

Q3. How does the number of people flying between different locations change over time?

3.1. Python Analysis for Q3

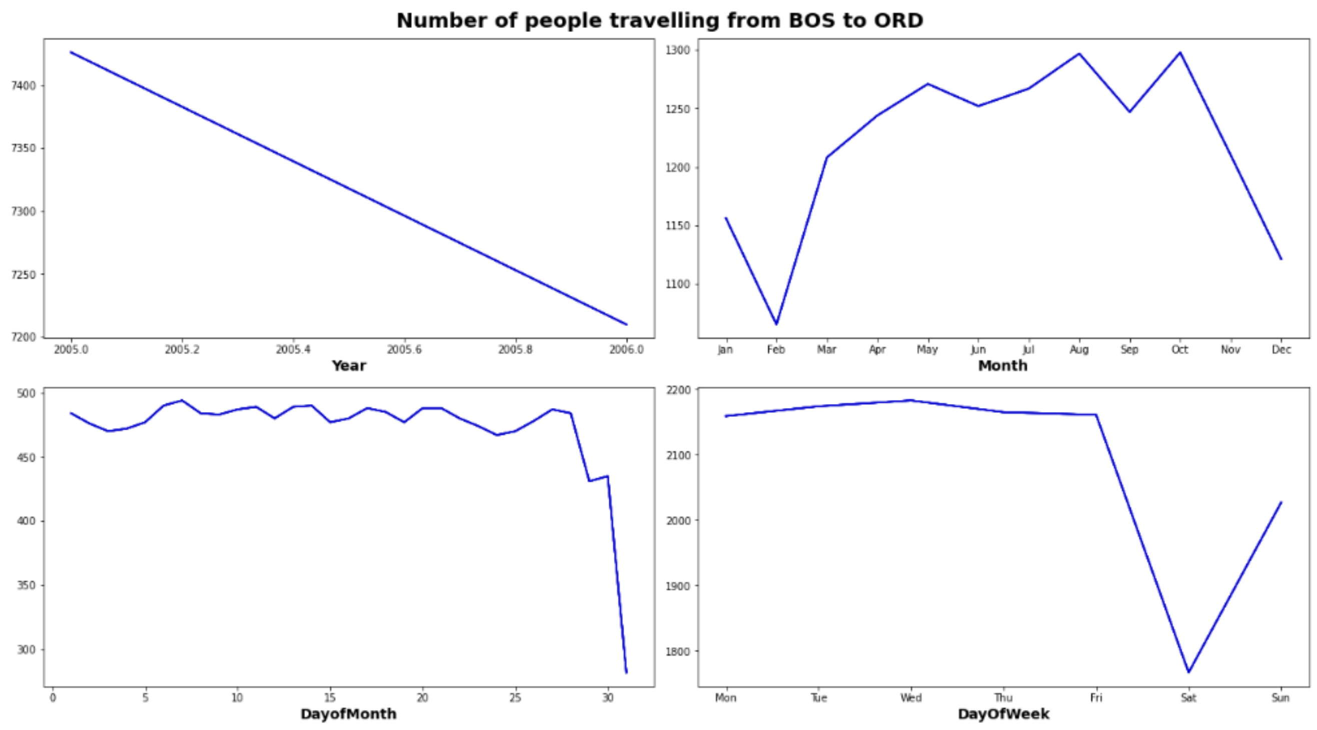
To begin, we subset our main dataframe, df, to only display time-related variables and the origin and destinations of the flights.



	Year	Month	DayofMonth	DayOfWeek	Origin	Dest
0	2005	Jan	28	Fri	BOS	ORD
1	2005	Jan	29	Sat	BOS	ORD
2	2005	Jan	30	Sun	BOS	ORD
3	2005	Jan	31	Mon	BOS	ORD
4	2005	Jan	2	Sun	ORD	BOS

The variables “BOS”, “ORD” represents the airport IATA code and in this case “BOS” represents Boston while “ORD” represents Chicago according to the column ”city” from the airport dataset which we imported from “airports.csv”.

As the dataset from our main dataframe, df, is extremely large, we would only be considering people flying from the city of Boston (BOS) to Chicago (ORD).



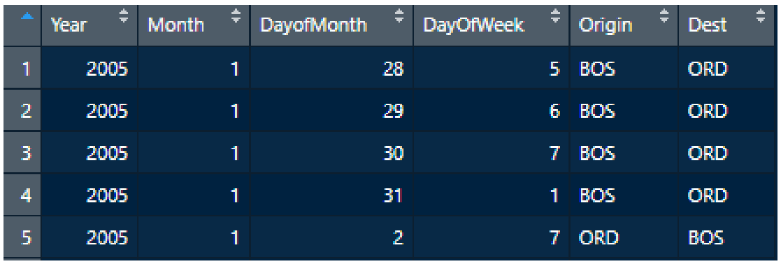
The figure shows 4 different line plots to represent the number of people travelling from Boston to Chicago over various time-related variables. Our analysis is as follows:

- The number of people travelling from BOS to ORD decreased from 2005 to 2006.

- The number of people traveling from BOS to ORD is at its lowest in the month of February while at its peak in October
- The number of people travelling from BOS to ORD is rather consistent from up till the 25th of the month whereby there is a sharp decrease.
- The number of people travelling from BOS to ORD is higher during the weekdays compared to the weekend. Saturday has the least number of people doing so.

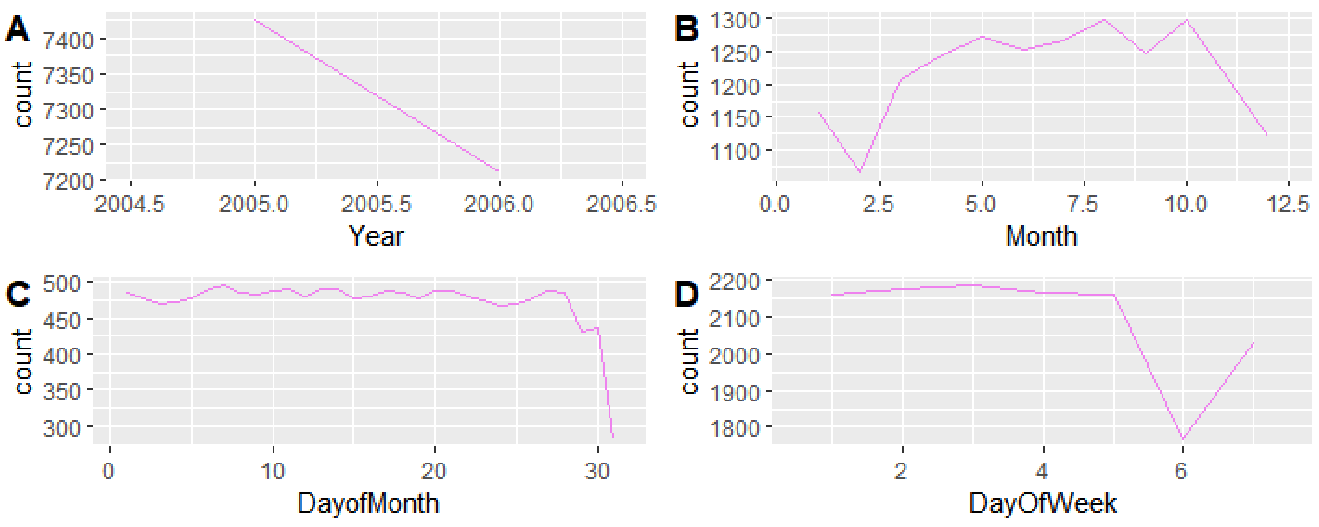
3.2. R Analysis for Q3

We start out analysis in R by subsetting our main dataframe, df, to only display the time-related variables (Year, Month, Day of Month, Day of Week) along with the origin and destination locations.



	Year	Month	DayofMonth	DayOfWeek	Origin	Dest
1	2005	1	28	5	BOS	ORD
2	2005	1	29	6	BOS	ORD
3	2005	1	30	7	BOS	ORD
4	2005	1	31	1	BOS	ORD
5	2005	1	2	7	ORD	BOS

Next, we plot the 4 different variables and the count of people travelling from BOS to ORD over our time variables:

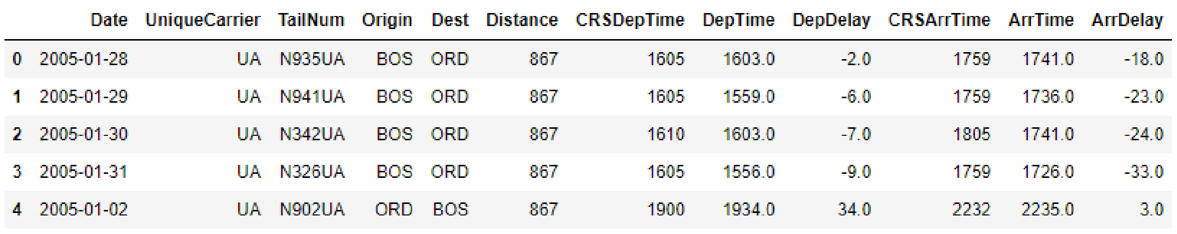


Visually, our figure plotted in R is similar to our figure plotted in Python. Hence, we are able to come to the same conclusion as in Python. The analysis can be found in **Chapter 3.1** of the report.

Q4. Can you detect cascading failures as delays in one airport create delays in others?

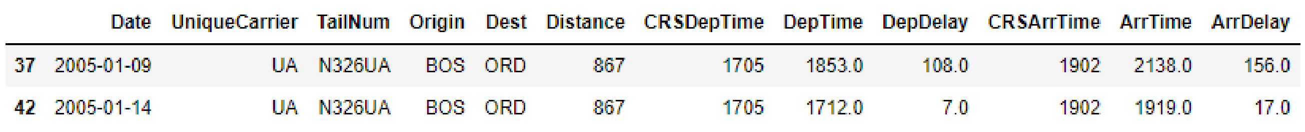
4.1. Python Analysis for Q4

To begin, we make a copy of our original main dataframe, df, and make a copy of it with the **copy()** function. Next, we add a 'Date' column to the dataset which displays the date in the yy-mm-dd format. Then, we subset the dataframe we made a copy of to only display the following columns:



	Date	UniqueCarrier	TailNum	Origin	Dest	Distance	CRSDepTime	DepTime	DepDelay	CRSArrTime	ArrTime	ArrDelay
0	2005-01-28	UA	N935UA	BOS	ORD	867	1605	1603.0	-2.0	1759	1741.0	-18.0
1	2005-01-29	UA	N941UA	BOS	ORD	867	1605	1559.0	-6.0	1759	1736.0	-23.0
2	2005-01-30	UA	N342UA	BOS	ORD	867	1610	1603.0	-7.0	1805	1741.0	-24.0
3	2005-01-31	UA	N326UA	BOS	ORD	867	1605	1556.0	-9.0	1759	1726.0	-33.0
4	2005-01-02	UA	N902UA	ORD	BOS	867	1900	1934.0	34.0	2232	2235.0	3.0

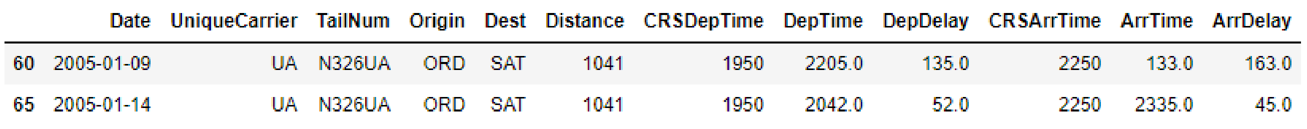
Next, we further subset the new dataframe to only show flights with TailNumber “N326UA”, Arrival and Departure delays that are greater than 0.



	Date	UniqueCarrier	TailNum	Origin	Dest	Distance	CRSDepTime	DepTime	DepDelay	CRSArrTime	ArrTime	ArrDelay
37	2005-01-09	UA	N326UA	BOS	ORD	867	1705	1853.0	108.0	1902	2138.0	156.0
42	2005-01-14	UA	N326UA	BOS	ORD	867	1705	1712.0	7.0	1902	1919.0	17.0

This figure displays flight data for the plane with Tail Number “N326UA” on 2005-01-09.On 2005-01-09, Flight "N326UA" was scheduled to travel from 'BOS' to 'ORD'. Flight "N326UA" was scheduled to depart 'BOS' at 1705 (5.05 pm) but the actual departure time was at 1853 (6.53 pm). There was a departure delay of 108 minutes. Flight "N326UA" was scheduled to arrive at 'ORD' at 1902 (7.02 pm) but the actual arrival time was at 2138 (9.38 pm). There was an arrival delay of 156 minutes.

Next, we check what happened at the ‘ORD’ airport.

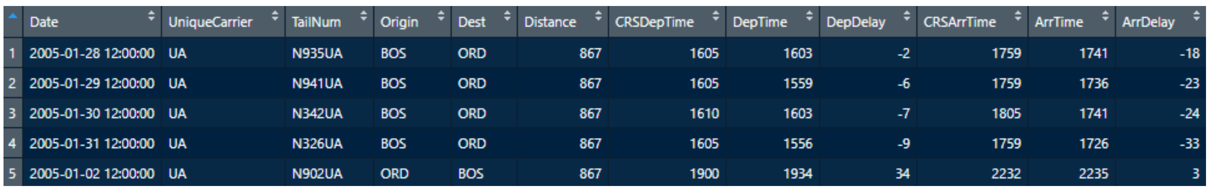


	Date	UniqueCarrier	TailNum	Origin	Dest	Distance	CRSDepTime	DepTime	DepDelay	CRSArrTime	ArrTime	ArrDelay
60	2005-01-09	UA	N326UA	ORD	SAT	1041	1950	2205.0	135.0	2250	133.0	163.0
65	2005-01-14	UA	N326UA	ORD	SAT	1041	1950	2042.0	52.0	2250	2335.0	45.0

This figure displays flight data from ‘ORD’ airport on 2005-01-09.At 'ORD' airport on 2005-01-09, Flight "N326UA" that was scheduled to depart from 'ORD' to 'SAT' was delayed by 135 minutes. The scheduled departure time was 1950 (7.50 pm) but the actual departure time was 2205 (10.05 pm).This shows that the departure delay in the 'BOS' airport on 2005-01-09 caused a departure delay in the 'ORD' airport on the same date.

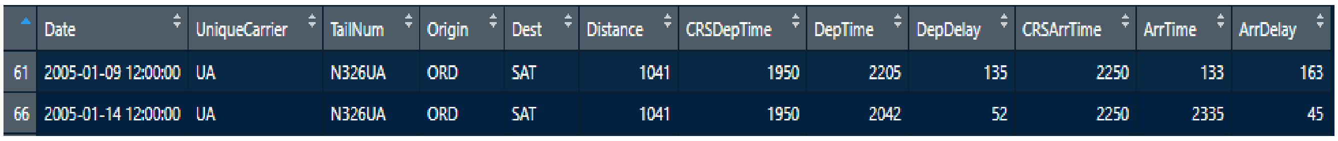
4.2. R Analysis for Q4

We begin our analysis in R by adding a date column to the dataset and subsetting the columns required for our analysis. The variables used are as follows:



	Date	UniqueCarrier	TailNum	Origin	Dest	Distance	CRSDepTime	DepTime	DepDelay	CRSArrTime	ArrTime	ArrDelay
1	2005-01-28 12:00:00	UA	N935UA	BOS	ORD	867	1605	1603	-2	1759	1741	-18
2	2005-01-29 12:00:00	UA	N941UA	BOS	ORD	867	1605	1559	-6	1759	1736	-23
3	2005-01-30 12:00:00	UA	N342UA	BOS	ORD	867	1610	1603	-7	1805	1741	-24
4	2005-01-31 12:00:00	UA	N326UA	BOS	ORD	867	1605	1556	-9	1759	1726	-33
5	2005-01-02 12:00:00	UA	N902UA	ORD	BOS	867	1900	1934	34	2232	2235	3

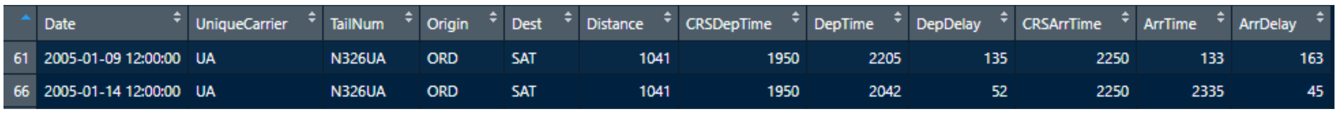
Next, we further subset the dataframe to only show planes with TailNum ‘N326UA’ and Arrival Delay & Departure Delay > 0. This shown in the figure below:



	Date	UniqueCarrier	TailNum	Origin	Dest	Distance	CRSDepTime	DepTime	DepDelay	CRSArrTime	ArrTime	ArrDelay
61	2005-01-09 12:00:00	UA	N326UA	ORD	SAT	1041	1950	2205	135	2250	133	163
66	2005-01-14 12:00:00	UA	N326UA	ORD	SAT	1041	1950	2042	52	2250	2335	45

This figure obtained from R is similar to the one in Python and interpretation of the data can be found in **Chapter 4.1** of the report.

Moving on, we look at data from ‘ORD’ airport on 2005-01-09.



	Date	UniqueCarrier	TailNum	Origin	Dest	Distance	CRSDepTime	DepTime	DepDelay	CRSArrTime	ArrTime	ArrDelay
61	2005-01-09 12:00:00	UA	N326UA	ORD	SAT	1041	1950	2205	135	2250	133	163
66	2005-01-14 12:00:00	UA	N326UA	ORD	SAT	1041	1950	2042	52	2250	2335	45

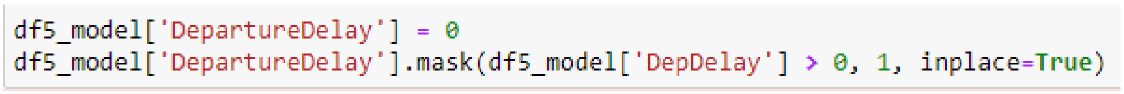
Both figures extracted from R is similar to Python and interpretation of the data for both is similar and can be found in **Chapter 4.1** of the report.

Q5. Use the available variables to construct a model that predicts delays.

5.1. Prediction model in Python

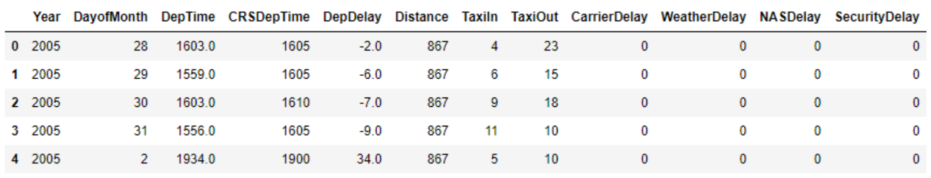
Firstly, we subset our dataframe to select only the variables used in our model. Then, we further subset the dataframe to only contain the first 10,000 rows.

Next, we create a column[‘DepartureDelay’] that contains 2 values, 0 and 1. If integer values found in the ‘DepDelay’ column is greater than 0 than it would be assigned a value of 1(Delay), else it would be 0 (No Delay). This step is shown below:



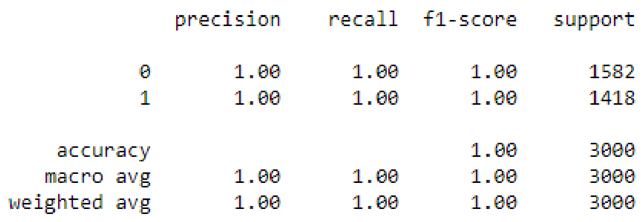
```
df5_model['DepartureDelay'] = 0
df5_model['DepartureDelay'].mask(df5_model['DepDelay'] > 0, 1, inplace=True)
```

Next, we define our independent and dependent variables. Our dependent variable was the [‘DepartureDelay’] column we created earlier that have a binary value of 0 and 1. Our independent variables are as follows:



	Year	DayofMonth	DepTime	CRSDepTime	DepDelay	Distance	TaxiIn	TaxiOut	CarrierDelay	WeatherDelay	NASDelay	SecurityDelay
0	2005	28	1603.0	1605	-2.0	867	4	23	0	0	0	0
1	2005	29	1559.0	1605	-6.0	867	6	15	0	0	0	0
2	2005	30	1603.0	1610	-7.0	867	9	18	0	0	0	0
3	2005	31	1556.0	1605	-9.0	867	11	10	0	0	0	0
4	2005	2	1934.0	1900	34.0	867	5	10	0	0	0	0

Then, we split our data into trainset and testset with a split ratio of 70% trainset and 30% testset. Next, we fit the logistic regression module with our data and perform predictions on the testset. The performance of our prediction model will be evaluated with the classification report module from **sklearn.metrics**. The results are as follows:



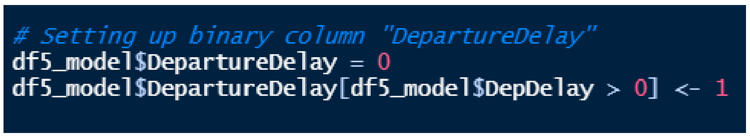
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1582
           1       1.00      1.00      1.00      1418

    accuracy                           1.00      3000
   macro avg       1.00      1.00      1.00      3000
weighted avg       1.00      1.00      1.00      3000
```

The high accuracy score of close to 100% suggest that the variables used to predict Departure Delays for our models are highly correlated or there is a possibility of overfitting our data.

5.2. Prediction model in R

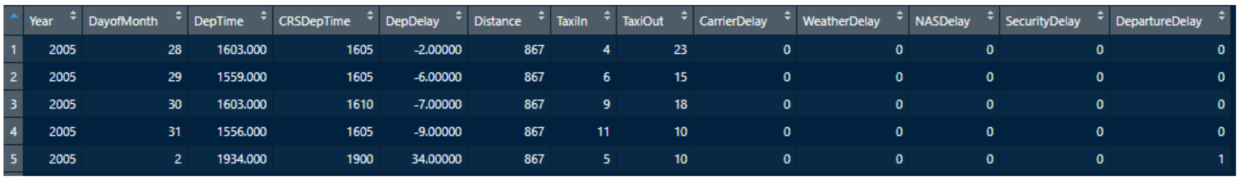
The steps we would undertake to develop our model in R would be pretty similar to the one in Python. Firstly, we subset our dataframe with the variables. Then, we further subset the dataframe to only contain the first 10,000 rows.

Next, we impute all the null values found in the dataset with their mean and setup our binary column [‘DepartureDelay’] that has an outcome of 0 (No Delay) or 1 (Delay). This step is shown below:



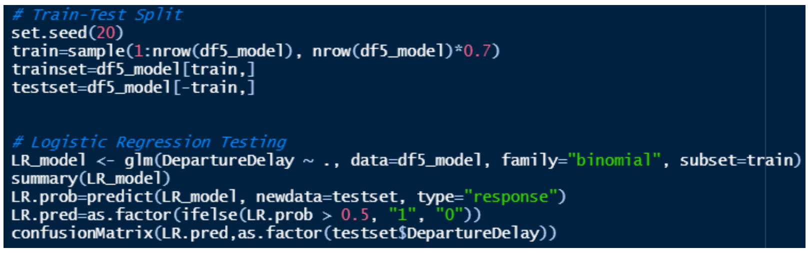
```r
# Setting up binary column "DepartureDelay"
df5_model$DepartureDelay = 0
df5_model$DepartureDelay[df5_model$DepDelay > 0] <- 1
```

Our dependent variable would be the binary column ‘Departure Delay’ as stated above. Here are our independent variables as shown in R:



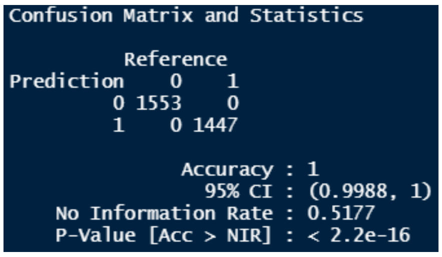
	Year	DayofMonth	DepTime	CRSDepTime	DepDelay	Distance	TaxiIn	TaxiOut	CarrierDelay	WeatherDelay	NASDelay	SecurityDelay	DepartureDelay
1	2005	28	1603.000	1605	-2.00000	867	4	23	0	0	0	0	0
2	2005	29	1559.000	1605	-6.00000	867	6	15	0	0	0	0	0
3	2005	30	1603.000	1610	-7.00000	867	9	18	0	0	0	0	0
4	2005	31	1556.000	1605	-9.00000	867	11	10	0	0	0	0	0
5	2005	2	1934.000	1900	34.00000	867	5	10	0	0	0	0	1

Then we split our data into a trainset and testset with a split ratio of 70% train and 30% test. Lastly, we apply the Logistic Regression model into our dataframe and perform predictions on the testset.



```r
# Train-Test Split
set.seed(20)
train=sample(1:nrow(df5_model), nrow(df5_model)*0.7)
trainset=df5_model[train,]
testset=df5_model[-train,]


# Logistic Regression Testing
LR_model <- glm(DepartureDelay ~ ., data=df5_model, family="binomial", subset=train)
summary(LR_model)
LR.prob=predict(LR_model, newdata=testset, type="response")
LR.pred=as.factor(ifelse(LR.prob > 0.5, "1", "0"))
confusionMatrix(LR.pred,as.factor(testset$DepartureDelay))
```

Finally, we evaluate the performance of our model with the **confusionMatrix()** function in R. Our results are as follows:



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1553    0
         1    0 1447

               Accuracy : 1
                 95% CI : (0.9988, 1)
    No Information Rate : 0.5177
    P-Value [Acc > NIR] : < 2.2e-16
```

The high accuracy score of close to 100% suggest that the variables used to predict Departure Delays for our models are highly correlated or there is a possibility of overfitting in our data.

References

Navlani, A., 2019. *Understanding Logistic Regression in Python Tutorial*. [online] https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python. Available at: <https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python> [Accessed 1 March 2022].

Narkhede, S., 2018. *Understanding Confusion Matrix*. [online] https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. Available at: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62> [Accessed 1 March 2022].

James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. *An Introduction to Statistical Learning: With Applications in R*. 2nd ed. Springer.

Chew, C., 2020. *Artificial intelligence, Analytics and Data Science*. 1st ed. Singapore.