UNIVERSITY OF LONDON

THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

Coursework Report- Python and R

Module – Programming for Data Science (ST2195)

Student ID- 210473416

Page Count – 10 pages excluding page 1 and page 2.

Table of contents

Introduction

The 2009 ASA statistical computing and graphics data expo consisted of flight details from October 1987 to April 2008 of departure, arrival, route travelled, airport etc. For the analysis, the 2006 and 2007 datasets were used. The analysis was done using both R and Python programming languages. In addition, other .csv files on airport and plane data were used when answering the questions.

The content of this report includes visualizations, approaches and conclusions of each task, and are discussed in detail below.

Importing and cleaning data

The 2006 and 2007 datasets were merged into one dataset consisting of a total of 14595137 observations and 29 variables. The number of null values in each column was checked and it was identified that 98% of the rows in the cancellation column had null values. Therefore, the cancellation column was removed entirely. Additionally, rows which had null values were removed and this removed a total of 31,6047 rows out of 14595137 which seemed plausible to proceed with considering the dataset is extremely large.

Considering DepTime, CRSDepTime, ArrTime and CRSArrTime following the 24-time format, rows which consisted of time more than 2359 were removed. Hence, bound checking was performed to make sure no values were out of bound.

The cleaned data was saved to a new variable called the *cleaned_data.csv* and was used in every question for analysis.

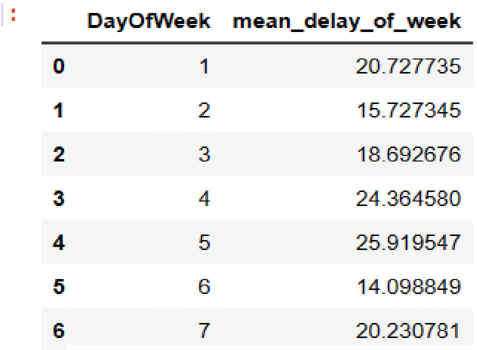
It is noticeable that the ArrDelay is the sum of the delay types such as the weather delay, Nas delay etc. Furthermore, the TailNum is considered unique for each flight when answering the following questions.

Question 1: When is the best time of day, day of the week and time of year to fly to minimise delays?

For this question a new column was added which was the sum of the DepDelay and ArrDelay which was saved under a variable called the *sum of delays.* This was used to identify the best time of day, day of the week and time of year to fly by grouping it with respect to the suitable categories and minimizing the mean sum of delays for each group.

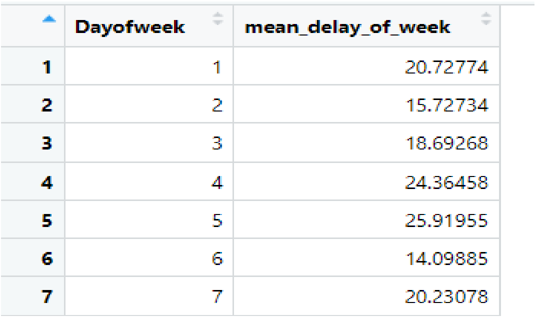
Best day of the week –

The DayOfWeek are numbered from 1 to 7, where 1=Monday and 7= Sunday. To find the best day of the week to fly to minimize delays the DayOfWeek column was grouped by the numbers from 1 to 7 and then the mean of the sum of delays for each group was computed. The DayOfWeek with the lowest mean of the sum of delays is considered as the best day of the week to fly.



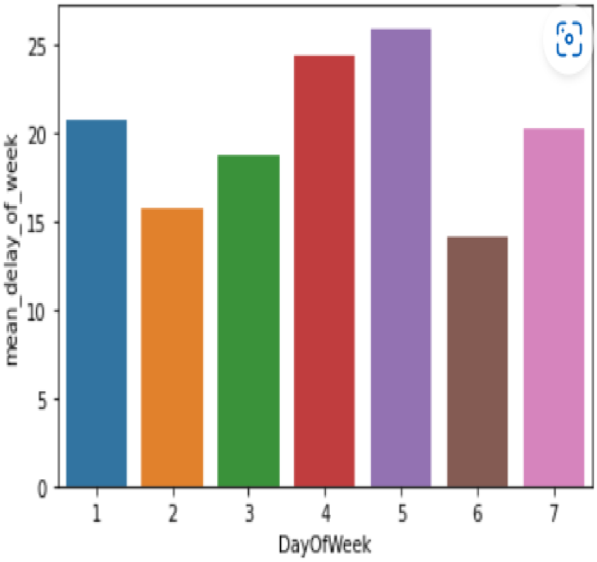
	DayOfWeek	mean_delay_of_week
0	1	20.727735
1	2	15.727345
2	3	18.692676
3	4	24.364580
4	5	25.919547
5	6	14.098849
6	7	20.230781

Day of week groupby table in python

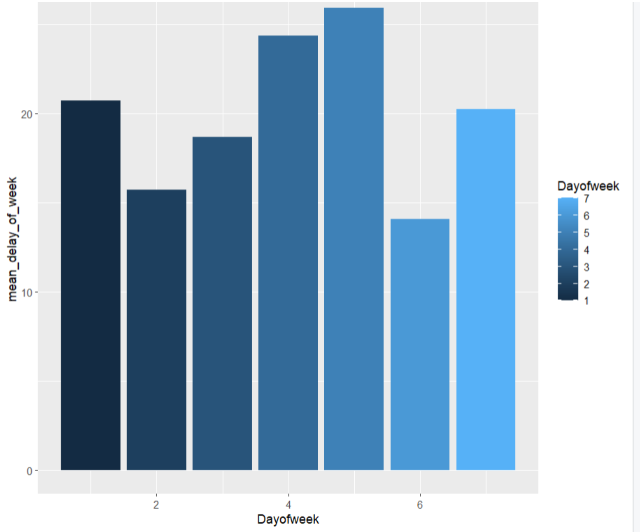


	Dayofweek	mean_delay_of_week
1	1	20.72774
2	2	15.72734
3	3	18.69268
4	4	24.36458
5	5	25.91955
6	6	14.09885
7	7	20.23078

Day of week groupby table in R



Day of week bar plot in python

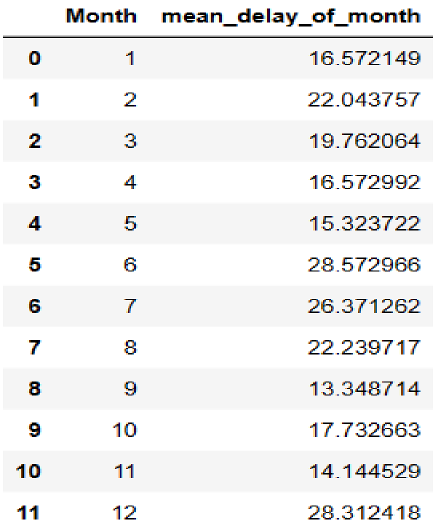


Day of week bar plot in R

From the above graphs and data frames, it is visible that the best day of the week to fly is Saturday as it has the lowest mean sum of delays with a value of **14.099**.

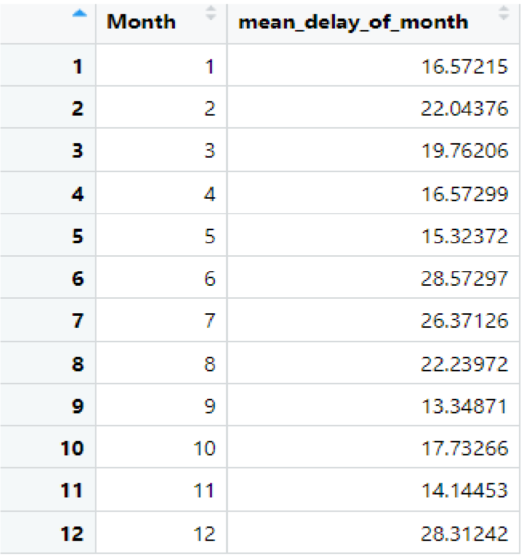
Best time of the year –

An identical approach is followed to identify the best time of the year. Each month is labelled from 1 to 12 where 1=January and 12=December. The best month to fly is considered as the best time of the year to fly. The data is grouped by the months and the mean sum of delays are computed for each month. The Month with the lowest mean sum of the delays is considered as the best month of the year.



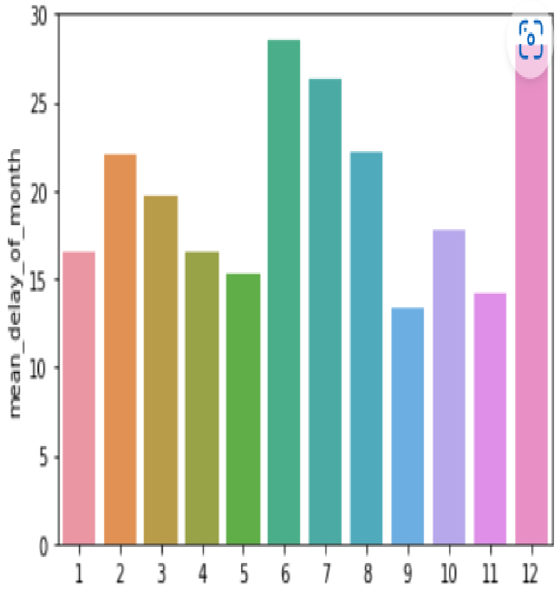
	Month	mean_delay_of_month
0	1	16.572149
1	2	22.043757
2	3	19.762064
3	4	16.572992
4	5	15.323722
5	6	28.572966
6	7	26.371262
7	8	22.239717
8	9	13.348714
9	10	17.732663
10	11	14.144529
11	12	28.312418

Month groupby table for Python

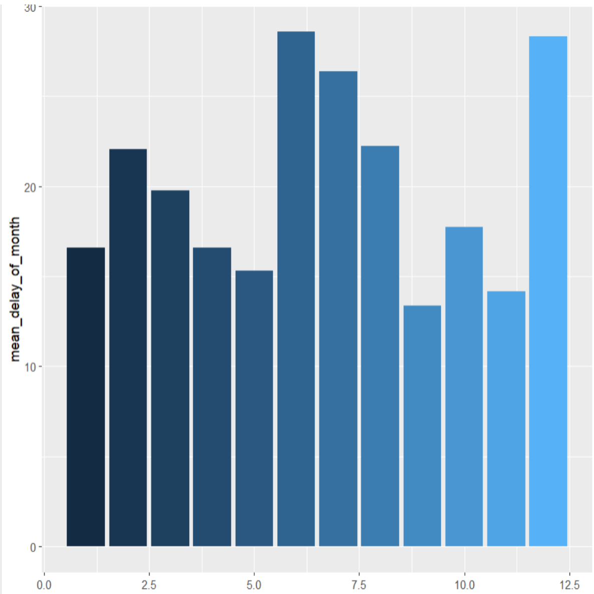


	Month	mean_delay_of_month
1	1	16.57215
2	2	22.04376
3	3	19.76206
4	4	16.57299
5	5	15.32372
6	6	28.57297
7	7	26.37126
8	8	22.23972
9	9	13.34871
10	10	17.73266
11	11	14.14453
12	12	28.31242

Month groupby table for R



Month Bar plot in Python



Month Bar plot in R

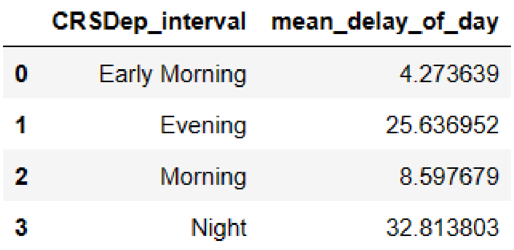
The above graphs and data frames portray that September is the best month to fly since it has the lowest mean sum of delays with a value of **13.349**. May and November also have a low mean sum of delay. Hence, could also be considered as best months to fly.

Best time of the day –

A very similar approached was used for this part. The only difference is that the time of the day was binned into four categories consisting of Early morning, Morning, Evening and Night. This was done using the CRSDepTime. The data is then grouped by the above mentioned four categories and the mean sum of delays are calculated for each group. The category with the lowest mean sum of delays is the best time of the day to fly.

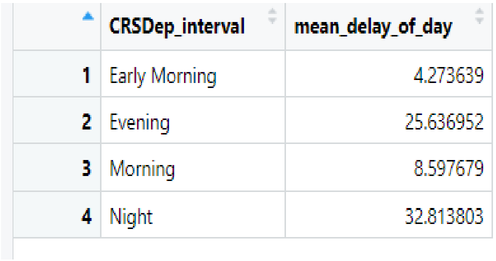
The categories were divided as follows:

Early morning	CRSDepTime<600
Morning	600<=CRSDepTime<1200
Evening	1200<=CRSDepTime<1800
Night	1800<=CRSDepTime<= 2359



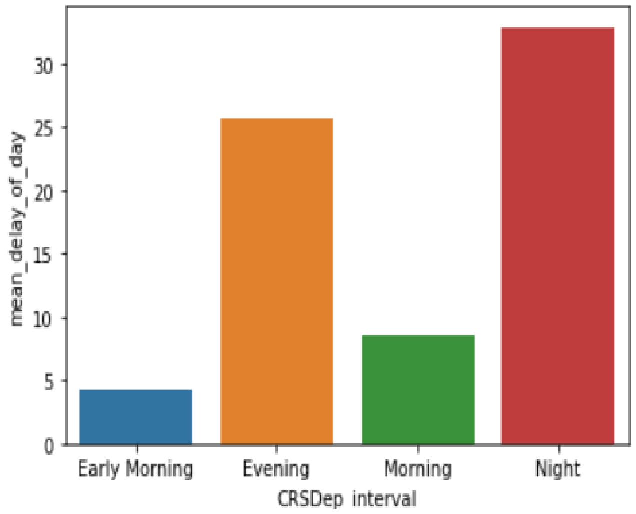
	CRSDep_interval	mean_delay_of_day
0	Early Morning	4.273639
1	Evening	25.636952
2	Morning	8.597679
3	Night	32.813803

Group data for CRSDep_interval in Python

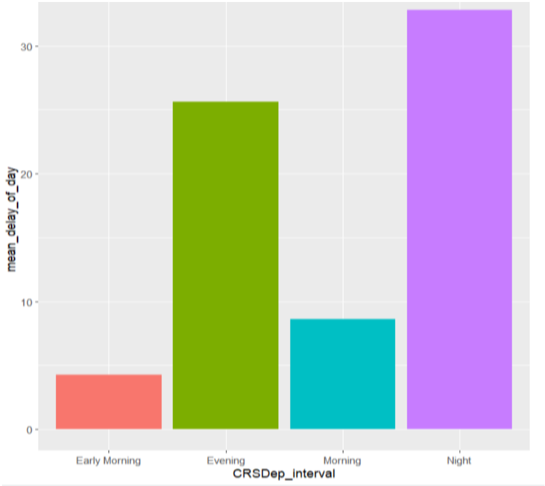


	CRSDep_interval	mean_delay_of_day
1	Early Morning	4.273639
2	Evening	25.636952
3	Morning	8.597679
4	Night	32.813803

Group data for CRSDep_interval in R



Bar plot for CRSDep_interval in Python



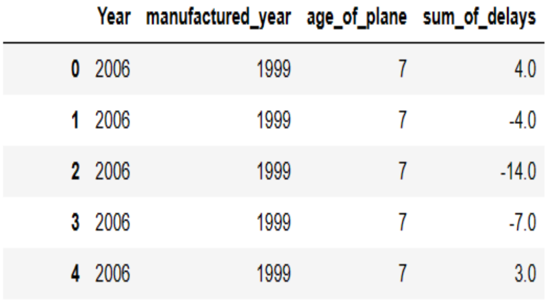
Bar plot for CRSDep_interval in R

The above graphs and data frames illustrate that the best time of the day to fly is Early Morning as it has the lowest mean sum of delay with a value of **4.274**.

Question 2: Do older planes suffer more delays?

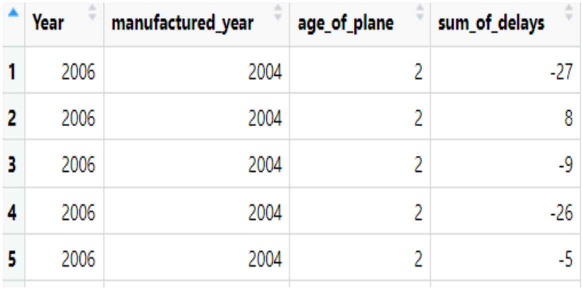
Since the age of plane is considered, the *plane-data.csv* dataset was merged with the *cleaned_dataset.csv* (Check page 3) on the TailNum column. All rows which have null values and manufactured year equal to “0000” were removed. A column for the age of plane was added, which is the difference between the manufactured year and the Year. In this analysis the age of plane varied from 0 to 51 and age of plane equal to zero was not considered.

Moreover, the sum _of_ delays column was added, which is the sum of the ArrDelay and the DepDelay as mentioned in Question 1.



	Year	manufactured_year	age_of_plane	sum_of_delays
0	2006	1999	7	4.0
1	2006	1999	7	-4.0
2	2006	1999	7	-14.0
3	2006	1999	7	-7.0
4	2006	1999	7	3.0

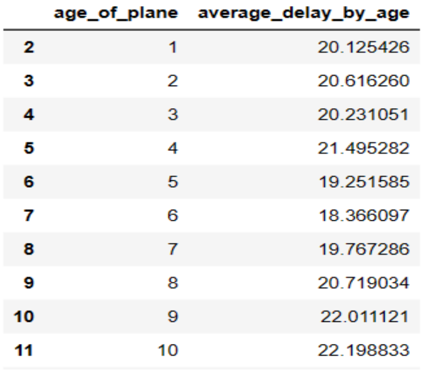
Subset of the dataframe in Python for the manufactured year 1999



	Year	manufactured_year	age_of_plane	sum_of_delays
1	2006	2004	2	-27
2	2006	2004	2	8
3	2006	2004	2	-9
4	2006	2004	2	-26
5	2006	2004	2	-5

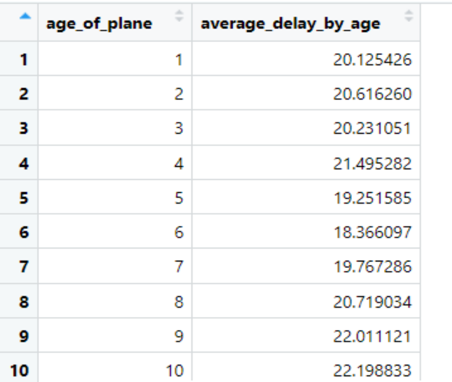
Subset of the dataframe in R for the manufactured year 2004

The data was then grouped by the age of the plane and the mean sum of the delays for each age group was calculated.



	age_of_plane	average_delay_by_age
2	1	20.125426
3	2	20.616260
4	3	20.231051
5	4	21.495282
6	5	19.251585
7	6	18.366097
8	7	19.767286
9	8	20.719034
10	9	22.011121
11	10	22.198833

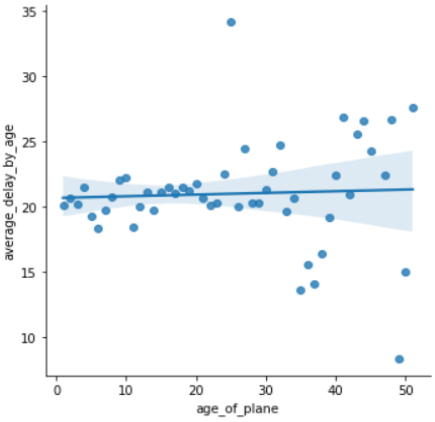
Subset of the dataframe in Python



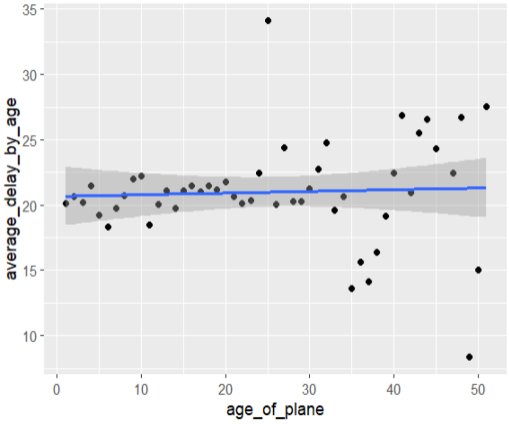
	age_of_plane	average_delay_by_age
1	1	20.125426
2	2	20.616260
3	3	20.231051
4	4	21.495282
5	5	19.251585
6	6	18.366097
7	7	19.767286
8	8	20.719034
9	9	22.011121
10	10	22.198833

Subset of the dataframe in R

A lm plot was plotted to check if there's a linear relationship between the age of the plane and the average delay by the age of the plane.



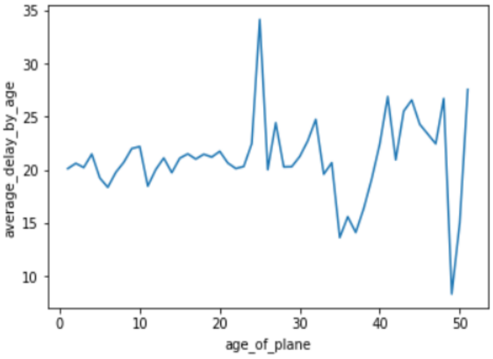
Lm plot in Python



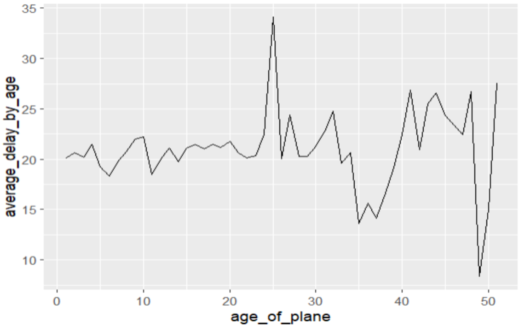
Lm plot in R

The graph depicts that the line is almost horizontal, and it was calculated and found that it has a correlation co-efficient of **0.04794**, which is almost zero. Hence, it can be concluded that there's no linear relationship between the age of the plane and the delays.

To check the overall variation of the age of the plane and the average delay by the age of the plane a line plot was constructed.



Line plot in Python



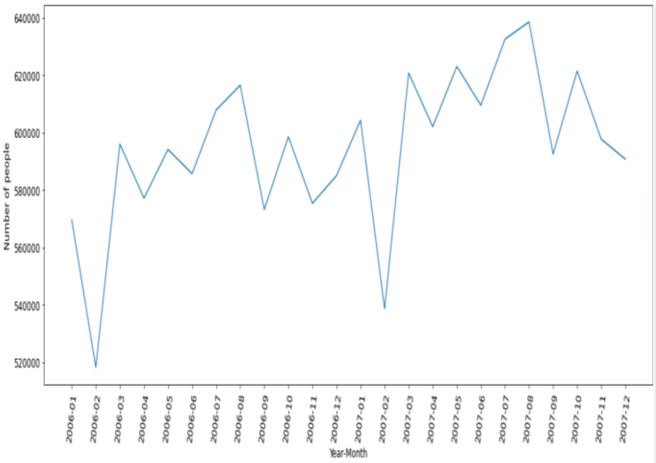
Line Plot in R

The line plot portrays that there is no clear trend between the age of the plane and the average delay by age. This further emphasizes that there is no relationship between the age of the plane and the delays.

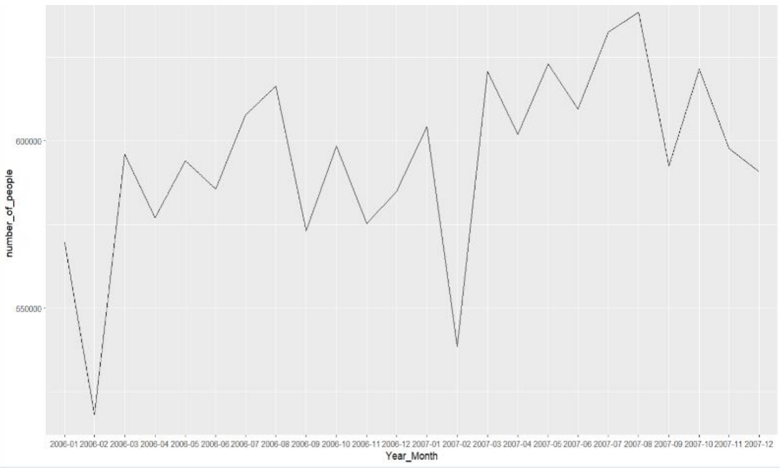
Question 3: How does the number of people flying between different locations change over time?

Since there is no data about the number of people flying, the number of flights is taken as a substitute. The cleaned dataset is merged with the airports.csv dataset on the Origin column to form a merged dataset. The Year and the Month column was then combined into one column (Year-Month) in the date-time format. Afterwards, the data was grouped based on the Year-Month and the size of each group was found. The size of each group is considered as the number of flights in each group, which in turn means the number of people flying in each group.

A line plot was plotted to demonstrate the relationship between the number of people flying and the Month.



Line plot in Python

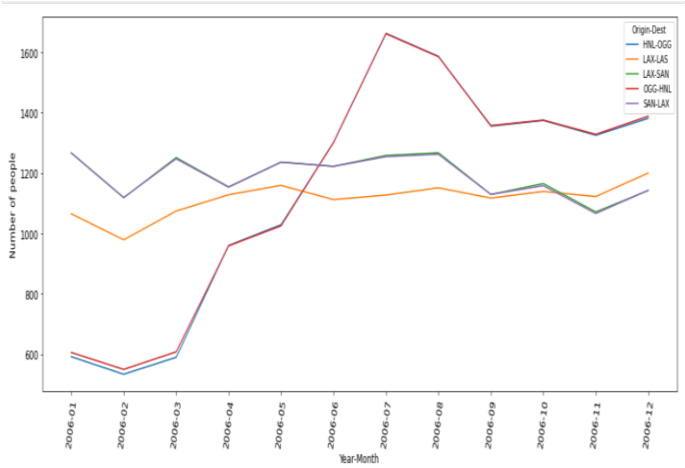


Line plot in R

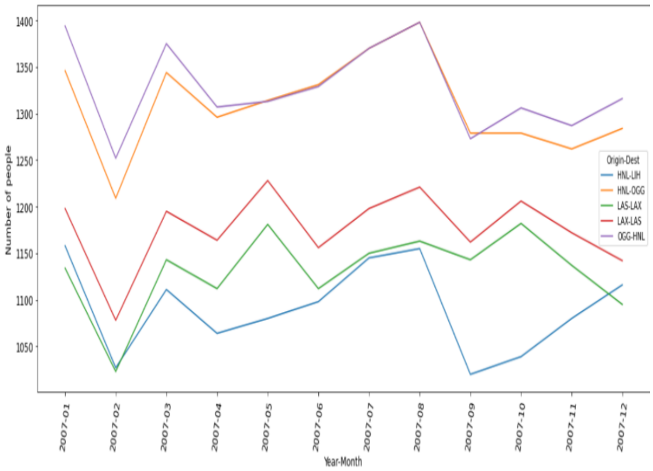
The graphs illustrates that there is a large drop in the number of people flying in February in the year 2006 and 2007. Also, the number of people travelling the most in each year is in the month of July.

The Origin and the Dest column were joined in the format of "Origin-Dest", and the data was separated for the year 2006 and 2007. "Origin-Dest" indicates a particular route. For each year the data was grouped by the Origin-Dest and the size of each group was found indicating the number of people flying in that particular route (Origin-Dest). The Origin-Dest(route) containing the highest five number of people flying were considered for further analysis.

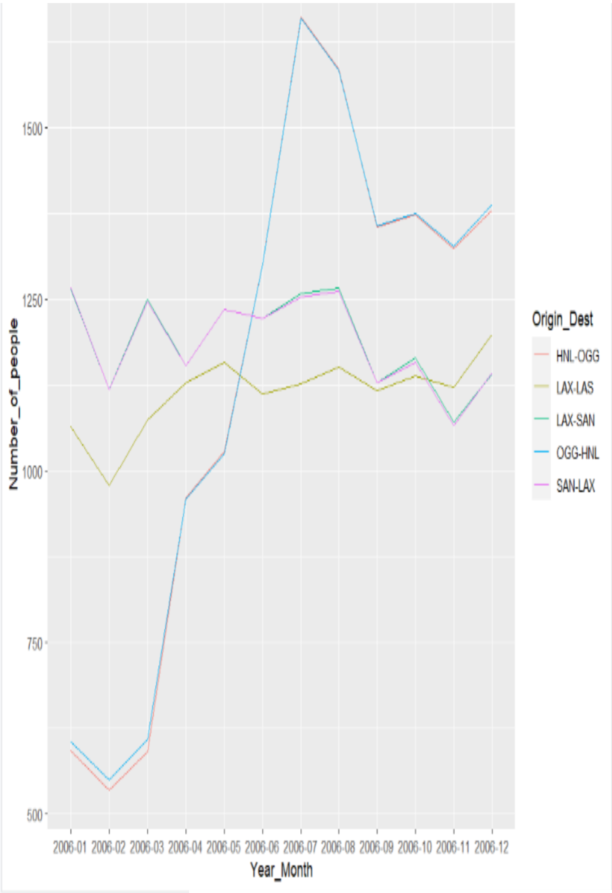
A line plot was plotted to show the relationship between the number of people flying and the Month based on the year and the Origin-Dest selected.



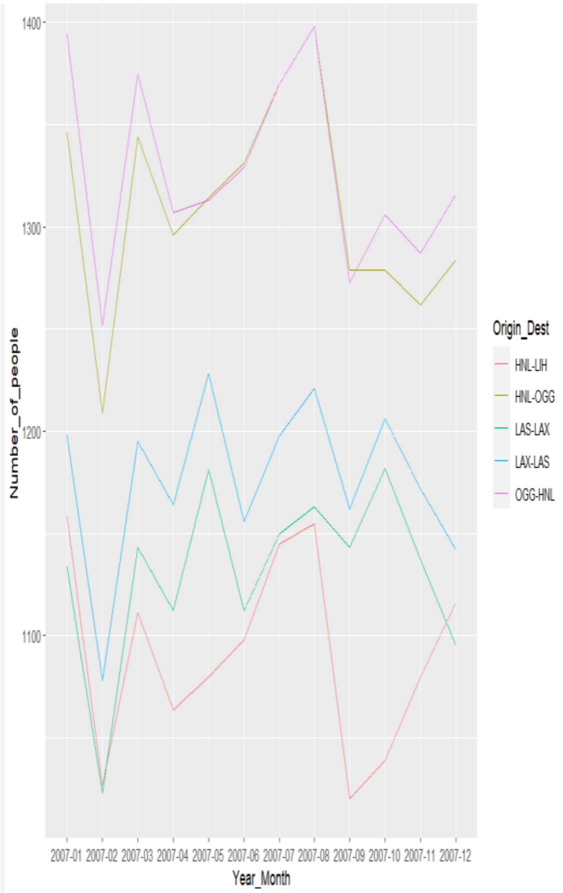
2006 lineplot in Python



2007 lineplot in Python



2006 Line plot in R



2007 Line plot in R

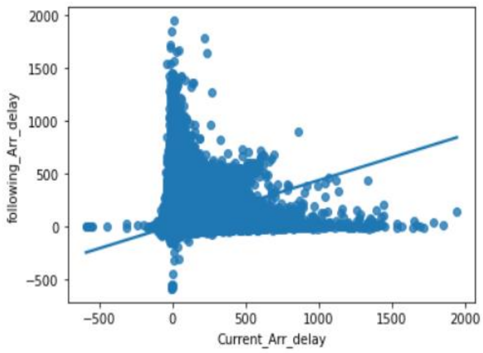
The graph clearly depicts that the Origin-Dest with the highest five number of people flying for 2006 are HNL-OGG, LAX-LAS, LAX-SAN, OGG-HNL and SAN-LAX. For 2007 its HNL-LIH, HNL-OGG, LAS-LAX, LAX-LAS and OGG-HNL.

Based on the graph, it is noticeable that at the end of the year in 2006 the most frequently used route is OGG-HNL. However, in 2007 the route which was frequently used overall was OGG-HNL. Based on the Origin-Dest selected, for 2007 the HNL-LIH route was used the least. For 2006, the least used route was initially HNL-OGG and OGG-HNL, but after the mid of the year it became the most frequently used route.

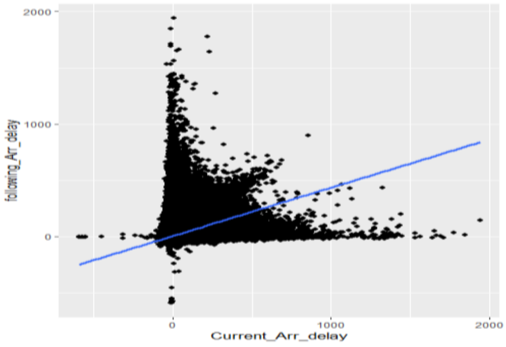
Question 4: Can you detect cascading failures as delays in one airport create delays in other airport?

The cleaned_dataset.csv dataset was used for this question and the DepDelay and ArrDelay were used in terms of the delays. A DateTime column (Timestamp) was made using the Date and the DepTime. The values were then sorted based on the TailNum and the DateTime column. Furthermore, the rows with TailNum = 0 were removed. Rows were filtered out so that the Dest of a particular flight matched the Origin of the next consecutive flight and the TailNum of a particular flight matched the TailNum of the next consecutive flight. A lagged column for the ArrDelay was added so that the ArrDelay for a particular row is considered as the current_Arr_delay and the lagged ArrDelay for the particular row is the following_Arr_delay. Moreover, rows with null values were removed.

A scatter plot was constructed between the current_Arr_delay and the following_Arr_delay and it was found that there is a positive linear relationship between the two with a correlation co-efficient of **0. 4294**. This concludes that the current delay does affect the next flights delay, hence is a cascading failure.

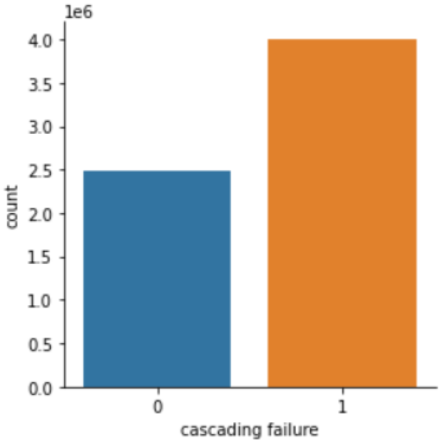


Scatter plot in Python

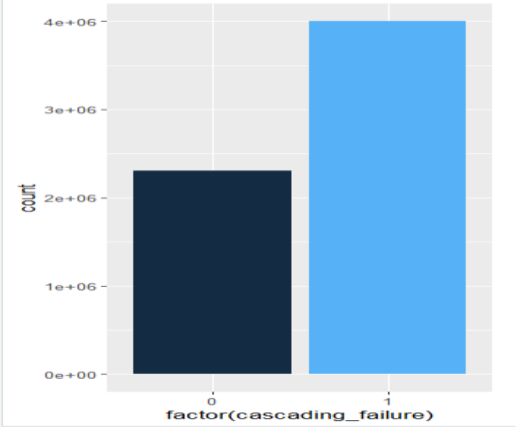


Scatter plot in R

To analyse this further, a catplot was plotted and rows which had a current_Arr_delay greater than zero was used in this analysis. An assumption was made, which is if the current_Arr_delay is greater than zero and the following_Arr_delay was greater than zero for a particular row it was considered as a cascading failure. A value of **1** was indicated if it was a cascading failure, if not, a value of **0** was indicated.



Catplot in Python



Catplot in R

The graphs depicts that more proportion of the flights suffered from cascading failure. This concludes that most of the planes which had a current_Arr_delay greater than zero caused the following flight to also have a delay greater than zero. Hence, this further proves that the delays in one airport creates delays in other airports resulting in a cascading failure.

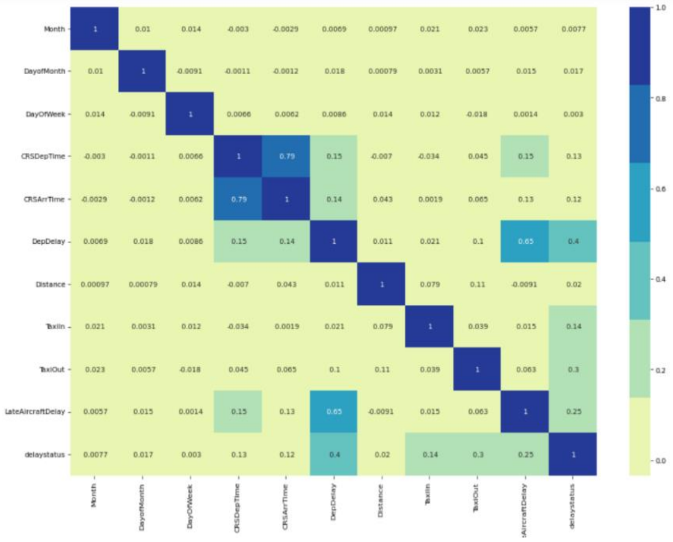
Question 5: Use the available variables to construct a model to predict the delays.

In this question, the analysis predicts the occurrence or non-occurrence of a departure delay using a classification logistic regression. The delay which is used in this regression is the ArrDelay and a new column called the delaystatus was added which was calculated from the ArrDelay. If the ArrDelay was greater than zero, the delaystatus was assigned a value of **1**, if not greater than zero, a value of **0** was assigned. Hence, the delaystatus is a binary column and will be the dependent categorical variable for this model. The count of the occurrence of **1** and **0** was similar (for **0**-*7626703*, for **1** – *6642225*), hence the data can be considered as not being imbalanced.

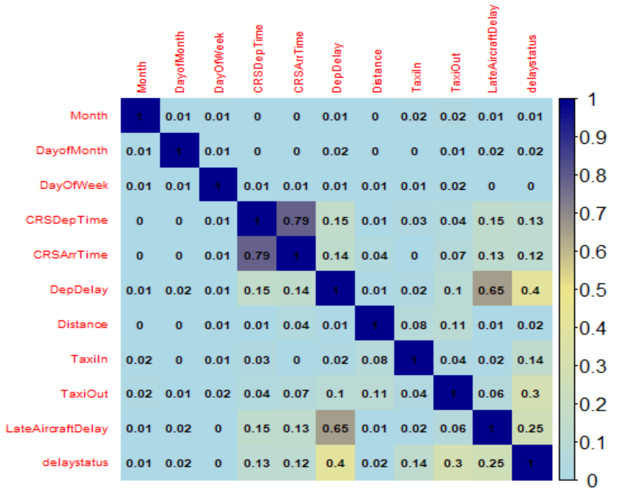
In a regression, the dependent variable must be highly correlated with the explanatory variables, however, the correlation within the explanatory variables must be low. Considering this, features such as UniqueCarrier ,FlightNum ,TailNum , Cancelled , Diverted , Year , DepTime , ArrTime ,Origin , Dest , ActualElapsedTime and AirTime,CRSElapsedTime were removed. The WeatherDelay, NASDelay, CarrierDelay , and SecurityDelay were also removed as it would not be known until the plane lands in the destination airport.

The *cleaned_dataset.csv* dataset is used in this question.

Using the remaining features, a correlation heat map was plotted to check for variables which were significantly correlated to the delaystatus.



Correlation heatmap in Python



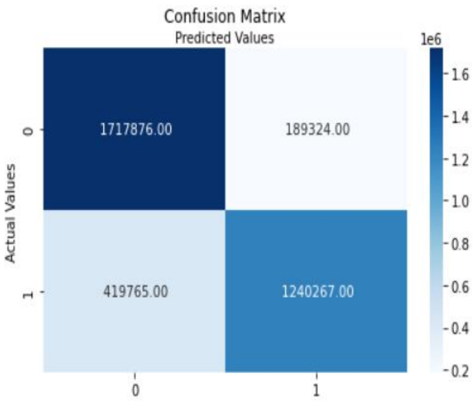
Correlation heatmap in R

For the model if the correlation between the delaystatus and a feature was more than 0.1, the feature is considered as being significantly correlated with the delaystatus. Therefore, it was used in the model for prediction as an explanatory variable.

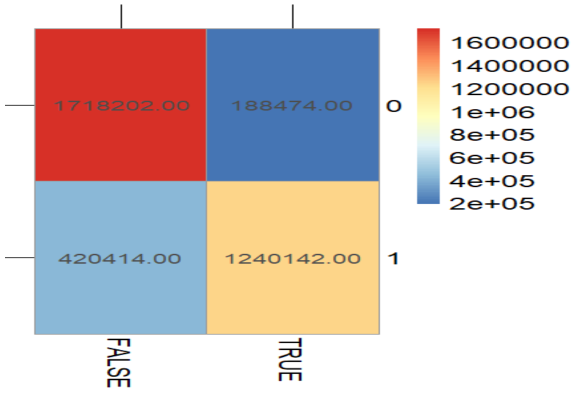
Developing the Model and prediction:

To build the model, the data was split into test data and train data. Where the set for training was allocated **0.75** of the data and the set for testing was allocated **0.25** of the data. Both the test set and the train set are mutually exclusive to obtain a reliable value for the test error. In order to bring data points close together and to create a better frame of reference when training the model, the data was scaled using the standard scaler. Furthermore, a specific code was used to ensure that the same random values were generated if the code was run again.

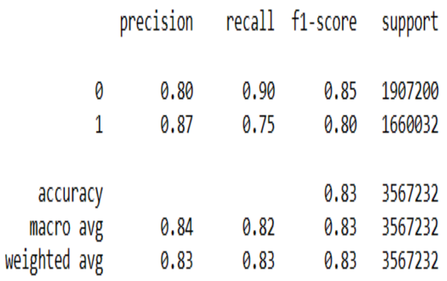
A confusion matrix which summarizes the prediction results based on a specific classification problem (which in this case is: Delays (**1**) or No Delays (**0**)) was used for this model.



Confusion Matrix in Python



Confusion matrix in R



```
              precision    recall  f1-score   support

           0       0.80      0.90      0.85   1907200
           1       0.87      0.75      0.80   1660032

    accuracy                           0.83   3567232
   macro avg       0.84      0.82      0.83   3567232
weighted avg       0.83      0.83      0.83   3567232
```

Classification report in Python

Accuracy- (1240142+1718202)/ (1240142+1718202+188474 +420414) =**0.829**

Recall – 1240142/ (1240142+420414) = **0.747**

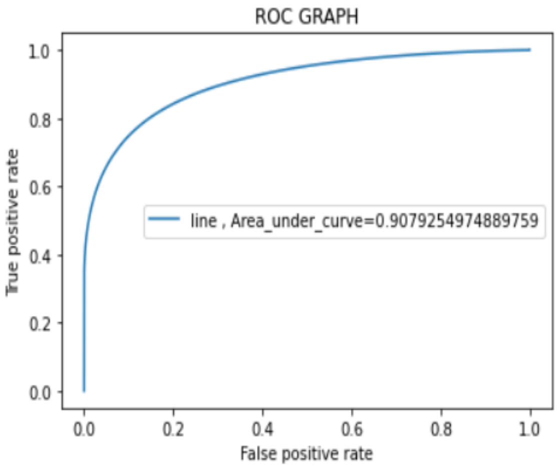
Precision - 1240142/ (1240142 + 188474) = **0.868**

Manually constructed classification report for R

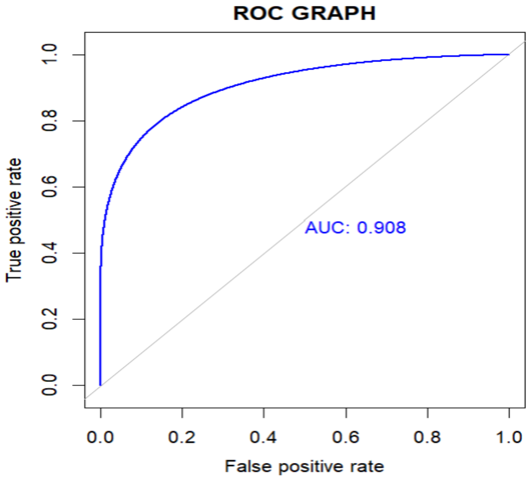
It is noteworthy that the random values generated for R and python are different. This changes the values in the confusion matrix although it is the same dependent variable and the explanatory variable.

The above graphs illustrates that the accuracy score obtained in python from the sklearn metrics and for R manually calculated is **0.83.** Hence, it supports the model being accurate as it is close to 1.

To check the performance of the binary classification model, which in this case is the delaystatus, a ROC curve was plotted, and the AUC (area under the curve) was obtained. Higher the AUC, better the model as the model is better in predicting the value **0** if the true value is **0** and **1** if the true value **1.**



ROC and AUC in Python



ROC and AUC in R

From the above visualization, it is visible that the AUC (area under the curve) in Python and R is **0.91** rounded to two decimal places (almost close to 1). This indicates that the model is good at predicting delays, which means that, mostly the true values have been predicted as true and the false values have been predicted as false.

The ROC curve of a 45-degree line is often taken as the baseline performance of a model as it gets absolutely nothing from the features. It is noteworthy that the graph obtained is above the 45-degree line. This supports that the model has good performance.

In conclusion, the model used to predict the occurrence of delay is good as it has satisfied most of the conditions required to be present in a good model.