Report

Gabriele Matini 1934803

1 Exploratory Data Analysis

Our final objective is to predict accurately if a certain flight will have a delay bigger than 15 minutes or not, thus any preliminary analysis will be conducted with this motivation in mind. In particular we will be distinguishing between two types of data: the time data, that tells us information about the timing of the flights (expected arrival time, actual arrival time, etc...), and other data that may or may not have important correlations with the delays (weather conditions, the airline, number of miles...). In total the dataset is composed of exactly 3 million data points. In the study many graphs have been plotted, not all of them are available here for the sake of brevity, but they are available in the EDA_PLOTS folder. Looking for the correlations, we will talk about only those that were found to be at least moderately strong (more than 0.5 or less than -0.5), and that we think will help in our task. However, we also report here the calculated correlation matrix to give a complete view:

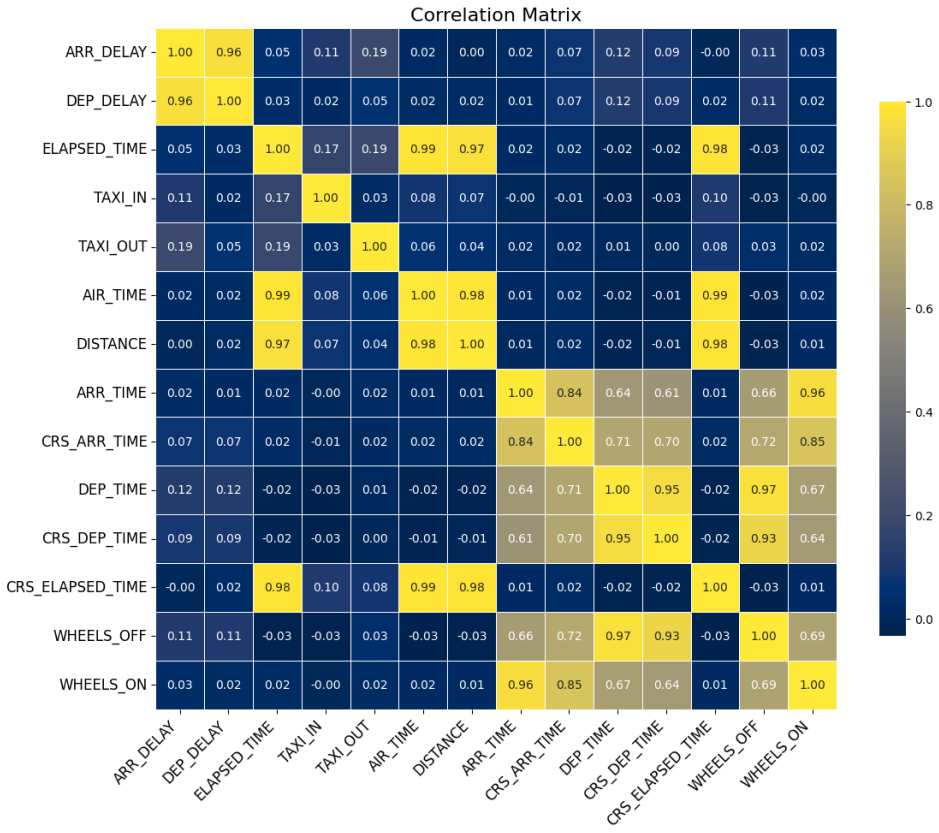


Figure 1: We can see the strong correlations in yellow.

1.1 The time related data

First let us list all of these information fields, many of them give us more information on the nature of the delay.

1. CRS_DEP_TIME, DEP_TIME, DEP_DELAY: these are scheduled departure times, actual departure times and departure delays in this order. From the dataset description we also see that the mean departure delay is ~ 10.12, with a standard deviation of ~ 49.25. A high correlation between DEP_TIME and CRS_DEP_TIME (~ 0.95) indicates that usually flights do not have much trouble departing in time.

2. CRS_ARR_TIME, ARR_TIME, ARR_DELAY: scheduled arrival time, actual arrival time, arrival delay. Notice that the arrival delay is essentially what the model will try to predict, in particular if this delay is bigger or smaller than 15 minutes. From the dataset description we see that the mean arrival delay is ~ 4.26, wile the standard deviation is ~ 51.17. The correlation between CRS_ARR_TIME and ARR_TIME is high, but it is lower than that for the departure (~ 0.84), indicating that most of the delay happens after the departure. Note that, both departure and arrival delays are not necessarily the subtraction of scheduled from actual time, as the dictionary.html file would suggest, as some flights can be delayed by days. Following, there are the delays distributions:

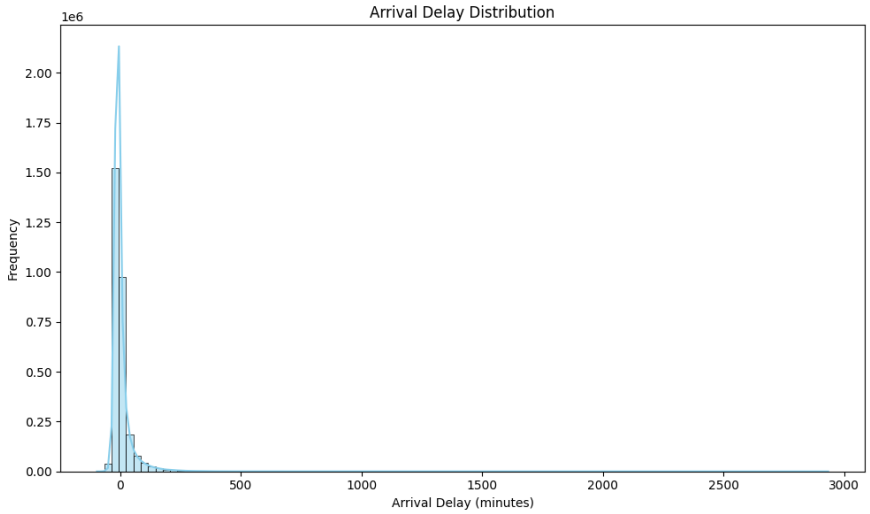


Figure 2: Distribution of arrival delays. The majority of values are centered around 0.

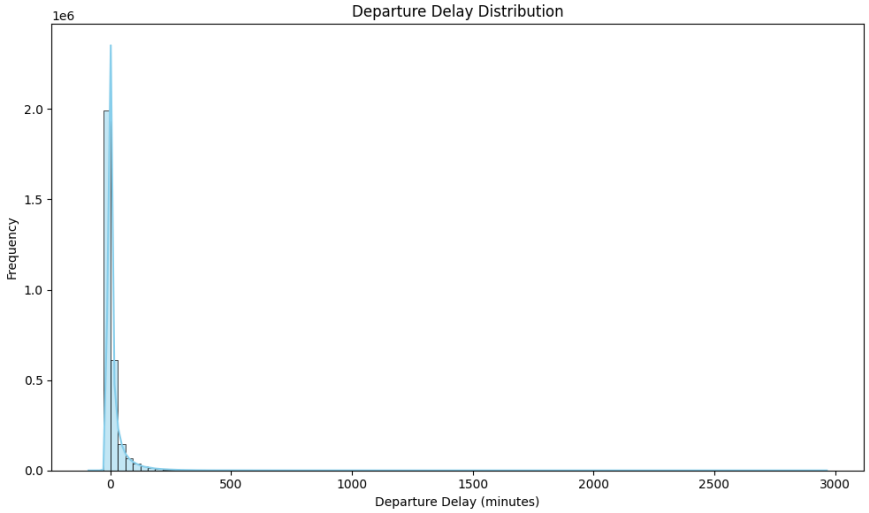


Figure 3: Departure delay distribution. Again, the points seem to concentrate around 0. The distribution is similar to the arrival delay distribution, hinting to a strong correlation between the two.

DEP_DELAY and ARR_DELAY have a huge correlation (~ 0.96), which is expected.

3. CRS_ELAPSED_TIME, ELAPSED_TIME: scheduled elapsed time from departure to arrival, actual elapsed time from departure to arrival. No correlation between elapsed time and departure delay was found. The correlation between estimated (CRS) elapsed time and real elapsed time is ~ 0.98, proving that usually the planes manage to do the flight itself almost without delays (if the flight itself takes a scheduled time of 1 hour, it will probably be 1 hour, although other reasons might cause a delay).

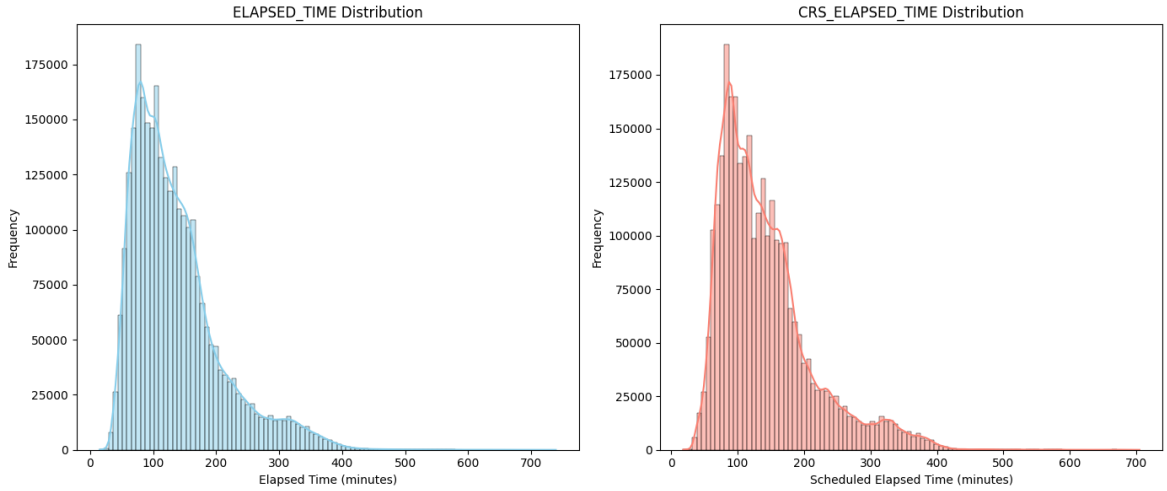


Figure 4: The high correlation shows that most of the times a flight does not encounter many problems to comply with its scheduled elapsed time.

4. AIR_TIME, TAXI_IN, WHEELS_OFF, WHEELS_ON, TAXI_OUT: time spent in the air, time spent driving on the ground before taking off, the exact moment at which the plane leaves takes off, exact moment the plane touches ground and time spent driving on the ground on arrival. Particularly, the WHEELS_OFF and WHEELS_ON times are not very useful, and as such will be eliminated. A suspiciously high correlation was found between AIR_TIME and ELAPSED_TIME (~ 0.987). Later, we will see that indeed the AIR_TIME is the main component constituting the total ELAPSED_TIME.

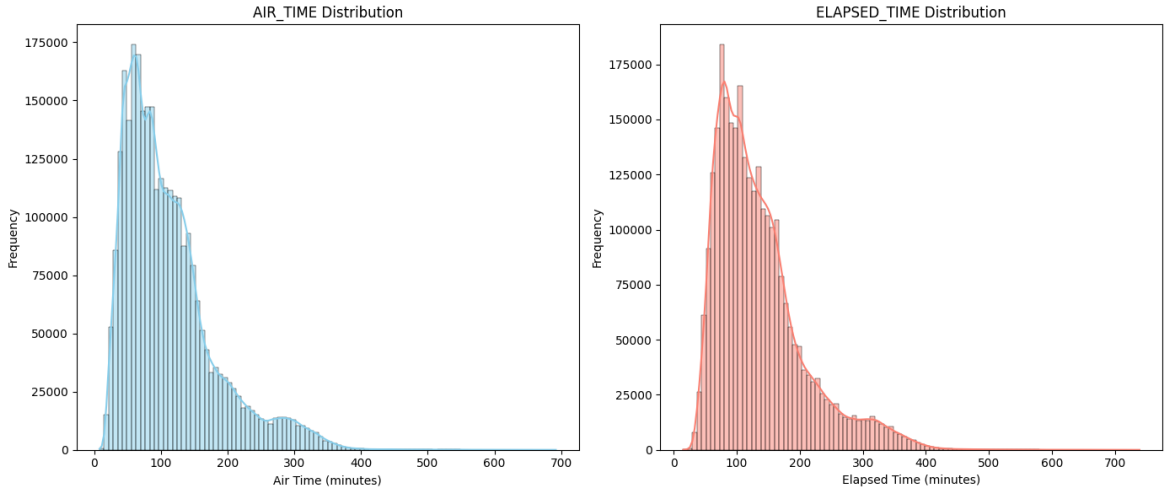


Figure 5: Air time and Elapsed time distributions. The comparison highlits the reason there is such a high correlation.

5. DELAY_DUE_CARRIER, DELAY_DUE_WEATHER, DELAY_DUE_NAS, DELAY_DUE_SECURITY, DELAY_DUE_LATE_AIRCRAFT: various amounts of delays. These values are missing for most of the rows.

First, let's analyze the relationships between these time data fields. We proceed by testing some hypotheses and finding if they indeed hold true withing the dataset, while other facts can be inferred just by looking at the rows of the dataset:

1. First, as stated by the dictionary.html file, the delays are just the difference between the scheduled time and the actual time. However, this does not mean that, given ARR_TIME, CRS_ARR_TIME

and others we are able to infer the ARR_DELAY field: some flights may be even delayed for more than two days, and there is no way to know if the other time fields are relative to the same day, the next day, or n days in advance compared to the scheduled departure day. It is in fact very easy to see that there are delays in the dataset of more than 2000 minutes, so more than 1 day has passed for the delay.

2. The WHEELS_OFF time is calculated by adding the TAXI_OUT minutes to the actual DEP_TIME. Similarly, the WHEELS_ON time is calculated by removing the TAXI_IN minutes from the actual ARR_TIME.

3. The ELAPSED_TIME is nothing more than the AIR_TIME + TAXI_IN + TAXI_OUT. This holds true for all rows apart from just 3 (out of 3 million). We conclude these are anomalous rows and remove them.

4. ELAPSED_TIME and DEP_DELAY and ARRIVAL_DELAY do not contribute to measuring the same things, since the elapsed times is sometimes bigger or smaller than what would be the difference in minutes between ARR and DEP.

5. The "DELAY_DUE" fields give occasional information about the type of delay, although only 533863 rows have these information. Also, they do not necessarily make up the total delay (departure and arrival), so they only provide partial info about any delays. They also come all together, that means that these "DELAY_DUE" fields are either all present, or non of them is present. Since they provide too few information, there is no way to infer these values when not present, and since less than a quarter of these points actually have these values, I decided to remove these features, as they would probably just slow down the training without providing much information to the ML model and possibly also act as noise.

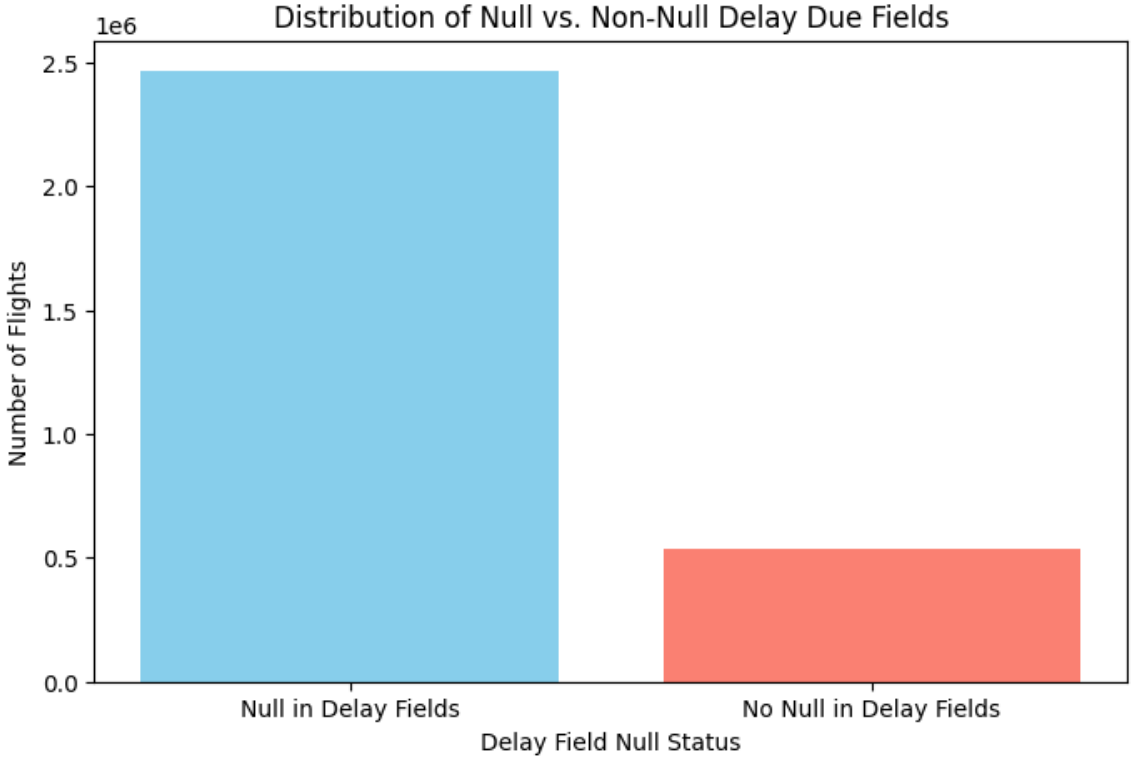


Figure 6: The amount of data points having delay due values with respect to those who don't: as we can see less than a quarter of the points actually has those values: these are around 533863 out of 3 million. It needs to be stressed that all the blue column data points do not have any of the "delay due" values.

6. Further notes: the CRS fields are always present, and when the other time values are missing they are always missing because the flight was either canceled or diverted. There are only two rows out of 3 million that escape this rule and we have to conclude they are some kind of anomaly and will be eliminated.

We then enumerate all the rows that are missing any of the above fields (apart from the DELAY_DUE fields), for a total of 86198 points. Following, there is a number of facts discovered on these points:

1. Most of these flights were canceled, precisely 79140. These flights obviously never provide any information on delays, and as such the chosen approach is to remove them from the dataset.

2. The remaining 7056 flights were all diverted, they are all missing information on elapsed time and on the arrival delays, but some do have information on scheduled arrival and actual arrival times. The adopted approach is to eliminate all of these tuples, since, as seen earlier, there is no way to understand if the hhmm fields (ARR_TIME, CRS_ARR_TIME,...) refer to the same day, so there is no way to infer the actual arrival delay. It's interesting to note that these points would be a type of outliers anyways: the delays that would be calculated, even with the most charitable interpretation (same day or next day assumption), is most of the time massive compared to other points.

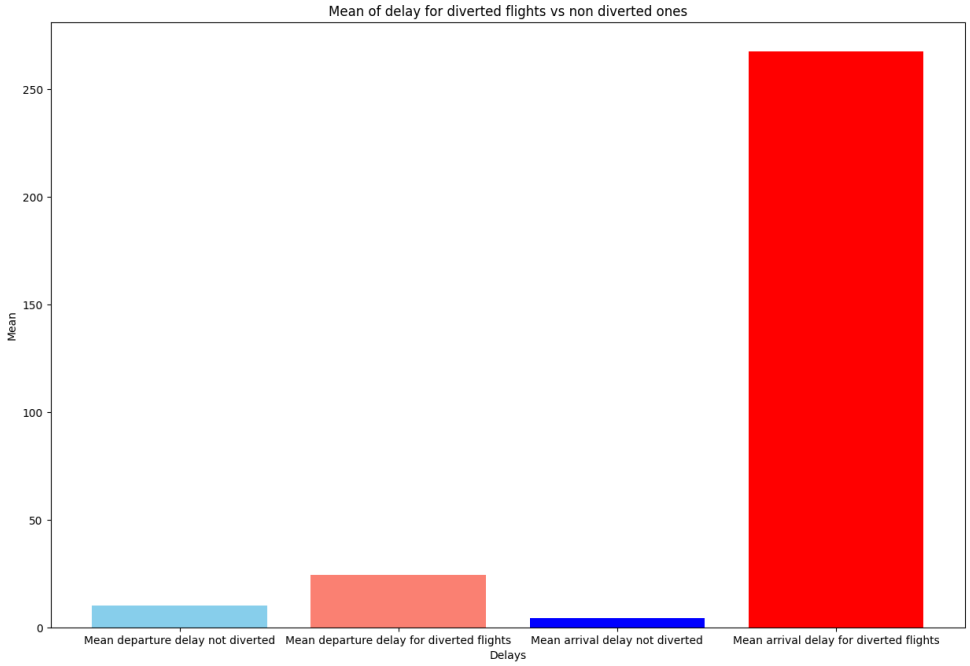


Figure 7: The mean of delays for diverted flights vs non diverted flights, where the delays for the diverted flights were estimated assuming that the ARR_TIME and CSR_ARR_TIME are distant at most 1 day. As demonstrated here, diverted flights experience on average more delay overall and a huge amount of arrival delay. Based on this distribution, we can expect any classification model to predict that a diverted flight will almost always be delayed more than 15 minutes. It's important to note that these means are calculated out of a wildly different amount of diverted flights vs non diverted (non diverted are many more in orders of magnitude).

1.2 The other fields

The other fields are:

1. FL_DATE, FL_NUMBER: date and identifier number of flight. The flight date is not useful to infer anything about the delay, and so it will be eliminated.
2. AIRLINE, AIRLINE_DOT: readable versions of the AIRLINE_CODE and DOT_CODE. These readable versions are not useful for our ML purposes, so they will be removed.
3. AIRLINE_CODE, DOT_CODE: unique carrier code, unique airline code. These two fields are contained in each other, so for any value of AIRLINE_CODE, there is the correspondent value of DOT_CODE and vice-versa.
4. ORIGIN, ORIGIN_CITY: origin airport and origin airport with city name. We remove ORIGIN_CITY, since the flights themselves are airport-to-airport.
5. DEST, DEST_CITY: destination airport and destination with city name. We remove DEST_CITY, since the flights themselves are airport-to-airport.
6. DISTANCE: distance in miles of the flight. As one would expect, there is a huge correlation between flight time and distance (~ 0.98).

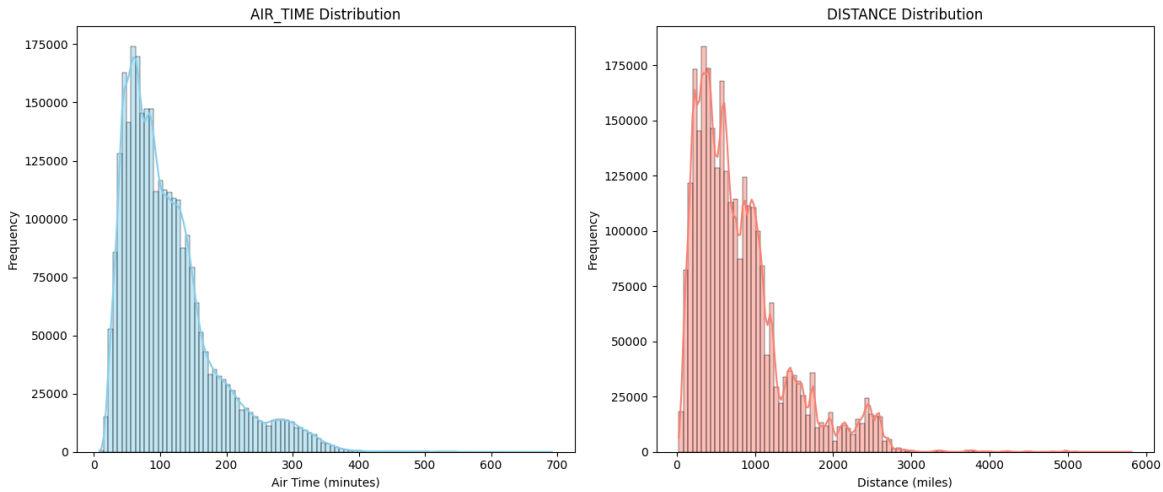


Figure 8: Distributions of air time and distance, showing their correlations. This is probably the most obvious correlation, and it gives also information on how many miles a plane usually does in what amount of time (for example, the distribution tells us that a plane can do 1000 miles in around 100 minutes).

7. CANCELLED, CANCELLATION_CODE, DIVERTED: they signify canceled flights and diverted flights. Since we remove those flights, these fields lose their utility and thus they are removed.

A preliminary analysis reveals that no rows are missing in these fields, and that none of these fields constitute by themselves a unique identifier of a row. Furthermore, a study was conducted on the airlines, taking the airline codes, to see if they have any meaningful differences in averages and standard deviation for both departure and arrival delay:

Figure 9: As we can see, there are significant changes, meaning that the airlines might be a predictor of lateness.

Figure 10: Standard deviation values for different airlines. Again we have confirmed the trend that some airlines just seem to have less problems than others.

1.3 Other outliers

Apart from the diverted flights, we analyze the distribution of outliers in ARR_DELAY and their percentage with respect to all the points in the dataset:

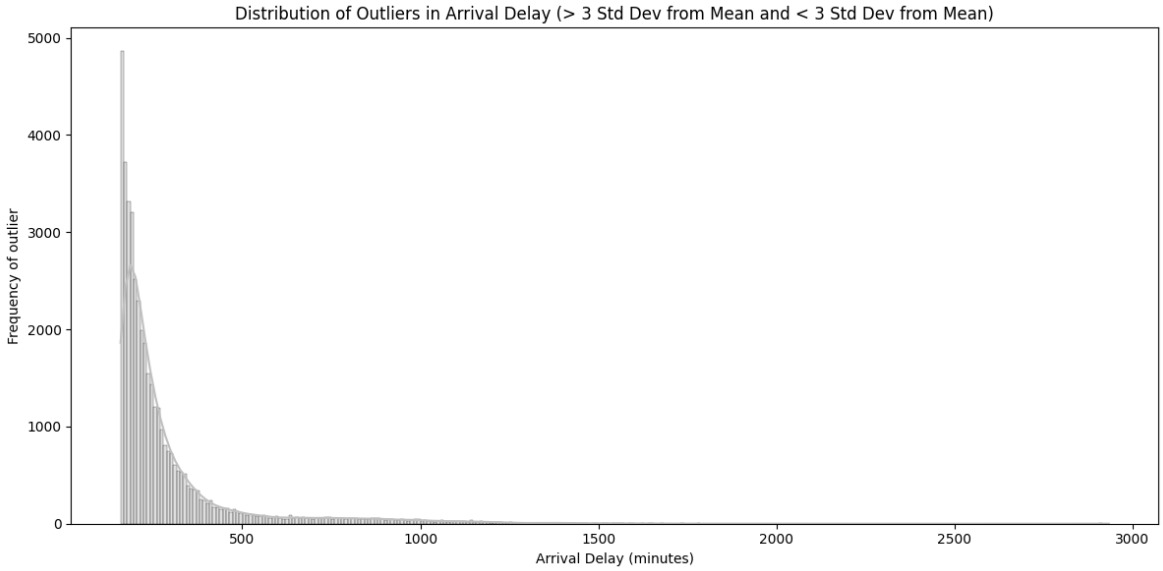


Figure 11: The distribution of outliers reveals two important facts: the majority of them concentrates around the mean + 3*stddev value, and there are no outliers that have a value $v \leq mean - 3 * stddev$. This is not surprising, since we can hardly expected a flight to be more than two hours early.

The total number of such outliers is 42370, which constitute $\sim$ 1.45% of all the data points left after removing the NULL delay values.

2 The ML task

Remembering that we are trying to predict whether a flight will be later more than 15 minutes or not, we are trying thus to predict something about the ARR_DELAY field. We decide to construct the vector by encoding (when needed) the AIRLINE_CODE, the ORIGIN and DEST airports. The final vector is assembled by using all the fields we didn't remove until now. Finally, the dataset is expanded with a LABEL column, which takes as value 1 if the flight arrival delay is more than 15 minutes, 0 otherwise. It's also important to note that not all screenshots and graphs are present for brevity reasons, but they are all present in the ML_PLOTS folder. Before starting, it needs to be

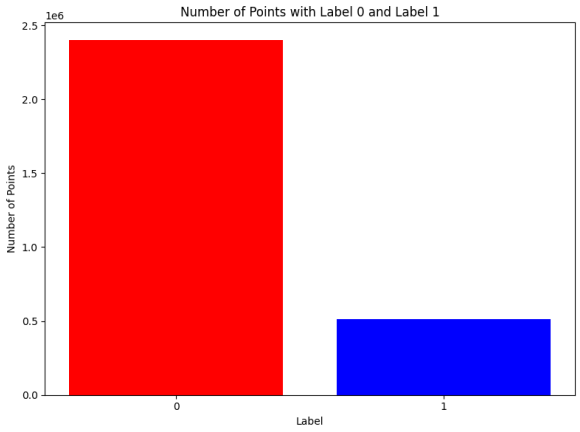


Figure 12: Labels distribution withing the dataset.

explained that an important problem was normalization with the mean put at true, which exploded the feature vector from a sparse vector into a dense one, so the computation time would explode. In the end, I resorted to use the standard scaler with normal rescale. We will see three different ML models, and also analyze the impact of feature normalization on these models. In particular, we use standard scaling, so we subtract the feature by the mean and divide by the standard deviation. For each model, we calculate 5 main parameters, other than plotting the ROC:

1. AUC (Area under curve): this calculates the area under the ROC curve, and it gives a measure of the true positive rate vs the false positive rate.

2. Accuracy: the accuracy is nothing more than the number of true positives and true negatives found divided by all the predictions. Let us call Tp the number of true positives, Tn the number of true negatives, Fp the number of false positives, and Fn the number of false negatives. $Accuracy = \frac{Tp+Tn}{Tp+Fp+Tn+Fn}$

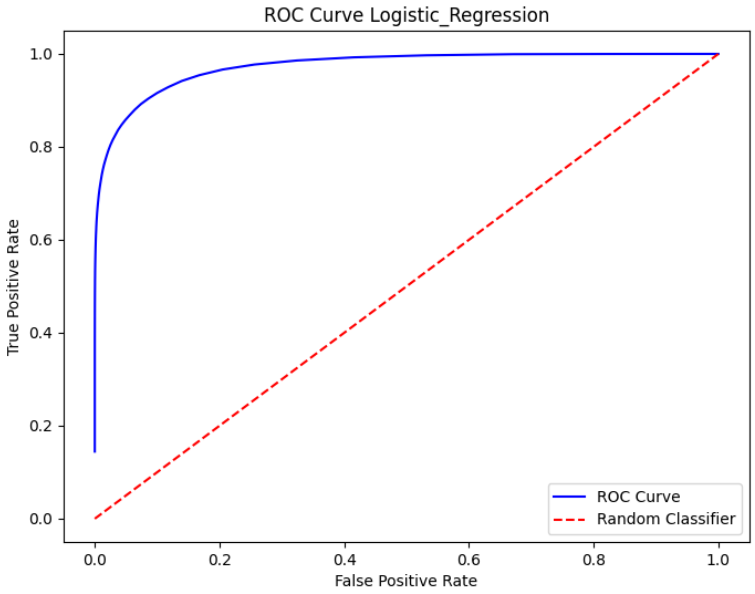
3. Precision: the number of true positives over the number of all positives the model has predicted. $Precision = \frac{Tp}{Tp+Fp}$

4. Recall: the number of true positives over all the positives the model should have predicted. $Recall = \frac{Tp}{Tp+Fn}$

5. F_1 Score: armonic mean of precision and recall. $F_1 = \frac{Precision*Accuracy}{Precision+Accuracy}$

2.1 Logistic Regression

The first model uses logistic regression, which classifies algorithms based on a threshold, using the sigmoid to map any real value to a probability value. It also optimizes a logarithmic loss function value. The used hyper-parameters were a regularization parameter penalties (0.1 and 0.01 were the tried values) and the elastic net parameter to use L1, L2 and a mix of L1 and L2 regularizations.
It's interesting to note that logistic regression does not seem to benefit at all from normalization: the results for all metrics and confusion matrix are exactly the same, as documented in the ML_PLOTS folder.



[h]

Figure 13: Logistic regression ROC curve: as we can see the AUC will be almost 1.

```
Confusion Matrix and metrics:
                  Predicted Positive   Predicted Negative
Actual Positive        57677                 45358
Actual Negative        396                  478878
Logistic_Regression AUC: 0.9734555322482553
Logistic_Regression F1 Score: 0.9122226991203951
Logistic_Regression Precision: 0.9275807357438357
Logistic_Regression Recall: 0.9214265965320817
Logistic_Regression Accuracy: 0.9214265965320818
```

Figure 14: The confusion matrix reveals the weaknesses of the logistic regression: while it does a good job of predicting the true negatives (thus it has a low rate of false positives), it does a poor job when it comes to predict positives, since almost half of the actual positives are predicted negatives.

2.2 Random Forest

The second model is a random forest, which takes the majority result out of a series of decision trees as classification result. The used hyper-parameters are the number of trees (10 and 20) and the maximum tree depth (5, 10, 15 splits). As we can see, performances are worse, however, things improve by normalizing the data:

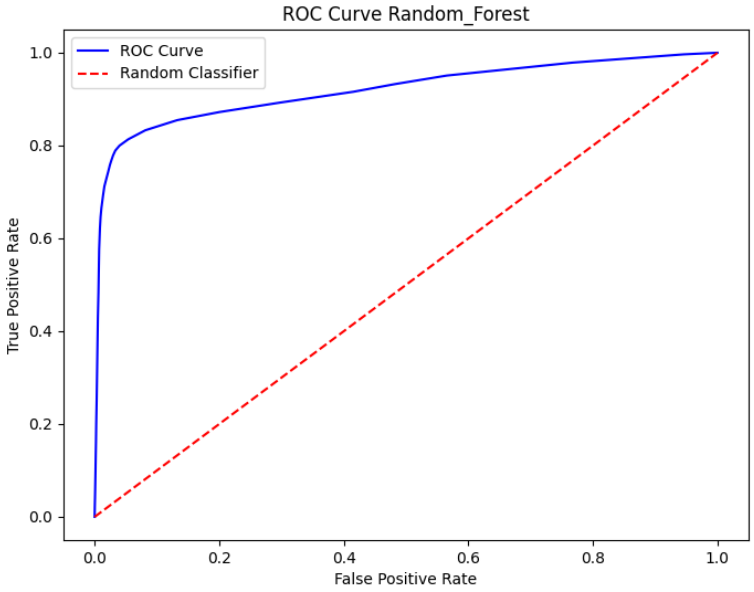


Figure 15: The random forest does a poorer job than the logistic regression in either predicting true negatives or true positives, thus resulting in a worse AUC.

```
Confusion Matrix and metrics:
                 Predicted Positive   Predicted Negative
Actual Positive        350                 102685
Actual Negative        19                  479255
Random_Forest AUC: 0.9200799435011262
Random_Forest F1 Score: 0.7446003680150899
Random_Forest Precision: 0.8456582125945313
Random_Forest Recall: 0.8236262877613088
Random_Forest Accuracy: 0.8236262877613089
```

Figure 16: The confusion matrix reveals that the random forest has a harder time predicting the true positives than Logistic Regression, resulting in worse performances.

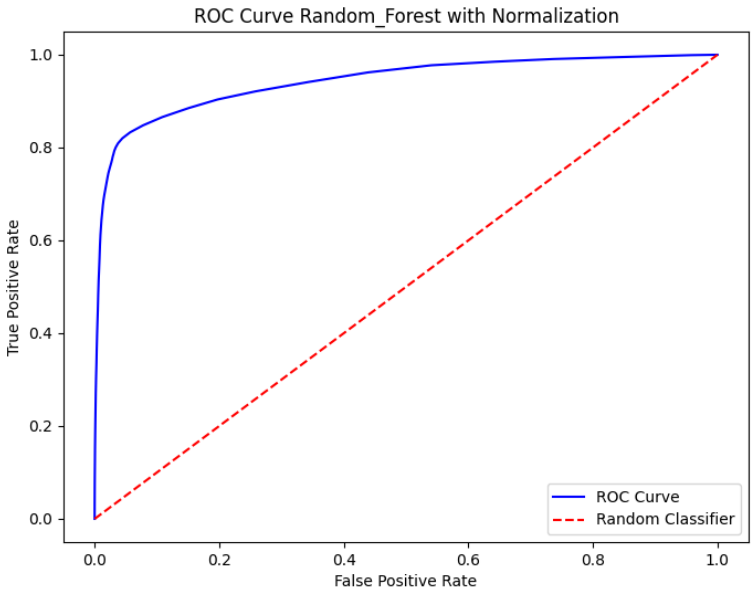


Figure 17: Normalization slightly improved the AUC, as the ROC bends better.

```
Confusion Matrix and metrics with Normalization:
                  Predicted Positive   Predicted Negative
Actual Positive        41004                 62031
Actual Negative        1888                 477386
Random_Forest AUC: 0.9429915733384889
Random_Forest F1 Score: 0.8708519751757811
Random_Forest Precision: 0.8975627726399207
Random_Forest Recall: 0.8902318185018607
Random_Forest Accuracy: 0.8902318185018607
```

Figure 18: As we can see, normalization helped a lot with true positives prediction, improving overall performances on all metrics.

2.3 Gradient Boosted Trees

Since random forests underperformed with respect to the other model, it felt appropriate to try a more advanced technique with respect to trees usage: gradient boosted trees are a series of trees built sequentially, each one trying to minimize the error of the previous tree. This effectively creates a loop where each prediction gets updated using the last prediction, and a decision tree is fit to the residuals that represent the error between the last prediction and the target value. The used hyper-parameters were the depth (10) of the trees and the maximum number of iteration (10). An important specification is that this model gave problems because training took too long, so the hyper-parameters were chosen with the idea of training a more lightweight model. In any case, the GBT model still took a long time. This time, normalization of data was applied (since it did benefit the random forest, it will probably benefit the GBT model too).

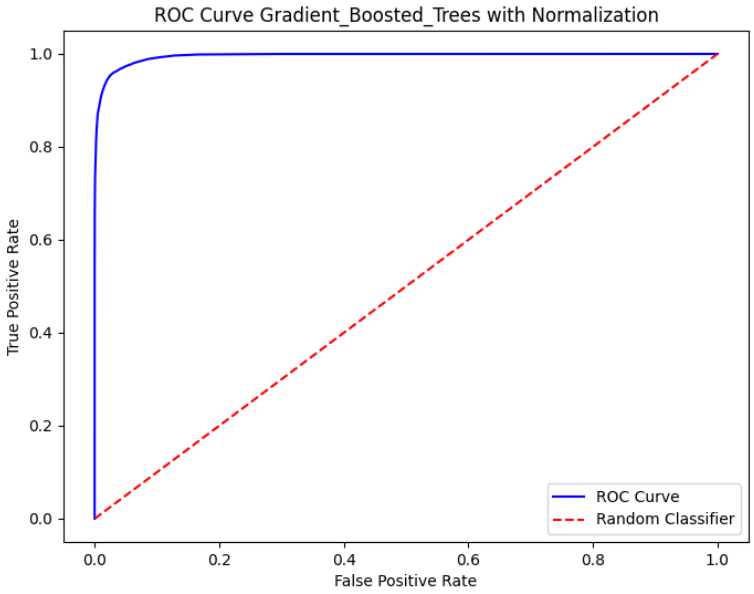


Figure 19: The gradient boosted trees technique awards us an even better result than logistic regression, the AUC being nearly 1.

```
Confusion Matrix and metrics with Normalization:
                  Predicted Positive   Predicted Negative
Actual Positive        92956                 10079
Actual Negative        4478                 474796
Gradient_Boosted_Trees AUC: 0.995466829655378
Gradient_Boosted_Trees F1 Score: 0.974724631146911
Gradient_Boosted_Trees Precision: 0.9747591205728366
Gradient_Boosted_Trees Recall: 0.9750012450434391
Gradient_Boosted_Trees Accuracy: 0.9750012450434391
```

Figure 20: Even if slightly worse at recognizing true negatives, the model has become incredibly accurate in recognizing the true positives, resulting in better performances overall.

2.4 Multilayer Perceptron

The MLP training came last, since it took too long to train and there was no time, only 1 epoch on the entire dataset was achieved. However, the code is the same as the other models, and the idea is to have a single layer NN with all the needed input neurons (784, 1 per feature), 32 neurons in the hidden

layer and two neurons in the output layer, which represent the probabilities and are thus needed for the binary classification. Only one epoch didn't give exciting results, but we are able to report them nonetheless:

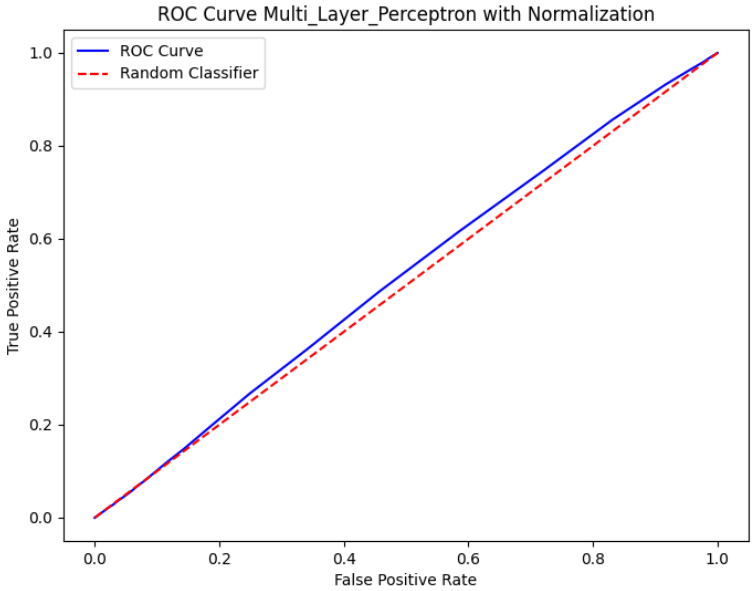


Figure 21: If nothing else, we can see that a neural network is far from a lightweight model from this task: the MLP trains each epoch on the entire train set, for a total of around 2 hours of training. Compare that with Logistic Regression, that took less than two hours to complete, we get the picture that NN may not be the fastest option, training-wise.

```
Confusion Matrix and metrics with Normalization:
                    Predicted Positive   Predicted Negative
Actual Positive        228                 102807
Actual Negative        1325                 477949
Multi_Layer_Perceptron AUC: 0.4996661053999717
Multi_Layer_Perceptron F1 Score: 0.7429762700859456
Multi_Layer_Perceptron Precision: 0.7033352684988071
Multi_Layer_Perceptron Recall: 0.8211739815115343
Multi_Layer_Perceptron Accuracy: 0.8211739815115342
```

Figure 22: Again, the data is not able to tell us much about the performance, but does provide us information on the fact that it takes many more times to train the MLP on the train set. Interestingly enough, the confusion matrix tells us that the MLP achieves performances similar to the random forest without normalization, hinting at some potential of becoming a very precise model in the future epochs.

2.5 Conclusions

The study needs more time, both to assess anything more on the MLP architecture, and to study how normalization impacts the GBT model. However, we can conclude that, between logistic regression, random forest and gradient boosted trees, the latter is the best option, while the second is the worst performing one among all five metrics. The MLP architecture would also probably be able to perform quite well, given enough epochs and the possibility of trying different architectures and choosing the best one, so it has the most potential. These would definitely be the future improvements and projects to realize for the project.