Problem 3 Report

Davide Fortunato 1936575

November 2024

1 EDA and features selection

The first step was about understanding the dataset so I needed key statistics to uncover potential relationships between the variables and to gain insights into their distributions. I started by generating summary statistics using the PySpark method `describe()`, which provided an overview of the numerical attributes, as shown in Figure 1.

summary	Departure delay (Minutes)	Arrival delay (Minutes)	CRS elapsed time (Minutes)	Elapsed time (Minutes)	Distance (Miles)	Taxi in (Minutes)	Taxi out (Minutes)	Air time (Minutes)	Delay due to weather (Minutes)
count	2922356.0	2913802.0	2999986.0	2913802.0	3000000.0	2920056.0	2921194.0	2913802.0	533863.0
mean	10.1233	4.2609	142.2758	136.6205	809.3616	7.679	16.643	112.3108	3.9853
stddev	49.2518	51.1748	71.5567	71.6758	587.8939	6.2696	9.1929	69.7548	32.4108
min	-90.0	-96.0	1.0	15.0	29.0	1.0	1.0	8.0	0.0
max	2966.0	2934.0	705.0	739.0	5812.0	249.0	184.0	692.0	1653.0

Figure 1: Summary statistics for numerical attributes

Next, I visualized the data through various plots to deepen my understanding. One of the most useful visualizations was the correlation matrix, which helped me identify relationships between different variables and uncover any unexpected correlations. Additionally, I plotted the bar chart of average delays per airline and a chart displaying the top 20 routes based on average arrival delays. These visualizations are shown in Figures 2, 3, and 4.

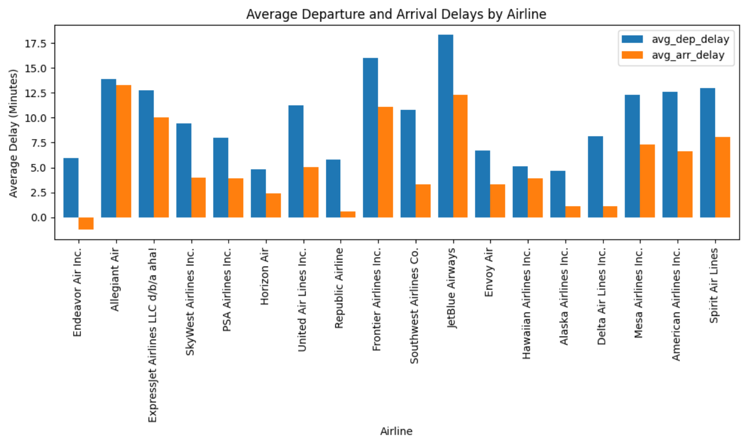

Figure 2: Average departure and arrival delays by Airline

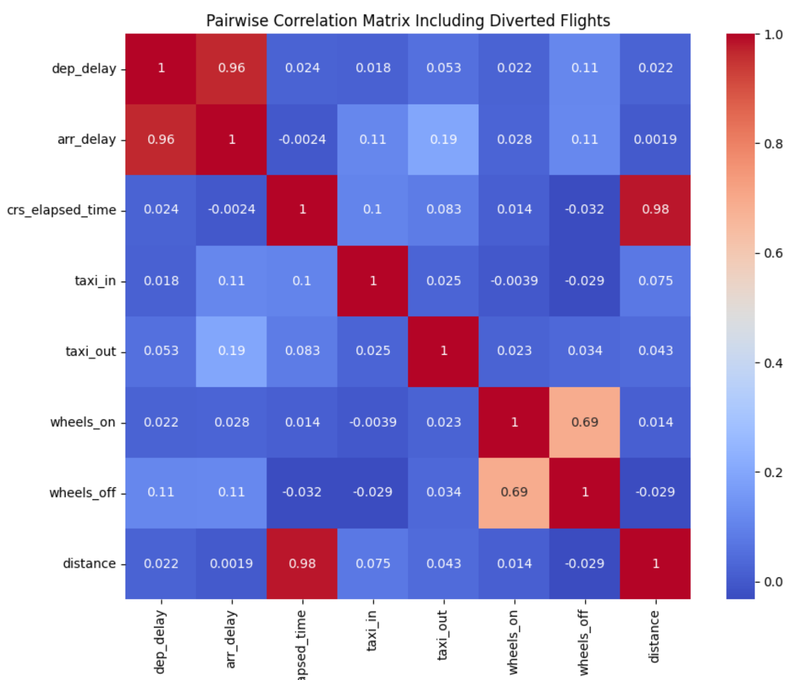

Figure 3: Pairwise correlation matrix

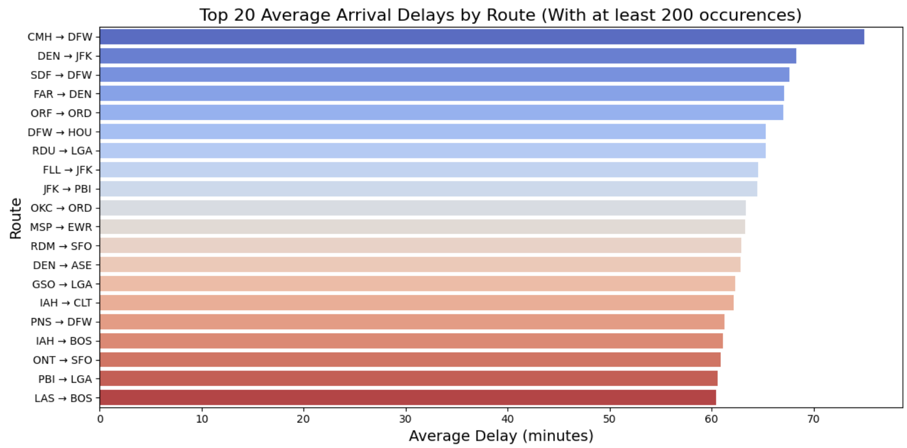

Figure 4: Average arrival delays by Route

The goal of these analyses was to determine the most relevant features for predicting flight delays. I experimented with different combinations of features, but the most general one is: [dep_time, dep_delay, taxi_out, wheels_off, distance, airline_vec, origin_vec, dest_vec, dot_code, crs_elapsed_time, diverted]. I decided to include taxi_out and wheels_off varibles thanks to the correlation matrix which highlighted a slight correlation with the arr_delay variable. The features ending with vec are categorical variables encoded as numerical vectors, enabling them to be processed by machine learning models. To ensure clean data, I removed any rows containing null values in one of the feature columns.

2 Model building

To address the task requirements, I developed two models: a Logistic Regression model and a Random Forest model. For hyperparameter tuning, I utilized the `ParamGridBuilder` as recommended, constructing grids to explore a range of parameter combinations. Additionally, I implemented a 5-fold cross-validation strategy to ensure robust evaluation of model performance across different data splits. In this section, I will detail the hyperparameter combinations evaluated for each model and the corresponding results obtained during the tuning process.

2.1 Logistic Regression

The hyperparameter `regParam` controls the degree of regularization applied to the model. Smaller values (e.g., 0.01) allow the model greater flexibility, potentially leading to overfitting, while larger values (e.g., 0.5) impose stronger regularization to mitigate overfitting but may cause underfitting. The `elasticNetParam` is another critical parameter that defines the balance between

L1 (Lasso) and L2 (Ridge) regularization (a value of `0.0` applies pure L2 regularization, while a value of `1.0` applies pure L1 regularization). To evaluate the model's performance, I tested the following hyperparameter grid:

- `regParam`: [0.01, 0.1, 0.5]
- `elasticNetParam`: [0.0, 0.25, 0.75]
- Data split: 80% training, 20% test

The best model achieved the following results:

```
Logistic Regression AUC: 0.9621069489541492
Logistic Regression Accuracy: 0.9419781991578117
Logistic Regression Precision: 0.9426388765937813
Logistic Regression Recall: 0.9419781991578117
Logistic Regression F1-score: 0.9354
Logistic Regression Confusion Matrix:
+-------+----------+------+
|delayed|prediction| count|
+-------+----------+------+
|    1.0|       1.0| 72595|
|    0.0|       1.0|  3466|
|    1.0|       0.0| 30361|
|    0.0|       0.0|476583|
+-------+----------+------+
```

To test the model's sensitivity to feature selection and dataset partitioning, I also tried to:

- Remove the features `diverted` and `crs_elapsed_time`.
- Adjust the train-test split to 75% training and 25% test.

The model's performance with these changes was as follows:

```
Logistic Regression AUC: 0.9615549616392448
Logistic Regression Accuracy: 0.9357207392197125
Logistic Regression Precision: 0.937908271373589
Logistic Regression Recall: 0.9357207392197125
Logistic Regression F1-score: 0.9306028309011558
Logistic Regression Confusion Matrix:
+-------+----------+------+
|delayed|prediction| count|
+-------+----------+------+
|    1.0|       1.0| 84568|
|    0.0|       1.0|  2659|
|    1.0|       0.0| 44297|
|    0.0|       0.0|598976|
+-------+----------+------+
```

resulting in a slightly less but still great AUC and accuracy score.

2.2 Random Forest

Since this model was more problematic, I made more attempts here because I was never really satisified with the performances of the mdoel I was having. So, I'm gonna list all my attempts:

- numTrees: [8, 20, 35]
- maxDepth: [5, 8, 10]
- Data split: 80% training, 20% training
- Whole set of features

I got

```
Random Forest AUC: 0.9270323020402981
Random Forest Accuracy: 0.8241764650388933
Random Forest Precision: 0.8547244784670128
Random Forest Recall: 0.8241764650388933
Random Forest F1-score: 0.7455136451686987
Random Forest Confusion Matrix:
+-------+----------+------+
|delayed|prediction| count|
+-------+----------+------+
|    1.0|       1.0|  4451|
|    0.0|       1.0|    75|
|    1.0|       0.0|124371|
|    0.0|       0.0|602949|
+-------+----------+------+
```

and feature importances resulted in

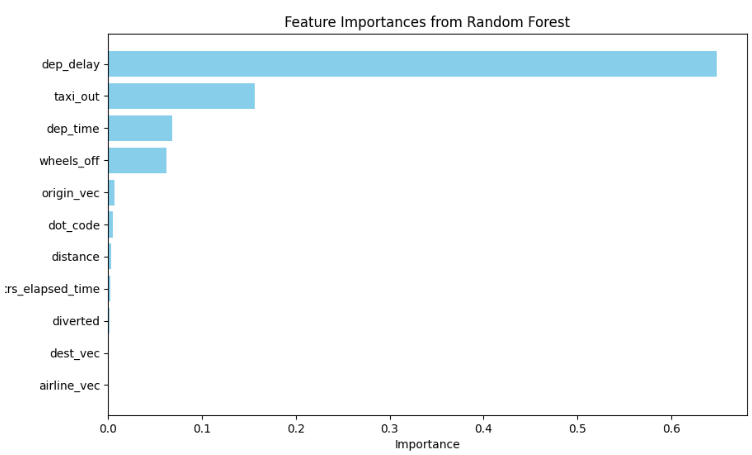

Figure 5: Caption

Another approach was

- numTrees: [13, 24, 40]
- maxDepth: [7, 10, 12]
- Data split: 80% training, 20% training
- Whole set of features

```
Random Forest AUC: 0.9221477296239632
Random Forest Accuracy: 0.8544420716803458
Random Forest Precision: 0.8716418672245342
Random Forest Recall: 0.8544420716803458
Random Forest F1-score: 0.8104175700198025
Random Forest Confusion Matrix:
+-------+----------+------+
|delayed|prediction| count|
+-------+----------+------+
|    1.0|       1.0|   598|
|    1.0|       0.0|102358|
|    0.0|       0.0|480048|
|    0.0|       1.0|     1|
+-------+----------+------+
```

with feature importances

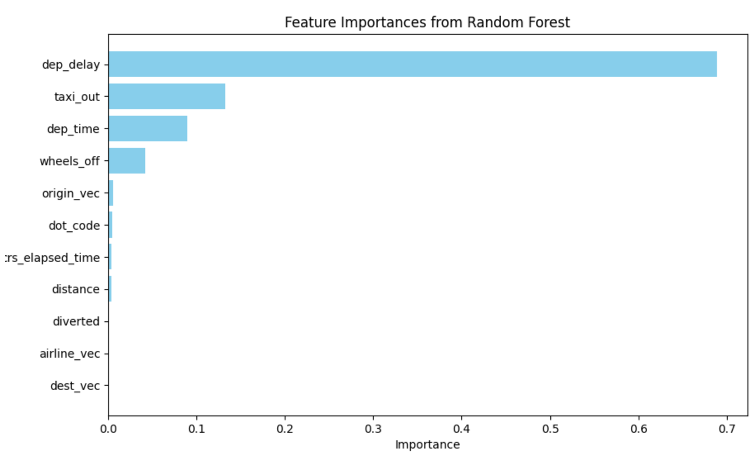

Figure 6: Caption

Although this configuration showed a slight improvement in F1-score, the model continued to struggle with identifying true positives effectively. This prompted a change in approach.

- numTrees: [8, 20, 35]
- maxDepth: [5, 8, 10]
- Remove from features `diverted` and `crs_elapsed_time`
- Data split: 75% training, 25% training

```
+-------+----------+------+
|delayed|prediction| count|
+-------+----------+------+
|    1.0|       1.0| 65591|
|    0.0|       1.0|  5149|
|    1.0|       0.0| 63898|
|    0.0|       0.0|596418|
+-------+----------+------+

Random Forest AUC: 0.9324548995802148
Random Forest Accuracy: 0.9055516950821825
Random Forest Precision: 0.9074788639749408
Random Forest Recall: 0.9055516950821825
Random Forest F1-score: 0.8938942981778376
```

This final configuration resulted in a significant improvement in both recall and F1-score, making it the best-performing Random Forest model. Removing the less-informative features and adjusting the training/test split likely contributed to this enhanced performance.

2.3 ROC curves plotting and problems faced

While I was able to plot both ROC curves successfully during the first attempt,

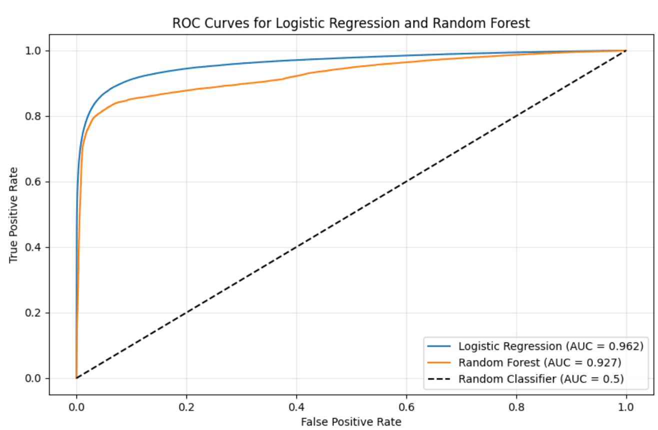

Figure 7: Caption

I encountered issues in subsequent tries. Specifically, I received the following error message:

```
 ERROR PythonRunner: Python worker exited unexpectedly (crashed)
java.net.SocketException: Connection reset by peer: socket write error
```

Unfortunately, due to time constraints, I was unable to investigate the underlying cause of this error. However, this issue prevented further plotting of the ROC curves, limiting the ability to compare the models visually after the initial successful plot.

3 Conclusions

This exercise has been both challenging and rewarding. Not only did I have to grasp the underlying theoretical concepts, but I also had to face other issues like Spark installation and the difficulties plotting ROC curves. Despite these obstacles, I am happy with my work. in particular, Logistic Regression model performed very well and while the Random Forest model did not achieve the same level of performance, after some hyperparameters fine tuning reached a decent level of performance also in the detection of true positives.