Data Mining Homework 2

Alessio Maiola

1 Exercise 1 and 2

I delivered 5 scripts. The first script implements exercise 1.1, storing in a TSV file all the products from the research on amazon.it using the "COMPUTER" keyword.

The other scripts implement tf-idf using Python dictionaries, spark RDDs and an equivalent implementation using spark DataFrames. I also stored the inverted index in a file as requested (using a manual serialization for better visualization). The average running time per query for the three implementations are $0.0196s$, $462.855s$ and $39.963s$ respectively.

The last script contains the StringPreprocessor class, which is used by the above three scripts to preprocess product descriptions.

A screenshot and a CSV file containing the results of each query were provided. For simplicity, I limited the results to the Top 20 documents.

I tried very short queries like: "Lenovo" and "Mouse Wireless", a longer query with clear requirements like "Laptop ACER ssd 256 GB 15.6 pollici ram 16 gb intel i7 windows 11" and a query containing the exact description of a product "NOTODD PC Portatile Laptop Win11 12GB 512GB 1TB SSD Espansione, Notebook 16 Pollici Celeron N5095 (fino a 2.9Ghz) || Ventola di Raffreddamento || 5G WIFI || 1920 * 1200 2K Schermo Doppio- Viola".

In my opinion, the results of the queries are all quite accurate. Of course, the results provided hold for all three implementations, as the results of tf-idf are all equal. However, although the relative order is the same, the cosine similarity computed by Spark DataFrames is slightly different due to numerical approximations.

2 Exercise 3

My approach in solving this exercise consisted of 5 phases.

2.1 Initial analysis of the fields

In this phase I analyzed the fields of the dataset and dropped unuseful columns: "AIRLINE", "AIRLINE_DOT", "ORIGIN_CITY", "DEST_CITY", which all contained redundant information, already present in other fields.

I instead kept both "AIRLINE_CODE" and "DOT_CODE", because they encoded the same information in different ways.

Then I checked the presence of NULL values in the dataset. Except for the last 5 fields, which are mostly NULL, most tuples with NULL values in the ARR_DELAY field belonged to canceled or diverted flights (86196/86198). They represented a considerably small portion of the whole dataset, hence I dropped all such tuples and also the "CANCELLED", "DIVERTED" and "CANCELLATION_CODE" fields.

2.2 Anomaly Detection

In this phase I checked the presence of anomalies in the dataset. The dataset contained clear relationships among the following fields:

- DEP_TIME - CRS_DEP_TIME = DEP_DELAY
- ARR_TIME - CRS_ARR_TIME = ARR_DELAY
- WHEELS_OFF - DEP_TIME = TAXI_OUT
- ARR_TIME - WHEELS_ON = TAXI_IN
- TAXI_OUT + AIR_TIME + TAXI_IN = ELAPSED_TIME
- the sum of the last 5 fields (which, when present, provide additional information about the reasons of the arrival delay) must be equal to ARR_DELAY itself.

This was a bit challenging, as the timestamps were represented as integer/float values and needed to be converted and preprocessed, when they referred to two consecutive days, in order to avoid fake anomalies. Only 18 tuples with anomalies were detected and dropped.

After this phase I also dropped the last 5 fields of the dataset as they were mostly NULL and contained redundant information.

2.3 Exploratory Data Analysis

In this phase, I first identified the outliers in the dataset for the field ARR_DELAY. This was achieved using the Z-score of the field, which consists in computing the normalized values by subtracting the mean and dividing by the standard deviation. To identify the outliers, I checked when the absolute value of the Z-score exceeded the value 3.

I identified 42356/2913784 (1.45%) outliers, all with a positive delay.

Secondly, I started checking relationships between the fields. I initially focused on numerical fields, looking at their range, average and standard deviation. Some similarities were easy to identify between pairs of fields, which were in fact confirmed by the correlation matrix.

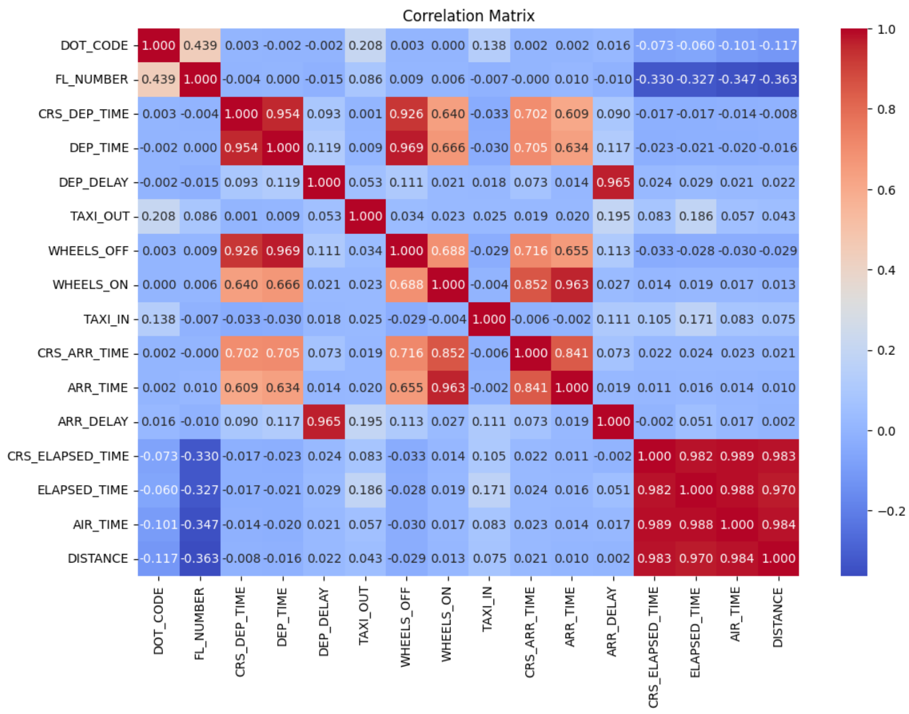


Figure 1: Correlation Matrix of numerical fields

The scheduled and actual times for departure, arrival or the elapsed times are all obviously highly correlated. Moreover, there is also a strong correlation between departure and arrival times. The elapsed times are also clearly highly correlated with the distance and the air time. Furthermore, these four fields exhibit a weak negative correlation with the airline code and an even stronger negative correlation with the flight number, suggesting that some airlines and flights cover shorter distances.

However, what I was mostly interested in was finding correlations with the arrival delay, that would allow to easily predict it. Besides some weak correlations with the TAXI_IN and TAXI_OUT fields, I found a very strong correlation with the departure delay. This makes this feature the most important one to predict the arrival delay.

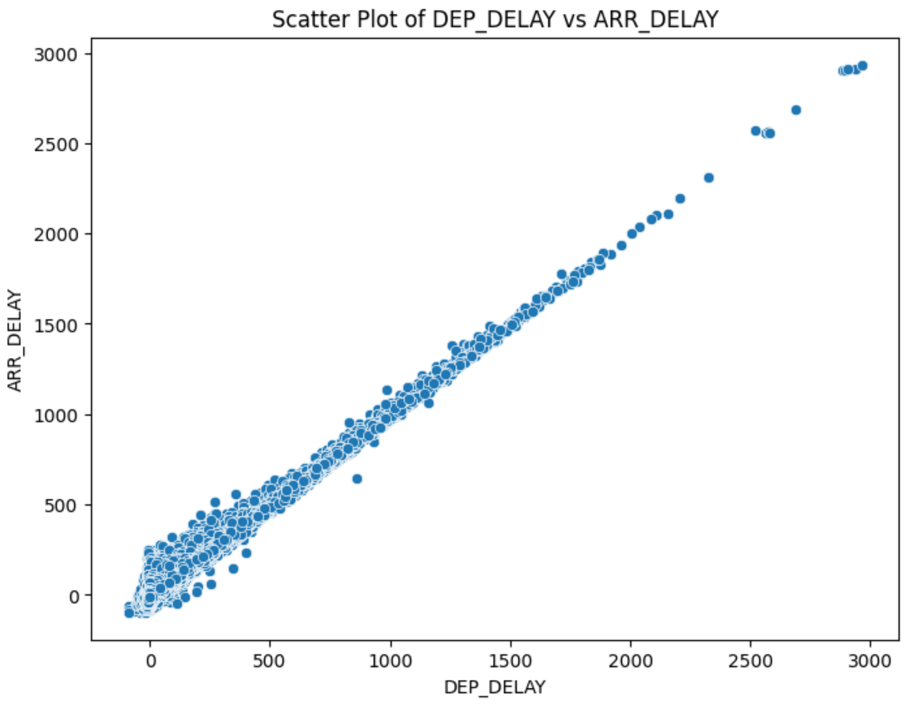


Figure 2: Scatterplot of arrival and departure delays

Lastly, I focused on nonnumerical fields, but unfortunately I did not identify neither strong correlations nor additional useful information to predict the arrival delay.

2.4 Model development

I preprocessed nonnumerical data by transforming dates into integer values and one-hot encoding the strings in the "AIRLINE_CODE", "ORIGIN" and "DEST" fields.

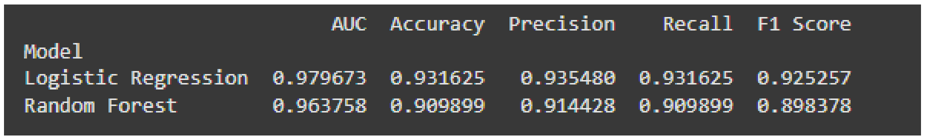
Then I removed from the dataset the ARR_TIME and WHEELS_OFF fields, which would allow to compute the target field ARR_DELAY, and also WHEELS_ON and AIR_TIME because of their redundancy. Additional details on why I kept all the other fields are available in the notebook.

Finally, I implemented the required machine learning models and I used the appropriate classes for 5-fold cross-validation and evaluation. I tested all the code using a subset of 100 samples from the original data and made sure that all models had a similar running time.

Unfortunately, my test was not representative of the real running time of the models. The time needed to train the Logistic Regression and the Random Forest was 1 hour and 5 hours respectively. The Gradient Boosted Trees and

Multilayer Perceptron training did not terminate even after 12 hours, even with less hyperparameters, hence I was not able to put their results in my report. However, running them in a cluster (if I only had one) could make their training feasible.

Anyway, both model converged and the logistic regression got the best performances.



```
                          AUC  Accuracy  Precision    Recall  F1 Score
Model
Logistic Regression  0.979673  0.931625   0.935480  0.931625  0.925257
Random Forest        0.963758  0.909899   0.914428  0.909899  0.898378
```

Figure 3: Metrics evaluated on models' predictions

The convergence of the models is demonstrated by the large area under the ROC curve, which is the metric maximized during training.

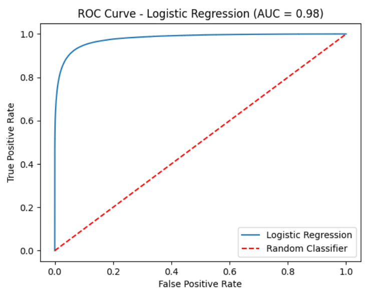


Figure 4: Logistic Regression

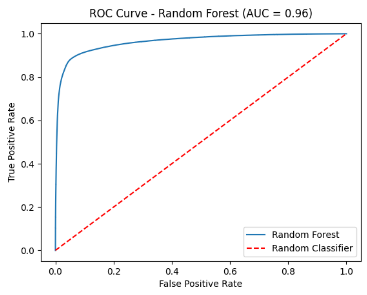


Figure 5: Random Forest

Figure 6: Confusion matrices of the models

From the confusion matrices, it is evident that both models tend to predict 0 due to the imbalance in the dataset. Consequently, the most frequent errors are false negatives.

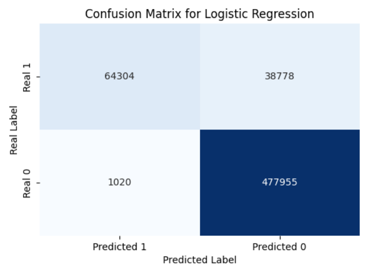


Figure 7: Logistic Regression

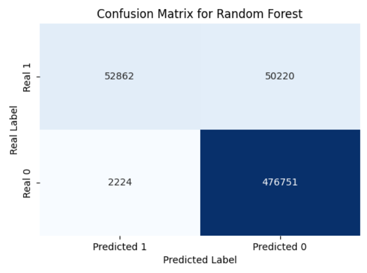


Figure 8: Random Forest

Figure 9: Confusion matrices of the models

For the random forest (and also for gradient boosted trees) it is possible to get the importance of each feature for the predictions. Not surprisingly, the highest importance was achieved by the most strongly correlated features with the target field.

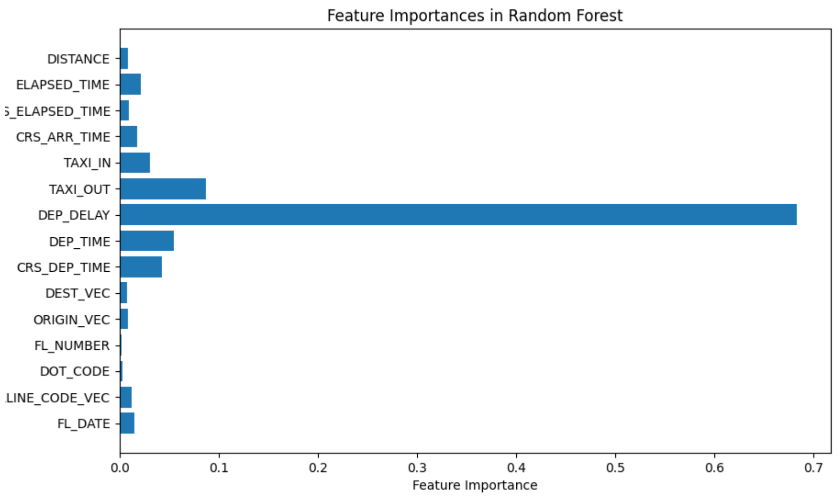


Figure 10: Feature importances in Random Forest

2.5 Error analysis

In the last phase I tried to analyzed possible reasons for the mistakes of the models, besides the imbalance in the dataset. I discovered that in the logistic regression, most mistakes happen when ARR_DELAY and DEP_DELAY are not correlated. Selecting only wrong predictions of the logistic regression, the correlation between the fields becomes much lower (from 0.965 to 0.26, while in Random Forest errors the correlation is still 0.93). This demonstrates how logistic regression deeply relies on this correlation to predict the arrival delay.

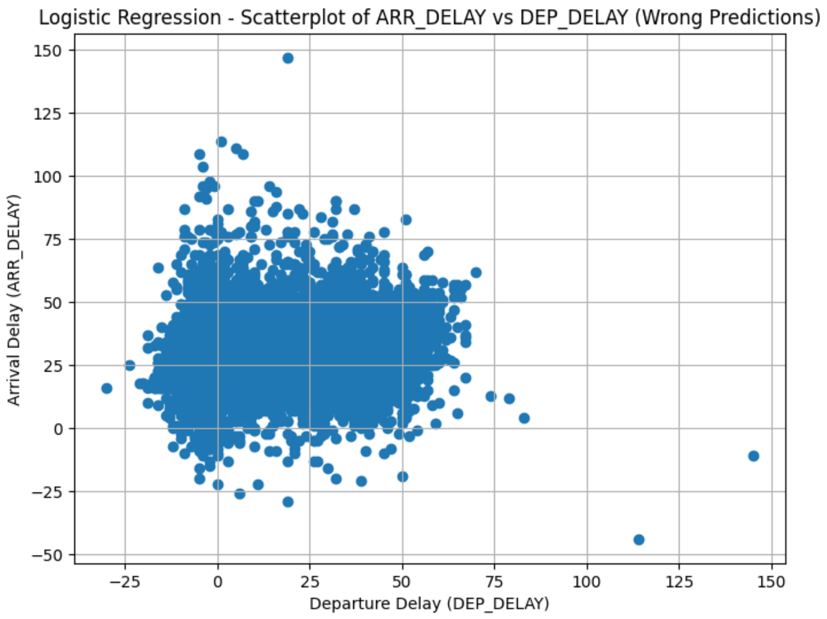


Figure 11: ARR_DELAY vs DEP_DELAY in logistic regression's errors

Conversely, while logistic regression's wrong predictions contain no outlier, those of the random forest contain a higher percentage of outliers than in the whole dataset (1.97% instead of 1.45%).

Therefore I concluded that Logistic Regression is more robust to outliers, but it relies more on highly correlated fields, often overlooking other relevant information.