Flight Delays: A Big Data Approach

Catarina Rocha
University of Porto
Faculty of Engineering
Porto, Portugal
up201505702@up.pt

Joaquim Carneiro
University of Porto
Faculty of Engineering
Porto, Portugal
up200203159@up.pt

João Pedro Pêgo
University of Porto
Faculty of Engineering
Porto, Portugal
up199502401@up.pt

Fabiana Rodrigues
University of Porto
Faculty of Engineering
Porto, Portugal
up202100810@up.pt

João Patrício
University of Porto
Faculty of Engineering
Porto, Portugal
up202100812@up.pt

Pedro Gouveia
University of Porto
Faculty of Engineering
Porto, Portugal
up202100813@up.pt

Abstract - The present paper addresses a well-known flight delays problem, with high impact to the overall air business stakeholders. Since, the amount of collected data in this business has been increasing on a yearly basis, the development of new strategies to retrieve knowledge from this high-volume data arises. This paper aims to explore a Big Data approach, based on Apache Spark parallel computing capabilities, with main focus on the data treatment, understanding, learning, and performance stages. Firstly, a Resilient Distributed Dataset (RDD) approach was used for data ingestion, followed by a query based Exploratory Data Analysis (EDA). Then, two predictive Machine Learning (ML) models were developed for classification and regression. The performance of the models was evaluated by an area under ROC of 0.598 for the classifier model and RMSE of 0.093 for the regression model. The overall pipeline performance was evaluated, where the best hardware configuration presents a running time of 14 minutes.

Keywords— flight delays, big data, apache spark, machine learning

I. Introduction

This document describes the process and results for building a predictive model on Spark to determine whether a flight with a delayed departure will be able to recover and arrive on time. This process is explained in three parts:

- Ingestion: data is profiled, filtered, validated and transformed to meet the requirements of the predictive model; this includes the handling of missing values and outliers.
- Data Understanding and Learning: data is explored to provide insights in terms of the explanatory and target variables of the model; alternative predictive models are trained and evaluated for the task.
- Performance Analysis: hardware analysis is performed, where different session configurations in terms of the amount of memory, instances, cores and parallelism are tested; the different scenarios performance was evaluated based on running time.

Figure 1 illustrates this process as a workflow of activities in conjunction with the Spark techniques employed.

II. Related Work

Due to the negative impact that flight delays can cause in a company's name, together with additional financial losses, a variety of studies related to the subject can be found in the literature. Studies can be divided into two different groups.

The first group where the problem is addressed as a classical ML problem, not taking into consideration the running time performance. Here a high variety of algorithms might be found, as well as, their related performance, from Support Vector Machines, DT, Random Forest, and Gradient Boosted Tree ([1][2][3][4][5]). The second group with a more realistic approach, that considers to be in the presence of a big data problem, taking that into consideration when performing the data treatment and prediction stages ([6][7]). Concerning the prediction stage, similar algorithms to the ones presented above are also used.

Overall, after evaluating both groups' approaches, best prediction models are obtained when ensemble methods are applied, together with meteorological features (which in our case were not available).

III. Data Ingestion

A structured dataset of several years of flight data [8] was used containing the following fields: fl_date, op_carrier, op_carrier_fl_num, origin,dest, crs_dep_time, dep_time, dep_delay, taxi_out, wheels_off, wheels_on, taxi_in, crs_arr_time, arr_time, arr_delay, cancelled, cancellation_code, diverted, crs_elapsed_time, actual_elapsed_time, air_time, distance, carrier_delay, weather_delay, nas_delay, security_delay, late_aircraft_delay.

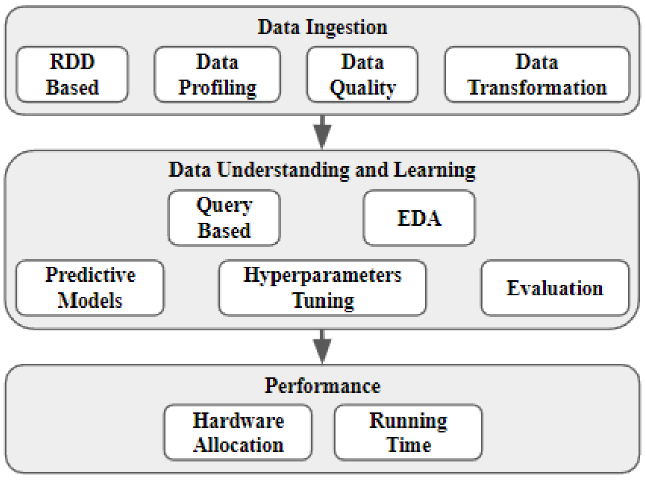

***Figure 1** - Workflow of activities.*

For the purpose of this work, only a subset of the available dataset was used. Table 1 details the instance inclusion criteria.

***Table 1** - Instance inclusion criteria.*

Criteria	Description
1	Flight is 2018
2	Flight with a delayed departure
3	Flight not diverted
4	Flight not cancelled

Additionally, not all attributes were used as some were considered either redundant or irrelevant for the predictive model (e.g., delay attributed to the weather conditions without knowing the weather conditions themselves).

The attributes included in this work are detailed below and, in general, they capture the flight date, operator, route, distance and all the related timings as illustrated in Figure 2.

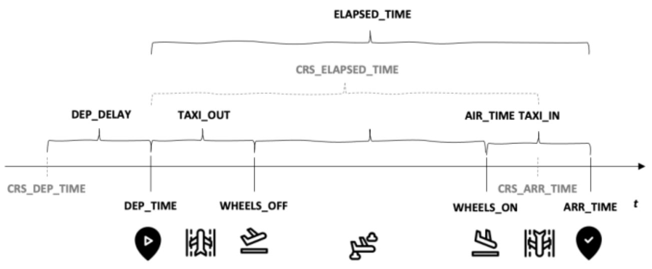

Figure 2 *- Flight timings.*

Data Profiling

Data profiling processes [9] were applied to determine the metadata discovery, to understand what information could support and answer the questions presented in this paper. It was composed by identifying variables and fields by exploring cardinalities, data types, and applying functions using Resilient Distributed Dataset (RDD), such as count, count distinct, mean, min, max, variance and covariance for example.

The functions applied to profile the data allowed to identify the cardinalities - number of rows, null values and distinct values -, value distributions - histograms, constancy - patterns e data types -, basic types and domain classes.

Statistical relationships between variables were tested to a better understanding of the existing correlations in the dataset. High correlations results were obtained from the analysis of the dataset, allowing to identify redundant columns of data.

During the data profiling phase, a smaller dataset, statistically compatible with the original datasets, were produced - with ten thousand rows of data -, in order to reduce time consumption when developing and allowing to test and run the process locally.

Data Quality

The data quality process applied:

- data type conversion based on data schema;
- filtering instances with relevant missing attributes;
- filtering cancelled, diverted and on-time departure flights;
- exclusion of outliers in the ARR_DELAY field.

Data Transformation

Once the previous steps were finished and the data was ready to be used in the data transformation process, RDD loaded dataset was used to apply the following transformations:

- creation of calculated items: REL_INIT_DELAY, T_LATE, T_REL_ELAP_TIME, DEP_HOUR_IN_DAY and ARR_HOUR_IN_DAY;
- creation of derived attributes: FL_DATE_DAY_IN_WEEK, FL_DATE_DAY_IN_MONTH and FL_DATE_MONTH;
- exclusion of attributes not needed for analysis;
- filtering data not relevant for the proposed analysis.

After performing the computation, in the data ingestion phase, the RDD dataset was converted to a dataframe dataset to proceed in the Machine Learning analysis.

Table 2 shows the output of calculated and derived fields, and Table 3 shows the fields that originate Table 2.

Table 2 *- Calculated and derived fields.*

Attribute	Description	Type
REL_INIT_DELAY	(DEP_DELAY + TAXI_OUT) / CRS_ELAPSED_TIME	double
T_LATE	1 If ACTUAL_ELAPSED_TIME - CRS_ELAPSED_TIME > 15; Otherwise = 0	long
T_REL_ELAP_TIME	ACTUAL_ELAPSED_TIME / CRS_ELAPSED_TIME	double
FL_DATE_DAY_IN_WEEK	Weekday of FL_DATE	long
FL_DATE_DAY_IN_MONTH	Day of FL_DATE	long
FL_DATE_MONTH	Month of FL_DATE	long
DEP_HOUR_IN_DAY	DEP_TIME /100	double
ARR_HOUR_IN_DAY	CRS_ARR_TIME /100	double

Table 3 *- Origin of calculated and derived fields.*

Attribute	Description
DEP_DELAY	Total Delay on Departure in minutes
TAXI_OUT	The time duration elapsed between departure from the origin airport gate and wheels off
ACTUAL_ELAPSED_TIME	Actual time amount made in the flight trip, in minutes
DEP_TIME	Planned Departure Time in minutes
CRS_ARR_TIME	Planned Arrival Time in minutes
CRS_ELAPSED_TIME	Planned time amount needed for the flight trip in minutes

IV. Data Understanding and Learning

The predictive model built in this work is intended to determine whether a flight with a delayed departure will be able to recover and arrive on time. This can be addressed as a classification problem (delayed vs on-time) or as a regression problem (how much delay).

Both approaches were followed and analysed in this work, and used the Spark implementations of classification and regression decision trees. The choice relates mainly to the known algorithm's ability to consume unnormalised and categorical data as well as its robustness to outliers [10].

The explanatory attributes used in the predictive mode are listed in Table 4.

Table 4 *- Model's explanatory attributes.*

Attribute	Description
Day in Week	Day of the week (1..7) (int.)
Day in Month	Day of the month (1..31) (int.)
Month	Month (1..12) (int.)
Hour in Day	Departure hour in day (0..23) (int.)
Arrival Hour	Arrival hour in day (0..23) (int.)
Initial Detail	Initial delay relative to estimated flight time (int.)
Est. Elap. Time	Estimated flight duration (int.)
Distance	Distance between airports (int.)
Operator	Carrier operator (cat.)
Origin	Origin airport (cat.)
Destination	Destination airport (cat.)

To better understand the possible interactions between the target and the explanatory attributes, a simple bivariate analysis was carried out.

SparkSQL was used to query data, and according to Spark's official documentation, "Spark SQL is Apache Spark's module for working with structured data." [11]. This module enables the possibility to access structured data by using SQL or a DataFrame API, and both approaches provide the same operations as relational query languages, [12]. In addition, both approaches provide a common way to access a variety of data sources [11].

Through Python, queries were created using both approaches, focusing on understanding the data in the sense of delays, and analysing how those queries perform. They are:

SQL Programmatically queries:

- Top 10 delays flights, classified by time of arrival delay

All queries below filter if the flight is classified as late, which is a result field from the Data Ingestion process:

- Top 10 count of late flights
- Top 10 relative delayed flights per day in the month
- Top 12 relative delayed flights per month
- Top 10 relative delayed flights per day in the week. The code below shows this example:

```
spark.sql("""select FL_DATE_DAY_IN_WEEK,
avg(T_REL_ELAP_TIME) AVG_REL_DELAY from DF where
T_LATE = 1 group by FL_DATE_DAY_IN_WEEK order by
FL_DATE_DAY_IN_WEEK asc""").show(10)
```

Dataframe queries:

- Top 10 carriers on average delays, for the flights classified as late
- Top 10 origin airports on average delays, for the flights classified as late
- Top 10 flight routes. The code below shows this example:

```
df.groupBy('ORIGIN','DEST').count().orderBy('count
',ascending=False).show(10)
```

The predictive models were built as a multi-stage process resorting to Spark ML Pipelines [14]. This process consists in i) Data preparation, ii) Model training and evaluation [15] and iii) Hyperparameter tuning [16]. Additional stages' details, and pseudocode are presented below (all the underlined operations are classification and regression shared).

i) Data preparation: since categorical attributes cannot be directly handled by a DT algorithm, an indexing operation was performed [17].

ii) Model training and evaluation: both DT classification and regression models were trained and tested using 80% and 20% of the dataset, respectively. Concerning the evaluation stage, area under ROC [18], and Root Mean Square Error (RMSE) [19], were used as classification and regression evaluators, respectively.

iii) Hyperparameter tuning: a grid search approach for model selection was adopted, in which all the hyperparameters configurations were generated. For both classification and regression algorithms, maximum depth, maximum bins, information gain, minimum instances per node, minimum information gain, and minimum weight fraction per node hyperparameters were tuned.

Classification and Regression Pseudocode

```
1 → StringIndexer(features)
2 → Features selection
3 → VectorAssembler(features)
4 → DecisionTreeClassifier(features, labels)
5 → DecisionTreeRegressor(features, labels)
6 → ParamGridBuilder(hyperparameters)
7 → BinaryClassificationEvaluator(area under ROC)
8 → RegressionEvaluator(RMSE)
9 → TrainValidationSplit(trainRatio = 0.8)
10→ Pipeline definition and deployment
11→ Model evaluation
```

Based on the pseudocode above, the best classification and regression model hyperparameters, and the related evaluation criteria are presented in Table 5.

Table 5 *- Best models hyperparameters and evaluation.*

Hyperparameters	DT Classif.	DT Reg.
Max.Depth	20	5
Max. Bins	360	32
Information Gain (Impurity)	Gini	Variance
Min. Instances per Node	1	1
Min.Information Gain	0	0
Min. Weight Fraction Per Node	0	0
	Area Under ROC	**RMSE**
	0.598	0.093

For visualisation a third party library [20] for DT visualisation was used. Within the time frame of this project it was not possible to run the dtreeviz on pyspark, though we were successful to run it on laptop computers. The visualization results for a DT classification model computed for a smaller dataset (10k records) are presented in Figure 3.

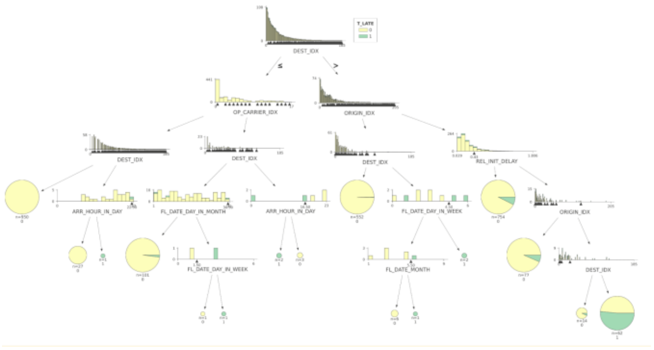

***Figure 3** - Classification Decision Tree.*

V. Runtime and Scalability Analysis

A performance analysis was done for the Spark SQL section of this work as well as for the machine learning component. Regarding queries and following the approach from ([13], [21]) it is noticeable that queries become much faster only for the cases of grouping, if they are reading data from a cached table, as shown in Table 6.

***Table 6 –** Comparison of the running times by using (or not) a cached table.*

Query Type	W/O Cache (s)	W Cache (s)
Select	0.045	0.048
Count	9	9
Group By	13.43	**0.57**
Group By w/ case when condition	9.62	**0.85**
Top 10 delayed per day in week	10	**0.8**

The data in cache is faster to be retrieved in cases of data grouping. Caching can help in reducing the cost of recovery in this scenario because the data is already available in cache and there is no need to calculate them again, which happens when retrieving data from memory, ([12][7]). For other functions such as count and select, there is not much difference of time because the action is just to retrieve and show the data, and no calculation inside the dataframe is needed.

The first batch of runs was performed in personal computers to assess the capacities of the algorithm running as expected. The classification and regression model and the visualization tool performed as expected for a dataset with 10k records. When using the same code in the pyspark cluster it was verified that not only dtreeviz could not be installed there as the regression model failed to output results. Due to time restrictions, for the presentation of the work we decided to advance with the runtime and scalability analysis only for the classifier decision tree model. Table 7 summarises the cluster configuration scenarios used and the performance in terms of time. For these tests the operations included were ingestion, SQL queries and ML models. Figure 4 shows the average running times for the four configuration parameters considered.

***Table 7 –** Scenarios config settings and running times.*

Memory (Gb)	Instances	Cores	Parallelism	Time (min.)
1	2	2	2	34
1	4	2	2	23
1	4	4	2	32
2	2	2	2	26
2	2	2	3	27
2	2	2	4	28
2	4	2	2	15
2	4	4	2	18
4	2	2	2	24
4	4	2	2	**14**
4	4	2	2	23
4	4	4	2	15

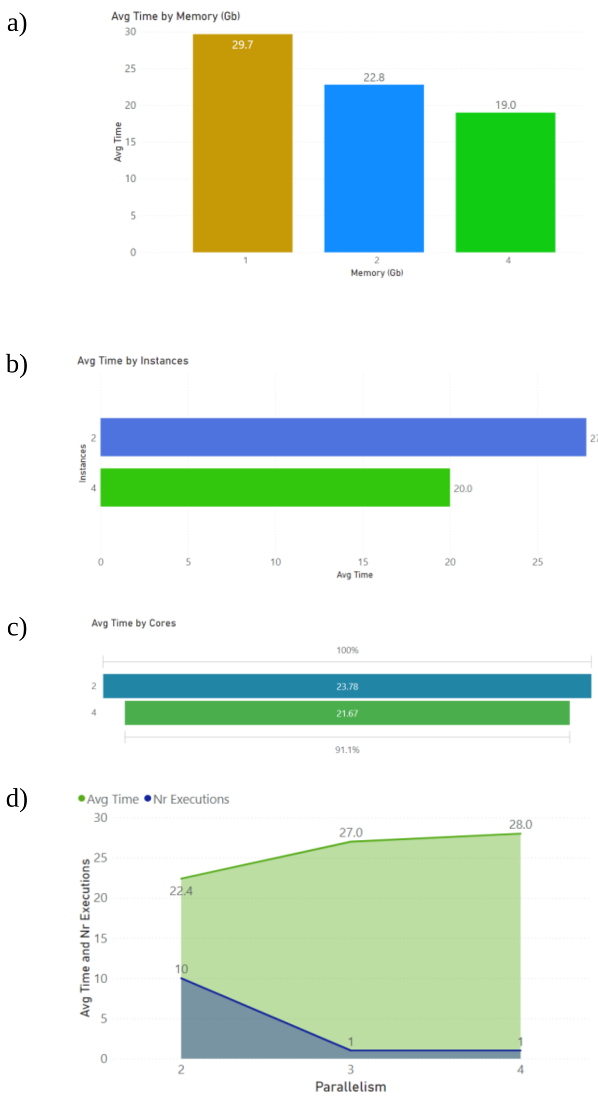

***Figure 4 –** Running times for different config settings*

A third round of tests was executed after the public presentation of the first results. For this last set of runs, the RDD were kept in cache through all the computations, which allowed for successful computation of the regression model, whose hyperparameters are summarized in Table 5. It was planned to have a set of 3 repeated computations for each config setting (1,2,3 Gb; 2,4 instances; 2,4 cores; 2,4 parallelism). Yet, probably due to the increased memory demand, only 4 configurations were successfully run, whose running times are presented in Table 8. The results are not directly comparable to the ones from Table 7 since here we had the two ML models working. Yet, it is clear that for the same the configuration, different but similar running times are obtained.

Table 8 – *Scenarios config settings and running times for second set of runs.*

Memory	**Instances**	**Cores**	**Parallelism**	**Time**	**Av. Time**
[Gb]	**[-]**	**[-]**	**[-]**	**[min]**	**[min]**
4	2	2	4	45.4	43.3
4	2	2	4	44.5	
4	2	2	4	40.0	
4	2	4	2	41.0	40.7
4	2	4	2	40.2	
4	2	4	2	40.8	
4	4	2	4	30.0	31.4
4	4	2	4	33.5	
4	4	2	4	30.7	
4	4	4	4	36.3	30.9
4	4	4	4	31.0	
4	4	4	4	25.3	

VI. CONCLUSION

The objective of using big data to train and test two machine learning algorithms was successfully implemented. The process used data ingestion, data profiling using RDD, and SQL querying and machine learning using dataframes. A classifier and a regression decision tree model were implemented. Several configuration scenarios were considered for the cluster, using running time as performance indicator.

Concerning the learning stage, although a high variety of predictive algorithms are already developed, one-class classifiers are still not available, which could be helpful to explore other learning strategies, probably with an increased performance (suitable for imbalance datasets). Concerning hyperparameters tuning, only one optimization strategy is currently available (grid search), which is time consuming and less optimised when compared with other search strategies, random search and genetic algorithms, for instance.

REFERENCES

[1] S. Mokhtarimousavi and A. Mehrabi. (2022). Flight delay causality: machine learning technique in conjunction with random parameter statistical analysis. IJTST.

[2] Y. Tang. (2021). Airline flight delay prediction using machine learning models. ICEBI 2021.

[3] N. Kalyani, G. Jeshmitha, Bindu Sri Sai U., M. Samanvitha, J. Mahesh. (2020). Machine learning model - based prediction of flight delay. IEEE. 2020 Fourth International Conference on I-SMAC.

[4] https://medium.com/analytics-vidhya/using-machine-learning-to-predict-flight-delays-e8a50b0bb64c, visited in May 2022.

[5] https://medium.com/@pedrodc/building-a-big-data-machine-learning-spark-application-for-flight-delay-prediction-4f9507cdb010, visited in May2022.

[6] https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/4292307494740474/1474730470433170/6190017855571830/latest.html, visited in May 2022.

[7] https://www.dbta.com/Editorial/Trends-and-Applications/Spark-and-the-Fine-Art-of-Caching-119305.aspx

[8] https://www.kaggle.com/datasets/yuanyuwendymu/airline-delay-and-cancellation-data-2009-2018, visited in April 2022.

[9] Liu, Z., & Zhang, A. (2020). Sampling for Big Data Profiling: A Survey. IEEE Access, 8, 72713–72726. https://doi.org/10.1109/ACCESS.2020.2988120

[10] João M. Moreira et. al, 2019, A General Introduction to Data Analytics, Wiley.

[11] Apache Spark project. http://spark.apache.org/sql/.

[12] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. 2015. Spark SQL: Relational Data Processing in Spark. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (Melbourne, Victoria, Australia) (SIGMOD '15). Association for Computing Machinery, New York, NY, USA, 1383–1394. https://doi.org/10.1145/2723372.2742797

[13] https://medium.com/analytics-vidhya/analyzing-data-and-performance-tuning-of-apache-spark-engine-1dfdf4e2c705

[14] https://spark.apache.org/docs/latest/ml-pipeline.html, visited in May 2022.

[15] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.tuning.TrainValidationSplit.html, visited in May 2022.

[16] https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.ml.tuning.ParamGridBuilder.html?highlight=paramgridbuilder, visited in May 2022.

[17] https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.StringIndexer.html, visited in May 2022.

[18] https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc, visited in May 2022.

[19] https://en.wikipedia.org/wiki/Root-mean-square_deviation, visited in May 2022.

[20] https://github.com/parrt/dtreeviz, visited in May 2022.

[21] Uta, Alexandru & Ghit, Bogdan & Dave, Ankur & Boncz, Peter. (2019). [Demo] Low-latency Spark Queries on Updatable Data. 2009-2012. 10.1145/3299869.3320227.