UNIVERSITÀ DI PISA

Distributed Data Analysis and Mining

Alberto Bessone
Sandro Cantasano Martino
Ondrej Krasnansky
Stefano Toresan
Luca Santolillo

Flight Status Prediction

Academic Year 2023/2024

Contents

1 Introduction **2**

2 Data Understanding and preparation **2**
2.0.1 First Phase 2
2.0.2 Second Phase 3
2.0.3 Explorative Analysis 3

3 Clustering: Insight from K-Means **4**

4 Classification **6**
4.1 ArrivalDelayGroups 6
4.2 ArrivalDelay15 8
4.3 Cancelled 9

5 Regression **9**

1 Introduction

This document concerns the final work for the analysis of "Flight Status Prediction" from Kaggle. The main purpose of this work is to identify a model and a pattern that allows us to say why a flight could be cancelled or not and which are the main motivations for which an airline is delayed.

For this purpose, we have gone through a classification task and a regression task in order to better represent the motivation of such a possible situation. The dataset contains data of 28 Airline companies, flighting from and to 370 different locations in 52 States in the United State of America.

2 Data Understanding and preparation

Our dataset is composed of 5.689.512 rows and 131 columns (in the original document of presentation). Every row is a flight, besides some general information (we will talk about it later), every flight is targeted as "Cancelled" or "not Cancelled" which, with the "Diverted", are the two main target variables of our dataset.

2.0.1 First Phase

As a starting point of Data Understanding, we decided to analyze every feature by looking also on the .html file joined to the folder of the dataframe, which gave us some information connecting to them. In a brainstorming manner, we dropped some features motivating the decision: for example, the features "DepDelay" and "DepDelayMinutes" are the same, the differences between them are that the second one shows 0 if there is a difference in minutes between scheduled and actual departure time. Instead, DepDelay shows a negative number in case the aircraft departs before the scheduled time. For this reason, we decided to maintain just DepDelay because it was much informative. We tried to model the remaining features in this manner.

For another block of features, we discovered the presence of some internal errors, for example in "Flights" and "CarrierDelay" columns. For another block of features, we noticed that every column with the prefix "Div" contained errors. In fact, for this reason, with this lack of informations, we were also obligated to drop the possible target variable "Diverted" since the strictly connected information was missed. Other kinds of information were the distance of the flight, the origin, the destination, some information from a CRS(a sort of black box of the aircraft) and some identification from two distinct Organizations like *IATA*(the international organization in the field of flight) and *DOT*(the American organization in the field of flight). The two different organizations assign an ID to the carrier for the same purpose, so maintaining both could be redundant. Since the IATA identifiers have been assigned to different carriers over time, we discovered this pattern and we decided to drop all the features with the prefix "IATA".

After this phase of dropping, we remained with 42 columns. The remained columns were composed of 13 non-numeric columns and 29 numeric columns. We went through the **Missing value analysis** and we discovered that 15 features contained from 80.000 to 100.000 rows with nan values. In the first attempt, since this quantity was about two per cent of the total amount of flight in the data, we decided to drop the rows. This action led us to drop the entire label "True" of the target variable "Cancelled" and, since this, we understood that the lack of information in that sense resulted in the cancelled flight, for which no data was gathered after the cancellation was announced.

Obtained a cleaned dataframe, we plotted the distribution of the numerical columns in order to make some visual interpretation; we noticed that no columns tracked a normal distribution and for this reason we decided to **normalize** our numerical column. Before passing to the Correlation analysis, we transformed some features like CRSDepTime, DepTime, ArrTime(all features written in hh-mm format) in a numerical format(count of minutes starting from the midnight). We decide to go for the **correlation analysis** splitting our dataframe in two (the categorical columns couldn't be analyzed). The analysis led us to drop some features:

- CRSElapsedTime: is identical to AirTime
- ActualElapsedTime: is identical to Airtime

- CRSdepTime_inMinutes: is identical to DepTime_ InMinutes
- CRSArrTime_InMinutes: is identical to Arrtime_inMinutes

2.0.2 Second Phase

Since we discovered the presence of a missing value (look at 2.0.1 above), we decided to drop the rows since it was just 2% of the total amount. As we anticipated, this led us to lose one of the two labels of the target variable "*Cancelled*", compromising the next Classification task.

Since this, we came back from several steps and we decided to drop the **columns** instead of the rows, resolving the problem of massive missing values.

Also the second phase was done with **Correlation analysis**, a massive dropping was done and, effectively, the end of this two-phase produced two different dataframe.

We decided to pursue both, maintaining the first dataframe (obtained from the 2.0.1 explaination) for the Regression task, instead of the second dataframe(obtained from the **second phase**) for the Classification task because the first preprocessing was much focused on preserve the *structure of the data* as it was designed and this would have been important for the regression(looking to the final features remained), instead the second preprocessing was much focused on preserve the *target variable* crucial for the classification task.

2.0.3 Explorative Analysis

We have done some analysis connected on the feature WheelsOn, shown in figure 1. As we know, that feature concerns the time in which a carrier touches the airstrip and it ends when the wheels break away.

As we can see in figure 2, we discovered that the airports with the highest values of average WheelsOn are big airports of metropolitan cities, while the lowest values are encountered for little airports in the outskirts of the cities.

Another interesting analysis is connected to the airlines with the biggest WheelsOn, the main one is the *Southwest Airlines* which, strangely, is the biggest Low-cost airline company, and it seems strange since this company has to be as much efficient as it can.

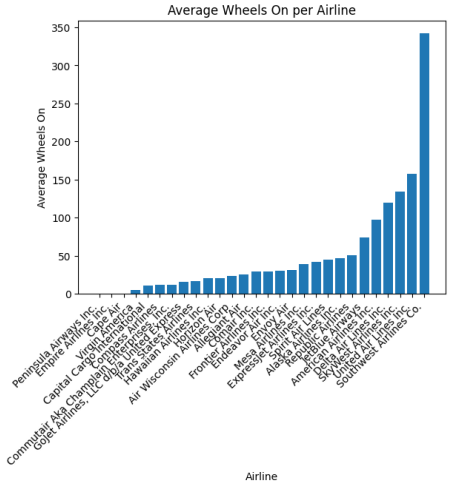


Figure 1: Airlines's WheelsOn

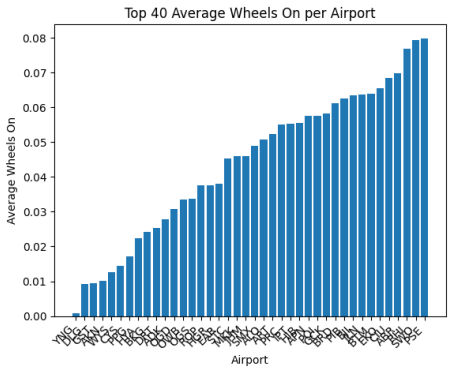


Figure 2: Top 40 airports with minimum average WheelsOn

3 Clustering: Insight from K-Means

During this step, we initially used the preprocessed dataset and filtered it, so as to obtain a dataset containing only records with a delay at both departure and arrival. After a few drops of unnecessary variables, several attempts were made at *outliers detection* but, in the end, it was deemed appropriate not to eliminate outliers. Next, features were chosen on which to perform clustering with K-Means ("*scaled_ DepDelay*", "*scaled_ AirTime*", "*scaled_ Distance*","*scaled_ ArrDelay*","*scaled_ DepTime_ InMinutes*", "*scaled_ ArrTime_ InMinutes*", "*scaled_ WheelsOff_ InMinutes*","*scaled_ WheelsOn_ InMinutes*") Finally, the *WSSE* has been calculated for each K and then, the *elbow method* has been plotted, and the *silhouette score* as k varied. As also visible in the Figure 3, k=*4* was chosen for the K-Means.

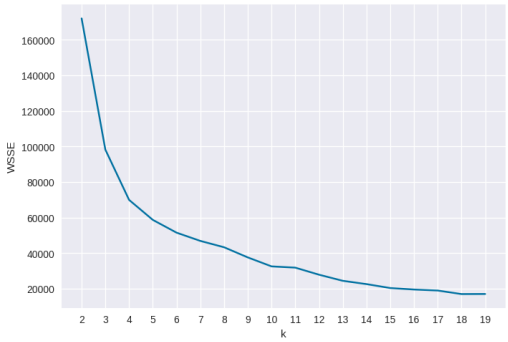


Figure 3: Elbow Method

Once the clustering results were obtained, we analysed the different clusters according to the associated categorical and numerical variables. As far as numericals features are concerned, the dataset being inclusive of outliers, we thought of choosing the *median* as the representative value of the cluster. In any case, we tried to understand the results and give a "proper label" to each cluster, and the results obtained are in figure 1:

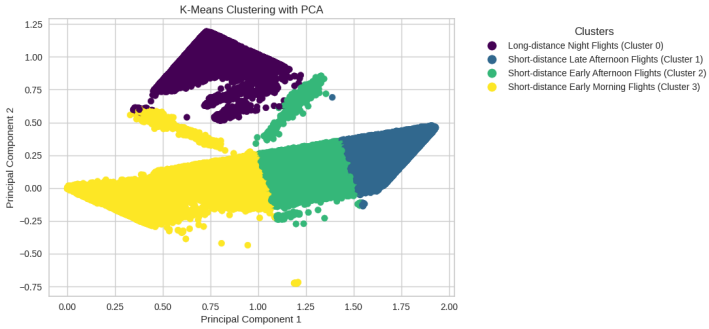
- **Long-distance Night Flights**: Departure in the evening and arrival late evening or early morning, mainly Q3 (Jul-Aug) Q4(Nov-Dec).
- **Short-distance Late Afternoon Flights**: Departure late afternoon and arrival in the evening, mainly Q4(Oct-Nov-dec) and Q3 (Aug-Sept).
- **Short-distance Early Afternoon Flights**: Departures late morning-early afternoon and arrival in the afternoon, mainly Q4(Oct-Nov-Dec) and Q3 (Aug-Sept).
- **Short-distance Early Morning Flights**: Departure early morning and arrival late morning, mainly Q4(Oct-Nov-Dec) and Q3 (Aug-Sept).

Table 1: KMeans Clustering Results

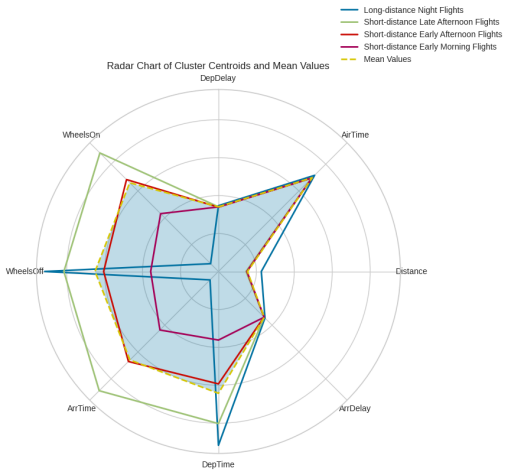
Category	Long-distance Night Flights	Short-distance Late Afternoon Flights	Short-distance Early Afternoon Flights	Short-distance Early Morning Flights
N° Records	96244	531314	494619	308207
Airline (TOP 5)	Southwest (33.05%) United (13.75%) Jetblue (13.68%) American (7.95%) Spirit (4.71%)	Southwest (33.56%) United (9.31%) SkyWest (7.06%) American (6.73%) Jetblue (6.55%)	Southwest (29.75%) United (9.33%) SkyWest (8.93%) American (7.16%) Jetblue (5.83%)	Southwest (25.45%) United (9.87%) SkyWest (8.96%) Jetblue (7.08%) American (6.90%)
Origin (TOP 5)	DEN (7.05%) LAX (6.72%) LAS (6.52%) ORD (5.78%) SFO (4.95%)	ORD (5.25%) DEN (4.52%) ATL (4.05%) EWR (3.39%) DFW (2.65%)	ORD (5.25%) DEN (4.52%) ATL (4.05%) SFO (3.45%) LAS (2.89%)	ORD (5.76%) DEN (4.18%) ATL (3.96%) LAX (3.25%) EWR (2.87%)
Destin (TOP 5)	BOS (4.60%) EWR (4.22%) JFK (3.65%) MCO (3.34%) FLL (3.23%)	ORD (5.25%) DEN (4.52%) EWR (3.39%) LAX (3.20%) ATL (4.05%)	ORD (4.71%) DEN (4.09%) EWR (3.90%) ATL (3.30%) SFO (2.85%)	ORD (5.68%) DEN (4.45%) SFO (3.79%) ATL (3.78%) LAX (3.30%)
Month (TOP 5)	August (12.34%) July (11.14%) December (9.81%) June (9.38%) November (9.34%)	August (10.85%) November (10.74%) December (10.37%) October (10.11%) September (9.73%)	November (11.66%) December (11.51%) October (10.99%) August (9.89%) September (9.28%)	November (11.74%) December (11.69%) October (10.34%) August (10.15%) September (9.05%)
DayOfWeek (TOP 5)	Thursday (17.30%) Friday (17.20%) Monday (15.36%) Sunday (14.92%) Wednesday (13.65%)	Friday (16.87%) Thursday (16.59%) Monday (15.14%) Sunday (14.93%) Wednesday (13.67%)	Monday (16.28%) Friday (16.27%) Thursday (14.64%) Tuesday (13.96%) Wednesday (13.20%)	Monday (16.28%) Friday (16.27%) Thursday (14.64%) Tuesday (13.96%) Wednesday (13.20%)
DepTimeBlk (TOP 5)	22:00-22:59 (26.49%) 21:00-21:59 (25.13%) 20:00-20:59 (19.27%) 19:00-19:59 (10.08%) 18:00-18:59 (06.55%)	17:00-17:59 (21.79%) 18:00-18:59 (19.05%) 19:00-19:59 (18.30%) 20:00-20:59 (12.41%) 16:00-16:59 (11.46%)	14:00-14:59 (17.55%) 12:00-12:59 (17.39%) 13:00-13:59 (16.60%) 15:00-15:59 (15.12%) 11:00-11:59 (13.22%)	08:00-08:59 (18.14%) 09:00-09:59 (17.02%) 07:00-07:59 (16.22%) 10:00-10:59 (14.98%) 06:00-06:59 (13.74%)
ArrTimeBlk (TOP 5)	00:01-05:59 (42.08%) 23:00-23:59 (33.34%) 22:00-22:59 (10.66%) 21:00-21:59 (4.91%) 06:00-06:59 (3.60%)	21:00-21:59 (20.08%) 20:00-20:59 (19.28%) 19:00-19:59 (18.56%) 22:00-22:59 (14.11%) 18:00-18:59 (13.86%)	16:00-16:59 (19.01%) 17:00-17:59 (16.71%) 14:00-14:59 (16.03%) 15:00-15:59 (16.02%) 13:00-13:59 (12.21%)	11:00-11:59 (18.29%) 10:00-10:59 (17.56%) 12:00-12:59 (14.84%) 09:00-09:59 (14.51%) 08:00-18:59 (10.24%)
DepDelay (median)	50 minutes	31 minutes	24 minutes	20 minutes
AirTime (median)	128 minutes	90 minutes	93 minutes	91 minutes
Distance (median)	977 miles	610 miles	632 miles	611 miles
ArrDelay (median)	46 minutes	29 minutes	23 minutes	20 minutes

As can be seen from the table 1, the late autumn-early winter months are the months in which most delays occur. We have already discussed above the characteristics of each cluster with regard to departure and arrival time slots. Now, however, let us analyse the numerical features in detail. The *Short-distance Late Afternoon Flights* cluster despite having a median *Distance* (610 miles) and median *AirTime* (90 minutes) similar to the *Short-distance Early Morning Flights* (611 miles & 91 minutes) and *Short-distance Early Afternoon Flights* (632 miles & 93 minutes) clusters, has a median *DepDelay* value 55% higher than *Early Morning* (31 minutes vs. 20 minutes) and about 29% higher than *Early Afternoon* (31 minutes vs. 24 minutes). We can sort the clusters in ascending order of *DepDelay*, obtaining *Short-distance Early Morning*, *Early Afternoon*, and *Late Afternoon flights*. Considering that the distribution of months is almost identical for each cluster, we can consider the *DepDelay* as a systemic delay that accumulates over the hours of the day. Furthermore, given the greater distribution of delays in the winter months, it could be assumed to be due to bad weather. Finally, looking at the days, it can be observed that it is always *Monday*, *Wednesday* and *Friday* on which the most delays accumulate. This could be related to a greater number of flights during these days. Interestingly, Sunday never features among the TOP 5 DayOfWeek. Finally, note the most peculiar and interesting cluster: *Long-distance Night Flights*. As we mentioned earlier, this is a cluster with a predominance of summer months (August-July-June) and is characterised by late evening departures and night-time arrivals. What stands out is that the median *DepDelay* is 50 minutes, the median *AirTime* is 128 minutes and the median Distance is 977 miles. Therefore, it can be said that this is a cluster with long-distance flights.

As observable in the figure 4a, some values of the *Long-distance Night* clusters overlap with both *Late Afternoon* and *Early Morning* (probably mid-distance flights with overlapping schedules for the different clusters). Finally, in the figure 4b , we can see that the centroids are well spaced apart and that the most peculiar is *Long-distance Night*, while the centroid that is closest to the average value of the delay dataset is *Short-Distance Early Afternoon*.



(a) 2D PCA with Clusters



(b) Radar Chart with Centroids+Mean

4 Classification

In this section we have implemented a few classification algorithms in order to predict 3 different variable. The first target variable was used **'ArrDel15'** which are flights that have delay in range of 15 minutes. Other task was to create a multiclass label using **'ArrivalDelayGroups'** which is the feature that contains intervals from -15 to 180 minutes of delay labeled from -2 to 12. As there were a lot of classes and really imbalanced we have decided to create a **'ArrivalDelayGroups_merged'** which contains just 4 different classes. In the figure 10 we are presenting the distribution of this target variable.

The third and the last prediction was computed on the variable **Cancelled** using the first type of dataset where we have kept all missing values in order to keep this class. In order to handle the large dimensionality of the data and the fact that the two classes were very unbalanced (98% of values for class 0, 2% of values class 1), we opted for undersampling technique that because is the fastest and the best by computational complexity.

4.1 ArrivalDelayGroups

For the prediction of ArrivalDelayGroups was not necessary to do a more detailed preprocessing as the dataset was already cleaned in the previous sections of this project. For this task was used the dataset mentioned in the previous sections where all missing values were dropped such that *Cancelled* and *Diverted* columns were dropped. The only important thing to mention was the decision to drop **'ArrDel15'** and **'ArrivalDelayGroups'** (not merged) this obviously because of its effective correlation. We were constraint to keep *'DepartureDelayGroups'* despite its correlation of around 63 % as this feature was crucial for predicting arrival delays this statement can be confirmed in the figure 6, where we can see that feature importance of this feature is much higher with respect to others (in the presented figure, the data has been visualized on a logarithmic scale). Also, we have computed correlation matrix from which we could decide to drop all highly correlated items like *'Flight_Number_Marketing_Airline'*,*'DistanceGroup'* and *'scaled_AirTime'*. Another important thing to mention is the label encoder used for *'OriginState'* and *'OriginCity'* in order to have them numerical. All other non numerical features were dropped. In the table 2 we can observe the original and a new merged classes with their corresponding interval description.

Classification algorithms picked for this tasks were Multilayer Perceptron, Random Forest, Logistic Regression, LinearSVM and Gradient Boosting Classifier. In order to compute GBClassifier and LinearSVM we have implemented OneVsRest strategy which is doing various binary classifications among the multiclass label.

Original Class	New Class	Description
-2	Class 0	Early more than 15 min
-1	Class 1	Early 0-15 min
0,1	Class 2	Delay 0-30 min
Rest	Class 3	Delay more than 30 min

Table 2: Original classes to merged classes

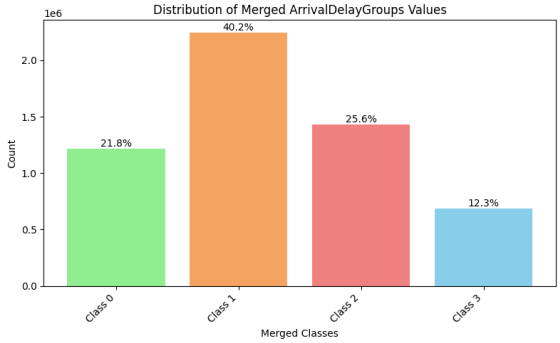


Figure 5: Correlation Matrix

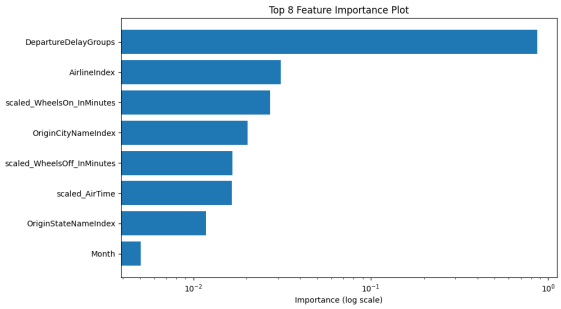


Figure 6: Feature Importance (RF)

Model	**Accuracy**	**F1-score**	**Parameters**
LinearSVC	0.51	0.37	linear, maxIter = 10, param = 0.1
GBTClassifier	0.57	0.54	maxDepth = 3, numIter = 10
RF	0.58	0.56	numTrees = 15, maxDepth = 10
LR	0.50	0.40	multinomial, max iteration = 10, reg_parameter= 0.1
MultilayerPerceptron	0.40	0.14	layers=[12, 10, 5, 4], blockSize=128, seed=1244

Table 3: Best Performances - LinearSVC, GBTClassifier, RF, LR , MultiLayerPerceptron

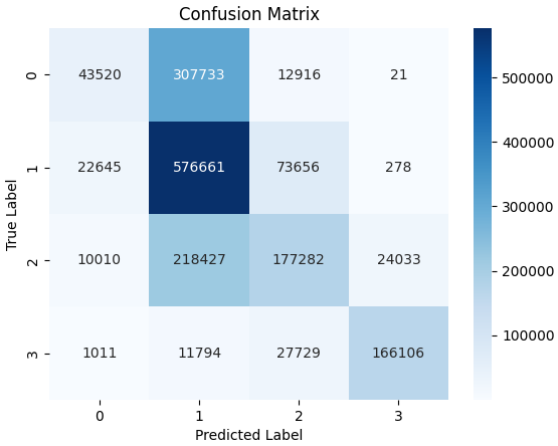


Figure 7: RF Confusion Matrix

In the table 4, we are providing the full classification report of the best trained Random Forest Classifier. We can observe that class 3 has the best F-score, as its precision and recall are all above 80%. The worst prediction was made on class 0 where recall falls into 12 %. This can be more specifically seen in the figure 7 where on its confusion matrix is clearly seen the bad true positive prediction of this class. Unfortunately, it can be concluded that the result of multiclass classification was not fully satisfactory.

Table 4: Classification Report for RandomForest Classifier

Class	Precision	Recall	F1-Score	Support
0	0.56	0.12	0.20	364,190
1	0.52	0.86	0.65	673,240
2	0.61	0.41	0.49	429,752
3	0.87	0.80	0.84	206,640
Accuracy			0.58	1,673,822
Macro Avg	0.64	0.55	0.54	1,673,822
Weighted Avg	0.59	0.58	0.53	1,673,822

4.2 ArrivalDelay15

The first dataframe was used in this type of classification. Next we focused on the binary variable ArrDelay15 which indicates whether an airplane has arrived within the 15-minute delay or is more than 15 minutes late. This feature from our point of view could be useful as a model, as it can predict with good accuracy whether an airplane will be on time or late, although it cannot classify how many minutes an airplane is actually late. To perform this task, we used 4 classification models that performed very well: MultilayerPerceptron, Linear SVM, Logistic Regression, and Decision Tree. the best results were obtained from multilayer perceptron and decision Tree which have significantly better values in the f1 score, also having good precision and high recall. As with the prediction of ArrivalDelayGroups, the most important and incisive feature is that of Departure Delay, which affects more than 80 percent in the final result.

Table 5: Performance Metrics for ArrivalDelay15

Model	Target Accuracy	Average F1 Score	Parameters
Multilayer Perceptron	0.92	0.87	layers = 12, 6, 4, 2, blockSize=128
SVM	0.88	0.74	maxIter=10, regParam=0.1
Logistic Regression	0.83	0.57	maxIter=20, regParam=0.2
Random Forest	0.93	0.88	numTrees=15, maxDepth = 10, 'entropy'

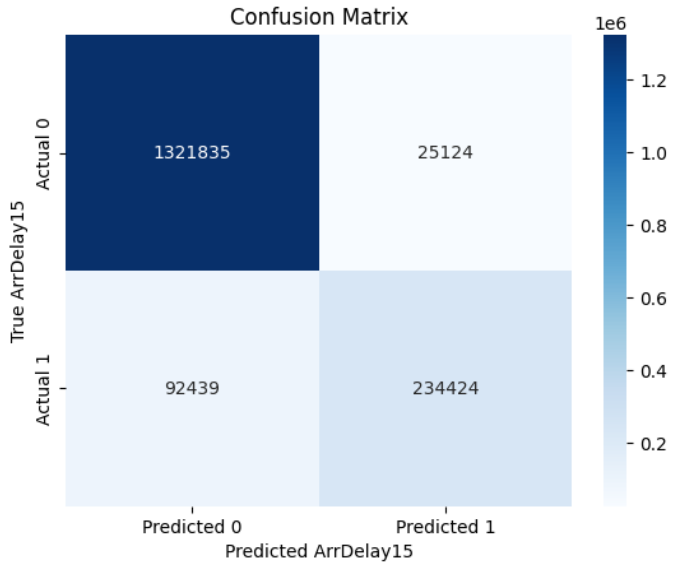


Figure 8: ArrDelay15 confusion matrix - Decision Tree

4.3 Cancelled

In this section, as mentioned earlier, we decided to use the second dataframe to predict Cancelled flights, after undersampling on the dataframe to equalize the number of values on which to train the model we attempted to use two classifiers MultilayerPerceptron and decisionTree. These were chosen because of the computational complexity of the dataset and the simplicity of these two classifiers to operate with a multiclass target variable. However, due to the lack of completeness of the dataframe features, the results were not extremely satisfactory, arriving at performances such as those shown in the table below. The Decision Tree performed better on all evaluation parameters, but does not achieve performance that would make the model appreciable on a large scale. From the features importance calculation, it can be seen that the airport of origin. day of the month and month are the columns that most influence the result. In contrast, other values such as departure and arrival time are not significant in determining whether a flight will be canceled or not. From this we can infer that whether or not a flight is canceled is little dependent on the hourly traffic at an airport, and is much more related to the location of the departure airport is by the day it departs, probably due to the weather and organizational conditions at the departure airport.

Cancelled Target	**Accuracy**	**Average F1 Score**	**Parameters**
Multilayer Perceptron	0.58	0.55	layers = [12, 6, 4, 2], blockSize=128, seed=1234
Decision Tree	0.73	0.73	maxBins=400,maxDepth=16, impurity = 'entropy'

Table 6: Performance Metrics- Cancelled Target

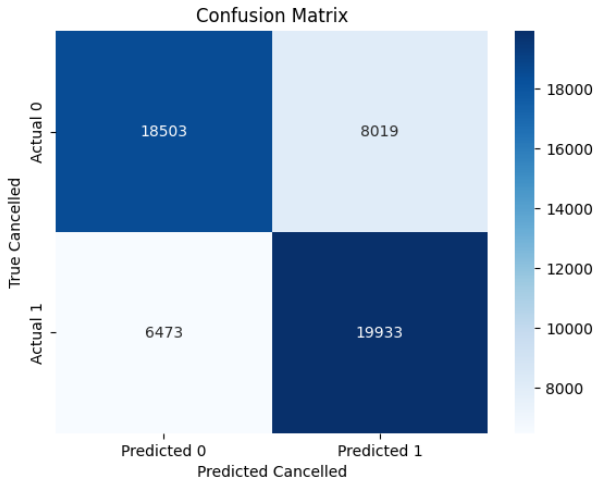


Figure 9: Cancelled confusion matrix - Decision Tree

5 Regression

We tried to model a Regression task to predict the Delay time of the vector. We tested the algorithm of **Linear Regression, Random Forest Regressor** and **Gradient Boosting Regressor**.

All the algorithms were optimized using the *ParamGridBuilder* which is a builder provided by pyspark that implements a grid-search model selection for the parameter of the algorithms. Unfortunately, the computational cost was huge so we abandoned this strategy to test a series of parameters and define the best ones. Only for Linear Regression, we were able to implement a "for loop" that would allow us to identify the best match between two hyperparameters. We set as a target variable **DepDelay** after being normalized and, instead, as an input feature we used ['Month', 'DayofMonth', 'DayOfWeek', 'ArrDel15', 'scaled_Distance', 'scaled_WheelsOff_InMinutes', 'scaled_WheelsOn_InMi
For this selection, we first implemented a correlation analysis using Pearson correlation, so as not to include related variables. We decided to use just these ones for a matter of necessity, the robust

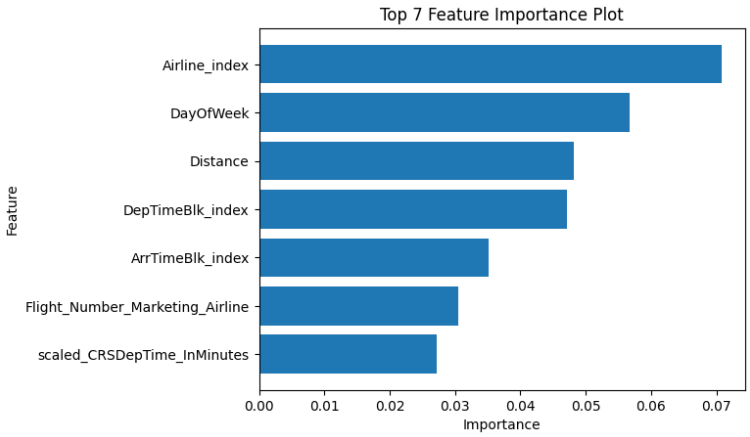


Figure 10: Cancelled features importance - Decision Tree

dropping of the pre-processing has forced us to choose just a few useful and meaningful resources. Every algorithm was evaluated using the three main metrics generally used for regression like **Root Mean Square Error, Mean Square Error** and **R$\hat{2}$**. In table 11 we show the parameters that we tried to analyze but that we could not get, in the last column we show the parameters actually used:

Algorithm	Parameter analyzed	Best parameter
Linear Regressor	maxIter = [50, 100] epsilon = [1.5, 2.0, 2.2]	maxIter = 20 epsilon = 1.5
Random Forest Regressor	maxDepth = [5, 10, 15] numTrees = [20, 30, 40]	maxDepth= 5 numTrees=20
Gradient Boosting Regressor	MaxDepth = [5, 10, 15]) maxIter = [20, 40]	MaxDepth = 10 maxIter = 20

Figure 11: Hyperparameter tuning - Regressors

As we exploit in the notebook, the linear regression wasn't tested using any kind of regularization because we believe that the numerosity of the features passed through input in the algorithms is quite low to allow us to avoid this kind of implementation.

The output of our algorithms is shown in the table 12:

Algorithm	RMSE	R^2	MSE
Linear Regressor	0.0097460	0.298218	0.00009498
Random Forest Regressor	0.0095564	0.325254	0.0000913
Gradient Boosting Regressor	0.009513	0.33133	0.0000305

Figure 12: Performances - Regressors

As you can see, the error seems to be absolutely low, quite equal to zero. For this reason, we move to evaluate the presence of overfitting. For this purpose, we tried both to display the learning curve, in order to better understand the motion of the error across the iterations and the error on the train and test set.

The second option seems not to be much more computationally efficient and faster, so we decided to abandon this technique. So we decided to calculate the error on the training and test set to see if there was any difference. We have done this analysis either using the RMSE metric and the R$\hat{2}$ metric and the results seem to be good, the errors in both train and test (in every algorithm) is good and convergent to the same point. Both errors in the train and test set go around 0.011 and 0.015.

Also the features importance of the Random Forest Regressor was inspired and it turns out from figure 13 that, above the seven features taken into account, ArrDel15 specifically if that flight has a delay greater than 15 minutes, was very important with the 87% of the total(the others are much less important), the distance doesn't look to be very important and this is strange.

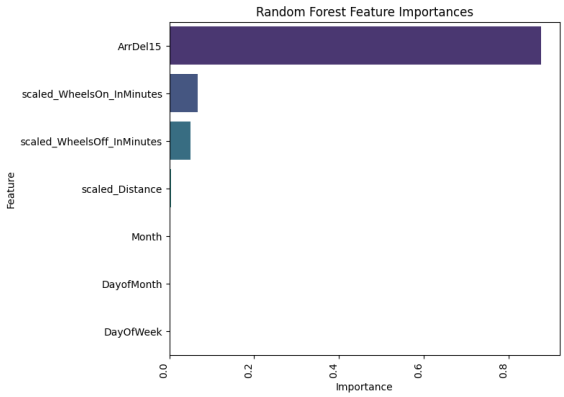


Figure 13: Feature Importance - Random Forest Regressor

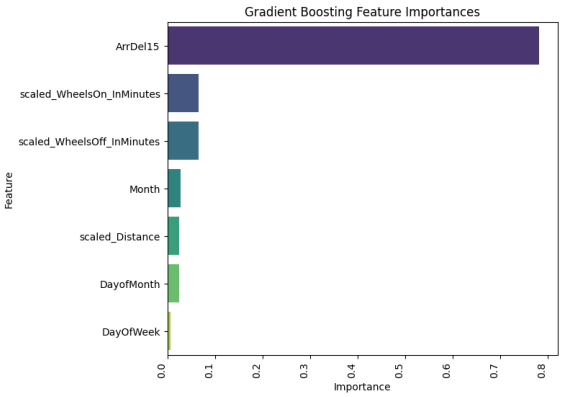


Figure 14: Feature Importance - Gradient Boosting Regressor

Furthermore, also Gradient boosting regressor was queried from the features importance task and we discovered in figure 14 a different opinion in this algorithm that, maintaining the main importance in ArrDelay15, it reserves also importance on Wheels On and Wheels Off(like for Random Forest) but also (anche se minima) a small importance also for the remaining features.

The **intercept** and **slope** for each feature were calculated for the Linear Regression. The intercept is 0.3271 and instead, the slope of the variables are [-2.8425861321413467e-05,-4.204061188538922e-06,3.4038584187264763e-06,0.015677330088980014,0.0006366600449219104,0.004025979600466158,-0.0031189900634890936].

The results show that some slopes have negative value and some positive, the problem is that they do not have a marked value, they are still very small and close to zero therefore it is difficult to define an appropriate modelling derived from these results.