Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Spark practical work

Big Data

20th december 2020

Jorge acosta hernández, Isabel Adarve Álvarez, jesús rodríguez corrochano

Data Science Master degree

Index

[1 Introduction 1](#_Toc59384290)

[2 Variables 1](#_Toc59384291)

[2.1 Original Data 1](#_Toc59384292)

[2.2 Forbidden Variables 2](#_Toc59384293)

[2.3 Excluded and selected variables 2](#_Toc59384294)

[2.4 Variable transformation and new variables created 4](#_Toc59384295)

[3 Machine Learning Techniques 6](#_Toc59384296)

[3.1 Linear Regression 6](#_Toc59384297)

[3.2 Decision Tree Regression 6](#_Toc59384298)

[3.3 Random Forest Regression 7](#_Toc59384299)

[4 Validation Process 8](#_Toc59384300)

[5 Final Evaluation 9](#_Toc59384301)

[6 How To Compile And Execute The Application 9](#_Toc59384302)

[7 Conclusion 9](#_Toc59384303)

[8 References 10](#_Toc59384304)

# Introduction

Have you ever been stuck in an airport because your flight was delayed and wondered if you could have predicted it if you had had more data? This is the main objective of our study: create an application to predict the flight delay.

The data that we have used consists of flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008.

# Variables

## Original Data

The original dataset [[1]](#Ref1) contains 28 variables [[2]](#Ref2) , plus the target variable ‘ArrDelay’, which are:

|  |  |
| --- | --- |
| Name | Description |
| Year | 1987-2008 |
| Month | 1-12 corresponding to January to December |
| DayofMonth | 1-31 |
| DayOfweek | 1-7 corresponding to Monday to Sunday |
| DepTime | actual departure time, in local time and in format hhmm |
| CRSDepTime | scheduled departure time, in local time and in format hhmm |
| ArrTime | actual arrival time, in local time and in format hhmm |
| CRSArrTime | scheduled arrival time, in local time and in format hhmm |
| UniqueCarrier | code of the carrier, corresponding to a supplemental dataset ‘carriers.csv’ [[3]](#Ref3) |
| FlightNum | flight number |
| TailNum | plane tail number |
| ActualElapsedTime | actual flight elapsed time, in minutes |
| CRSElapsedTime | scheduled flight elapsed time, in minutes |
| AirTime | air flight time, in minutes |
| **ArrDelay** | arrival delay, in minutes. It is the target column |
| DepDelay | departure delay, in minutes |
| Origin | code of the origin airport, corresponding to a supplemental dataset ‘airports.csv’ [[3]](#Ref3) |
| Dest | code of the destination airport, corresponding to a supplemental dataset ‘airports.csv’ [[3]](#Ref3) |
| Distance | in miles |
| TaxiIn | time of taxi in (before landing), in minutes |
| TaxiOut | time of taxi out (after landing), in minutes |
| Cancelled | if the flight was cancelled or not |
| CancellationCode | reason for cancellation (A = carrier, B = weather, C = NAS, D = security) |
| Diverted | 1 = yes, 0 = no |
| CarrierDelay | in minutes |
| WeatherDelay | in minutes |
| NASDelay | in minutes |
| SecurityDelay | in minutes |
| LateAircraftDelay | in minutes |

Table 1. Description of the variables

## Forbidden Variables

Some of the variables explained in the previous section cannot be used, which are:

* ArrTime
* ActualElapsedTime
* AirTime
* TaxiIn
* Diverted
* CarrierDelay
* WeatherDelay
* NASDelay
* SecurityDelay
* LateAircraftDelay

## Excluded and selected variables

We decided to drop some or the remaining variables, which are:

* DayOfMonth
* DayOfWeek
* CRSDepTime
* CRSElapsedTime
* FlightNum
* TailNum
* Origin
* Dest
* Cancelled\*
* CancellationCode

The ones that we are going to keep are:

* Year
* Month
* DepDelay
* DepTime
* TaxiOut
* UniqueCarrier
* Distance
* CRSArrTime

This selection is based in the following explanation.

Related to the dates, we have ‘Year’, ‘Month’, ‘DayOfMonth’ and ‘DayOfWeek. ‘Year’ feature does not provide information if we use one dataset, but if we perform the analysis with more than one dataset, it may be relevant, therefore we keep it. ‘DayOfMonth’ does not provide relevant information because it resembles, for example, the 1st of january with the 1st of november; it resembles two rows that are clearly different. The ‘DayOfWeek’ is also useless, as we can see in the following graphics; it represents the mean of the delay per flight that has been in each day of the week. We can see a minimum variation between the different days.

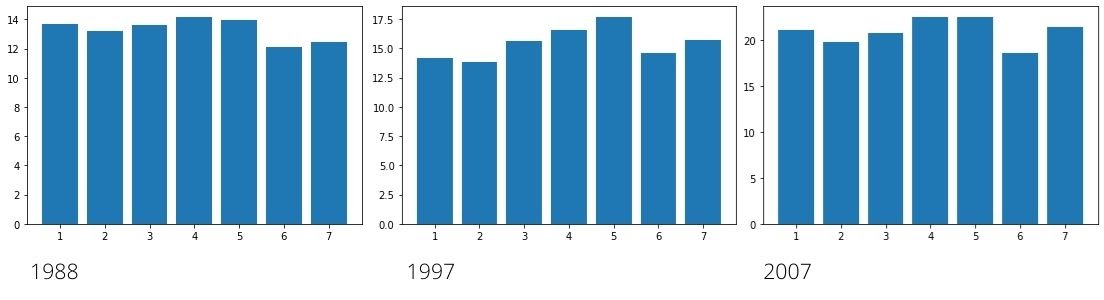


Figure 1. Delay per flight in each day of week.

Also we can see in the following graphics that almost the same happens with the feature ‘Month’; there is a relevant variation between the different months, so that we keep these two features. The graphics represent the mean of the delay per flight that has been in each month.

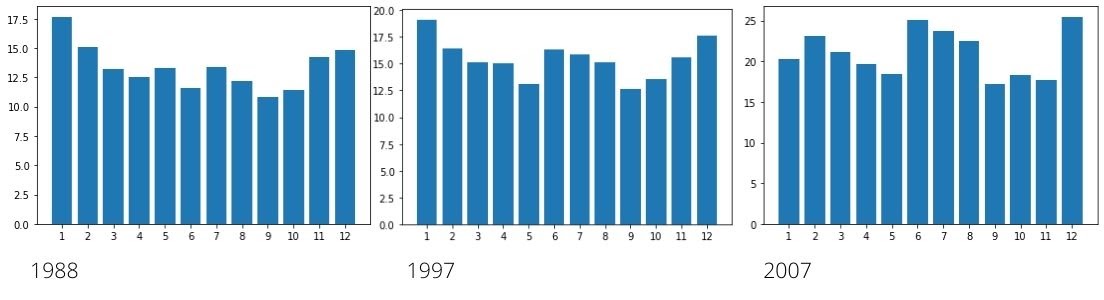


Figure 2. Delay per flight in each month.

Related to the flight departure we have several data, ‘CRSDepTime’, ‘DepDelay’, ‘DepTime’ and ‘TaxiOut’. We can see that the three first columns are redundant because knowing two of them we can obtain the remaining column, so we discard ‘CRSDepTime’ and we keep ‘DepDelay’ and ‘DepTime’, both of them give useful information for our study. The delay in departure ('DepDelay') could become a delay in arrival, which is what we want to predict.

We decided to keep the ‘ArrTime’, actual arrival time, because we realised in a previous analysis of the variables that the later is the arrival time of a flight, the more prone to suffer delay is.

Looking at the variables ‘Distance’ and ‘CRSElapsedTime’, we see that these variables are highly correlated since one refers to the flight distance and the other to the planned flight duration. That is, they are both giving us practically the same information about how long the flight is going to be; this is related to our target column since increasing its duration increases the possibility of being delayed. Therefore we decided to keep only one of them, the 'Distance' feature, and discard the other one, 'CRSElapsedTime'.

We also consider the features ‘Origin’ and ‘Dest’ useless for our study. ‘Origin’ and ‘Dest’ are string features with more than *200* classes, so we could only apply one hot encoding to split them into different integer columns, but the result would be almost *500* with practically vectors of zeros.

We keep ‘UniqueCarrier’, which is a string feature with between *10* and *20* classes that applying one hot encoding is splitted into the same number of columns than classes. In the following graphics, that represents the mean of the delay per flight that corresponds to each one of the flight carriers, we can see a relevant variation between the different carriers.

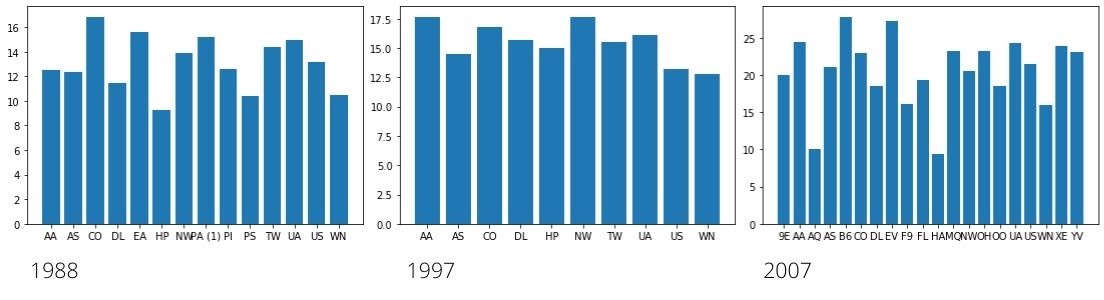


Figure 3. Delay per flight in each Carrier.

The feature ‘TailNum’ is the aircraft’s identity code, so we consider it does not provide any information related in some way with the ArrDelay.

The feature ‘Cancelled’ is used to filter the rows containing non-cancelled flights, because the cancelled ones do not provide information to our study. After that, we drop ‘Cancelled’ and ‘CancellationCode’ columns.

## Variable transformation and new variables created

We have made several transformations in the dataset. The first one is to replace the ‘NA’ string values into null values; ‘NA’ is not recognised as a common ‘na’ or ‘null’ value, so that it cannot be dropped and it needs to be replaced in order to do it.

We also convert strings to integer in the following columns:

* ArrDelay (target variable)
* DepDelay
* Distance
* CRSArrTime
* TaxiOut
* DepTime

The ‘TaxiOut’ feature has given some problems because it is not always filled with values, sometimes it is filled with ‘NA’ strings. We initially considered that the correct thing to do was dropping the column if the whole one was filled with null values, but we finally decided to replace the null values with the mean of the values corresponding to all the datasets (with real values and no ‘NA’ strings), which is *15*.

The ‘DepTime’ and ‘CRSArrTime’ variables have a time format of hhmm, that we do not consider useful for our study. With this format, the difference between the hours 7:59 and 8:00 is *41* minutes (*800 - 759 = 41*), but instead the real difference between 7:59 and 7:58 is one minute (*759 - 758 = 1*), so we decided to change this hhmm format to a minutes format. This is, for example, we transform *759* to *7 x 60 + 59 = 479* so we have that these columns are integers that go from *0* to *23 x 60 + 59 = 1439*.

The last transformation that we have made is applying the One Hot Encode to the categorical variables:

* Month
* Year
* UniqueCarrier

We drop these columns after creating the new variables:

* MonthOHE
* YearOHE
* UniqueCarrierOHE

The One Hot Encode splits each of the categorical variables into a vector with as many elements as classes have. The vector is filled with *0* and *1*. For example, if the value in the column ‘Month’ is *1*, the vector in the new variable ‘MonthOHE’ will be *(1, 0, 0, …, 0).*

# Machine Learning Techniques

To predict the flight delay we are going to apply three different machine learning techniques.

## Linear Regression

Linear Regression is a machine learning algorithm that predicts continuous values rather than trying to classify them into categories, as is the case of our target column ‘ArrDelay’. To do that, the model uses the known parameters, which are correlated with the target column, to make a continuous and constant slope which is used to predict the target column.

The linear regression model can be represented by the following equation:

where Y is the predicted value (target column), is the bias term, are the parameters, and are the feature values.

The goal of this algorithm is to find the parameters that minimize the error function

When the parameters are found, the line defined by the equation is called a regression line, and the errors are the residuals.

## Decision Tree Regression

The decision tree regression is a greedy algorithm that performs a recursive binary partitioning of the feature space. The tree predicts the same label for each bottommost partition. Each partition is chosen greedily by selecting the best split from a set of possible splits. The algorithm does not consider all the possible splits because there may be a several number of them, but just the number of splits previously fixed, (we have to tune it according to our dataset) named as MaxBins, using a heuristic algorithm to select the splits to consider. The best split is the one that maximizes the information gain at the tree node. This information gain is calculated using the impurity of the nodes and the following formula:

Being D the dataset of size N, splitted (s) into two datasets, and of sizes and .

The impurity of a node is defined as:

where is the label for an instance, N is the number of instances and μ is the mean given by

The recursive tree construction is stopped at a node when the node depth is equal to the fixed threshold (in our case *10*), or no split candidate leads to an information gain greater than the fixed threshold.

## Random Forest Regression

The Random forest regression is an ensemble of regression decision trees. This algorithm consists of training a set of regression decision trees (in our case we use *20*) separately. The algorithm injects randomness (bootstrap) into the training process so that each decision tree is a bit different. Apart from this randomization, regression decision trees are trained in the same way as we have already explained before. To make a prediction on a new instance, the algorithm averages the predictions from its set of regression decision trees.

So as in the regression decision tree case, the parameter we have to tune according to our dataset is the number of splits to consider in each growing step.

# Validation Process

We are facing a regression problem so in order to measure / validate / compare the different machine learning models, we use two metrics:

* Root Mean Square Error (RMSE): this is a way to measure the error of a model in predicting continuous target variable. It is defined as:

Where N denotes the number of observations, denotes the predicted value for the i observation and is the observe/real value for the i observation

* Coefficient of Determination (): this coefficient represents the percentage of the target variable variation that is explained by the model. It is defined as:

Where N denotes the number of observations, denotes the predicted value for the i observation, is the observe/real value for the i observation and denotes the mean value of the target column.

Looking at the formulas of each one of the measures, we have that these measures are inversely proportional, that is, if the RMSE coefficient decreases, increases, and vice versa. Therefore comparing several models taking into account one of the measures previously mentioned will be similar to compare them taking into account the other measure.

To select the most suitable hyperparameters for our data for each of the machine learning techniques applied, we use a validation process known as cross-validation, which consist in:

1. Configuring a set of values for each of the parameters that we want to set.
2. Selecting a model performance metric to compare the different models; we will use the Root Mean Square Error (RMSE) explained in the previous section.
3. Dividing the dataset in k parts. For each one of the possible parameters combinations, the cross-validation makes k analysis, using k-1 parts of the dataset in each one of them to train it, and the remaining part to test it. So, for each of the parameter’s combinations, we obtain the average of the RMSE measures corresponding to each of the k analysis performed.
4. The model (parameter’s combination) with less average of RMSE is chosen.

In this validation process we have decided to use the RMSE as the suitable measure to evaluate the performance of the different models, instead of the measure even though they are equivalent, just because our main objective is to predict the flight delay so the error made in this task is what we want to highlight.

# Final Evaluation

Finally, in order to validate and evaluate the performance of the different implemented machine learning algorithms in our application, we apply it with the dataset corresponding to the year *2008*. We have obtained the following results:

* For the Linear Regression technique that we defined in the [section 3.1](#_Linear_Regression), we obtain that the suitable hyperparameters setting is the following: Elastic net = *0* and Reg Param = *0.1*. Regarding the coefficients associated with each variable, it should be noted that of the continuous variables, the variable 'DepDelay' is the one with the highest coefficient, therefore more important when predicting the flight delay as we could expect, and 'TaxiOut' also has a high coefficient compared to the rest. Regarding the categorical variables (to which we have applied the one hot encode), we see that some of the 'UniqueCarrier' classes have very high coefficients, highlighting the fact already predicted in the [Figure 3](#F3) that some of these classes have associated a longer delay. Applying this model we obtain a Root Mean Square Error (RMSE) equal to *10.79*.
* For the Decision Tree Regression algorithm defined in the [section 3.2](#_Decision_Tree_Regression), we obtain that the best configuration of hyperparameters is MaxBins = *30*. We obtain a RMSE = *18.85*.
* For the Random Forest Regression algorithm explained in the [section 3.3](#_Random_Forest_Regression), we obtain that the best configuration of hyperparameters ir MaxBins = *30*. We obtain a RMSE = *20.83*.

For this dataset we can see that the algorithm that better perform according to the Root Mean Square Error is the Linear Regresion. In addition, in this model we can check that the coefficients associated to each variable agreed with the previous analysis that we have done in the [section 2](#_Variables).

# How To Compile And Execute The Application

The use of the requirements.txt file is required to install the dependencies in a venv using pip.

The input data can be placed in any directory, you just need to give the path to the python script, for example ‘python3 main.py /home/user/data/1987.csv’. It is also possible to give more than one file to the script, for example ‘python3 main.py 1987.csv 2005.csv’.

# Conclusion

From our point of view, the realization of this work has been a great end point to the Big Data course, since we have been able to put into practice what we have learned in addition to serving us as an introduction to a new programming language for us such as spark, implemented in python using pyspark.

During the performance of this work, all team members have improved our skills in analysis, selection and transformation of variables, as well as performing data cleaning and implementation of different machine learning algorithms to predict a continuous variable, also configuring its parameters. Proof of this knowledge is the application developed, where we have had to select the variables to use, treated columns or rows with null elements, categorical variables or with formats other than those usually used.

# References

[1] Variable descriptions of the dataset, <http://stat-computing.org/dataexpo/2009/the-data.html>

[2] Dataset ‘Data Expo 2009: Airline on time data’, <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/HG7NV7>

[3] Supplemental data, <http://stat-computing.org/dataexpo/2009/supplemental-data.html>

[4] Spark guide, <http://spark.apache.org/docs/latest/ml-guide.html>

[5] Pyspark guide, <http://spark.apache.org/docs/latest/api/python/pyspark.ml.html>