Universidade Politécnica de Madrid

Master Universitario en Ingenieria Informatica

Arrival Delay of Commercial Flights

Teachers:
Antonio Latorre De La Fuente
Pablo Toharia Rabasco
Jesus Montes Sanchez

Authors:
Daniel Reis
Afonso Castro

February 10, 2018, 1º Semester
Academic Year 2017/2018,

Abstract

This report aims to give a brief explanation of the work we developed for the course "Big Data". The project consisted in developing a model for predicting the arrival delay of commercial flights using Spark's Machine Learning Library (MLib). We were given access to data from commercial flights from 1987 to 2008 to achieve this task.

With this project, we wanted to broaden our knowledge in the Machine Learning field, as well as learn to use new technologies such as Spark and Scala. We believe we achieved both our objectives and our teachers', when they proposed to us this practical work.

Contents

1 Variables selected for the model

Among the 29 variables that the dataset contains, 1 is the target variable for the prediction model. 10 of them contain information that is unknown when the plane takes off and therefore can't be used for predicting the delay in commercial flights. Which means 18 variables remain. Out of those, the following variables were picked:

- **Month**, **DayofMonth** - If the flight is in the winter, for instance, there is a bigger possibility of snow, which means a bigger possibility of variance in flight time. Thus, three important variables for the model.
- **DayOfWeek** - Some days of the week are busier than others. Busy weekdays can lead to things like delays in departure time.
- **DepTime**, **CRSDepTime** - The departure time, when compared to the predicted departure time, is a key factor in determining the arrival delay of the flight.
- **UniqueCarrier** - The company operating the flight is important in determining the arrival delay, since some companies have stricter policies than others in order to avoid arriving late.
- **CRSElapsedTime** - Bigger flights lead to bigger variation of flight duration.
- **DepDelay** - If the plane is late, chances are it will arrive late too. Thus, also a crucial variable for determining arrival delay.
- **Origin**, **Destination** - Weather of any of the locations, or how well the airports function, can influence arrival delay.
- **TaxiOut** - The time the plane takes to take off can also directly influence flight delay.
- **Distance** - Bigger distance leads to bigger possibility of problems and variance.
- **Cancelled** - This variable is used not because it influences arrival delay, but because a cancelled flight didn't arrive and thus, shouldn't be considered. So, it's used to remove cancelled flights.

2 Variables excluded from the model

Among the 18 eligible variables, 14 were picked. Which leaves us with 4 variables that weren't. They are:

- **Year** - The year the flight is in doesn't influence the arrive delay of the flight.
- **FlightNum** - The number of the flight is just an identifier of the flight, it doesn't make sense to include it in the model.
- **TailNum** - Like the flight identifier, the plane identifier doesn't directly tell us anything useful about the plane. However, if a plane arrives late to a certain destination, that will most likely influence its subsequent flights. Unfortunately, the process required to make the model support this would be computationally very expensive, so we ended up not implementing it for the delivery.
- **Cancellation Code** - All that matters for us is that if the flight was cancelled. If it was, it is not considered in the model. The reason is irrelevant for us.
- **CRSArrTime** - The expected arrival time of the plane can be determined through the expected departure time and the expected duration of the flight and is therefore unnecessary.

Like mentioned earlier, 10 variables were forbidden because they cannot be known when the plane takes off and thus, are not used in the model. They are:

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

3 Description of the variable transformations performed

Some variables are not usable in their raw state. Hence, they need to be transformed to be useful for the model.

The variables **DepTime**, **CRSDepTime**, **CRSArrTime**, **CRSElapsedTime** represent time in the hhmm format. After some experimentation, we determined that using the minutes was detrimental for the model, so we took them out and use a function created by us to parse the hours from those columns. Variables that represent numbers (like **Month**, for instance) were converted to Integers.

The variables **Month**, **DayOfMonth**, **DayOfWeek**, **Origin**, **Dest**, **UniqueCarrier**, **CRSDepTime**, since they are discrete variables, were indexed and converted into categorical variables.

These variables were also normalized in order to produce better and more accurate results.

4 Machine Learning Technique Used

The Machine Learning technique that our group decided to use was **Random Forest**. Random Forest is an ensemble algorithm (which means, an algorithm that makes use of multiple algorithmsto obtain better performance) that makes use of several decision trees in order to reduce the risk of overfitting. Said decision trees are trained separately, so the training can be done in parallel. Randomness is injected into the training process so that each decision tree has some slight variations. Then the predictions of each tree are combined, reducing variance and improving the quality of the final result.

4.1 Parameters

The following parameters are used for the algorithm:

- **numTress**: 100 - Number of trees in the forest.
- **maxDepth**: 9 - The maximum depth of each tree of the forest. Increasing the depth makes the model more expressive and powerful. However, deep trees take longer to train and are also more prone to overfitting.
- **maxBins**: 2700 - Bins are ordered sets of split candidates. This value is approximately the square root of the total size of the dataset, since it is the standard the literature suggests.
- **featureSubsetStrategy**: auto - Defines the strategy for choosing features for the model. We decided to let the algorithm choose the best strategy.

4.2 Reasons for choosing the algorithm

The first thing we immediately recognized is that we were facing a **supervised learning** problem, since we have example values for the target variable. Most of our variables are discrete and the ones that aren't can be easily split. Although methods such as linear regression and Naïve-Bayes also work with discrete variables, we felt that **decision trees** were more appropriate for our problem. So, we decided to use them.

However, decision trees are very prone to overfitting. Which led us to use the **Random Forest** algorithm, since the multiple-tree approach means the resulting tree is much less likely to fall into overfitting.

5 Validation Process

The dataset was split into a training set and a test set. The train set is composed of 70% of the dataset, whereas the remaining 30% go to the test set.

After the training set is used to generate the model, said model is applied to compute a value for the *ArrDelay* for the test set. After that, the computed values are compared with the values for the *ArrDelay* present in the dataset, using the following model evaluation error metrics:

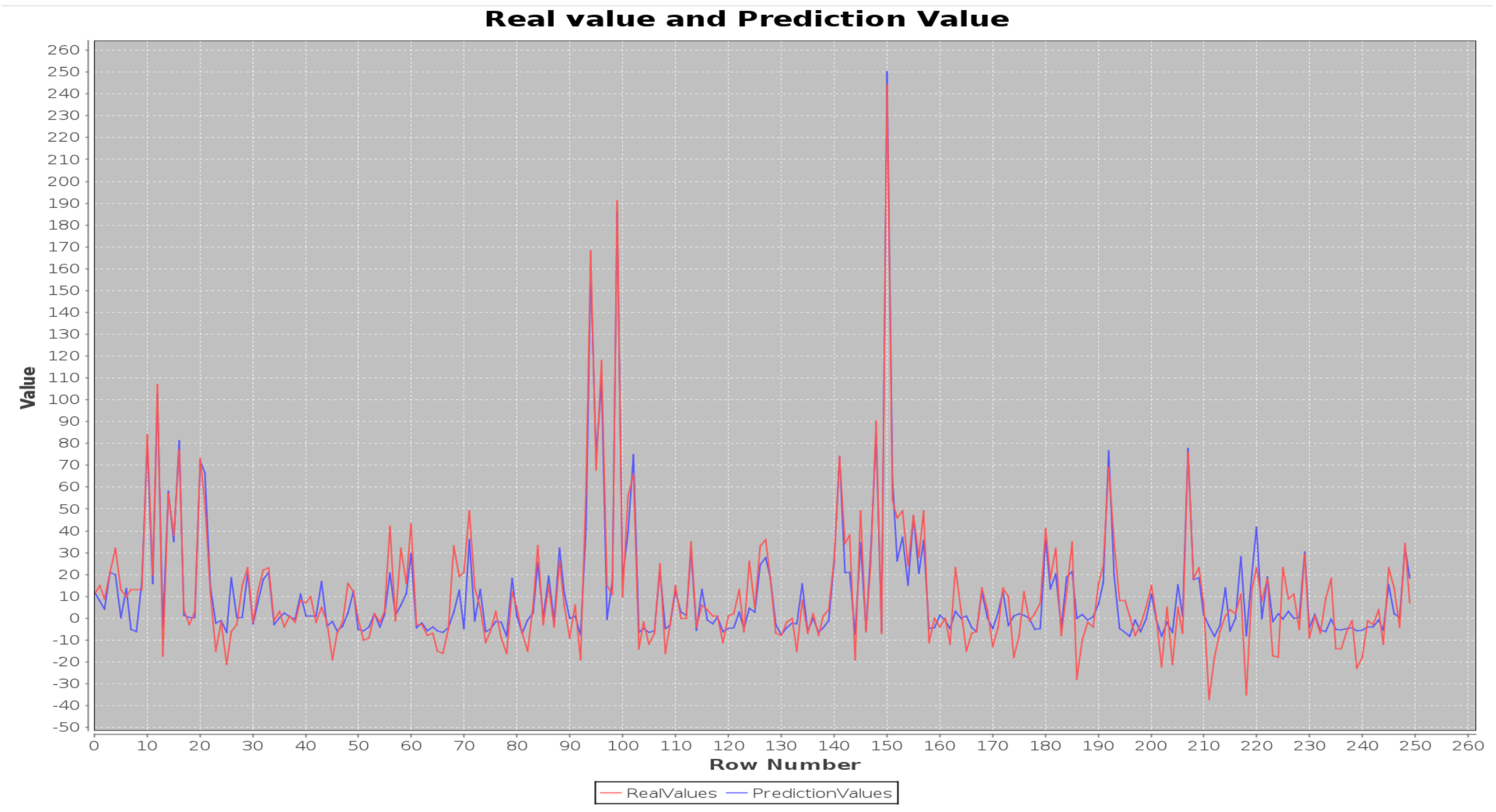
- **Mean Square Error (MSE)** - Corresponds to the average of the squares of the errors. The errors are the difference between the expected and the computed results.
- **Root Mean Square Error (RMSE)** - Corresponds to the square root of the mean square error. Both this metric and the Mean Square Error are very prone to big errors, which are highly undesirable in this scenario, hence the choice of MSE and RMSE over, for instance, Mean Absolute Error.
- **R^2** - Varies between 0 and 1 and determines how much can the independent variables (which means, the variables used to infer the model) explain the results obtained for the dependent variable (the *ArrDelay*). This metric allows us to understand the variance between the computed values and the dataset values.
- **Explained Variance** - Measures how much does the model account for the dispersion of the variables in the data set. Like R^2, this metric also allows us to understand the variance between the computed values and the dataset values.

6 Evaluation of the prediction model

After the model has been predicted, it must obviously be validated. In section 5 we listed the model evaluation error metrics we are using. We got the following results for each metric:

- **Mean Square Error**: 150.26539159560804
- **Root Mean Square Error**: 12.25827849233358
- **R^2**: 0.8640003733339979
- **Explained Variance**: 1492.3143310481785

In order to better visualize the results obtained, we decided to make a small plot with the first 250 results of the test. In this plot we can observe the value of ArrDelay that effectively had flight, as well as the forecast made by the algorithm.



As we can see from the values obtained and the plot, the model was capable of predicting flight delay with success. The group considers both an RMSE of 12.3 and R^2 of 86.4% good results, since they mean the errors are not too bug and that there is not too much variance in the results. That can be confirmed by looking at the plot, where we can see exactly that.

7 Instructions on executing the application an input data

In order to execute the application, one must have Java 8 and Scala 2.11.8 installed. Once they are installed, the user just needs to create the project using the defined *pom.xml*,build the project and run it. To run the project, all that has to be done is run the “Main” scala script file.

In regards to input data, the dataset to use should be defined in the value ”TEST” in the file */src/main/scala/explorer/FileExplorer.scala* where it’s possible to define several files in order to make a faster switch between datasets.

8 Final conclusions

After concluding this project, we consider we understood all the topics asked and managed to solve the problem proposed successfully.

Developing our project allowed us to broaden our knowledge in topics such as Scala, Machine Learning, Clustering and Spark, and both of us enjoyed developing it thoroughly.

We believe we achieved all the expected goals. The project also increased our interest in the Big Data area. Overall, both of us believe the experience was very positive.

We can also pinpoint some possible improvements that we could do to the model, like, for instance, the one referred in the **TailNum** variable in section 2.