Department of Informatics

Big Data Project 1

A Big Data and Machine Learning estimation of Flight Delays

Authors:
Alexander Sandström, Eliam Kourie, Karl Rosengren

December 5, 2021

Contents

1 Variables

1.1 Variables Included

The project started on square-one with discussions of what variables would be relevant for predicting delay and without under- or overfitting. We also had to exclude variables which would mean an increase in the complexity of our work, without the model reaping any significant rewards. In the end we decided to include the following variables:

ArrDelay As the intention of the project is to create a model which can predict Arrival Delays of a commercial flight, it was obvious that this parameter would be included in the model and also used as the target for our model's prediction.

Month As a month strictly describes what part of a year that is refered to by it's name and with this includes a change in weather depending on what month of the year it is we found this to be a valuable variable. Since heavy winter weather could be much harder to fly and land a flight in on the contrary to nice calm spring weather for instance.

Day of the week The day of the week was also included since we consider it probable that some days have heavy traffic, e.g Sunday where a lot more people are traveling than on a tuesday.

CSRDepTime Considering the Departure Time we had a similar line of thought compared to the day of the week, there should be a certain hour of the day that has more flights scheduled than another, we could call this the rush-hour. The rush-hour should have a higher probability of delays as the flight schedule should be more complex.

Carrier To identify any differences by specific airlines we included the UniqueCarrier variable to give our model the ability to learn that a specific airline has more delays than another and therefore increase the model's ability to predict the Arrival Delay.

CSRElapsedTime The elapsed time of the flight should be a variable of value as when the length of the flight increases there should also be more things that could go wrong which could lead to more delays.

TaxiOut TaxiOut could have an effect as delays caused by weather or airport management would show in this variable.

Distance Longer distance flights might be more prone to delay.

1.2 Variables Excluded

The 9 variables which we could not use we excluded with creating a list of the variable names and dropping them.

Cancelled Flight There is no need to check the delay of a cancelled flight as we do not know if the flight ended up having a delay at all or was incredibly fast. As we have zero information on what the delay status of the flight was, we decided that it is of zero value to keep it in the model.

Cancellation Code Since cancelled flights are not used in the model a cancellation code is not of any significance.

Day of the month Although we included day of the week and month we decided to not include the Day of the month since for example the 15th of August and 15th of December are nothing alike when it comes to weather or anticipated amounts of flights.

Year The year of a flight was not included as a delay is not dependent on the year besides for any technical advancements during later years. Despite this, it is indirectly included since there is data from a wide range of years that our model is using for the predictions. Although our reasoning regarding his stays the same, we did discuss how some years where different kinds of crisises occured could have an effect on delays.

Flight Number A specific number that correlates to the specific flight on a specific day could not be related to a delay. When taking this in to consideration we decided that it is not something that would be useful for us.

Tail Number Although there could be minor differences in plane models and/or if any aircrafts have had a decrease in performance, we decided to not include this due to it being of little significance as we assume that an industry standard of aircraft performance is used.

Departure Delay The Departure Delay is a part of the Arrival Delay, as such we decided that the Arrival Delay is sufficient for our model. Specifically as the Arrival Delay is our target variable we do not want a variable that is directly correlated related to it.

1.2.1 Forbidden Variables

Besides the variables above that we decided to exclude ourselves, there were ten forbidden variables that according to the project description should not at any time be used in the model.

1. ArrTime
2. ActualElapsedTime
3. AirTime
4. TaxiIn
5. Diverted
6. CarrierDelay
7. WeatherDelay
8. NASDelay
9. SecurityDelay
10. LateAircraftDelay

2 Descriptions and Justifications

2.1 New Variables

There are no new variables that we've added to the model besides variables that were already available in the datasets. For example, we wanted to map weather data to the origins and destinations provided in the data. This is something we would have wanted to do if there was time to further improve our model.

2.2 Variable Transformations

We appropiately changed the time variable of the departure time which is available as a string e.g 11:45 in the datasets to be represented as integers. We also indexed the "Unique Carrier" variable to be able to add them to the model.

2.3 Machine Learning Technique

We used both regression in two different forms; Linear Regression and Random Forest. The linear regression was added as a simple machine learning model but ran the risk of underfitting as it couldn't use all the variables we wanted.

We ended up using the Random Forest Regressor. This model creates multiple decision trees and takes the average out of them, reducing the risk of overfitting that normal decision trees would cause. This model would also allow us to use categorical values.

2.4 Validation Process

The validation was done through comparing the measured error through the root mean square error (RMSE) and comparing the measured error to the standard deviation of the actualt arrival delays. The error was within one standard deviation of the variance off ArrDelay.

2.5 Selection of input data

We tried with different subsets of the data but ended up running the model on the latest datasets, 2006 and 2007, as 2008 only contained data for the four initial months of the year and would thus not provide accurate predictions for datasets with more months than four.

3 Final Evaluation of the model

As for the final valuation of our model we ran into a few problems and have thought of what could be done if more time was at our hands. First of all there were problems with downloading the larger datasets from 2005 and 2007 which led to our results being entirely based on the year 2008. The problem with this beside not having a lot of data, the data of 2008 is not even complete, as it contains no data from the first four months of the year. Something which ultimately leads to the model not having sufficient information to work properly.

The following is our correlation matrix for the used variables and as we can tell from it most of it has very low correlation and as such should add something of value to our model with new datapoints.

Month	DayOfWeek	CRSDepTime	UniqueCarrier	CRSElapsedTime	Distance	TaxiOut
1.0	-0.00074	-0.001	-0.0035	0.002	-0.024	-0.003
-0.00074	**1.0**	0.006	0.015	0.017	-0.0104	0.005
-0.001	0.006	**1.0**	-0.034	-0.036	0.002	-0.003
-0.0035	0.015	-0.034	**1.0**	0.9800	0.1349	0.039
0.002	0.017	-0.036	0.9800	**1.0**	0.084	0.015
-0.024	-0.010	0.002	0.134	0.084	**1.0**	0.1349
-0.003	0.005	-0.003	0.039	0.015	0.1349	**1.0**

With this information we can conclude that we should not exclude any of them, although with more time it would be possible adding additional variables of value, such as weather data, to see if that could improve the prediction power of our model.

The method we choose to measure the error was the Root Mean Squared Error (RMSE) which resulted in an error of 37.094848224182634. This error is by our calculations rather high and should be improved.

The choice of decision trees in our predictions and not linear regression is due to the fact that we have some categorical variables in the model and also chose the Random Forest clustering to avoid overfitting of our model.

With all of this in mind, there is a lot for us to improve on the model and with more time this could be done but as of now there is not much for us to do besides being happy with the results and what we've learned during this project.

4 Instructions on How To Use Our Model

The program is written in Scala as a SBT project and is run most easily through packaging and running it through spark's submit script (see in further detail below). It can also be run directly from the spark-shell for a more interactive experience - although a deeper understanding of the different program components and their methods is necessary for this. The program can be used either can union data from multiple input files into a single data frame as long as the files are in the same directory.

4.1 Instructions on how to build and execute with spark-submit

The following instructions assumes that the program files are already downloaded through the project submission and that the necessary programs, i.e. scala, spark and SBT are already installed. First comes a general description of the different steps required and then follows a list of the commands in order.

4.1.1 Input files

Firstly, the input files have to be downloaded from `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/HG7NV7` and placed in a directory preferably close to the project folder as the program uses the relative path.

4.1.2 Build with SBT

Then, SBT should be used to build the jar-file using the sbt package command.

4.1.3 Submit to spark

After that, the program can be submitted to the spark-submit script using the appropriate arguments (examples shown further below). The first program argument should be the directory in which the files are in, and the following ones the files you want to use for the model. The program only takes CSV-files provided to the program without filename extension (".csv").

An example submit to local machine, with data files placed in the project directory under directory flight_data, would be: spark-submit –master local target/scala-2.12/sparkapp_2.12-1.0.0.jar flight_data/ 2002 2005 2007.

4.1.4 Example

1. Download the project and enter the project directory
2. Package using sbt package
3. Submit using spark with desired model parameters: "spark-submit –master local target/scala-2.12/sparkapp_2.12-1.0.0.jar path_to_directory file file"

The program will print the feature columns used, followed by a few lines of the prediction dataframe where the model has tried to predict the target variable. Then the RMSE for the model will be printed.

4.2 Spark-shell

Using the spark-shell can be a bit cumbersome when dealing mith many parts of a program but is still a really good tool to during development as you don't have to build and submit the program each time you change something small. Using the program through the shell requires a bit more knowledge about the methods themselves so it is suggested to have the source code at hand while doing so.

To use the app through the spark-shell

1. Download the project and enter the project directory
2. Enter the spark-shell using spark-shell
3. Load the files ML and PreProcess into the shell
4. Use val df = PreProcess.loadDF(spark, path) to load a single file or val df = PreProcess.loadDF(spark, Array(path_to_directory, file, file))
5. Create an instance of ML using val ml = ML(spark, df, "ArrDelay", List("UniqueCarrier")
6. Run the different models using for example: ml.randomForest(0.8,0.2) or ml.linearRegression

5 Conclusion

Although we would have wanted to spend more time refining the model our measures of correlation between the input variables showed quite good results in the end.

6 Personal Remarks

6.1 Alexander Sandström

Using the framework was a bit tough at the start, but while learning more and more about the toolset it became easier and also more fun. It was especially nice using the spark-shell to quickly see results when changing the model although it required some restructuring allowing the shell's SparkSession to be used. All in all a fun and intense project, while there also was plenty more we would have liked to do which time did not allow!

6.2 Eliam Kourie

According to me it was a tough but very suitable project, this led to us using our current knowledge of statistics, Machine Learning and common sense to put together a valid model. It has been a great project and if I could do it again I would like to fix the problem with the datasets to be able to run the model for a bigger dataset to achieve better results.

6.3 Karl Rosengren

This has been a very interesting course and an even more interesting project. In the beginning things were tough as to figure out what we were able to do in a, for a majority of the group, new framework. As we progressed further and further with the project things eased up and we were able to work more freely. We would like to have done even better for our final version of the model but there was not quite enough time for us. Hopefully next time!