Spark Practical Work. Big Data

Group 2
Pablo Crucera Barrero
Javier Gallego Gutiérrez
Júlia Sánchez Martínez
Master's Programme in Data Science

December, 2021

Contents

1 Introduction

Big data has become a field of study of particular interest due to the exponential growth in data generation in the last years. Bigger volumes of information imply computational costs both in volume and space that are not manageable using conventional algorithms. It is for this reason that Big Data frameworks like Hadoop and Apache Spark, which allow performing operations over data with several purposes, have emerged in the last decades and are acquiring more importance every time.

In this work, we will implement a prediction model of the arrival delays of airplanes in US domestic flights given some input variables relating to the flight information prior to the plane landing. For that, we will use machine learning techniques from the MLlib library of Apache Spark in Scala.

2 Preprocessing and Feature Selection

The data is presented in different CSV files, each of them containing information about all the recorded domestic flights for a different year. The list of all the input variables is presented in Table 1 [4].

Variable	Description	Type	Format
Year	Year of the flight departure.	Int	YYYY (continuous integer, range [1987, 2008])
Month	Month of the flight departure.	Int	MM (continuous integer, range [1, 12]
DayOfMonth	Day of month of the flight departure.	Int	DD (continuous integer, range [1, 31]
DayOfWeek	Day of week of the flight departure.	Int	1=Monday - 7=Sunday (continuous integer, range [1,7])
DepTime	Actual departure time, in local time.	Int	HHMM (discontinuous integer, range [0, 1439])
CRSDepTime	Scheduled departure time, in local time.	Int	HHMM (discontinuous integer, range [0, 1439])
ArrTime	Actual arrival time, in local time.	Int	HHMM (discontinuous integer, range [0, 1439])
CRSArrTime	Scheduled arrival time, in local time.	Int	HHMM (discontinuous integer, range [0, 1439])
UniqueCarrier	Unique carrier code.	String	Original code.
FlightNum	Flight number.	Int	Original flight number (continuous integer)
TailNum	Tail number.	Int	Original tail number (continuous integer)
ActualElapsedTime	Actual elapsed time, in minutes.	Int	MM (continuous integer)
CRSElapsedTime	Scheduled flight duration, in minutes.	Int	MM (continuous integer)
AirTime	Actual time in the air during the flight, in minutes.	Int	MM (continuous integer)
ArrDelay	Actual arrival delay, in minutes. Target variable.	Int	MM (continuous integer)
DepDelay	Actual departure delay, in minutes.	Int	MM (continuous integer)
Origin	IATA airport code of the origin of the flight.	String	Original code.
Dest	IATA airport code of the destination of the flight.	String	Original code.
Distance	Distance between origin and destination, in miles.	Int	Distance in miles (continuous integer)
TaxiIn	Taxi time in the destination airport, in minutes.	Int	MM (continuous integer)
TaxiOut	Taxi time in the destination airport, in minutes.	Int	MM (continuous integer)
Cancelled	1 if the flight was cancelled, 0 otherwhise	Int	0 or 1 (boolean)
CancellationCode	Reason for cancellation (A = carrier, B = weather, C = NAS, D = security)	String	Original cancellation code (char)
Diverted	1 if the flight was diverted, 0 otherwise	Int	0 or 1 (boolean)
CarrierDelay	Air carrier delay, in minutes.	Int	MM (continuous integer)
WeatherDelay	Delay due to weather, in minutes.	Int	MM (continuous integer)
NASDelay	National Aviation System delay, in minutes.	Int	MM (continuous integer)
SecurityDelay	Delay due to security reasons, in minutes	Int	MM (continuous integer)
LateAircraftDelay	Delay due to late arrival of the flight aircraft, in minutes.	Int	MM (continuous integer)

Table 1: Input variables, their descriptions, types and encodings. In red, the forbidden variables. In yellow, the target variable.

The first thing that our code does is to parse the possible options. To select the CSV files the program reads a file chooser is shown. We make all the appropriate comprobations of the files: they must exist and we demand them to satisfy the columns schema presented in Table 1 [4].

From now on, we will explain the experiments carried out using two files of the full data set ("2002.csv" and "2008.csv") as input files. The results obtained and decisions made to keep a subset of variables are subject to these data sets. In the final code we have just implemented the consequences derived from these experiments. Nevertheless, the checkups for the forthcoming results can be computed as an optional input.

The first thing that we did was to remove forbidden variables (those that remain unknown before the plane lands), as well as other attributes that we considered to be useless. In particular, these are "TailNum", for which there were too many missing values, and "Cancelled" and "CancellationCode", which represent flights that have been cancelled and whose arrival delays make no sense to consider.

Rows containing any missing values were also removed: they are not suited for feature analyses nor regression models. In addition, variables that represent day times were transformed from the HHMM format to a continuous spectrum of minutes since 00:00. The justification for this is that the models that we will use will try to map the input variables to a set of predictions of the arrival delay, which fall in the continuous real domain, through a set of continuous functions. Therefore, we need all variables to be continuous in their domain.

Categorical variables ("Origin", "Dest" and "UniqueCarrier") were transformed into numerical ("Origin_cat","Dest_cat" and "UniqueCarrier_cat") via string indexing. This is done because for regression models, it is necessary to have all variables as numerical, and there are two main approaches for doing that: one-hot encoding (that is, introducing a dummy binary variable for every possible category of the original categorical variable) and string indexing (replacing each category of the original categorical variable by an integer). Although the first one seemed to suit better a regression model, it also implied introducing too many new variables, which would increase the regression model computational costs significantly, whereas the second method would preserve the original number of variables. We do no expect these new variables to be important in the regression model since the indexing follows an arbitrary assignment of the numerical values, but we will not discard them until a proper variable analysis is done.

Other variable removals were considered due to the (intuitively) collinear nature of some variables. For instance, one could think that "CRSArrTime" could be obtained from the sum of "CRSDepTime" + "CRSEllapsedTime", and therefore, that it could be useless. However, we refused to remove them because of two reasons: scheduled departure and arrival times are expressed in local time and there is an upper bound of 1439 for these variables, meaning that there is no perfect linear dependency between the three mentioned columns.

Once all variables were converted to numerical, they were standardized and stored as rows in a new column of the data set called "scaledFeatures". The target variable was excluded from this process.

The last step of the preprocessing was to randomly split the data in two: randomSplit command in Scala allows users to select a percentage of random instances that will be included in the training set, and the remaining are assigned to the test set. The default options are 70% for the training set and the remaining 30% for the test set.

3 Data Analysis

Once data was ready for being fed into a model, a data analysis was carried out with two purposes:

- Outliers removal for the training set, so that very unlikely instances do not influence the regression model.
- Feature selection for the machine learning model for reducing computational costs and reducing the output error.

3.1 PCA and Outliers removal

The filtering of the outliers was done by excluding those instances with significantly high distances from the mean vector. The criterion that we established was to remove rows with a Mahalanobis distance from the mean vector that exceeds the 3rd quartile by 1.5 times the interquartile range. Mahalanobis distance calculation was created as an external function in our code.[6]

Since the Mahalanobis distance calculation requires the computation of the inverse matrix of the covariance matrix, the latter cannot be singular. However, this condition cannot be guaranteed because the Year column could have variance 0 (only an input CSV file containing flights that departed the same year) or there could be very strong linear dependencies between columns.

We overcame this problem by computing the Mahalanobis distance of the instances according to their principal components, to ensure that the components are linearly independent and that the maximum variability in the data has been preserved. We set the number of principal components to 3, but this is a parameter that could be tuned.

3.2 Correlation Matrix

Now that outliers have been removed, we wanted to see the actual correlation in our features. This could be interesting for feature selection, as well to propose a regression model that fits the nature of our data.

For doing that, we computed the correlation matrix of the scaled features and the arrival delay. We mostly looked at the correlation of the scaled features with the "ArrDelay" column. We do not mind the other correlations because in prediction models they do not play a role. The results are displayed below:

TaxiOut	***CRSDepTime***	*UniqueCarrier_cat*	***Month***
0.30823444139640505	**0.09917346583896178**	0.013855711023797953	**-0.05358366909575097**
FlightNum	*CRSElapsedTime*	***DepTime***	*Dest_cat*
0.013893878804416457	**-0.031084426377904303**	**0.15322326399023536**	0.007570658007747436
DayOfWeek	***Year***	*DayOfMonth*	***CRSArrTime***
-0.0062119841156679885	**0.08496942740774**	-0.011862660291578434	**0.09239863012953628**
Origin_cat	*Distance*	***DepDelay***	*(ArrDelay)*
-0.014251866640069897	-0.0323355149030031	**0.900790673943624**	(1)

Table 2: Correlation of the different variables with "ArrDelay". In bold, the variables that have an absolute value of the correlation with "ArrDelay" higher than 0.05

We see a strong dependency between the arrival delay and the departure delay, which induces us to think that a linear regression model could be a good idea. We decided to keep only those variables showing a correlation with the arrival delay greater or equal than 0.05 in absolute value (those which are in bold in Table 2).

3.3 Variable summary

Allowed variables that have been excluded from the regression model and reasons for removing them:

Variable	Reason for excluding it
Cancellation code	Incompatible with the target variable.
Cancelled	Incompatible with the target variable.
CRSElapsedTime	Did not show enough correlation with the target column.
DayOfMonth	Did not show enough correlation with the target column.
DayOfWeek	Did not show enough correlation with the target column.
Dest	Its numerical representation did not show enough correlation with the target column.
Distance	Did not show enough correlation with the target column.
FlightNum	Did not show enough correlation with the target column.
Origin	Its numerical representation did not show enough correlation with the target column.
TailNum	Was full of NA values.
UniqueCarrier	Its numerical representation did not show enough correlation with the target column.

Table 3: List of allowed variables that are not used in the regression model and reason why they are excluded

Initial variables that have been kept for the final model and transformations performed over them:

Variable	Transformation(s)
CRSArrTime	HHMM to minutes + standardization
CRSDepTime	HHMM to minutes + standardization
DepDelay	Standardization
DepTime	HHMM to minutes + standardization
Month	Standardization
TaxiOut	Standardization
Year	Standardization

Table 4: List of allowed variables that are used in the regression model and transformations that they have been set to

Created variables and their explanation:

Variable	Meaning	Utility
pca-features	The numerical value of the k first principal components of an observation, with k = 3.	Dimensionality reduction without minimal loss of information. Orthogonalization of the components for computation of distances in the training split.
mahalanobis	Mahalanobis distance from an observation to the mean vector.	Outlier detection in the training set.

Table 5: List of new variables created with their meaning and utilities in the code

4 Machine Learning

All machine learning models and validation measures have been implemented with the MLlib library of Spark. In this section we will explain how the models work, the selected parameters and the validation measures taken into account.

4.1 Linear Regression

Linear regression assumes a linear relationship between input variables, also called predictive or explanatory variables, and the response variable. This model fits a linear equation to observed data and predicts the output by minimizing the sum of the squares of the deviation from each point to the predicted line, thus obtaining the best-fitting one [1].

When talking about regression analysis, regularization plays a main role in penalizing complex models, implemented for reducing overfitting, by putting network weights small.[7] Elastic net regularization combines the penalties of ridge regression and lasso regression, two model tuning methods, to get the best of both [5].
We have implemented the predefined linear regression model in Scala with 10 as the number of maximum iterations and the elastic net parameter 0.8. We have chosen these values because they are the ones used for all the examples in the Spark documentation and have proven to give good results for our problem.

4.2 Decision Tree

Decision trees are widely used in classification and regression methods. These models are rule-based in the sense that data is continuously split according to certain values of the attributes. Finally we obtain a tree in which the leafs are the terminal nodes that predict the outcome and the branches are the decisions we take to split the data [2].
We have implemented this simple algorithm in Spark with the Decision Tree Regressor defined for Scala and without any additional parameters apart from the label column and the features column.

4.3 Validation measures

A key step in the development of any machine learning model is to assess the accuracy of its results. The regression metrics we have used to do so are RMSE, R^2 and R_a^2, which we will explain in the following [3]:

1. **Root Mean Squared Error**: This measure is simply the square root of the Mean Squared Error, which represents the average in the squared difference between the original output of the model and the predicted values. It measures the standard deviation of the residuals.

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{N}}$$

2. **Coefficient of Determination or $\mathbf{R^2}$**: Its value is between 0 and 1 and represents the proportion of variance in the dependent variable that is explained by the independent variables.

$$R^2 = 1 - \frac{MSE}{\mathrm{VAR}(\mathbf{y}) \cdot (N-1)} = 1 - \frac{\sum_{i=0}^{N-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{\sum_{i=0}^{N-1}(\mathbf{y}_i - \bar{\mathbf{y}})^2}$$

3. **Adjusted $\mathbf{R^2}$**: The coefficient of determination increases whenever a new variable is taken into account in the linear regression model. That is why we applied the adjusted R^2, to take into account the number of independent variables (k) and penalize more complex models. In the formula below n is the number of observations in the data set.

$$R_a^2 = 1 - \frac{(1-R^2)(N-1)}{N-k-1}$$

We have used the above scores for the reason that they are specific for regression. The adjusted R^2, which will be the one that will give us a real sense of the accuracy of the model is defined by MSE and R^2.

5 Results

For the selected variables and the two models described in section 4, we have obtained the results shown in the tables below.

	Linear Regression	**Decision Tree**
RMSE	10.32	15.88
R-squared	0.902	0.769
Adjusted R-squared	0.902	0.769

Table 6: Checkups with 2002 and 2008 files and all initial variables

	Linear Regression	**Decision Tree**
RMSE	10.82	17.86
R-squared	0.893	0.710
Adjusted R-squared	0.893	0.710

Table 7: Final results with 70% training and 30% testing

	Linear Regression	**Decision Tree**
RMSE	11.20	27.68
R-squared	0.920	0.512
Adjusted R-squared	0.920	0.512

Table 8: Final results with some files as training and the remaining years as testing

6 Application options and output

The initialization of our code allows different options. The execution command, after using "sbt package", works as follows:

$ spark−submit [spark−options] <PATH−TO−JAR−FILE> [options]

When running the code, an "output.txt" file is created with the experiment results (or an error message if the CSV files do not satisfy the requirements previously mentioned). Depending on the option you choose in the execution command the file will display different computations. The list of the allowed options are described in the following:

- -c: The same checkups that have been made for the feature selection in section 2 are run for any input data set. It only works with a 70-30 percentage split for training and testing, respectively. This checkups are done for all variables, before keeping just the final ones. See Table 6.
- -t <value>: The experiment splits randomly the training and test sets from the original input data sets. <value> denotes the fraction of instances that is destined to the test split, introduced as a float between 0 and 1. See Table 7.
- -s: Two file choosers are prompted to the user, one for training and the other for testing, so that the user can select which files will be fed as training input to the models and which will be used for the validation process. See Table 8.

These three options are mutually exclusive and the default one (no options provided) is the following:

$ spark−submit [spark−options] <PATH−TO−JAR−FILE> −t 0.3

7 Conclusions

Taking everything into consideration we can conclude that we have successfully achieved to implement machine learning algorithms to the regression problem that was posed to us. We have managed to predict the Arrival Delays of different US domestic flights using a very big data set in a decent computing time.

For the purpose of achieving the best predictor possible, we tried many regression models in Spark and the ones that gave better results are the ones explained in this report. Concretely, the best performing model was the Linear Regression. Taking only the variables listed in Table 4, we have been able to explain around 90% of the variance of the output. The Decision Tree was only able to explain around 70% of the output's variance. We have kept this variables choice because it gives very similar results to the model with all possible variables and it is significantly simpler.

We would not have been able to understand the model if it were not for the data analysis we performed. For example, thanks to the correlations computed between variables, we could see how the Departure Delay is the explanatory variable most correlated with the Arrival Delay, thus being key in the outcome prediction.

Despite the model working better with all original variables, the loss the in predictive ability of the model is completely negligible: the simpler model is a particular case of the more complex model where some estimated coefficients are set to 0. Therefore, no better results would have been obtained by only removing variables.

However, the increase in the RMSE and the reduction in the R-squared coefficients are quantitative and qualitatively unsignificant, meaning that the variable drop is justified due to computational cost reduction.

Further research would be very useful in order to improve the results. Some ideas that could be interesting to explore would be to create new variables that helped the predictive model and also to import information from external datasets. An example of a new variable could be the number of flight landings in a given moment. This way we could take air traffic of the destination airport into account at the moment of a flight's landing. Furthermore, if you search for reasons of flight delays, weather plays a main role. Importing information about the weather conditions of the given flights would most probably help in the prediction.

References

[1] Jason Bronwlee. *Linear Regression for Machine Learning*. 2016.

[2] Afroz Chakure. *Decision Tree Classification: An introduction to Decision Tree Classifiers*. 2019.

[3] Akshita Chugh. *MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better?* 2020.

[4] Eric Morris. *Different Types of Flight Delays*. 2019.

[5] Michał Oleszak. *Regularization: Ridge, Lasso and Elastic Net*. 2019.

[6] Gafar Matanmi Oyeyemi, Abdulwasiu Bukoye, and Akeyede Imam. "Comparison of Outlier Detection Procedures in Multiple Linear Regressions". In: (2015).

[7] Neelam Tyagi. *L2 and L1 Regularization in Machine Learning*. 2021.