Spark Practical Work Report

Big Data

Carolina Isabel Martins Neves and Pedro Duarte da Costa

Universidad Politécnica de Madrid
{carolina.mneves,p.duarte}@alumnos.upm.es

January 19, 2019

1 Introduction

This project was developed in the scope of the course Big Data. Its purpose is, given a big amount of data, to be able to correctly predict the arrival delay of commercial flights. To do so we are going to develop three different algorithms in a Spark application in Scala.

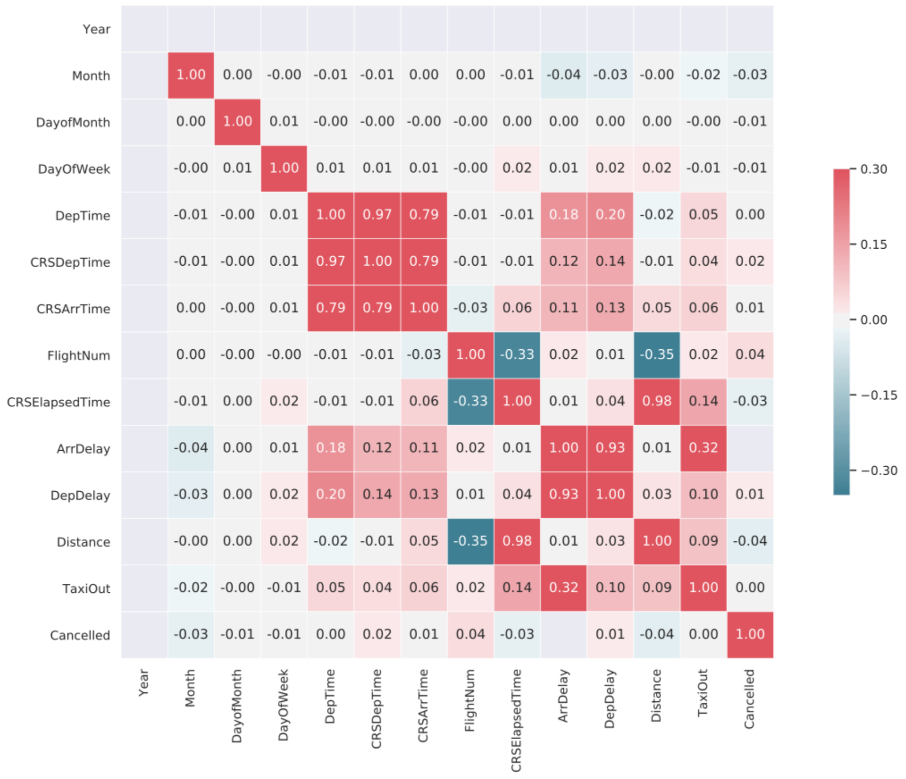
For our study we chose to work with only one of the years of the data-set, *2008.csv*.

2 Exploratory Data Analysis

As described in the previous section, our focus is to predict the value of ArrDelay. In order to properly perform this study, there are some variables from the dataset that can't be included (forbidden variables) since their values are only known once the plane has already taken off. If we used the forbidden variables, all the algorithms would return unreliable results. For this reason, all the columns ArrTime, ActualElapsedTime, AirTime, TaxiIn, Diverted, CarrierDelay, WeatherDelay, NASDelay, SecurityDelay and LateAircraftDelay were dropped.

Once the data was properly cleaned, we decide to perform some exploratory data analysis not only to understand our data, but also to try to gather as many insights before start working with it. All the code used for this is in the parser *eda.py*.

In order to use Linear Regression it's necessary to remove correlated variables to improve our model. For this reason, we did a correlation matrix with all the numerical variables of the data-set.



	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	CRSArrTime	FlightNum	CRSElapsedTime	ArrDelay	DepDelay	Distance	TaxiOut	Cancelled
Year														
Month		1.00	0.00	-0.00	-0.01	-0.01	0.00	0.00	-0.01	-0.04	-0.03	-0.00	-0.02	-0.03
DayofMonth		0.00	1.00	0.01	-0.00	-0.00	-0.00	-0.00	0.00	0.00	0.00	0.00	-0.00	-0.01
DayOfWeek		-0.00	0.01	1.00	0.01	0.01	0.01	-0.00	0.02	0.01	0.02	0.02	-0.01	-0.01
DepTime		-0.01	-0.00	0.01	1.00	0.97	0.79	-0.01	-0.01	0.18	0.20	-0.02	0.05	0.00
CRSDepTime		-0.01	-0.00	0.01	0.97	1.00	0.79	-0.01	-0.01	0.12	0.14	-0.01	0.04	0.02
CRSArrTime		0.00	-0.00	0.01	0.79	0.79	1.00	-0.03	0.06	0.11	0.13	0.05	0.06	0.01
FlightNum		0.00	-0.00	-0.00	-0.01	-0.01	-0.03	1.00	-0.33	0.02	0.01	-0.35	0.02	0.04
CRSElapsedTime		-0.01	0.00	0.02	-0.01	-0.01	0.06	-0.33	1.00	0.01	0.04	0.98	0.14	-0.03
ArrDelay		-0.04	0.00	0.01	0.18	0.12	0.11	0.02	0.01	1.00	0.93	0.01	0.32	
DepDelay		-0.03	0.00	0.02	0.20	0.14	0.13	0.01	0.04	0.93	1.00	0.03	0.10	0.01
Distance		-0.00	0.00	0.02	-0.02	-0.01	0.05	-0.35	0.98	0.01	0.03	1.00	0.09	-0.04
TaxiOut		-0.02	-0.00	-0.01	0.05	0.04	0.06	0.02	0.14	0.32	0.10	0.09	1.00	0.00
Cancelled		-0.03	-0.01	-0.01	0.00	0.02	0.01	0.04	-0.03		0.01	-0.04	0.00	1.00

Fig. 1. Correlation matrix of all numerical variables of 2008.csv

As we can see from Fig. 7, dark shades represent positive correlation while lighter shades represent negative correlation. Besides the colors we also used the values of each correlation to get a better idea of the relation between the variables.

Since our target variable is ArrDelay, we are going to take especially attention to the relations with that variable. We can clearly see that the strongest correlations are with DepDelay (0.93) and TaxiOut (0.32). It also shows some, but much less, correlation with DepTime (0.18), CRSDepTime (0.12) and CRSArrTime (0.11).

To get a better understanding of these features we draw scatterplots between our target variable and all the variables that it showed some correlation with.

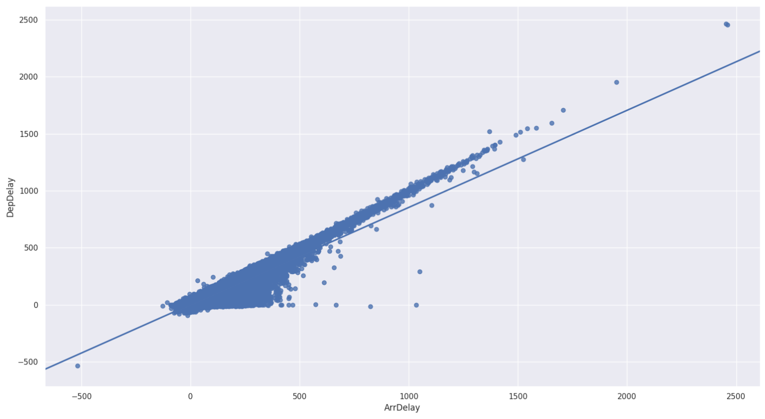


Fig. 2. Scatterplot between DepDelay and ArrDelay

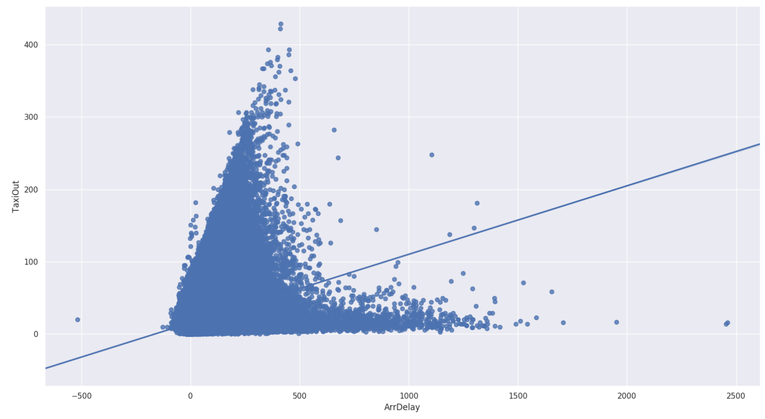


Fig. 3. Scatterplot between DepDelay and ArrDelay

As we can conclude from the images above, in Fig. 2 there's a much stronger correlation than in Fig. 3. We chose to run our model once with these variables since these are the ones with the strongest correlation, therefore, should provide the best results.

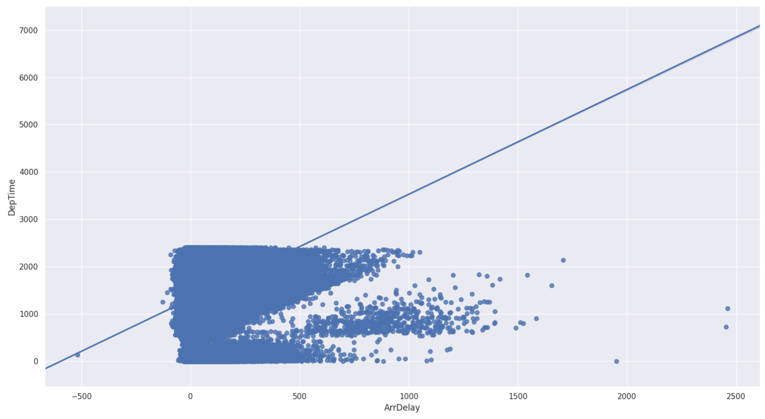


Fig. 4. Scatterplot between DepTime and ArrDelay

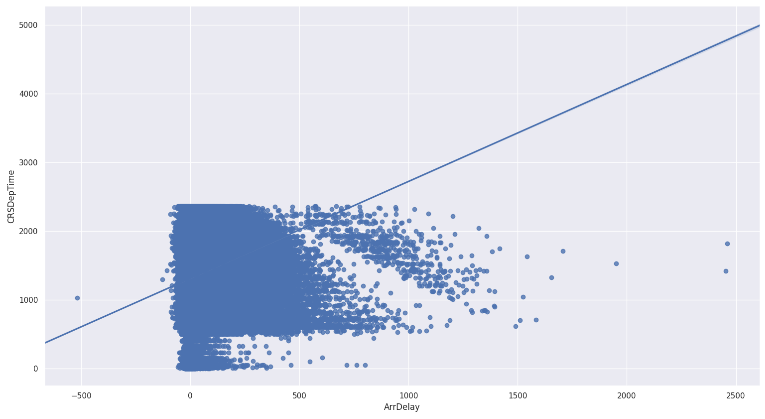


Fig. 5. Scatterplot between CRSDepTime and ArrDelay

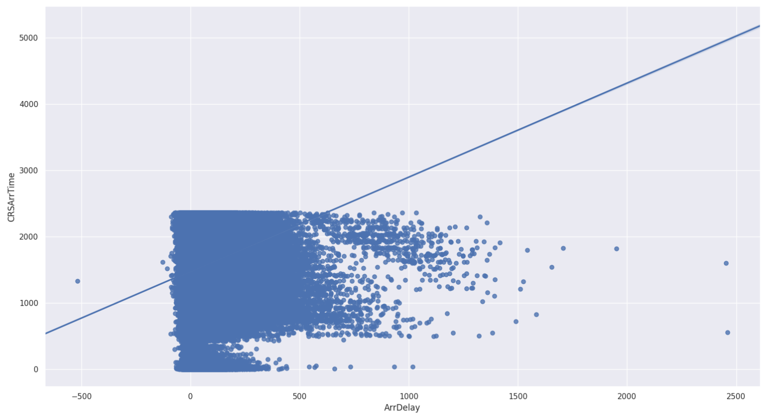


Fig. 6. Scatterplot between CRSArrTime and ArrDelay

From the three images above we can also see a positive correlation, yet not so strong if we compare with Fig. 2 and Fig. 3. For this reason, we also thought it would be a good idea to try out our model with these 5 variables plus the categorical variables and check if the results would be better.

3 Application

3.1 Machine Learning Techniques

Regression algorithms are machine learning techniques for predicting and estimating the relationship between variables. Whereas classification models identify which category an observation belongs to, regression models will return a numeric value. Therefore, for the purpose of this project, predict our continuous numerical target variable ArrDelay, we need to use regression algorithms.

Once the exploratory data analysis phase was completed, we were able to conclude that the dependent variable showed linear relations with independent variables of the data-set. Hence, a suitable model to our needs is Linear Regression.

Although, linear regression presented great result we decided to test other machine learning algorithms, such as, Random Forest Trees and Gradient-Boosted Tress.

Random Forest Trees and Gradient-Boosted Trees are techniques that are able to discover more complex dependencies at the cost of more time for fitting and therefore were worth being put to the test.

3.2 Loading and processing the data

The provided data is a csv and is being loaded by providing the full path of the file. Then the nominal features need to be transformed into something that the model can use by converting them into a numerical representation. A StringIndexer is used to map a string column of labels into a vector of numbers of length of the total unique string in that column and we apply that to every categorical feature. Then OneHotEncoder is used to wrap all unique values in a single vector in a new column of the data-set.

Finally we use a VectorAssembler, to assemble all the features into one vector called features to be used by the machine learning algorithms.

3.3 Creating and validating the model

As mention in 3.1, our application supports three different machine learning techniques for creating a model. The approach is very similar but some considerations were taken. All the data processing transformations as well as the machine learning training method is put into a pipeline combining all the steps into one transformation and making it easy to repeat all the transformations on new data.

Parameter optimization For building a model using linear regression, we perform a grid search to optimize the parameters of training, specifically we try different variations of elastic net regularization.

Since we are running on a single machine with limited resources we only used grid search for parameter optimization in the linear regression method since it was not possible to do the same for the other two algorithms in feasible time.

Training and validating From the initial exploratory data analysis we created 5 feature vectors to train our model. The feature vectors are as follows:

- $\mathbf{x}_1$: [DepDelay, TaxiOut, UniqueCarrier, DepTime, CRSArrTime, CRSElapsedTime, Distance, FlightNum, CRSDepTime, Year, Month, DayofMonth, DayOfWeek, TailNum, Origin, Dest]
- $\mathbf{x}_2$: [DepDelay, TaxiOut, Distance, FlightNum, CRSDepTime, Year, Month, DayofMonth, DayOfWeek, TailNum, Origin, Dest]
- $\mathbf{x}_3$: [DepDelay, TaxiOut, TailNum, Origin, Dest]
- $\mathbf{x}_4$: [DepDelay, TaxiOut, Origin, Dest]
- $\mathbf{x}_5$: [DepDelay, TaxiOut]

The final version of the application only considers $\mathbf{x}_4$ and $\mathbf{x}_5$ since these two seem to provide the essential information for a good prediction and allow to run the three machine learning algorithms in a single machine for a full year worth of data. For the purpose of testing, we used the five vectors in linear regression since it is a fast algorithm and to get a grasp on the insights of using each vector.

Then we drilled down to more useful features according to our exploratory data analysis.

Vector $\mathbf{x}_1$, represents all the features except the forbidden ones. $\mathbf{x}_2$, encompasses vector $\mathbf{x}_1$ minus features that are highly correlated between themselves and are therefore redundant, such as, DepTime and CRSArrTime which are highly correlated to CRSDeptime and CRSElapsedTime which is highly correlated to distance. Vector $\mathbf{x}_3$ includes vector $\mathbf{x}_2$ minus uncorrelated variables to the target, arrDelay. Vector $\mathbf{x}_4$ includes $\mathbf{x}_3$ minus TailNum. Vector $\mathbf{x}_5$ removes the categorical variables and is composed only of the variables that are highly correlated to the target.

Before training we split the data into a training and test set, 70% and 30% respectively.

For the linear regression method, since we specified several different variations of the model, a regression evaluator as well as a train test split were implemented to train the several models, with different set of parameters in the specified grid, and evaluate them. To do so we used spark's TrainValidationSplit class. Then applied the fit function using the train data-set to search the potential model space and returning the best performing model. In the case of Random Forest and Gradient-Boosted Trees we train directly the pipeline since it would take to much time and space to run and evaluate in all the possible configurations of theses methods and therefore we use the default ones.

Finally, we use the model to transform the test data-set and use a RegressionMetrics class to obtain useful information regarding the accuracy of the final model.

4 Results

To analyse and compare the results of our experiments we analysed three metrics, the Root Mean Square Error(RMSE), Mean Absolute Error and R squared. The results for the three machine learning algorithms are as follows.

Linear Regression					
Features Vector	***x1***	***x2***	***x3***	***x4***	***x5***
RMSE	9.7967	9.8506	9.8583	9.9618	10.8789
MAE	6.8871	6.8917	6.9304	7.0254	7.7834
RSquared	0.9351	0.9347	0.9344	0.9330	0.9201

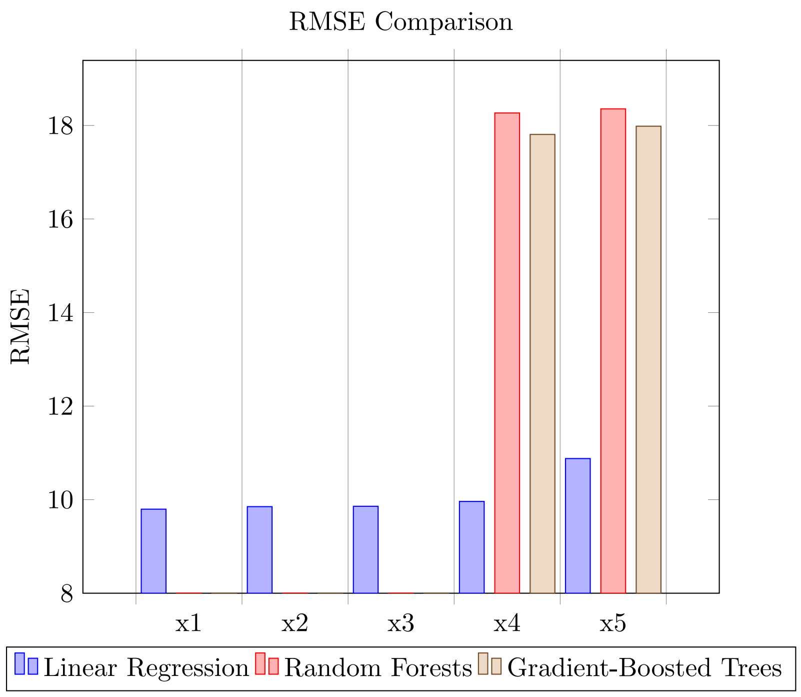
Table 1. Regression metrics for linear regression method.

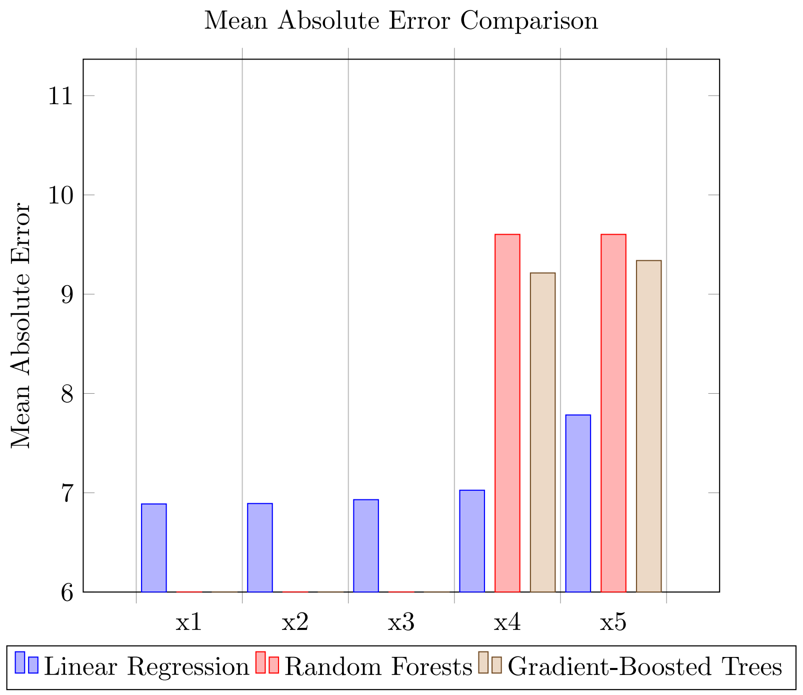
Random Forests		
Features Vector	***x4***	***x5***
RMSE	18.2667	18.3545
MAE	9.6023	9.3950
RSquared	0.7747	0.7726

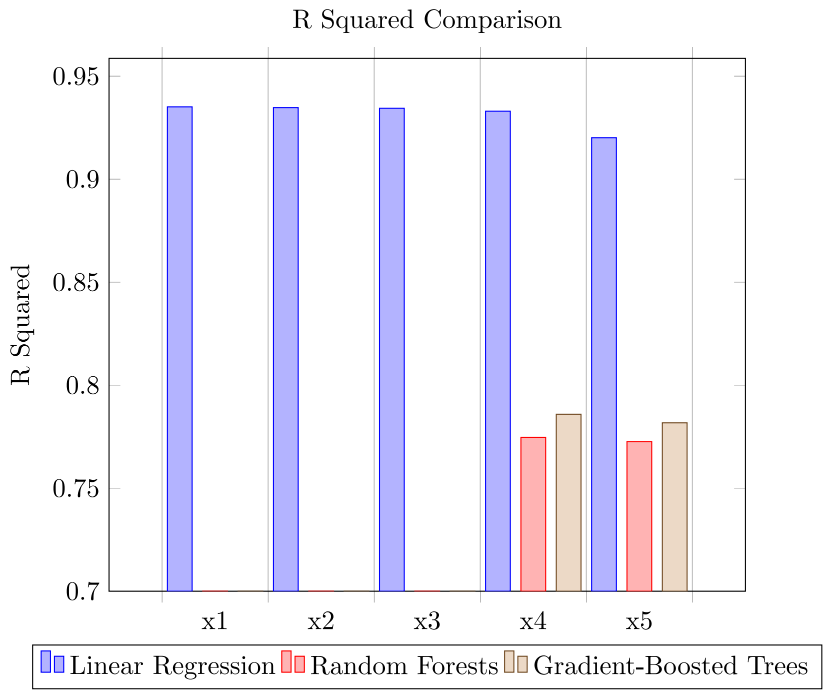
Table 2. Regression metrics for random forests method.

Gradient-Boosted Trees		
Features Vector	***x4***	***x5***
RMSE	17.8073	17.9833
MAE	9.2140	9.3390
RSquared	0.7859	0.7817

Table 3. Regression metrics for gradient-boosted trees method.




Mean Absolute Error Comparison
Mean Absolute Error
11
10
9
8
7
6
x1
x2
x3
x4
x5
Linear Regression
Random Forests
Gradient-Boosted Trees


R Squared Comparison
R Squared
0.95
0.9
0.85
0.8
0.75
0.7
x1
x2
x3
x4
x5
Linear Regression
Random Forests
Gradient-Boosted Trees

5 Conclusion

Before taking any solid conclusions we will first analyze the metrics we used to evaluate the algorithms.

1. **Root Mean Square Error(RMSE):** It measures the difference between values predicted by a model and the values observed. Considering this, the lower RMSE value is, the better is the model.
   Taking a look at the RMSE Comparison between the three models, 1º graph, we can plainly see that Linear Regression provided low values with all the five vectors used. Besides, by comparing it with the values of Random Forests and Gradient-Boosted Trees in the vectors $\mathbf{x}_4$ and $\mathbf{x}_5$, Linear Regression provided the lowest in both scenarios.
2. **Mean Absolute Error(MAE):** It refers to the mean of the absolute values of each prediction error on all instances of the test data-set, where prediction error is the difference between the actual value and the predicted value for that instance. Once again, the lower the MAE value is, the better is the model.
   By examining the Mean Absolute Error Comparison, 2º graph, we are able to verify that Linear Regression returned low values with all the vectors. Whereas Random Forests and Gradient-Boosted Tree's MAE are round 9, Linear Regression has a MAE of round 6.
3. **R squared:** Also known as the coefficient of determination, it is a statistical measure of how close the data is to the fitted regression line. It assess the goodness of fit of our regression model. The closer the value is to 1, the better is the model.
   Considering R Squared Comparison, 3º graph, Linear Regression produced values between the 0.9 and 0.95 to all of the five vectors studied. On the other hand, both Random Forests and Gradient-Boosted Trees returned values between 0.75 and 0.8. Although these results are not bad, the former's are better.

All in all, after comparing the values of each of the used metrics in the three algorithms, we can conclude that the model which proved to provide the best predictions was Linear Regression.

6 Running the application

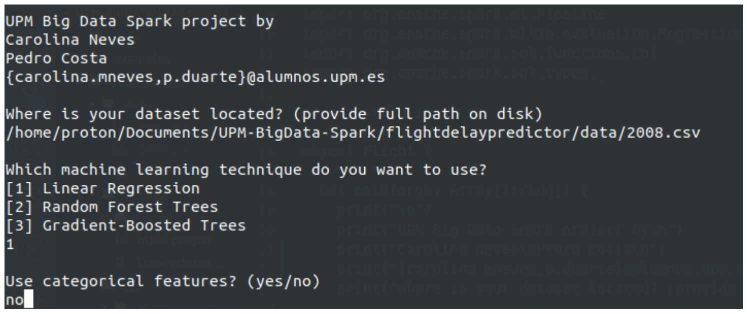
To run the application JDK8 is necessary, some errors were found while running with JDK11 due to compatibility issues.

We used sbt to compile the application with the command, sbt package, and spark submit to run the resulting jar file. In our machine we allocated 6gb to the driver memory allowing all the current options of the application to run in less than 1h for the 2018 data-set, this may vary on other machines.

The command t

The command used was:

$ spark-submit –driver-memory 6g –class FlightDelayPredictor.Flight flightdelaypredictor.jar –master local



```
UPM Big Data Spark project by
Carolina Neves
Pedro Costa
{carolina.mneves,p.duarte}@alumnos.upm.es

Where is your dataset located? (provide full path on disk)
/home/proton/Documents/UPM-BigData-Spark/flightdelaypredictor/data/2008.csv

Which machine learning technique do you want to use?
[1] Linear Regression
[2] Random Forest Trees
[3] Gradient-Boosted Trees
1

Use categorical features? (yes/no)
no
```

Fig. 7. Interface