Big Data Course: Project Report

Flight arrival delay predictor

Lorenzo Framba: lorenzo.framba@alumnos.upm.es
Ostap Kharysh: ostap.kharysh@alumnos.upm.es
Federico Rodigari: federico.rodigari@alumnos.upm.es

Department of Computer Science
Master EIT Digital in Data Science

December 19, 2020

Contents

1 Introduction

The last decade has opened in the age of big data and data economy, in which computational power and memory combined make it possible to extract from data important knowledge, insights, and potential. This results in significant challenges and opportunities that are challenging the domains of research, innovation and business. This article provides a solution to the basic data science problem of predicting the arrival delay time of a commercial flight, given a set of parameters known at the time of take-off. To accomplish the target, we applied different statistical algorithms implemented in Spark MLlib application using PySpark. The project has been carried out following most of the steps of CRISP-DM which is the open standard for data analysis that was specifically developed for dealing with massive datasets. All the data processing, model training, and validation is done exceptionally in Spark as it is required in this project.

2 Dealing with data

The first phase of the project drives the focus to identify, collect, and analyze the data sets that can help accomplish the project goals. After that, the data is prepared through transformations and other actions to present a final dataset for modelling.

2.1 Data understanding

In the first step of this phase we the dataset of 2004 year published by the US Department of Transportation and explored all the variables present in the dataset. Aligned with the scope, we firstly dropped the columns which values were forbidden by the project requirements. In particular, those variables are: *"ArrTime"*, *"ActualElapsedTime"*, *"AirTime"*, *"TaxiIn"*, *"Diverted"*, *"CarrierDelay"*, *"WeatherDelay"*, *"NASDelay"*, *"SecurityDelay"*, *"LateAircraftDelay"*. After a basic data exploration we also dropped two columns that we do not consider relevant for our analysis (*"TailNum"* and *"UniqueCarrier"*) as those rises the complexity of the analysis because of large amount of categories they contain. Also we don't believe that plane tale number and name of the airline will enhance the prediction of the arrival delay of the plane. At last we filtered out all the flights that were cancelled using *"CancellationCode"* and dropped it, remaining more than 99% of the existing data.

Once we removed the most intuitively useless variables, we performed a correlation analysis to better understand how the remaining variables are correlated with arrival delay which is presented in the dataset as ***"ArrDelay"***.

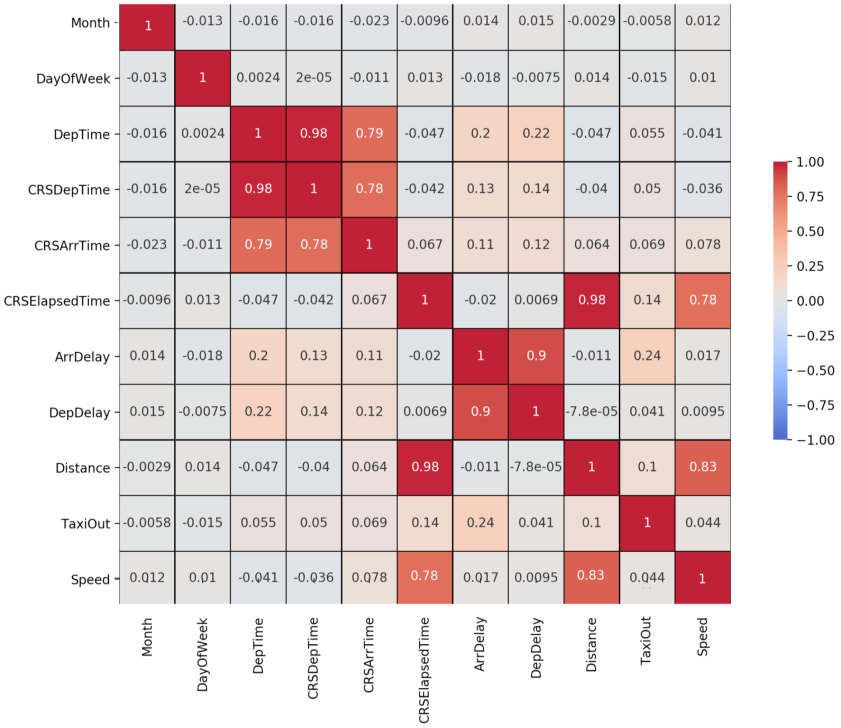

Figure 1: Correlation Matrix

When we analyze the correlation matrix we are focused in the variables involved in a relationship with our response variable *"ArrDelay"*. We can notice that the strongest correlations are with *"DepDelay"* (0.9), *"TaxiOut"* (0.24) and *"DepTime"* (0.2).

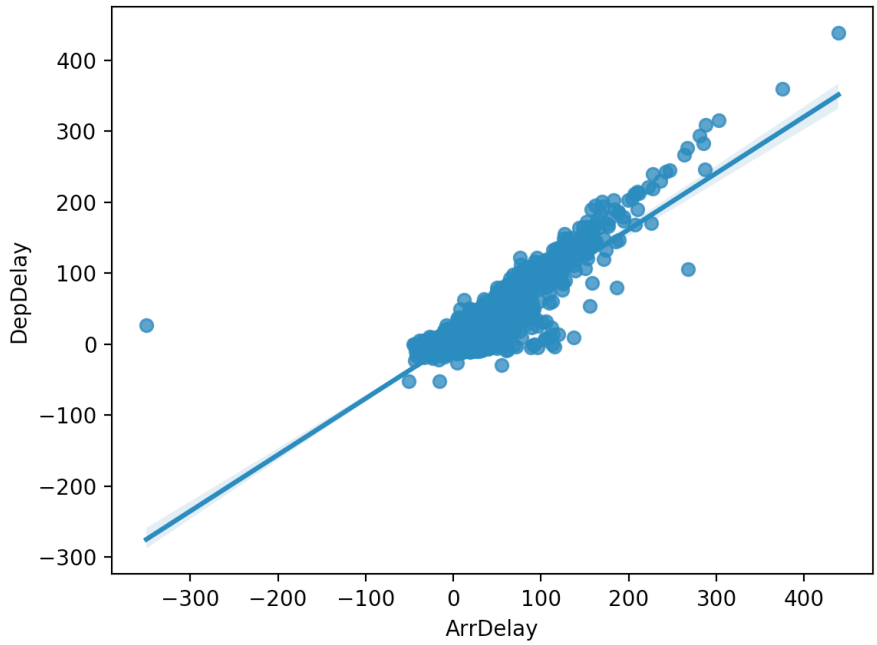

Figure 2: Correlation between "ArrDelay" and "DepDelay"

After exploring those strong relationships, we take into account the other variables that also shows some, but much less, correlation with the target variable: *"CRSDepTime"*(0.14) and *"CRSArrTime"* (0.11). *"CRSElapsedTime"* and *"Distance"* are not related with the target variable *"ArrDelay"* - but it may be useful to transform them to increase their prediction power. So we created a *"Speed"* variable as a result of division of *"Distance"* and *"CRSElapsedTime"* which basically means the expected speed of plane for a certain route. We expect that this variable should correlate with *"ArrDelay"*, as the higher is expected speed of the plane the more difficult it is to catch up with the schedule, hence, easier to arrive late. Also we create a new variable *"OrigDest"* as the combination of *"Origin"* and *"Destination"* variables, statistically speaking, we create an interaction between those variables. All in all, we end up with a variable which characterizes the flight route, because we believe that some of the airports are much busier than other that could result in a long waiting time for both taking off and landing. However, after several investigations, we decided to discharge this variable as it significantly contributed t0 the model complexity and didn't produce the appropriate results in B.

The last consideration is done about the variables that describe the time period: *"DayofMonth"* is removed as we move with a day of week and period of the day while the *"Year"* variable is dropped since the application is designed to work with one year at a time.

2.2 Data preparation

After the refinement described in the previous section, we moved into data preparation. In this phase we organized and tided up the data, removing what is no longer needed, replacing what is missing and standardising the format across all the data collected. In order to prepare the data, some transformations are required. We converted *"ArrDelay", "DepDelay", "CRSDepTime", "CRSArrTime", "DepTime"* to integer, as it will be appropriate for our further transformations we will mention below.

After those procedures, we finally selected the variable for the first step of processing that will enable us to perform our predictions of ***"ArrDelay"***.
The variable selected are the following:

- *"Month"*: even the correlation is low some of the months are more active in terms of the number of flight, so that factor may affect the target variable.
- *"DayOfWeek"*: some days of the week are busier than the others, that is why it is important to explore whether it has an effect on the arrival delay.
- *"DepTime"*: the different period of the day airports could have a different business level
- *"CRSDepTime"* and *"CRSArrTime"*: used to understand the period of day the plane will take off and land.
- *"Speed"*: the expected plane speed, as a result of division of *"Distance"* and *"CRSElapsedTime"*
- *"ArrDelay"*: the value we are trying to predict, target variable.
- *"DepDelay"*: the most important factor that is highly correlated with the target variable
- *"OrigDest"*: the flight root as a combination of *Origin* and *Destination*
- *"TaxiOut"*: the longer taxi out the more time is spent on the ground before the flight, which impacts the target variable.
- *"Cancelled"*: to take into account only successful flights.

At this point, some more transformations are needed. First of all, we statistically justified that precise time do not have much influence on the accuracy of *"ArrDelay"* predictions, but having categories of day time it improves the accuracy of the model. B.1.1. So using **Bucketizer** we transform these variables as *"CRSDepTime", "CRSArrTime", "DepTime"* taking specific time period of day: morning, afternoon, evening, night, as categories. Secondly, in order to use categorical variables (*"Month", "DayOfWeek", "CatCRSDepTime", "CatCRSArrTime", "IndOrigDest", "CatDepTime"*) in our models, we index their values using **StringIndexer** and transforme them via **OneHotEncoding**. Finally, we apply **VectorAssembler** to combine the variables in the vector of features and apply **MinMaxScaler** from Spark ML in order to rescale the continuous data within the range of [0, 1], so that we avoid an artificial predictive dominance of the larger numbers over the smaller numbers while fitting the model.

In this section we finished the data selections, preprocessing and transformation and ready to use them in our statistical models.

3 Modeling

In this phase we assess various models based on different modeling techniques. We, first of all, determined which algorithms to use addressing this problem, generated a test design by splitting the data between training, test and after that we built our models using the Pyspark documentation. Consequently, the models were assessed using the main statistics: R^2, MAE and RMSE.

3.1 Models

There are 4 models we decided to be appropriate to use for this problem. Here we share them and the reasoning behind chosing those:

- **Linear regression:** is always used as a benchmarking model for the prediction of continuous variables. It allows checking how linear interrelation between predictor and response variable helps to predict the response variable using a simple least squares approximations. However, it won't capture any non-linearities or interactions unless we assign them at the beginning.
- **Decision Tree regression:** are widely used models, which do not require feature scaling, and are able to capture non-linearities and feature interactions in the model using both categorical and continuous variables.
- **Random Forest regressions:** this model combines several Decision Trees, that should help to reduce the risk of overfitting when training the model Gradient boosted tree regression: is also combining several Decision Trees. In this case, it iteratively trains decision trees to reduce the loss function.
- **Gradient Boosted Tree regression:** is also combining several Decision Trees. In this case, it iteratively trains decision trees to reduce the loss function, hence, decrease the error.

From the model selection it could be inferred that with our analysis we want to give a chance to the tree-based algorithms to show their predictive power for this problem, that's why we prepared a broad selection of categorical data to try it out.

3.2 Variable selection and model tunning

As mentioned in 3.1, our application supports four different statistical models. We, basically, assess the performance of the tree-based algorithms against the benchmark - linear regression. Every data transformation as well model estimation is performed using Pipeline from Spark ML combining all the steps into an ordered workflow of actions. Also, we introduced a parameter tuning and cross-validation using **ParamGridBuilder** and **CrossValidation** from Spark MLlib. In our case, for the sake of reducing the training duration, the selection of parameters to tune is up to 3 with up to 3 elements to try for each model.

From the initial exploratory data analysis we created 5 feature vectors to train our models. The feature vectors are the following:

- Vector $\mathbf{x_1}$: *['DepDelay', 'TaxiOut']*
- Vector $\mathbf{x_2}$: *['DepDelay', 'TaxiOut', 'HotDepTime']*
- Vector $\mathbf{x_3}$: *['DepDelay', 'TaxiOut', 'HotDayOfWeek', 'Speed']*
- Vector $\mathbf{x_4}$: *['DepDelay', 'TaxiOut', 'HotDayOfWeek', 'Speed', 'HotMonth']*
- Vector $\mathbf{x_5}$: *['DepDelay', 'TaxiOut', 'Speed', 'HotDepTime', 'HotCRSCatArrTime']*

As one could infer, we are exploring the predictive power of categorical variables and *"Speed"* to enhance the predictive accuracy of the *"DepDelay"* and *"TaxiOut"* variables who have the highest predictive power among all the variables of the dataset.

4 Results

4.1 Results evaluation approach

To analyse and compare the results of our experiments we applied three metrics, the Root Mean Square Error(RMSE), Mean Absolute Error (MAE) and R^2 A. The target metric here is RMSE, as it shows how high is our predictive error to the flight arrival delay (*ArrDelay*). Despite similarity to RMSE we also use MAE as the measure of a true error. Because as we know from the nature of the RMSE, it gives relatively high values for the large errors, which is useful to take into account when exploring and tuning the models. That is why use it as a **RegressionEvaluator** for our problem, because we believe that high spikes of errors is highly undesirable in this prediction problem. In addition, we also use R^2 to explore the amount of variability explained by the selected variables, to insure that an addition of the data to the model (raising complexity) increases predictive power of the model.

4.2 Results comparison

After we run all our 4 models with all 5 feature vectors we obtained the corresponding results:

Metric	X1	X2	X3	X4	X5
R^2	0.848790	0.848985	0.848916	0.848916	0.849299
RMSE	13.357852	13.349252	13.352318	13.352318	13.335378
MAE	8.254204	8.248468	8.252661	8.252661	8.236516

Table 1: Metrics of Linear Regression

Metric	X1	X2	X3	X4	X5
R^2	0.652765	0.633599	0.655616	0.647514	0.723877
RMSE	20.242247	20.793397	20.158977	20.394735	18.050875
MAE	9.896120	10.453974	9.868963	10.084569	9.626383

Table 2: Metrics of Random Forest

Metric	X1	X2	X3	X4	X5
R^2	0.796384	0.795516	0.781501	0.781501	0.812682
RMSE	15.500741	15.533748	16.057278	16.057278	14.867443
MAE	8.088126	8.096620	8.860185	8.860185	8.017568

Table 3: Metrics of Decision Tree

Metric	X1	X2	X3	X4	X5
R^2	0.664063	0.667596	0.653461	0.653461	0.696640
RMSE	19.910200	19.805226	20.221959	20.221959	18.920217
MAE	9.681802	9.665054	9.701003	9.701003	9.456740

Table 4: Metrics of Gradient-Boosted Trees

We compared the results of the models we also apply visual analysis. We created the bar plots to represent each model outcomes for all the feature vectors on the x-axes and their magnitude in terms of the metric on the y-axes.

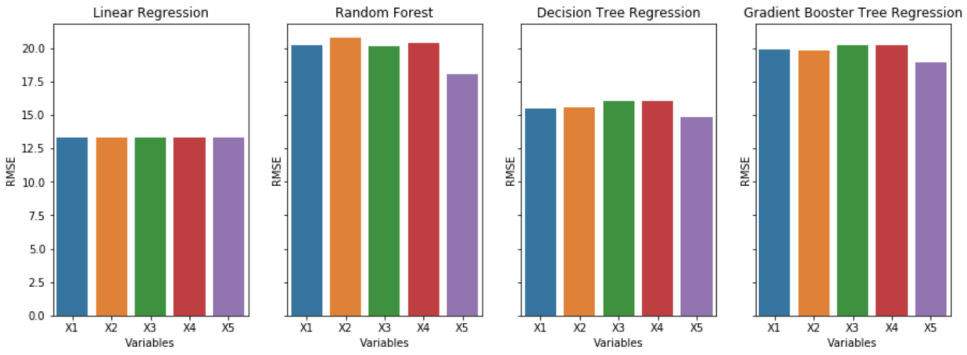

Figure 3: RMSE comparison

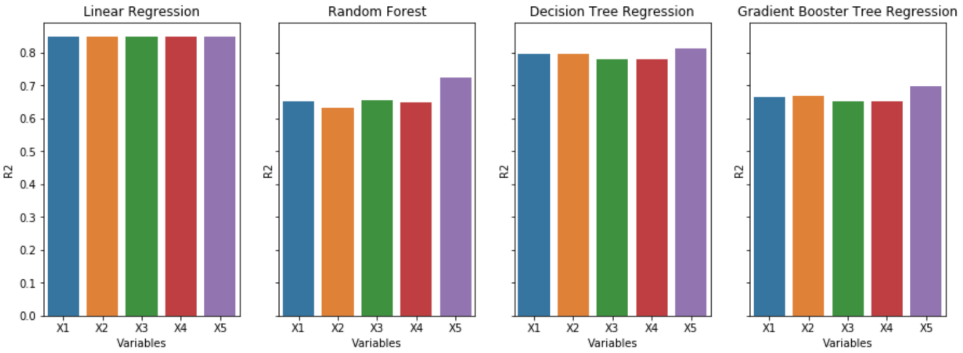

Figure 4: R squared comparison

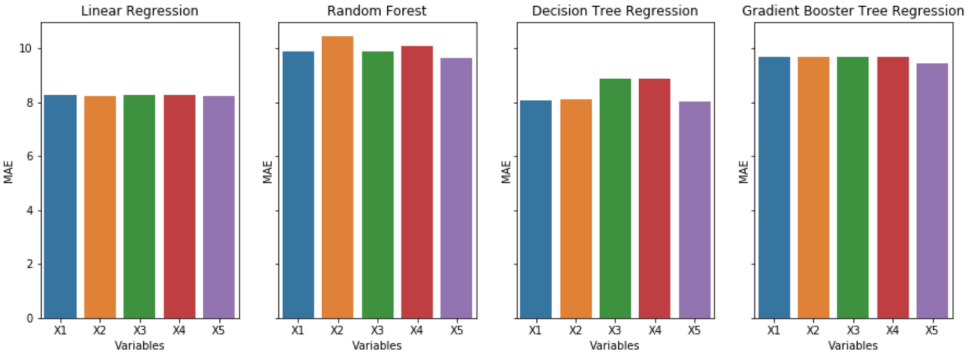

Figure 5: MAE comparison

5 Conclusion

Considering the results of the 4 statistical models, we conclude that for our a current problem of flight arrival delay prediction, the best solution is the *Linear Regression* model.

- The tree-based models were overall improved after the introduction of categorical variables and creation of the new continuous variable *"Speed"*. The best improvements and the lowest RMSE among them demonstrated *Decision Tree Regression*.
- Even there is no much improvements of linear model while increasing the model complexity it still performs much better then any of the tree-based models (For example $\mathbf{x_5}$ RMSE: *Decision Tree Regression* - 13.33 versus *Linear Regression* - 14.86 .
- The processing, transformation of variables, model training, cross validation, and tuning was conducted in Spark which largely reduced the computing duration of the dataset presented in this problem.

All in all, exploitation of categorical variables didn't significantly improve the tree-based models fitting power to outperform the *Linear Regression* for the problem of flight arrival delay predictions.

6 Running the application

The following instructions are provided to explain how to compile and execute the application:

1. Select the *–dataset* name file the *–path* at which the Airbus dataset is saved[1].

```
python main.py --dataset 'year.csv'
```

2. You have the option to choose the train/test split with *–split_size_train* [2]
3. You have the option to choose the dataset size with *–dataset_size* [3]
4. With *–model* You have the option to choose the ML model type between [4]:
   - *'linear_regression'*
   - *'decision_tree_regression'*
   - *'random_forest'*
   - *'gradient_boosted_tree_regression'*
   - *'all'* will train and test all the models and compare their respective R^2 / MAE / RMSE]
5. To select different variables to be trained, use *–variables* [5]
   - *'X1'*
   - *'best'*
   - *'all'*
6. To visualize the Scatterplots and the Correlation Matrix, set *–view True*. If *–model 'all'* is selected, you will also get a comparing bar chart that combines all the models' metrics at ones

A general approach to run the application could be:

```
python main.py --dataset 'year.csv' --model 'linear_regression' --split_size_train 80 --variables 'all'
```

[1] If is not specified, the program assumes the Airbus is in the same folder as the project itself. Make sure the name is *'year.csv'* and *year* is a 4 digit number from 1987 to 2008.

[2] default: *75 / 25*

[3] default: all the samples

[4] default: *linear_regression*

[5] default: 'X1'

A Used evaluation metrics

For comparing the results we were focused on the three metrics:

- **RMSE - Root Mean Square Error**: is defined as the square root of the average squared distance between the actual score and the predicted score. It is the most commonly used metric for regression tasks. As far as the lower this value is, the better is our model.
- **R^2 - R Squared**: is also called as coefficient of determination and gives us a measure of how well the actual outcomes are replicated by the model or the regression line. This is based on the total variation of prediction explained by the model shifting between 0 and 1.
- **MAE - Mean Absolute Error**: is the measure of the difference between the two continuous variables. The MAE is the average vertical distance between each actual value and the line that best matches the data. Once again, the lower the MAE value is, the better is the model.

B Previous variable selection and evaluation

From the initial exploratory data analysis we created 6 feature vectors to train our model. The feature vectors were as follows:

- Vector $\mathbf{x_1}$: *['DepDelay', 'TaxiOut']*
- Vector $\mathbf{x_2}$: *['DepDelay', 'TaxiOut', 'HotDepTime']*
- Vector $\mathbf{x_3}$: *['DepDelay', 'TaxiOut', 'HotIndOrigDest', 'HotDepTime']*
- Vector $\mathbf{x_4}$: *['DepDelay', 'TaxiOut', 'HotDayOfWeek', 'HotMonth', 'Speed']*
- Vector $\mathbf{x_5}$: *['DepDelay', 'TaxiOut', 'HotDayOfWeek', 'HotIndOrigDest', 'Speed']*
- Vector $\mathbf{x_6}$: *['DepDelay', 'TaxiOut', 'HotIndOrigDest', 'Speed', 'HotCRSCatDepTime', 'HotCRSCatArrTime', 'HotDepTime']*

B.1 Results

As you see from the results below we didn't accomplish any significant improvements in RMSE or R^2 using flight routes as *"HotIndOrigDest"* along with *"HotCRSCatDepTime"* do these variable were dropped from feature vectors $\mathbf{x_3}$, $\mathbf{x_5}$ and $\mathbf{x_6}$ since they rose the complexity of the algorithm without improving its performance.

B.1.1 Model metrics

The results for the three machine learning algorithms are as follows.

Metric	X1	X2	X3	X4	X5	X6
R^2	0.863443	0.858356	0.859397	0.863434	0.861890	0.857932
RMSE	12.305020	12.564305	12.472192	12.347650	12.361847	12.564910
MAE	7.846345	7.867043	7.864282	7.857177	7.867813	7.869502

Table 5: Metrics of Linear Regression

Metric	X1	X2	X3	X4	X5	X6
R^2	0.711323	0.661806	0.689347	0.647392	0.645006	0.693264
RMSE	17.870397	19.317928	18.480953	19.773551	19.980872	18.532997
MAE	9.632420	10.695816	10.391762	10.698236	10.972520	10.377750

Table 6: Metrics of Random Forest

Metric	X1	X2	X3	X4	X5	X6
R^2	0.739567	0.735974	0.742908	0.734263	0.737428	0.742681
RMSE	16.973668	17.068707	16.812446	17.165814	17.184172	16.974560
MAE	9.286871	9.276274	9.244541	9.320668	9.256949	9.235528

Table 7: Metrics of Decision Tree

Metric	X1	X2	X3	X4	X5	X6
R^2	0.677975	0.731742	0.734007	0.746609	0.740248	0.739396
RMSE	18.936929	17.405220	17.224535	16.830948	17.048796	16.961023
MAE	10.291720	8.919500	8.970977	8.799469	8.916626	8.897008

Table 8: Metrics of Gradient-Boosted Trees

B.1.2 Comparing metrics between the models containing discharged variables

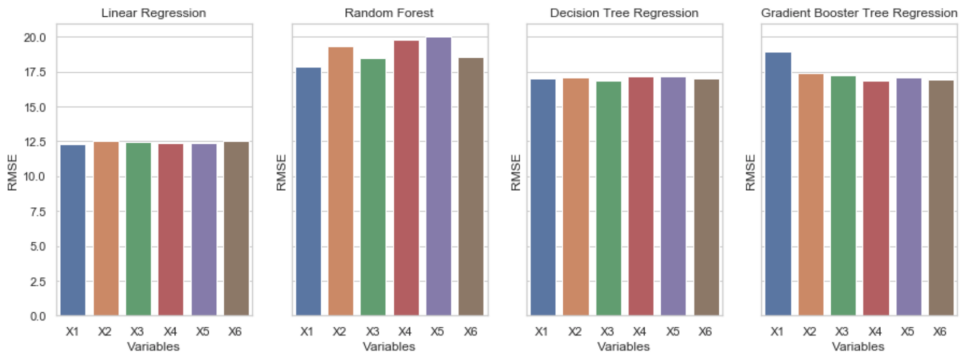

Figure 6: RMSE comparison

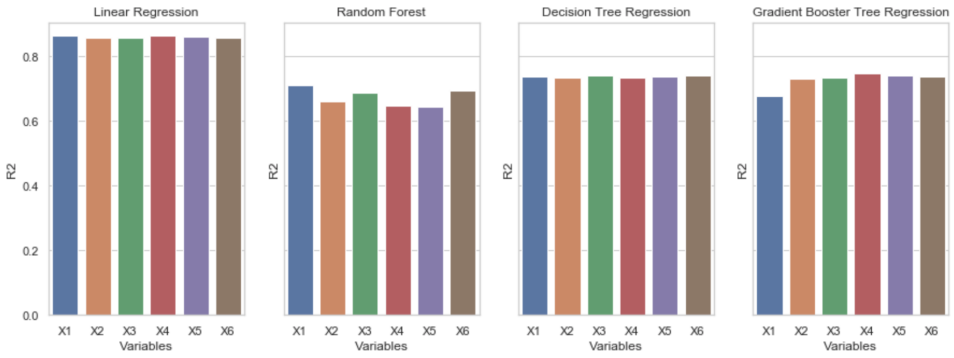

Figure 7: R squared comparison

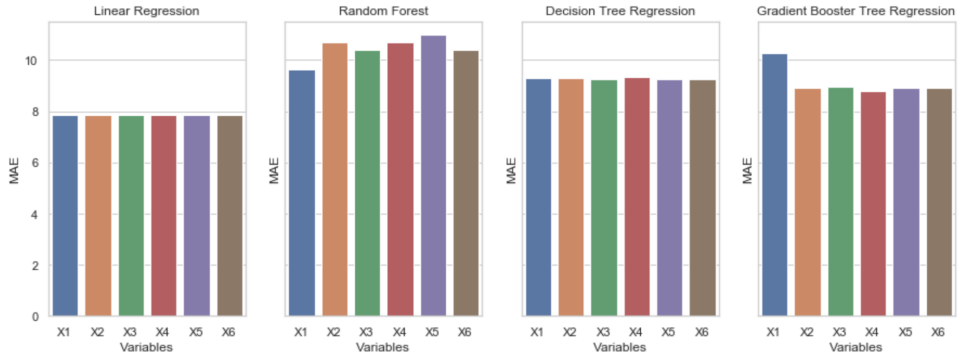

Figure 8: MAE comparison