Big Data Assignment

# Spark Practical Work

## Universidad Politécnica de Madrid

**Cecilia Peccolo, Karla Gonzalez Romero, Maximilian Boekels**

cecilia.peccolo@alumnos.upm.es
maximilian.boekels@gmail.com
karla.gonzalez@alumnos.upm.es

## 1 Selected Variables

| | |
|---|---|
| Year | Capture effect of long time trends of flight delays. |
| Month | Capture effect of the date of the flight on a delay, maybe the time of the year influences the probability of delay due to weather conditions. |
| DayofMonth | Capture effect of the date of the flight on a delay, maybe the time of the year influences the probability of delay due to weather conditions. |
| DayOfWeek | Capture effect of the week of the flight on a delay, maybe some days of the week are more prone to delays due to local regulations (e.g. in Germany, all stores are closed on Sundays). |
| UniqueCarrier | Capture effect of carrier on flight delay, maybe some carriers have better measures to prevent delays than others. |
| CRSElapsedTime | Capture effect of flight duration on delay, maybe long flights have a higher chance of being delayed due to more preparation. |
| ArrDelay | Target variable |
| DepDelay | Capture effect of departure delay on the flight. Late departure is probably positively correlated to late arrival. |
| Origin | Capture effect of origin airport, as maybe some airports have better measures to prevent delay. |
| Dest | Capture effect of destination airport, as maybe some airports have better measures to prevent delay. |
| TaxiOut | Capture effect of how long it takes to get the plane to the runway. Maybe if this takes longer than expected, the flight will have a higher chance of being delayed. |

## 2 Deleted Variables

| | |
|---|---|
| ArrTime | forbidden |
| ActualElapsedTime | forbidden |
| AirTime | forbidden |
| TaxiIn | forbidden |
| Diverted | forbidden |
| CarrierDelay | forbidden |
| WeatherDelay | forbidden |
| NASDelay | forbidden |
| SecurityDelay | forbidden |
| LateAircraftDelay | forbidden |
| CRSDepTime | Departure delay and time of day are already accounted for by DepDelay and DepHour and DepMinute. |
| CRSArrTime | Duration of flight and time of day are already accounted for by CRSElapsedTime and DepHour and DepMinute. |
| FlightNum | Just an identifier without additional information |
| TailNum | Just an identifier without additional information |
| Distance | The correlation with CRSElapsedTime is very high and apart from how long the flight is, this does not provide additional information and might harm the explainability of our model. |
| CancellationCode | Cancelled and diverted flights do not have an ArrTime so they cannot be estimated, hence these rows are removed from the dataset according to the value in Cancelled and Diverted. |
| Cancelled | Cancelled and diverted flights do not have an ArrTime so they cannot be estimated, hence these rows are removed from the dataset according to the value in Cancelled and Diverted. |
| DepTime | This value is split up into DepHours and DepMinutes. |

## 3 Variable Transformations

All categorical variables are encoded using a One-Hot Encoder as there is no ordinal relationship among them. The following Integer variables are also One-Hot encoded: **Year, Month, DayOfWeek, DayOfMonth**. They do not have a linear effect on the target variable ArrDelay, but the bivariate Analysis showed that there is an effect on the target, hence they are used as factors. The remaining Integer variables are normalized and none of the boolean variables remain in the model.

## 4 New Variables

| | |
|---|---|
| DepHours | Derived from DepTime to be able to handle hour and minute of departure separately. |
| DepMinute | Derived from DepTime to handle hour and minute of departure separately. |

## 5 Machine Learning Algorithm

Our objective was to develop a model capable of efficiently predicting the expected delays of airplanes. For this, the results of the exploratory data analysis have been taken into consideration. For example, some of the variables have a linear dependency with ArrDelay, such as DepDelay and TaxiOut, in the following referred to as *linear*, and some do not, such as Month or DayOfWeek, referred to as *non-linear*.

We chose the following two models, one that does not require the conditions of a linear model and one that uses linear properties.

- **Decision Tree Regression:** The first model we selected was the Decision Tree Regression. This model is an excellent predictor and is capable of yielding robust results without the necessity of a linear relationship between the response variable and the other variables. This fits our problem well since most of the variables do not have a linear relationship with the target variable.
  Initially, we selected the default tree from the *ml PySpark library*. Following this, we sought to identify the optimal tree using hyperparameter optimization techniques, exploring combinations of various parameters. We varied the maximum depth of the tree among the values of 5, 10, and 20. Additionally, we limited the maximum number of bins to 20, 30, and 40. This approach allowed us to fine-tune the decision tree to best fit our data, striking a balance between complexity and predictive power.

- **Linear Regression:** The second model adapted was Linear Regression. We chose this model as it allows great explainability and utilizes linear relationships between the dependent and the target variables. Especially with the linear relationship between the DepDelay, TaxiOut and ArrDelay, this model fits our problem very well. This model has also been optimized with hyperparameter tuning. The regularization parameter that can help to prevent overfitting by limiting the values of coefficients, was chosen from values 0.01, 0.1 and 1. The Elastic Net Mixing parameter, which determines the type of regularization applied, was chosen from values 0, 0.5 and 1. This allowed our model to utilize an appropriate regularization technique with an appropriate amount of regularization.

## 5.1 Validation

The validation of our models was performed using 5-fold cross-validation with the Root-Mean-Square-Error (RMSE) metric. Cross-validation was chosen to allow for hyperparameter optimization and the RMSE metric was chosen as it yields great explainability, independence of the target variable scale and sensitivity to large errors.

## 5.2 Evaluation

In our first evaluation, the Linear Regression model scored an RMSE of 12.75, improving over the Decision Tree model with an RMSE of 13.67.

After running the models for the first time, we analyzed the residuals of the Linear Regression model to identify linear patterns in the residuals which indicate problems in the model.

The results of the residual analysis showed, that there were still linear patterns in the data as well as a strong influence of outliers. To address this problem, we created an alternative version of the dataframe that removes outliers using the z-score. It takes the mean and the standard deviation of values into account. In our case, we removed all outliers with a z-score of 2 or higher, or -2 or lower, and trained an alternative Linear Regression model to see, if this helped to improve the performance.

This does not apply to the Decision Tree regressor, as it does not capture the linear relationships in the data and is not sensitive to outliers.

To develop further on the Linear Regression model, we created another two training datasets to be able to train and compare different models:

- The first dataset was a version where we one-hot encoded some of the non-linear variables to introduce them as factors to our model. This meant to address the problem of non-linear effects of these variables not being accurately represented by the linear model.

- For the second dataset, we created a trimmed version of the dataset that removed outliers which aimed to address the problem that Linear Regression models are prone to be heavily biased by influential outliers.

**Final Results:** By introducing One-Hot-Encoding, we managed to reduce the RMSE of the Linear Regression model to 11.82, which is quite a significant improvement. We managed to more accurately fit this data by factorizing some of the non-linear attributes with our adjusted model.

The model with the outliers trimmed did not introduce any improvement in performance. Instead, it increased the RMSE to 14.52.

Thus, our final best model is the Linear Regression Model with some of the non-linear variables One-Hot encoded, scoring an RMSE of 11.82.

## 6 How to run the Code

This section outlines the procedure for executing the provided PySpark application via the ‘spark-submit‘ command-line interface. This should be done using a Unix-based system, natively or through a virtual machine.

### 6.1 Prerequisites

Python must be successfully installed and properly configured on your system.

We recommend to create a python virtual environment using the following bash commands from the folder, where you want to create and activate your environment:

```
virtualenv venv
source venv/bin/activate
```

Listing 1: Create Virtual Python Environment

After successfully activating the virtual environment, move the requirements.txt file supplied to the current folder and install all the required packages using

```
pip install -r requirements.txt
```

Listing 2: Install Requirements

Alternatively, packages *pyspark*, *seaborn* and *scipy* can be installed manually using pip.

### 6.2 Execution Steps

1. Ensure `SPARK_HOME` is set to your Spark installation path and `${SPARK_HOME}/bin` is included in your system’s `PATH`.

2. Ensure that the virtual environment with all dependencies installed is activated.

3. Navigate to the Application Directory: Change the directory to where your `.py` application file and *correctly named* ***dataset.csv*** file reside.

4. Formulate the `spark-submit` command:

   ```
   spark-submit assignment.py
   ```

   Listing 3: Run the code

5. Monitor Execution (Optional): For cluster deployments, use the Spark UI or cluster manager’s monitoring tools to track your application’s progress.

## 6.3 Additional Notes

To facilitate the deployment of this Spark application in a cluster environment, it is essential to modify the SparkSession/SparkContext configuration within the code, particularly beginning from line 21. The initialization of the SparkSession, exemplified by **spark = SparkSession:.**, needs to be tailored to align with the specific requirements of the target cluster setup. This adjustment entails specifying the correct master URL and potentially adjusting other configuration parameters like executor memory and the number of cores, to optimize resource utilization and performance. Such configuration ensures that our application can efficiently leverage the distributed computing capabilities of the Spark cluster, thereby enhancing scalability and processing power.

Modify the ‘spark-submit‘ command according to your specific Spark cluster configuration and application requirements. Consult the Spark documentation for detailed options and parameters applicable to your environment.

The code was successfully tested on MacOS 12.7.2 and Google Colab, adjusting the import of the data file through Google Drive, with a sample size of 1000000 entries.