Big Data Spark Assignment

Group 5

Pueblas Núñez, Rodrigo

Martínez Capella, Ignacio

Burrell, David

1. Introduction

The purpose of this work is to apply Big Data technologies to a machine learning problem. A Spark application has been developed to predict the arrival delay of a series of US flights. All code has been created in an sbt project using the Scala programming language. The final prediction is performed using both linear regression and random forest models

2. Variables

As part of this project decisions were made about which variables are to be used and removed from the original datasets, what transformations could be applied to the variables and what new variables could be created.

Initial variable removal

The following variables were dropped from the dataset as indicated by the instructions:

- **ActualElapsedTime**
- **AirTime**
- **TaxiIn**
- **Diverted**
- **CarrierDelay**
- **WeatherDelay**
- **NASDelay**
- **SecurityDelay**
- **LateAircraftDelay**

In order to decide which of the remaining variables should be retained the summary statistics, the number of distinct values, and presence of null values was obtained for each.

In addition, we deleted both Year and the variable regarding cancelled flights.

- **Year:** it always has the same value for the whole dataset. Before deleting it, it is used to create the variable **Date.**
- **Cancelled:** all the rows with cancelled flights were deleted since we are not interested in having them. After filtering out these rows, the variable is deleted.
- **Cancellation code**: We are not interested in cancelled flights, less in their cancellation code.

Variable creation and transformation

Several variables were created created with the purpose of helping in the prediction process

- **isWeekeed**: This is used to determine if the **DayOfWeek** is on a weekday (Monday - Friday) to weekend day (Saturday and Sunday). This was created with the intention of capturing if each flight busier is during the weekend days.
- **CityOrigin**, **StateOrigin**, **CityDest**, **StateDest**: Using the supplementary data provided [1], more information was gathered about the airports the flights originated from and arrived at. Using the airport code the state and city of each airport was added into dataframe. The purpose of adding these variables is to capture the influence of the location of the airport, rather than just the airport itself.
- **Plane Age:** By using the supplementary data about the individual plans the age of the plane at the time of the flight. This was calculated by finding the difference between it's manufacturing year, and the year the flight occurred.
- **Date**: Combines the variables Year, Month and **DayOfMonth** in a new variable in unix timestamp format (time since 1/1/1970).

Although we had hope all of these variables would lead to better predictive power of the model, it was found that only the **Date** variable gave any greater insight and the rest were removed from the features.

Next transformations were applied to the variables so they were suitable for use. Below is a description of the variables that were transformed using custom logic:

- The original continuous variables have been converted to integers as they were being recognised as strings.
- Variables with "HHMM" format, as **DepTime**, **CRSDepTime** and **CRSArrTime** have been converted to minutes since midnight, as it is easier to interpret so that there are no steps in the data.
- As many of the time variables are inherently cyclical they are converted into their sine and cosine representations tp preserves this information. An example of this cyclical feature is DayOfWeek where 1 and 7 are closer to each other than days 1 and 5. For each value to have this transformation applied two new columns are created to store the values. For example applying the transformation to the DepTime column leads to the creation of the DepTime_sin and DepTime_cos columns. The full list of features that had this applied are; DepTime, CRSDeptTime, CRSArrTime, MonthDayOfWeek and DayOfMonth. More information about the theory behind this transformation can be found in the following references [2], [3].
- Any *null* values found in the **TaxiOut** column are set to be 15 minutes as this is the average time for flights to taxi out in the USA [4]

- The continuous variables have gone a normalization step before being fed to the regression model.

Next transformations were applied using the built in Spark methods as part of the model pipeline.

- String indexers were used to turn categorical features into indexed columns so they could be used in the model, e.g. **Origin** to **OriginIndex**.
- One hot encoders were used on the output of the string indexers when the variable was to be used in the random forest model, e.g. **OriginIndex** to **OriginIndexEncoded**.
- Vector assemblers were used to transform the input features into a single column for training and predicting. Different vector assemblers were used for different models due to the requirement of needing categorical features needing to be represented in alternative ways. The single column created is called “features”.
- A scaler was used after the creation of the “features” column to set integer values to a normalized range. This is necessary as the dataset contains variables with different ranges [5]. For example, **DayOfWeek** values range between 1 and 7, whereas **DepDelay** ones do so between -918 and 1618.

Variable selection

The next step was to select which variables were useful for predicting the class variable. The Pearson correlation coefficient relating each explanatory variable with **ArrDelay** was obtained and the explanatory variables were ordered according to these measurements.

A regression model was selected as the machine learning algorithm to be used. More information about this decision can be found in Section 3. The ordered variables were added to the regression model following a forward selection approach, starting with the one with the highest correlation coefficient. The process stopped when RMSE (one of our goal metrics) reached a threshold at around ~10.5.

The following variables have been included in the regression model:

Variable	**Pearson correlation with ArrDelay**
DepDelay	0.7945904564537559
TaxiOut	0.3073894720645663
DepTime_sin	-0.07021587587248983
DepTime_cos	0.07747367491821687

DayOfWeek_sin	-0.06543700093605814
OriginIndex	-0.02478981851099163
Date	-0.03063228106867741

3. Machine Learning

Techniques used

For this regression problem, we chose to compare two different methods: Linear Regression and Random Forest Regressor. Linear regression provides easy interpretation of the variables and their coefficients, while Random Forests are able to model nonlinearity in the data. Knowing which variables are the most important when predicting flight delays could be used in real scenarios to avoid or mitigate them.

Concerning the linear regression model hyperparameters, the maximum number of iterations has been set to 100, the regularization parameter has been set to 0.3 and the elastic net parameter, to 0.8. For the random forest model, we set the number of trees to 8, the maximum bins to 28 and the maximum depth to 8. Although we get better results with higher hyperparameters, it increased the complexity leading to a longer run time, thus we decided to simplify the model. The tuning of these parameters was done via trial and error until the results were satisfactory.

Validation

The dataset is split in two collections, one for training the model and the other for testing The spit is done randomly and the training dataset accounts for 70% of the original dataset, whereas the testing dataset is composed of the remaining 30% [6]. The measurements employed to assess the quality of the predictions are the coefficient of determination (R^2) and the squared and non-squared mean errors (RMSE and MSE). The R^2 is indicative of the quality of the model fit, which is determined by how well the regression predictions approximate the real values [7]. The MSE and RMSE are used as indicators of the error in the predictions, which is calculated by the difference between the real values and those predicted by the model [8].

To ensure that were no temporal lurking variable was affecting the results the models were trained and tested against several of the datasets made available. Since varied results were seen when giving different datasets the final models selected gave the best results on average across different time periods. Though as can be seen from the results below the

final models perform considerably better on more recent years, getting progressively worse the further back in time the supplied data is from.

Evaluation of Linear Regression Model

The final model had the following measurements in the following datasets:

Dataset	R^2	MSE	RMSE
1987	0.645827	233.785508	15.290046
1997	0.719186	221.23906	14.874107
2007	0.923238	117.320783	10.831471

As we see, the results are much worse for earlier years. On datasets from 2005 and later we achieve consistent results between 10 and 11 for RMSE.

Evaluation of Random Forest Regression Model

The final model had the following measurements in the following datasets:

Dataset	R^2	MSE	RMSE
1987	0.564526	287.45136	16.95439
1997	0.676206	255.101063	15.971883
2007	0.755359	373.905928	19.336647

We get worse results than for the linear regression model. This is in part due to using low random forest hyperparameters, in order to increase the performance.

4. Running the Program

Note: The following instructions require that **sbt** has been installed on the machine the program is being executed on.

To run the application first unzip the attached “BigData-Flights-2019” file and place it in a suitable location on your machine. Next rename the CSV file for processing to “InputData.csv” and place it the projects “app/data” folder located in the top level of the project directory. Then in a command terminal navigate so you are inside the “app” folder on your machine. Then run the following sbt commands:

$ sbt compile

```
$ sbt run
```

This will execute the program to perform the data frame transformations and the execution of the linear regression model, followed by the random forest model. The validation metrics described above will be shown for each model. Please note it has been seen the program can have a run time of up to 25 mins for the largest available datasets.

5. Conclusions

The developed application is able to read a dataset, analyse it, process its variables, train and test two classifiers, and inform of their performance. During the whole work we have tried to use the already implemented Scala functions, in order to make the execution faster.

Although the results shown are commendable we realise that these could be greatly improved. Using more variables improved the results, as was tested during the early versions of the models. However, this lead to models with increased complexity and long computational times.

6. References

1) American Statistical Association. Retrieved 18 December 2019, from http://stat-computing.org/dataexpo/2009/supplemental-data.html
2) Dossman, C. (2019). Top 6 errors novice machine learning engineers make. Retrieved 17 December 2019, from https://medium.com/ai%C2%B3-theory-practice-business/top-6-errors-novice-machine-learning-engineers-make-e82273d394db?
3) London, I. (2019). Encoding cyclical continuous features - 24-hour time. Retrieved 17 December 2019, from https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/
4) What Is Your Airport's Average Taxi Time?. (2016). Retrieved 18 December 2019, from https://www.planestats.com/aptot_2016mar
5) Jaitley, U. (2018). Why Data Normalization is necessary for Machine Learning models. Retrieved 18 December 2019, from https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029
6) Splitting the Data into Training and Evaluation Data. (2019). Retrieved 18 December 2019, from https://docs.aws.amazon.com/machine-learning/latest/dg/splitting-the-data-into-training-and-evaluation-data.html
7) Coefficient of determination. Retrieved 18 December 2019, from https://en.wikipedia.org/wiki/Coefficient_of_determination
8) Root-mean-square deviation. Retrieved 18 December 2019, from https://en.wikipedia.org/wiki/Root-mean-square_deviation