Machine Learning Engineer Nanodegree

Capstone Project - Predicting Delay on Flights in USA

Miguel Alemán

September 18th, 2018

I. Definition

Project Overview

In a globalized world travel is getting as easy as possible. It is possible to go from one country to another in less time and with less effort than 20 years ago. This makes even more evident the problems that exist in the travel process, in which one common problem is the delay of the flights. There are a few papers trying to understand the impact of this problem, which seems to affect travelers who are forced to wait several hours or even miss a connection flight, which is the worst scenario. This also affects companies and airports with increased block times on routes and higher carrier costs and airfares, and finally affecting other segments of economy such as industries that rely on air transportation to conduct business, and more.

This creates the opportunity for machine learning algorithms to make an effort and explain why does this happen and also, when this is more probable to happen and by this helping the customers to choose best flights and companies planning and avoiding the reasons why a delay can happen.

In this project a classification model was created in order to predict if a flight will be delayed or not based on real data collected from the The U.S. Department of Transportation in 2015, using features such as departure times, both planned and real, to help predicting flight delay based on location, hour, flight length and others. Two classifiers (Decision Tree and Random Forest) were trained and tested and only one got selected based on its performance.

Problem Statement

The goal is to create a classification model capable of predict when a flight is going to be cancelled, in order to do so the following steps will be executed:

1. Preprocess the data
2. Train the models
3. Compare the models
4. Select one model

The model selection will be based on its performance considering time of training, time of prediction, and two metrics Kappa and f1 score.

Evaluation Metrics

Since the data this projects uses is an unbalanced set (60% - 40%) the plain accuracy isn't the best metric, since a dummy classifier that classifies everything as the majority class will probably have a good accuracy. Because of this data distribution two scores will be used:

1. Cohen Kappa
2. F1 Score

About Cohen Kappa Score

Cohen's kappa: a statistic that measures inter-annotator agreement. It is a score that expresses the level of agreement between two annotators on a classification problem[1]. It is defined as

Kappa = (p_o - p_e) / (1 - p_e)

where p_o is the empirical probability of agreement on the label assigned to any sample (the observed agreement ratio), and p_e is the expected agreement when both annotators assign labels randomly. p_e is estimated using a per-annotator empirical prior over the class labels.

About F1 Score

The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal[2]. The formula for the F1 score is:

F1 = 2 *(precision* recall) / (precision + recall)

II. Analysis

Data Exploration

The dataset used in this project comes from the U.S. Department of Transportation, it is available from Kaggle[3] and contains the records for a year of flights. The dataframe contains 581+ millions of data points with 31 features each.

Brief description of the features

YEAR: Year of the Flight Trip

MONTH: Month of the Flight Trip

DAY: Day of the Flight Trip

DAY_OF_WEEK: Day of week of the Flight Trip

---

[1] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.cohen_kappa_score.html
[2] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
[3] https://www.kaggle.com/usdot/flight-delays

AIRLINE: Airline Identifier

FLIGHT_NUMBER: Flight Identifier

TAIL_NUMBER: Aircraft Identifier

ORIGIN_AIRPORT: Starting Airport

DESTINATION_AIRPORT: Destination Airport

SCHEDULED_DEPARTURE: Planned Departure Time

DEPARTURE_TIME: WHEEL_OFF - TAXI_OUT

DEPARTURE_DELAY: Total Delay on Depature

TAXI_OUT: The time duration elapsed between departure from the origin airport gate and wheels off

WHEELS_OFF: The time point that the aircraft's wheels leave the ground

SCHEDULED_TIME: Planned time amount needed for the flight trip

ELAPSED_TIME: AIR_TIME + TAXI_IN + TAXI_OUT

AIR_TIME: The time duration between wheels_off and wheels_on time

DISTANCE: Distance between two airports

WHEELS_ON: The time point that the aircraft's wheels touch on the ground

TAXI_IN: The time duration elapsed between wheels-on and gate arrival at the destination airport

SCHEDULED_ARRIVAL: Planned arrival time

ARRIVAL_TIME: WHEELS_ON + TAXI_IN

ARRIVAL_DELAY: ARRIVAL_TIME - SCHEDULED_ARRIVAL

DIVERTED: Aircraft landed on airport that out of schedule

CANCELLED: Flight Cancelled (1 = cancelled)

CANCELLATION_REASON: Reason for Cancellation of flight: A - Airline/Carrier; B - Weather; C - National Air System; D - Security

AIR_SYSTEM_DELAY: Delay caused by air system

SECURITY_DELAY: Delay caused by security

AIRLINE_DELAY: Delay caused by the airline

LATE_AIRCRAFT_DELAY: Delay caused by aircraft

WEATHER_DELAY: Delay caused by weather

Columns that contain null values

TAIL_NUMBER = 0.252978%

DEPARTURE_TIME, **DEPARTURE_DELAY** = 1.48053%

TAXI_OUT, **WHEELS_OFF** = 1.53026%

SCHEDULED_TIME = 0.000103%

WHEELS_ON, **TAXI_IN**, **ARRIVAL_TIME** = 1.58982%

AIR_TIME, **ELAPSED_TIME**, **ARRIVAL_DELAY** = 1.80563%

CANCELLATION_REASON = 98.4554%

CANCELLATION_REASON, **SECURITY_DELAY**, **AIRLINE_DELAY**, **LATE_AIRCRAFT_DELAY**, **WEATHER_DELAY** = 81.725%

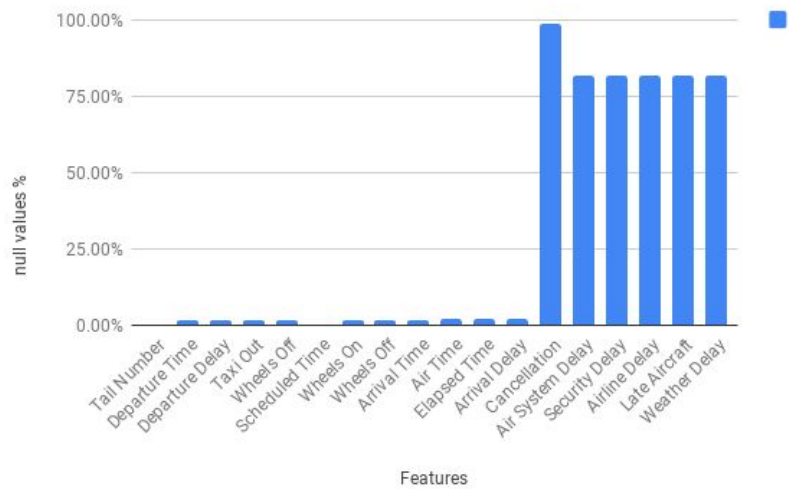

Fig 1 - Features containing null values and its percentage

Class balance

This dataset will be splitted in two classes:

DELAYED

NOT DELAYED

Every flight having a departure delay greater than 0 will be considered as **delayed**, while having 0 or less departure delay is considered **not delayed**.

During 2015 there were 2125618 flights delayed and 3607308 flights that were ahead of its departure time.

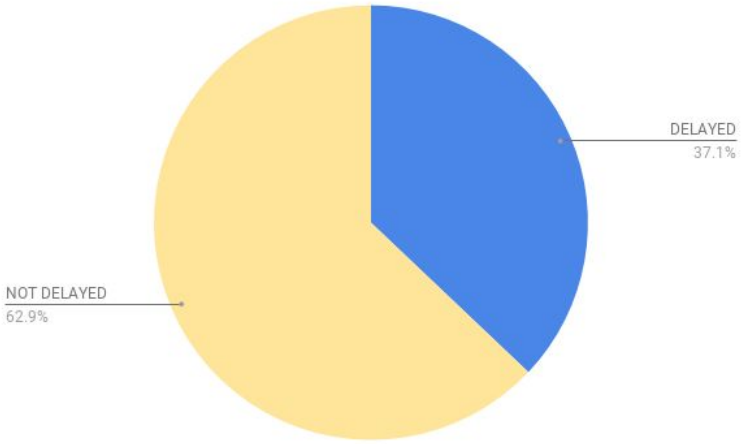

Fig 2 - Distribution of classes Delayed vs Not Delayed

Data specificity

Since the dataset contains a whole year of domestic flights records in the United States, the data can be too specific. This means that some features can have a wide domain, almost as if they were continuous features. Some of this features includes the FLIGHT_NUMBER and TAIL_NUMBER, they can be even unique for every flight so it is necessary to understand how they behave and how much specificity they carry.

In total there are 4898 unique tail numbers in 5819079 flights on the dataset. Meaning that 0.0841713955078% of the records are unique.

In total there are 6952 unique flight numbers in 5819079 flights on the dataset. Meaning that 0.119469077495% of the records are unique.

It can be possible that this two information have some correlation with other features of the dataset, maybe the flight number is related to the destination and origin.

It is seen on the dataset that even when the same flight number matches with the tail number the flight is different on it's several time related features, so it can lead to a different result (Delayed or Not Delayed).

Thus it was decided to keep both features for the classification.

Algorithms and Techniques

Given the nature of the dataset, having a lot of categorical features and a few numerical, the classifier needs to work well with both discrete and continuous data. Also, having the application of

this model as a product for web applications such as skyscanner, trivago, and other websites that uses this information to promote or list flights for sell, the classifier needs to perform quickly when predicting.

The two algorithms used in this project are a Decision Tree Classifier and a Random Forest Classifier.

About Decision Trees

A Decision Tree is a non-parametric supervised learning method used for classification and regression. As its name suggests, it is a tree of questions with binary answers (true or false) and each of this questions narrow the possible results until only one result is selected and so a decision is made[4].

Decision Trees are really simple to understand and to interpret. They use a white box model and their decisions can be easily explained since there are a lot of boolean choices. They perform well on both continuous and discrete data, and do not require a lot of computational resources for training and predicting.

About Random Forest Classifiers

A Random Forest is an ensemble model made with various Decision Trees that vote for a final decision. Each tree in the ensemble is built from a sample with replacement from the training set, and when the nodes are split during the construction, the split chosen is no longer the best among all features but the best of a random subset of features. This creates various trees that are capable of taking decisions but in a slightly different way, making the variance of this model decrease while the bias increases just a little, generating an overall better model than a single tree[5].

Since both methods are decision tree based their hyperparameters are pretty similar, and the main two that are going to be used in this project are:

1. Max Depth
2. Number of Estimators

Max Depth

Max Depth is the maximum depth of the tree, this being how many nodes are going to be generated until a decision is taken. Its default value generates nodes until all leaves are pure, causing the tree to overfit.

Number of Estimators

[4] http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[5] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

This is an exclusive for ensemble models and it is the total number of trees generated by the model. As a rule of thumb, increasing the number of trees in the ensemble will never increase the generalization error, but it can be computationally expensive[6].

Benchmark

The benchmark for this project is a Dummy Classifier[7], since it is the way every flight is classified today. Every flight is going to be classified as NOT DELAYED, since this is the majority class (60:40) and this gives an accuracy of 0.6049 and a F1 score of 0.7538.

The Dummy Classifier was trained with a sample of data of the first 10 weeks and splitted into a training and testing set of 67% and 33% accordingly.

III. Methodology

Data Preprocessing

This process was done in a jupyter notebook titled II - Analysis and it can be complemented by the blocks of code in it.

Cleaning Unobtainable Data

Since the objective for the project is to predict if a flight is going to be delayed or not, it is impossible to use information from the future, so the features that indicates events that have not occured yet will be discarded.

- DEPARTURE_TIME
- DEPARTURE_DELAY
- TAXI_OUT
- WHEELS_OFF
- ELAPSED_TIME
- AIT_TIME
- WHEELS_ON
- TAXI_IN
- ARRIVAL_TIME
- ARRIVAL_DELAY
- DIVERTED
- CANCELLED
- CANCELLATION_REASON
- AIR_SYSTEM_DELAY
- SECURITY_DELAY
- AIRLINE_DELAY

---

[6] https://arxiv.org/abs/1407.7502

[7] http://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html#sklearn.dummy.DummyClassifier.predict

- LATE_AIRCRAFT_DELAY
- WEATHER_DELAY

Cleaning Null Data Points

As shown on the fig 1 removing null data points will remove less than 0.5% of the information, which in this case is not significant due the size of the dataset.

Feature Engineering

The information of the dataset is usable, it contains features about the date of the flight, the duration of the flight in minutes, the origin and destination airport and even the airline and the tail number. But some of this information require previous treatment to make use in the classification models.

It is important to understand the domain of each feature and it's type.

- YEAR int64
- MONTH int64
- DAY int64
- DAY_OF_WEEK category
- AIRLINE category
- FLIGHT_NUMBER category
- TAIL_NUMBER category
- ORIGIN_AIRPORT category
- DESTINATION_AIRPORT category
- SCHEDULED_DEPARTURE int64
- SCHEDULED_TIME float64
- DISTANCE int64
- SCHEDULED_ARRIVAL int64
- LABELS category

The year, month and day features are highly related each other. In fact, since all data was collected from 2015 the year feature is constant, so it can be removed. The month and day are important, since it can be possible to have a high volume of delayed flights in winter because of snow storms and such, but the day and month itself don't contain readable information for the model since they are looked as individual variables. For example, if in the model exists a node that uses the day feature to make the split it will only look at the day to separate the data, and it is not the same to be in the 15th day of april and the 15th day of november. This dependence between the data can be translated to something that also represents time in a categorical manner by making them a single feature containing the week number of the year.

The only numeric features are:

- SCHEDULED_DEPARTURE
- SCHEDULED_TIME
- DISTANCE

- SCHEDULED_ARRIVAL

Since each of them is measuring units (time in minutes and distance in kilometers). Every other feature is categorical because there is not a numerical scale between them, there are only classes to belong or not.

The final dataset contains 11 features and 1 label, four of this features are continuous and 7 are discrete.

Implementation

This process was done in a jupyter notebook titled III - Methodology and it can be complemented by the blocks of code in it.

Preparing the Data for Training and Testing

To use the data for training and testing the models a train test split was used, using 33% of the data for testing and 67% for training.

The first attempt of training the dummy model was a failure because the categorical features can only be used after being converted to numerical features, so the get_dummies function from pandas is used to make the conversion. Each categorical feature is transposed into several features that contains the name of the main feature, an underline and then a value, and its value is converted to 0 and 1.

The second try was also a failure because of the size of the data. The transposed data generated a dataframe too big and require too much computational resources, so the solution was to reduce the dataset to only a few weeks of data instead of a year.

The training dataset contains 608163 data points, while the testing dataset contains 299544 data points.

Training the Dummy Classifier

A dummy classifier was trained using the most frequent strategy, this generates a classifier that always returns the same class, in this case the Not Delayed class.

The accuracy for this dummy classifier is 0.6049 while its F1 score is 0.7538.

Selecting the Best Model

The best model was selected based on its predicting time and two scores: F1 score and Cohen Kappa Score.

For this three helper functions were defined:

1. train_classifier - Trains a classifier and logs the time of the process.

2. predict_labels - Makes predictions, logs the time and evaluate the model with F1 score and Kappa score.
3. train_predict - Calls previous functions on both training and testing datasets.

Also to compare its performance of different sizes of datasets three subsets were created containing 10000, 20000, and 30000 data points.

Classifer 1 - Decision Tree

Training Set Size	Training Time	Prediction Time (test)	F1 Score (train)	F1 Score (test)	Kappa Score (train)	Kappa Score (test)
10000	2.5820	44.3490	1.0000	0.6723	1.0000	0.1476
20000	7.8360	43.9529	1.0000	0.6804	1.0000	0.1660
30000	12.9322	44.2589	1.0000	0.6885	1.0000	0.1814

Classifer 2 - Random Forest

Training Set Size	Training Time	Prediction Time (test)	F1 Score (train)	F1 Score (test)	Kappa Score (train)	Kappa Score (test)
10000	3.5879	51.3358	0.9903	0.6988	0.9750	0.1887
20000	8.2012	49.9041	0.9908	0.6998	0.9766	0.2013
30000	13.3635	51.3536	0.9898	0.7007	0.9741	0.2114

Fig 3 - Results of the comparison between the classifiers using datasets with different sizes

The training time for both models was almost the same but the prediction time was slightly slower in the Random Forest. While both classifiers shows signals of overfitting, having a perfect score on training sets, the testing score is better on the Random Forest.

Since the Random Forest is showing better results for almost the same performance, it is going to be selected as the best classifier for this project.

Refinement

For refinement a cross validation[8] was used, changing the two main hyper parameters for the Random Forest and using the Kappa Score as the scoring function. The biggest batch of the dataset was used, in this case the one containing 30000 data points.

The values for this parameters were:

- max_depth: 10, 20, 50, 100, 300
- n_estimators: 10, 20, 50, 100, 300

The selected values for the number of estimators was 20, while the max depth was 300.

The prediction time was 53.1027 seconds, almost the same registered during the model selection.

Both scores were highly better, the F1 score was 0.7248 and the Kappa score was 0.2291.

[8]http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html#sklearn.model_selection.RandomizedSearchCV

IV. Results

Model Evaluation and Validation

Comparing the Results with the Benchmark

Even after the hyper parameter tuning the obtained model performed worst than the dummy classifier, since the dummy obtained a F1 score of 0.7538 and the highest score obtained by the Random Forest was 0.7248.

Interpreting Kappa Score

The Cohen Kappa score is a metric that represents the amount of agreement between the classificator and the real results, and it ranges from -1 to 1, where negative values indicate not agreement at all and 1 being the maximum agreement between the two. In this case the Kappa score obtained by the selected model was 0.2291, which isn't a high score. A value of 0 represents the amount of agreement of random chance, so the obtained score actually shows some agreement. Cohen suggest that a value from 0.2 to 0.4 can be fair, but not substantial. It is important to note that any agreement less than perfect, represented by a score of 1, is also a metric for the reverse disagreement, indicating high levels of disagreement when the score is low[9].

Application of the Obtained Model

Since the model didn't perform well on the data, and even worse than a dummy classifier, the model shouldn't be used in a real environment and trusted without improvement.

Justification

There are a lot of things that can lead to a poor performance when using Decision Trees and Random Forest. One thing is the tendency to overfit, since the classifiers will be trying to be as deep as possible, it can easily create a dependency of the training data. The two classifiers used here showed high signs of overfitting, having almost perfect scores on the training sets. Using small trees should help control the overfit but in this case, when the cross validation was used a high value for the maximum depth was chosen, indicating that small trees didn't perform significantly good as well.

Also, high number of estimators tend to generalize more and as explained in the introduction of this project, a "rule of thumb" for the Random Forest is that the more trees are used the lower generalization error it will get[10]. In this scenario a small number of estimators were chosen by the cross validation process.

The obtained results can also be caused by the distribution of the classes, since the dataset is unbalanced (60:40), but the ratio is not that far. In cases when the distribution is 10:1 it is better to

[9] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/
[10] https://arxiv.org/abs/1407.7502

oversample or undersample the dataset to have balanced classes and not letting the classifier tend to classify everything as the majority class (as the dummy classificator did).

Finally the data collection can be leading to this results, when the data obtained have a possible random behaviour the tree wouldn't be able to generalize since all it sees is random points and can't understand the rules to classify.

In conclusion the model is not able to predict correctly enough instances accurately and can't be used in a real environment, so the problem indicated in the project was not solved, but it can serve as a stepping stone for other classifiers such as an XGBoost[11] or a mixed input Neural Network.

IV. Results

Reflection

The process used for this project was the following:

- A problem was selected and a public dataset was obtained
- The data was collected from the source (Kaggle - U.S. Department of Transportation)
- The data was analysed and preprocessed in order to be used by the classifiers
- Two classifiers were selected as possible models for solving the problem (Decision Tree and Random Forest)
- Data engineering was used to improve the data
- A benchmark was created to compare the final model
- The two classifiers were trained in samples of data and then compared
- One classifier was selected given it's performance
- The classifier was trained, tuned and evaluated

The hardest part for the implementation was to deal with the volume of the data. It looked like a very good idea to have a big dataset, but for binary classification small datasets are good enough and big datasets require a lot of computational resources. That lead to the point that was impossible to train the models using the whole dataset and only a small batch was used in the entire project.

It was really interesting to work with a real environment in mind since there were solid goals to achieve, having a small predicting time can make possible to a model like this to be used in production environments such as tourism websites that offer flights on a real market.

The final result wasn't the expected since the benchmark model that always predict every flight as not delayed performed better, but there were some big insights about the process that could be improved and also new ways to approach the problem.

Improvement

Definitely there are better ways to solve this binary classification problem. Starting by the sampling of the data, since in the project just the first 10 weeks of the year were used. A random selection

[11] https://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn

model can be used to sample the dataset and then used for training the models in a more generalized way. Also other classifiers can be tested because both models compared in the project were very similar (both works with decision trees). And lastly there were some hyper parameters that were left aside when doing the cross validation, like the criterion used to measure the information gain with the split.