Machine Learning Engineer Nanodegree

Capstone Project

Michael Moreira Cabral
June 14th, 2018

I. Definition

Project Overview

Supervised learning algorithm to classify aircraft turnarounds into different categories of towing behavior.

In the domain of airport operations, it is common to apply stochastic simulation and multi-criteria optimization to perform tasks such as evaluating passenger and baggage flows and aircraft stand allocation.

The proposed study aims to develop a method to enhance the accuracy of an aircraft stand allocation multi-criteria optimization algorithm, which in turn aims to mimic the allocation performed by human allocators from the airport operations team.

With a well calibrated stand allocation model, it is possible to evaluate the impact of new operations procedures and/or infrastructure expansion. For example, if one has a calibrated model, that correctly mimics the ability to allocate passengers in contact gates, it can be measured the impact, in terms of % of passengers processed in boarding bridges, of the construction of new bridges.

ANAC (Agência Nacional de Aviação Civil), Brazilian civil aviation authority, requires that a minimum of 95% of international passengers should be processed in boarding bridges. This is an important performance indicator, which the airport operator should carefully manage.

There are plenty of academic research regarding stand allocation algorithms, such as:

- The over-constrained airport gate assignment problem. H. Ding, A. Lim, B. Rodrigues, Y. Zhu. 2005.
- The use of meta-heuristics for airport gate assignment. Chun-Hung Cheng, Sin. C. Ho, Cheuk-Lam Kwan. 2012.
- Incorporation of recoverable robustness and a revenue framework into tactical stand allocation. Bert Dijk. 2016.

Using these algorithms, it is possible to create and implement rules that mimic the ones that are used by the human allocators from the airport operations team. Doing this, is expected that the final allocation performed by the algorithm should be similar to the one performed during the live operation. There are many groups of rules to be considered, such as:

- Geometric rules (max aircraft size per stand; adjacencies restrictions);
- Type of flight (the available contact gates depends on whether the flight is domestic or international);
- Business rules (to consider the preferable allocations of the airlines)
- Efficiency rules (to emulate the expertise of the human allocators)
- Boundary rules (as the simulator considers just one day, boundary rules are necessary)

The input to a stand allocation algorithm is a flight schedule, with the flight number, airline, arrival/departure times, aircraft type, aircraft registration, flight type and number of passengers. With the times of arrival and departure from each aircraft in the airport, it is possible to imagine “bars” (aircraft turnarounds) into the available lines (aircraft stands). The output of the algorithm can then be visualized with a Gantt chart.

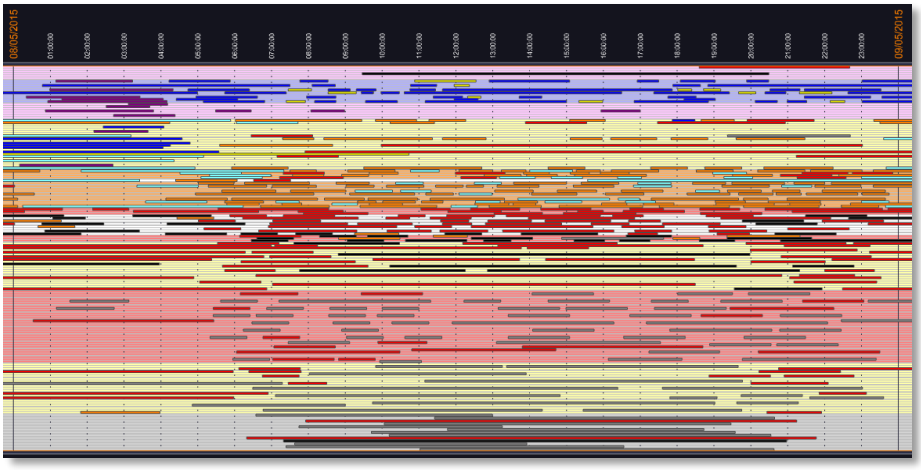

Figure 1 - Stand allocation Gantt chart

However, if you have an aircraft that arrives at 9AM and departs at 9PM it doesn’t necessarily means one will need to allocate a 12h long bar. Probably this turnaround will be broken in to pieces, namely, it will be broken into an arrival movement (first bar), a long stay movement, probably in a remote stand (second bar) and a departure movement (third bar). In order to allocate passengers in boarding bridges, one just need to take into account the first and third bars. These will be the ones to be allocated in the contact gates (more expensive infrastructure). The long stay part, that does not involve passengers processing, can be allocated in a remote stand (cheaper infrastructure).

Therefore, before applying the stand allocation algorithm, it is necessary to perform a preprocessing step, which is the break of the aircraft turnarounds with towings.

The problem with the stand allocation academic research is that they generally take this important preprocessing step for granted. Experience shows, though, that using rules of thumb like “break any movement with more than X hours of total turnaround time” will not be accurate and will lead to final results that are not close to what was done in live operation.

I personally faced this challenge when I was responsible to develop a simulation model, based on a stand allocation model, to estimate the impact of different operational procedures and infrastructure expansion in terms of % of passengers processed by boarding bridges.

Problem Statement

In order to successfully develop a stand allocation model, that can support many management decisions, it is crucial to also develop a preprocessing step, which is the break of the aircraft turnaround with towings.

Rule of thumb rules usually provided by the allocation personnel are not accurate. That is why a supervised learning classification model might be useful.

There are three categories of turnaround towing behavior:

Category	Label	Description
1	No towings	An aircraft arrives, is allocated in stand A and departs from the same stand.
2	Single tow	An aircraft arrives, is allocated in stand A, is towed to stand B and departs from this one.
3	Multiple tows	An aircraft arrives, is allocated in stand A, is towed to stand B, is towed again to stand C and departs from this one.

The classification model should be able to analyze turnaround input features and correctly predict which type of towing behavior it will present in live operation.

Datasets and Inputs

In order to perform this classification task, a dataset with 10.677 samples will be used. This dataset contains seven features and one label (*tow_type*).

Each sample of this dataset is a complete turnaround, namely, it contains data from the whole operation of an aircraft arriving and departing from the airport. The seven features are:

Feature	Description
acft_type	The ICAO aircraft type code (e.g. A320, E190, B737).
acft_cat	The ICAO aircraft category code (A, B, C, D, E, F).
in_block_hour	The hour when the aircraft in-block occurred (0 to 23).
off_block_hour	The hour when the aircraft of-block occurred (0 to 23).
total_time	Time span, in minutes, between off-block and in-block times).
turnaround_type	The first letter represents the arrival flight type and the second letter represents the departure one (D = domestic and I = international).
turnaround_qualifier	The first letter represents the arrival flight qualifier and the second letter represents the departure one (e.g. J = normal passenger flight; G = extra passenger flight; F = cargo flight).

The label is the column *tow_type*, which can assume the values 1, 2 and 3, as described in the previous section.

Based on my domain knowledge, these features are the best available candidates to determine which kind of towing behavior a turnaround will present. GRU Airport allowed the use of this dataset for the present study.

Solution Statement

A supervised learning algorithm will be developed, in order to predict which kind of towing behavior a turnaround will present, based on its features.

A training data set, labeled data from live operations and containing a set of turnaround features will be used to train three different learning algorithms: DecisionTreeClassifier, SVC and AdaBoostClassifier. The model with the best performance, in terms of F1 score, will be chosen and further tuned.

The final model will be able to predict the towing behavior based on turnaround features. The predicted classes will then be imported to the simulation software, allowing the development of more accurate stand allocation models.

Metrics

The optimal model will be the one with the highest F1 score. As this is a multiple-label classification problem, the final score will be the weighted average of the FI score of each class. The F1 score for each class can be calculated as:

$$F1 = \frac{(precision \times recall)}{(precision + recall)}$$

As the dataset is expected to be unbalanced in terms of its label (*tow_type*), F1 score seems a better option than accuracy. The best model will be the one that presents a good balance between precision and recall.

If the precision is very low, it means that the model is suggesting an unreasonably high number of tows. This would incorrectly increase the ability of the stand allocation model to allocate flights in contact gates, creating a positive bias, thus harming the calibration process.

If the recall is very low, it means that the model is suggesting an unreasonably low number of tows. This would incorrectly decrease the ability of the stand allocation model to allocate flights in contact gates, creating a negative bias, thus harming the calibration process.

II. Analysis

Data Exploration

The original dataset contains 10.677 samples, 7 features and 1 multiclass categorical label.

	acft_type	acft_cat	in_block_hour	off_block_hour	total_time	turnaround_type	turnaround_qualifier	tow_type
0	AT72	C	21	21	33	DD	JJ	No towings
1	A320	C	0	6	376	DD	JJ	Single tow
2	A320	C	0	7	435	II	JJ	No towings
3	B734	C	0	4	234	DD	AA	No towings
4	B738	C	0	6	396	ID	JJ	Single tow

Figure 2 - DataFrame created from the dataset (five first columns).

From the seven features, three are numerical and four are categorical:

```
acft_type              object
acft_cat               object
in_block_hour           int64
off_block_hour          int64
total_time              int64
turnaround_type        object
turnaround_qualifier   object
tow_type               object
```

Figure 3 - Data types of each column.

The in-block and off-block hours present an approximately uniform distribution (more details are presented in the exploratory visualization section).

The total time feature, however, presents a right-skewed distribution, clearly with the presence of outliers.

	in_block_hour	off_block_hour	total_time
count	10677.000000	10677.000000	10677.000000
mean	12.981830	13.480191	191.911773
std	6.216499	6.062116	239.438139
min	0.000000	0.000000	1.000000
25%	8.000000	9.000000	58.000000
50%	13.000000	14.000000	95.000000
75%	18.000000	19.000000	208.000000
max	23.000000	23.000000	5244.000000

Figure 4 - Numerical columns statistics

The Boeing 737-800 (B738) is the most widely used aircraft type in the airport. In terms of aircraft category, most are C.

Most of the turnarounds are composed of domestic (D) regular passenger (J) flights, both in arrival and in departure (DD and JJ).

In most turnaround there are not any tows.

	acft_type	acft_cat	turnaround_type	turnaround_qualifier	tow_type
count	10677	10677	10677	10677	10677
unique	28	4	4	26	3
top	B738	C	DD	JJ	No towings
freq	2607	9119	7727	9756	9137

Figure 5 - Categorical columns statistics

All the categorical features seem to be unbalanced in terms of their classes (more details are presented in the exploratory visualization section).

Exploratory Visualization

For the purposes of this study, the in_block_hour and off_block_hours can be considered relatively balanced. Except between 1AM and 4AM, which is a period of low traffic, the operations can be considered well distributed during the day. This is a typical characteristic of a congested airport, when the demand almost fully fills the capacity.

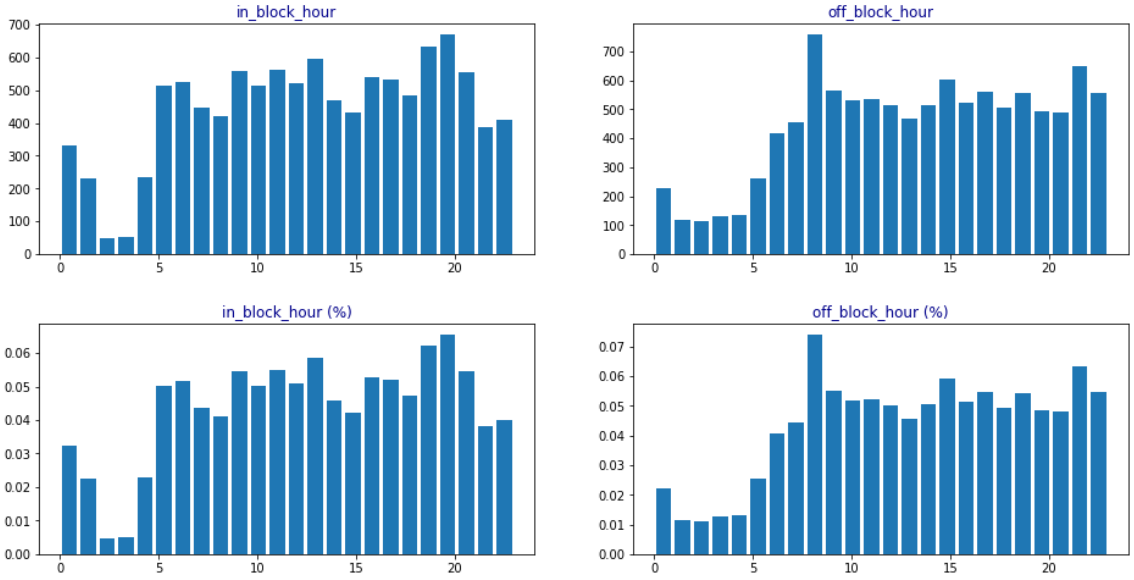

Figure 6 - in_block_hour and off_block_hour histograms

The total_time feature presents a right-skewed distribution, clearly with the presence of outliers. The boxplot shown in Figure 7 illustrates these outliers.

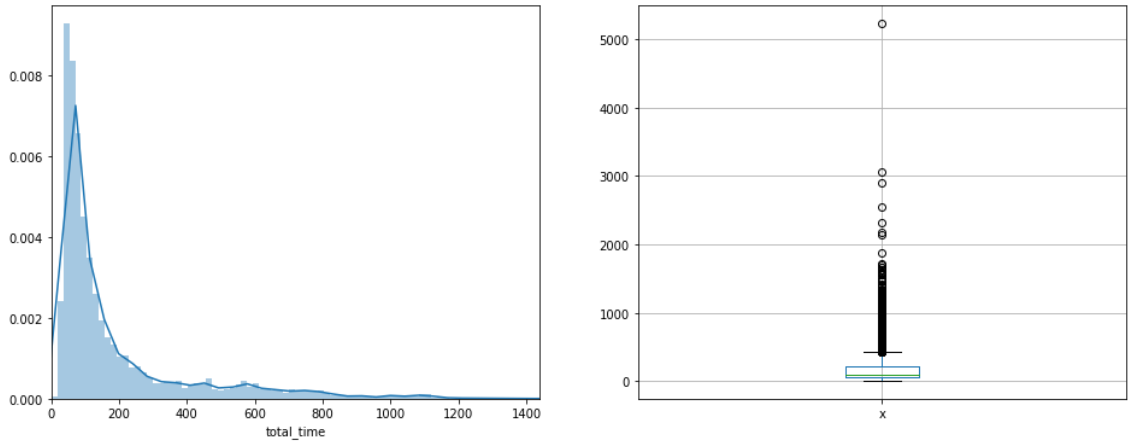

Figure 7 - total_time histogram and boxplot

There is an interesting relation between total_time and in_block_hour. It is common to have international long haul flights arriving in the morning (between 4AM and 6AM) and departing only at night. It is also common that flights that arrive after 9PM stay overnight and depart only in the next day.

Figure 8 shows that flights that arrive in these two intervals usually have longer turnaround times.

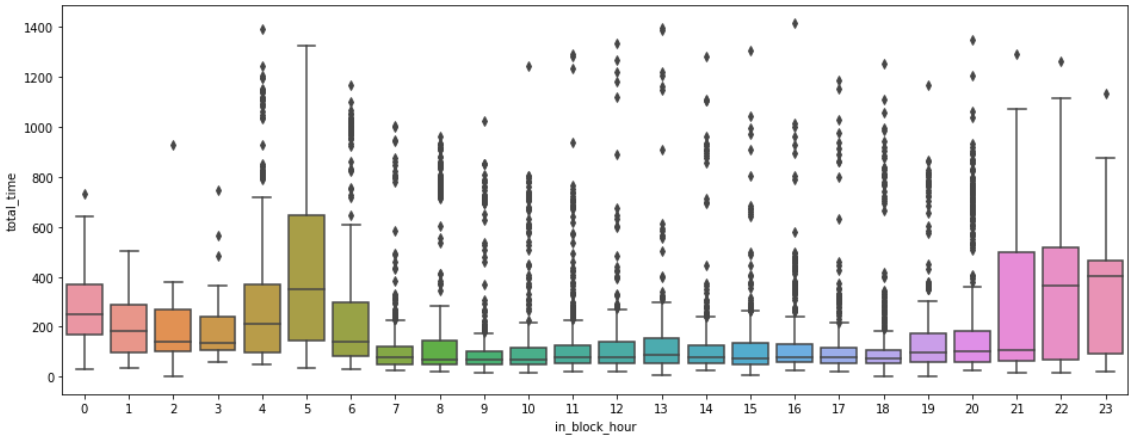

Figure 8 - total_time boxplots by in_block_hour

Most of the aircrafts operating in this airport are category C. These aircrafts are generally used for short and medium haul flights. For long haul flights, the most used are category E.

Most of the aircraft turnarounds connect two Brazilian locations. The aircraft arrives as a flight from a Brazilian origin and departs as a flight to a Brazilian destination (DD). It is also very common to have turnarounds connecting two international locations. In this cases, the airport acts like an international hub, connecting South America to the globe (II).

In most turnaround there are not any tows. The tow_type histogram in Figure 9 shows that the label is unbalanced in terms of its classes.

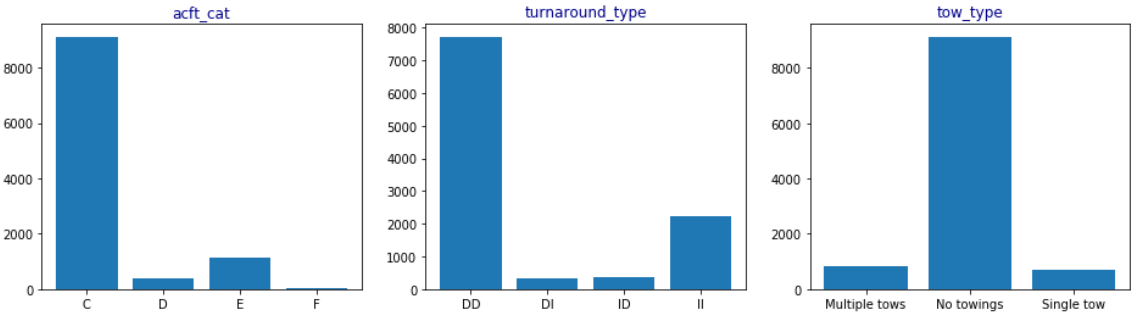

Figure 9 - acft_cat, turnaround_type and tow_type histograms

Aircrafts from Boeing and Airbus are the most used in the airport. The use of aircrafts from Embraer and ATR is also relevant. The histogram in Figure 10 shows the relative use of each type of aircraft.

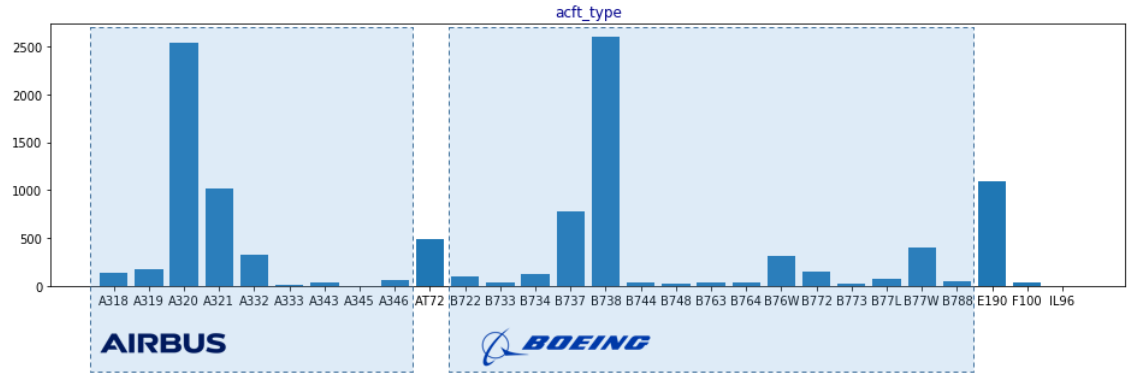

Figure 10 - acft_type histogram

The vast majority of the turnarounds qualified as JJ, namely, composed of a normal passenger arrival and a normal passenger departure. Other relevant qualifiers are:

- JG: regular passenger arrival + extra passenger arrival (GJ is the opposite)
- AA: non-regular cargo arrival + non-regular cargo departure
- MM: mailing arrival + mailing departure
- FF: regular cargo arrival + regular cargo departure

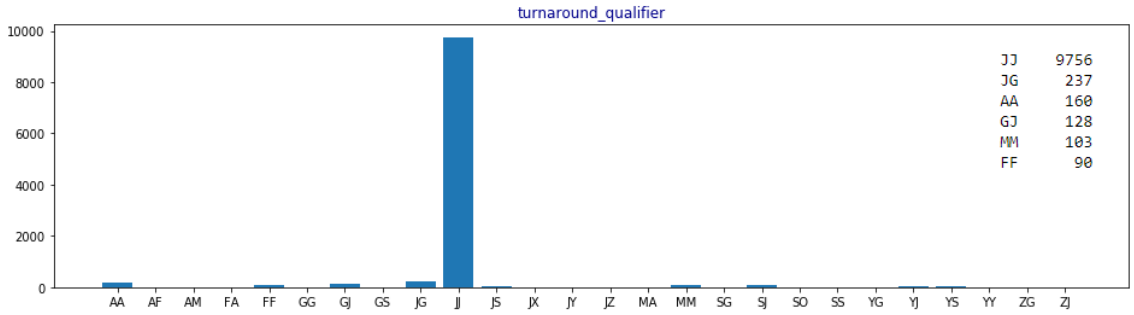

Figure 11 - turnaround_qualifier histogram

Algorithms and Techniques

Three sklearn classification models are going to be used: **DecisionTreeClassifier**, **SVC** and **AdaBoostClasisfier**. Below, for each of these algorithms, some of their strengths and weaknesses are described. Additionally, it is also explained why they may be good candidates to solve the proposed problem.

Much of what is exposed in this section is the result of research (please find more details about the references in the end of this report) and from previous valuable lessons and projects from Udacity.

Before applying each of the following learning algorithms, processes will be executed to remove outliers and generate dummies from the categorical features.

Decision Trees

Strengths: Trees are easy to understand and to visualize. Their outputs are rule-based, thus they can be translated into if-then-else rules and be incorporated in a broader range of softwares (as the airport simulator mentioned above). Other advantages, as listed in the scikit-learn documentation, are the little data preparation requirement and the ability to handle both numerical and categorical data.

Weaknesses: They can create over-complex trees, thus causing overfitting and they also create biased trees if some classes dominate. To avoid that, in the initial training and evaluation of the different models, the DecisionTreeClassifier will be trained with max_depth=10.

Decision trees are usually good candidates due to their simplicity and easiness to interpret. Even if their results are not the best, good information can be extracted from the model, like an estimate of the relative importance of each feature. Indeed, later in this report a list of the most important features for the classification task is presented.

Support Vector Machines

Strengths: One of the main strengths of SVMs, as mentioned in the Udacity's classes, is their ability to use different kernel functions in order to run the algorithm in other domains, in which

the data can be more easily separable. SVMs work with the concept of margin (they do not only look for lines that separate the data, but for the ones that do so with the biggest margins), so they are less prone to overfitting when comparing to other linear models. Another advantage, as listed by Kotu and Deshpande (2014), is the fact that, once an SVM model is built, small changes to the training data will not result in significant changes to the model coefficients as long as the support vectors do not change.

Weaknesses: As stated by Géron (2017), the training complexity is usually between $O(n^2)$ and $O(n^3)$. So it can get very slow when the number of training instances gets large. As mentioned by Kelleher and Namee (2015), they are not very interpretable, and, especially when kernel functions are used, it is very difficult to understand why a particular prediction has been made.

This model is a good candidate because, as described in the scikit-learn documentation, it is still effective in cases where the number of dimensions is large. As dozens of dummy variables are going to be created, this property might be useful.

Ensemble Method - AdaBoost

Strengths: As described by Géron (2017), in general, ensemble modeling reduces the generalization error that arises due to overfitting the training set. This is possible, as explained in the scikit-learning documentation, because several weak learners (less prone to overfitting) are combined to produce a powerful ensemble.

Weaknesses: The weak learner needs to be chosen carefully. If the weak learner itself is over-complex, the ensemble method may not avoid overfitting.

This model is a good candidate because it tends to boost the results of base learners. In its default configuration, AdaBoost's base estimator is the DecisionTreeClassifier. So as we have chosen decision trees as one of our candidate models, it may be a good idea to take AdaBoost too and compare them.

The three learning algorithms will be applied and the choosing decision will be based on the analysis of their learning curves. The chosen algorithm is then tuned varying the values of some of its main hyperparameters, using the grid search technique and the F1 weighted score.

Benchmark

The benchmark model is the one I developed at GRU Airport in 2015. Airport simulation softwares and techniques evolved rapidly in the aviation sector. Experts from Paragon and ARC – Airport Research Center, reviewed at that time our stand analytics model (supervised learning for towing classification + multi-criteria optimization for stand allocation).

The supervised learning task was performed by a simple decision tree algorithm, which provides rules that could be inserted in the simulation software.

At that moment, I did not know much about machine learning best practices. Today I have better ideas that for sure will enable more creative and efficient solutions. The new results can be compared to the legacy ones with the F1 score.

The weighted F1 score of the legacy model (see Figure 12) when applied to the dataset used in this project is 0.8961.

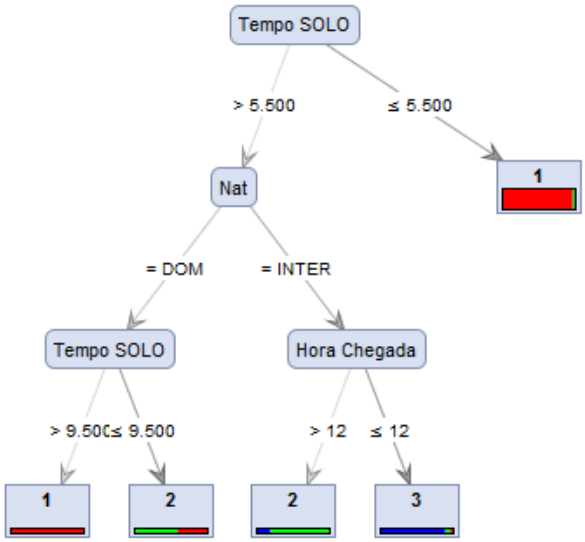

Figure 12 - GRU Airport towing classification model (legacy)

III. Methodology

Data Preprocessing

Missing Values

The *pandas.DataFrame.isnull().values.any()* was applied to confirm that there was not any missing values.

Outliers

As previously exposed, the total_time feature shows the presence of outliers. As the simulation studies usually consider 1 day of operation, it is a good practice to exclude flights with a total turnaround time greater than 24h (1440 minutes). Figure 3 presents the new histogram of this feature after the removal of its outliers.

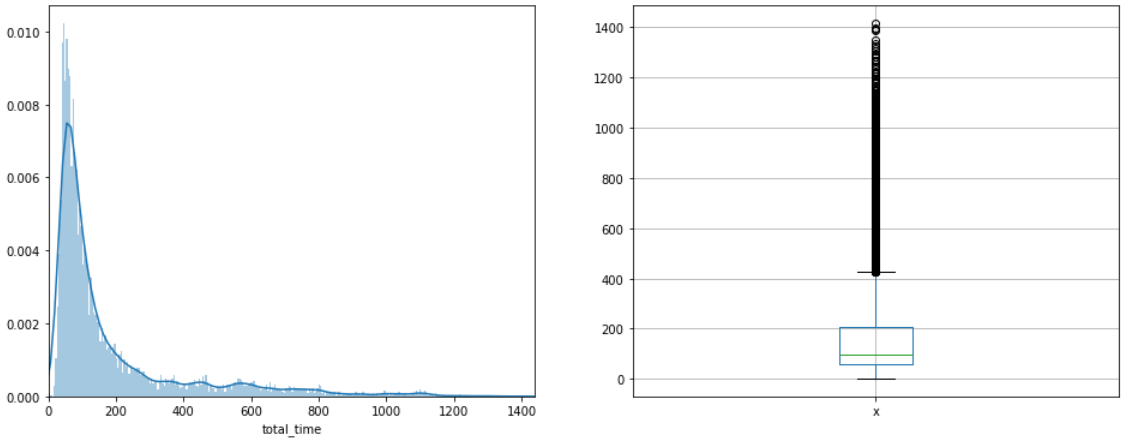

Figure 13 - Removal of turnarounds with total time greater than 1.440 minutes.

Most of the turnaround qualifiers are very rare. Some of them also cannot be used for prediction purposes. For example, the qualifier “Y” refers to diverted flights. This flight condition only takes place during live operation. Additionally, it rarely occurs, so it does not worth it to try to elaborate some way to predict them.

Only the six most frequent qualifiers were kept. Figure 14 illustrates the new histogram, after this qualitative outliers removal.

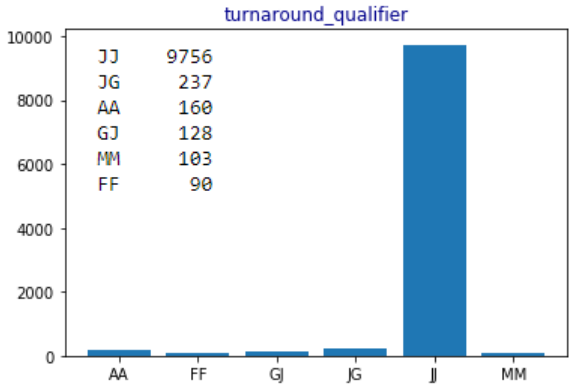

Figure 14 - Only the six most frequent qualifiers were kept.

The new transformed dataset was slightly reduced from 10.677 to 10.452 samples.

Dummy Variables

The original dataset is composed of seven features plus one label. From these seven original features, three are numerical and four are categorical.

The 4 categorical features were used to generate 42 dummy features.

	in_block_hour	off_block_hour	total_time	tow_type	acft_type_A318	acft_type_A319	acft_type_A320	acft_type_A321	acft_type_A332	acft_type_A333	...
0	21	21	33	No towings	0	0	0	0	0	0	...
1	0	6	376	Single tow	0	0	1	0	0	0	...
2	0	7	435	No towings	0	0	1	0	0	0	...
3	0	4	234	No towings	0	0	0	0	0	0	...
4	0	6	396	Single tow	0	0	0	0	0	0	...

5 rows × 46 columns

Figure 15 - DataFrame with the dummy variables.

The new transformed dataset contains 45 features (3 original + 42 dummies) plus 1 label.

Implementation

The dataset was shuffled and split into training (70%) and testing (30%) sets, stratified on the label.

The three classifiers and the f1_score modules were imported from sklearn. When creating the three classifiers, random states were defined. Two generic functions were created, to perform the fitting of the algorithms and the prediction plus scoring.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import f1_score

clf_A = DecisionTreeClassifier(random_state=1, max_depth=10)
clf_B = SVC(random_state=1)
clf_C = AdaBoostClassifier(random_state=1)

def train_classifier(clf, X_train, y_train):
    clf.fit(X_train, y_train)

def predict_score(clf, features, target):
    y_pred = clf.predict(features)
    return f1_score(target.values, y_pred, average='weighted')
```

Figure 16 - Importing the classifiers and the f1 scorer plus defining generic functions.

In order to better understand the behavior of the learning curves, each learning algorithm was used and evaluated for 12 training sets with incrementally larger amount of samples (from 600 to 7.200 samples). Figure 17 presents the results.

	clf_A_train	clf_A_test	clf_B_train	clf_B_test	clf_C_train	clf_C_test
600	0.991401	0.89069	0.97036	0.835459	0.960896	0.898575
1200	0.98861	0.899492	0.968872	0.861247	0.94108	0.904178
1800	0.985595	0.907522	0.951268	0.87413	0.933254	0.903525
2400	0.981505	0.918814	0.946906	0.880034	0.922729	0.893858
3000	0.975883	0.913676	0.941377	0.884895	0.917543	0.900692
3600	0.972353	0.914434	0.941029	0.89251	0.912945	0.897203
4200	0.967711	0.91821	0.943944	0.892332	0.918136	0.904176
4800	0.967632	0.913315	0.944456	0.895449	0.924207	0.913188
5400	0.964097	0.921862	0.94021	0.895233	0.91793	0.908954
6000	0.960353	0.92329	0.936193	0.89704	0.915253	0.911678
6600	0.964122	0.923117	0.936314	0.899787	0.918489	0.912047
7200	0.961608	0.919462	0.935844	0.897578	0.915992	0.913464

Figure 17 - Results from each training, prediction and scoring iteration.

It is possible to see that, for the largest training set size, SVCs performed poorer.

Decision Trees and AdaBoost had similar performances.

Figure 18 illustrates the learning curves from the three algorithms.

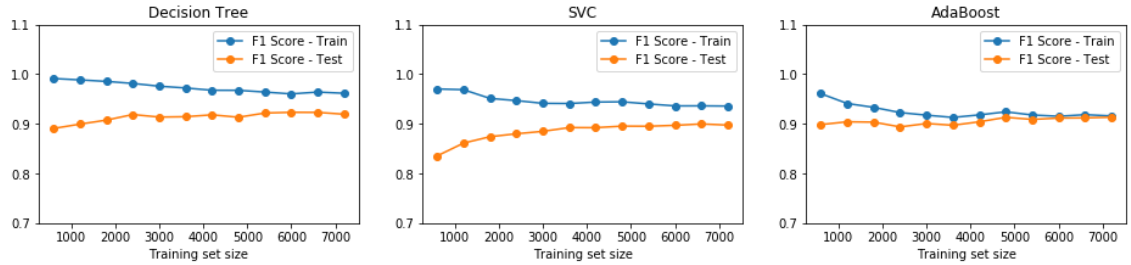

Figure 18 - Learning curves from the three learning algorithms.

As the SVC presented the worst performance, it was discarded.

As the performances from the Decision trees and AdaBoost were very similar, another characteristic was determinant for the choice of the model: the convergence between the training and testing scores.

It is valuable to be able to more accurately predict the expected performance in unseen data based on the training score. Therefore, after applying this tiebreak factor, the chosen model was AdaBoost.

However, as previously anticipated, the Decision Trees also contribute to the knowledge discovery process by listing the most important features for the classification task. Figure 19 presents the five most important features. Note that the total turnaround time is by far the most important.

feature	importance
total_time	0.631162
acft_cat_C	0.152133
in_block_hour	0.059915
off_block_hour	0.058742
turnaround_type_DD	0.025057

Figure 19 - The most important features according to the DecisionTreeClassifier.

Refinement

The Adaboost model was then tuned in terms of the number of estimators and the learning rate. The default values are 50 for the former and 1.0 for the latter.

The grid search technique was executed with 25 possible parameter combinations (5 possible values for each of the two tuning parameters).

Figure 20 presents the used code to configure and apply grid search cross validation.

```
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import make_scorer

parameters = {'n_estimators': [25, 50, 100, 125, 150],
              'learning_rate': [0.8, 0.9, 1.0, 1.1, 1.2]}

clf = AdaBoostClassifier(random_state=1)
f1_scorer = make_scorer(f1_score, average='weighted')
grid_obj = GridSearchCV(estimator=clf, param_grid=parameters, scoring=f1_scorer)
grid_obj = grid_obj.fit(X_train, y_train)
clf = grid_obj.best_estimator_
```

Figure 20 - Configuring and applying grid search cross validation.

The Adaboost model, before the grid search tuning, as can be seen in Figure 17, presented a F1 score of 0.9134.

The tuning process changed the number of estimators from 50 to 125 and kept the default value of the learning rate. The F1 score of the tuned model is 0.9154.

```
predict_score(clf, X_test, y_test)
```

```
0.91549619991063236
```

```
clf
```

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=125, random_state=1)
```

Figure 21 - Parameters values and score of the tuned model.

IV. Results

Model Evaluation and Validation

The final model was derived from:

- Domain knowledge feature selection. Part of the available features were previously rejected based on previous studies and general knowledge about airport operations.
- Insights, based on the legacy model and previous Udacity's projects, of which models could be good candidates to solve the proposed problem.
- Application of a framework that incorporates many of the best practices in the machine learning domain. Practices of which the author did not know at the time of the development of the legacy model.
- An implementation framework which included a careful data preprocessing step, use of an appropriate scoring metric (weighted f1 score instead of just accuracy), evaluation of different candidate models, cross validation, learning curve analysis and model tuning.

The results of the final model are aligned with the expectations and will for sure contribute to reduce the bias of our stand allocation simulation model.

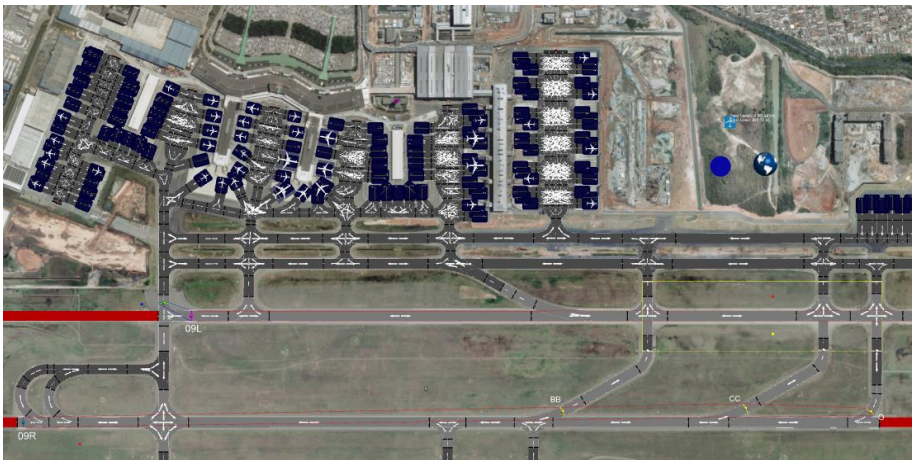

Figure 22 - Runway, taxiways, aprons and aircraft stands modelled in CAST Simulator

The final parameters values seems reasonable and the convergence.

From approximately 5.000 samples onwards, the training and testing scores showed a very stable convergence in the subsequent training set splits. This is a good indicator that, given a large enough dataset, the model will be robust independently of the training set split.

It is very nice that the model presented this kind of robustness with only a half of the provided dataset. For the airport operator, it is easy to provide datasets 20 to 40 times larger than the one used in this project.

Justification

The results of the final model were stronger than the ones from the legacy one.The weighted F1 score of the legacy model when applied to the dataset used in this project is 0.8961. In the case of the final model, the same scorer in the same conditions present a result of 0.9154.

This improvement justifies the update of the stand allocation model in order to reflect the more accurate towing rules. Before loading a flight schedule to the simulator, it should pass to the classifier predict method in order to get the towing behavior predicted label.

After that, the flight schedule should be loaded to the simulator, which will apply rules to read the predicted label and decide when and how to break the turnarounds.

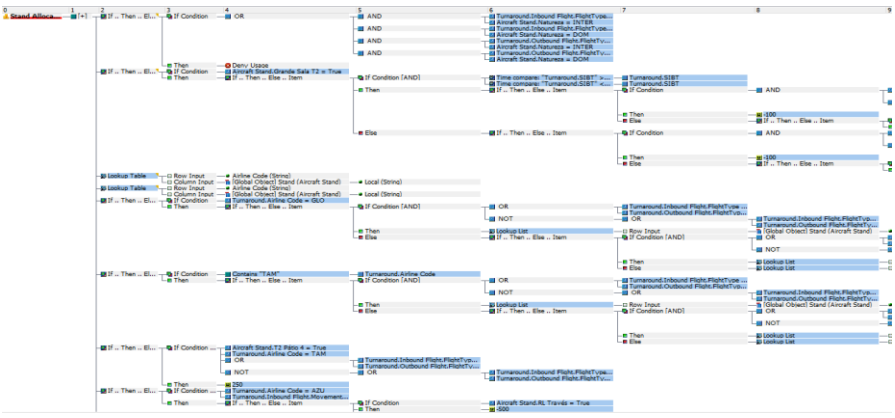

Figure 23 - An extract of towing and allocation rules in CAST Simulator

V. Conclusion

Free-Form Visualization

My free-form visualization, although simple, has a lot of meaning. It shows in the back the beginning of my learning process, using component based software, until the more recent proficiency in Python. That is a process in continuous improvement.

The available data have not changed a lot since 2015. However, it was intense the dedication in the last years to better understand the machine learning concepts and to “learn the meaning of learning”, thus avoiding many pitfalls.

Even the result of the worst candidate model, untuned, was best than the legacy model, that was used for years in the largest airport in South America. This shows the power of knowledge to add value for the individual and the society.

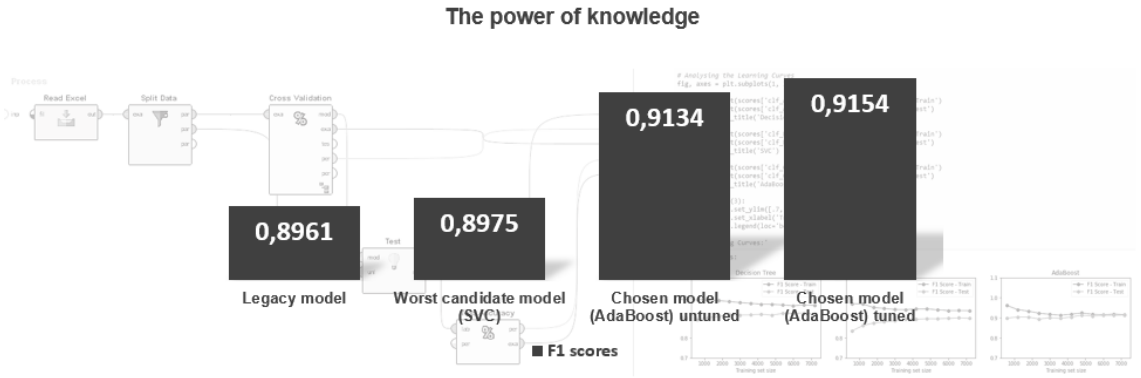

Figure 24 - The power of knowledge to add value.

Reflection

In my opinion, the most important quality about the project was the framework used:

- *Set up*
- *Data Exploration*
- *Feature Selection*
- *Data Preprocessing*
- *Training and Testing Data Split*
- *Training and Evaluating Models*
- *Choosing the Best Model & Model Tuning*

The exploratory visualization, part of the data exploration process, is a task that I think is really challenging. It is hard to discover the best visual ways to present and explore data. Many times, you try some tool, with great expectations, and the result is frustrating. It takes experience to improve the feeling.

The process of model choosing and tuning is also very intense. Each model has enough subtleties that would take you days for a complete understanding. Additionally, there are dozens of options for any given task. So it is always challenging to choose the best candidates.

Improvement

Certainly, there are many ways to improve the present project.

I would say that testing more learning algorithms is one of them. There are many classification algorithms that I am not yet comfortable to use. I intend to learn them soon, though!

Another possibility is to try new features. Actually, some of them could have been used in this project were it not for confidentiality issues. There are room for improvement though using filter or wrapping techniques for feature selection with some data that today the airport does not have access but will do in the near future.

References

Books

Kelleher, John; Namee, Brian. Fundamentals of Machine Learning for Predictive Data Analytics. MIT Press, 2015.

Kotu, Vijay; Deshpande, Bala. Predictive Analytics and Data Mining. Morgan Kaufmann, 2014.

Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly, 2017.

Websites

https://www.datasciencecentral.com/profiles/blogs/real-life-applications-of-support-vector-machines

https://medium.com/@ailabs/5-machine-learning-algorithms-and-their-proper-use-cases-a8cfd0cedb51

Other

Scikit-learn documentation and Udacity's classes.