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.

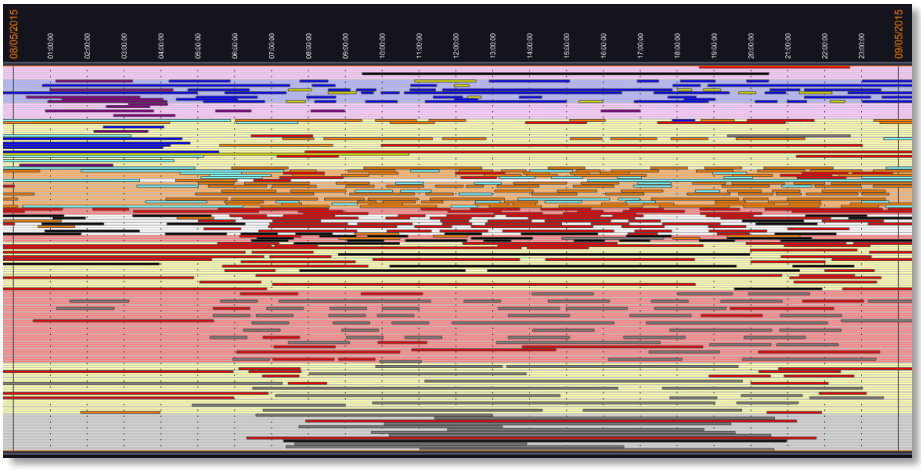

Figure 1 - Stand allocation Gantt chart

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

In this section, you will be expected to analyze the data you are using for the problem. This data can either be in the form of a dataset (or datasets), input data (or input files), or even an environment. The type of data should be thoroughly described and, if possible, have basic statistics and information presented (such as discussion of input features or defining characteristics about the input or environment). Any abnormalities or interesting qualities about the data that may need to be addressed have been identified (such as features that need to be transformed or the possibility of outliers). Questions to ask yourself when writing this section:

- *If a dataset is present for this problem, have you thoroughly discussed certain features about the dataset? Has a data sample been provided to the reader?*
- *If a dataset is present for this problem, are statistics about the dataset calculated and reported? Have any relevant results from this calculation been discussed?*
- *If a dataset is **not** present for this problem, has discussion been made about the input space or input data for your problem?*
- *Are there any abnormalities or characteristics about the input space or dataset that need to be addressed? (categorical variables, missing values, outliers, etc.)*

Exploratory Visualization

In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant. Questions to ask yourself when writing this section:

- *Have you visualized a relevant characteristic or feature about the dataset or input data?*
- *Is the visualization thoroughly analyzed and discussed?*
- *If a plot is provided, are the axes, title, and datum clearly defined?*

Algorithms and Techniques

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.

Decision trees are usually good candidates due to their simplicity and easiness to interpret. Even if their results aren't the best, good information can be extracted from the model, like an estimate of the relative importance of each feature.

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 WE ARE GOING TO GET DOZENS OF DUMMY VARIABLES, THIS MIGHT BE USEFULL, EVEN WITH OUR LARGE DATASET.

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.

In this section, you will need to discuss the algorithms and techniques you intend to use for solving the problem. You should justify the use of each one based on the characteristics of the problem and the problem domain. Questions to ask yourself when writing this section:

- *Are the algorithms you will use, including any default variables/parameters in the project clearly defined?*
- *Are the techniques to be used thoroughly discussed and justified?*
- *Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?*

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 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.

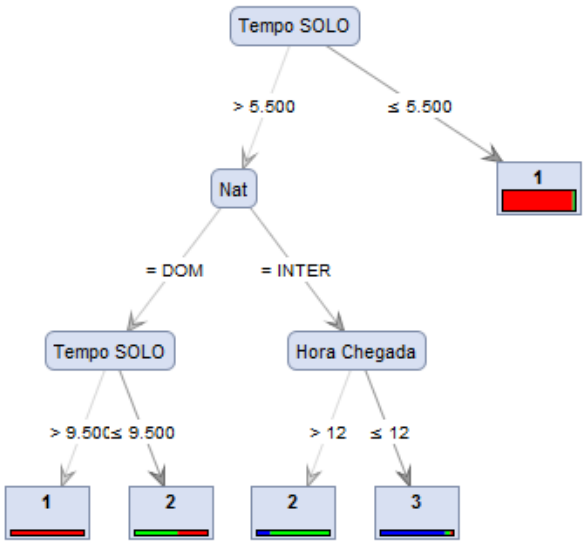

Figure 2 - GRU Airport towing classification model (legacy)

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.

In this section, you will need to provide a clearly defined benchmark result or threshold for comparing across performances obtained by your solution. The reasoning behind the benchmark (in the case where it is not an established result) should be discussed. Questions to ask yourself when writing this section:

- *Has some result or value been provided that acts as a benchmark for measuring performance?*
- *Is it clear how this result or value was obtained (whether by data or by hypothesis)?*

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section:

- *If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented?*
- *Based on the **Data Exploration** section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected?*
- *If no preprocessing is needed, has it been made clear why?*

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
- *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
- *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

- *Has an initial solution been found and clearly reported?*
- *Is the process of improvement clearly documented, such as what techniques were used?*
- *Are intermediate and final solutions clearly reported as the process is improved?*

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

- *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?*
- *Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?*
- *Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?*
- *Can results found from the model be trusted?*

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- *Are the final results found stronger than the benchmark result reported earlier?*
- *Have you thoroughly analyzed and discussed the final solution?*
- *Is the final solution significant enough to have solved the problem?*

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

- *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?*
- *Is the visualization thoroughly analyzed and discussed?*
- *If a plot is provided, are the axes, title, and datum clearly defined?*

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

- *Have you thoroughly summarized the entire process you used for this project?*
- *Were there any interesting aspects of the project?*
- *Were there any difficult aspects of the project?*
- *Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

- *Are there further improvements that could be made on the algorithms or techniques you used in this project?*
- *Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?*
- *If you used your final solution as the new benchmark, do you think an even better solution exists?*

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.

Before submitting, ask yourself. . .

- Does the project report you've written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?
- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?