Université Paris-Saclay

RAMP: Air Passenger Prediction

Meladianos Polykarpos
January 20, 2016

1 Introduction

Regression analysis is a statistical process for estimating the relationships among variables, in the hope of finding linear or non-linear projections of the feature space that can work well as function approximators.

In this project we will implement a system that deals with the prediction of the number of passengers for a given flight. In the first part we will briefly present the algorithms used for the analysis and discuss the processing stages of the system. In the second part we will present the results from the algorithms we tested, we will discuss and explain how we chose the best settings and algorithms in order to effectively predict the out-of-sample data.

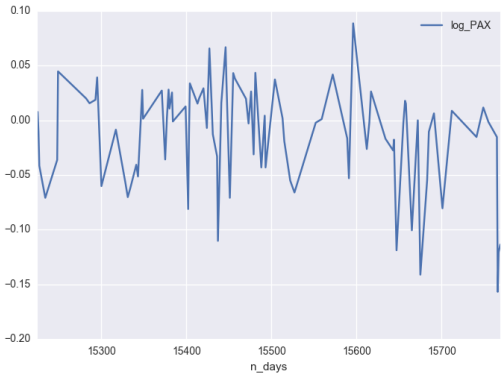
2 Feature Design

Feature Selection is of outmost importance for most machine learning tasks. In this section we will discuss how we chose the features based on the ones that were included on the initial dataset as well the ones based on the external dataset.

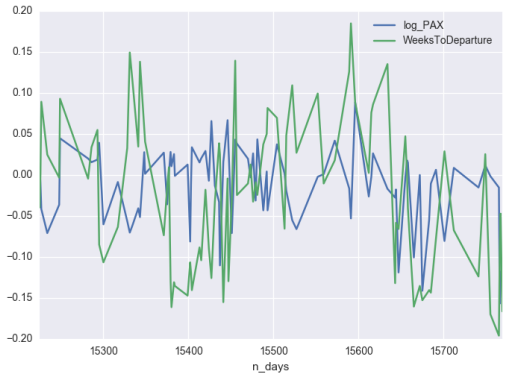
The initial data set contained the departure and arrival airport codes, the date of the flight the average and standard deviation of time the passengers booked their flights measured in weeks. While this are intuitively very informative features they may contain redundant information and they may fail to model the problem in detail.

Visualizing the pax with respect to the features helped in obtaining some meaningful information about the importance and role of each feature. For example in figure 1 we can see the pax plotted versus several features that were contained in the initial dataset. Diagrams like the one in the figure were produced for many more features but were not included in the report. An initial observation is that time can play a role as we can see that certain some weeks have less passengers. Another observation is that besides these certain timeframes, the number of passengers does not change a lot.

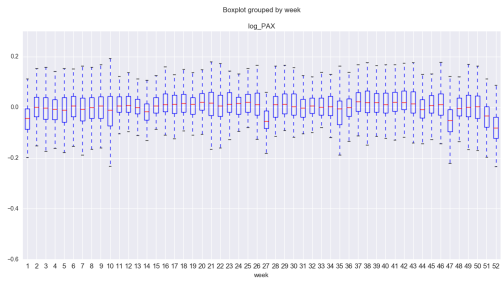
Weather data were also provided as an external source. Weather conditions may play a role for some passengers that may miss a fligh, but this is a very rare event, as flights are usually planned from passengers for business or leisure and people don't have the luxury to cancel any of those things. At the time of the booking the passengers are not able to know the weather since it is on average several weeks (11.46) before, as it can be seen from the data. The few passengers that book a flight the day before know the weather but the urgency behind their last-minute booking overshadows any bad weather that could discourage them. Our intuition was confirmed by testing the system with and without the weather data, where we observed that not only the data are not helpful but they also introduce a significant amount of noise, deteriorating the performance of the system. Our default out-of-the-box random forest regressor that was provided with the initial notebook scored a 0.9 RMSE cross-validation score with the weather data and 0.4 without the weather data, keeping other



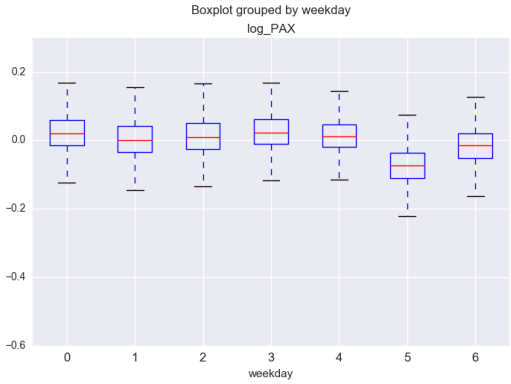
(a) 1a



(b) 1b



(c) 1b



(d) 1b

Figure 1: Visualizations of various Features compared with Pax

features and parameters unchanged.

So after dropping the external weather data, decisions had to be made on which features and in which form would describe the problem in the best manner. The first assumption made was that every route has it's own characteristics that do not rapidly change. It is trivial but when saying route we mean the directed departure-arrival combination. This initial assumption was strengthened by visualizing the log_pax versus the number of days per Route. In figure 1a we can see a diagram like this. But visualizing sampled diagrams is not enough for validating a hypothesis. The hypothesis at this point can be stated in the following manner: The series of log_pax versus time measured in days is stationary, with respect to the route. In order to test this assumption we perform the Augmented Dickey Fuller test using python's statsmodels.tsa.stattools toolbox. In 105 out of 125 unique routes in the dataset the test statistic was less than 10% of the critical value so the null hypothesis could be rejected and we assume that the series is stationary. In many of the remaining cases the value was very close although there were examples in which it was not the case. Ideally these cases should have been treated differently but for now we go on with this assumption and hope that the learning model will deal with the special cases.

Since the mean of the pax is unchanged for many of the routes we include it as a separate feature. While calculating the mean from the test data would have been valid in some cases, this always depends on how large is your test set. It could be the case that the test set is a single or very few predictions and so estimating route statistics for the test set is unreliable. So using the fact that the mean does not change per route, we include at test time the mean calculated from the reliable and large train set. This may help to construct a model of the form $y = x\_mean + n(t, Weeks\_to\_departure)$, where t represents features related to the date of the flight. By directly presenting the mean per route to the learning model, one can hope that it will be easier for system to model the noise seen in the above figure, as a non-linear combination of the other features rather than directly estimating the noisy function. However this is just an intuition and we did not force the model to have this form during training. The feature turned out to actually lower the rmse by 0,02 which is significant relatively to the improvement that other features gave us. Of course the improvement was not larger since the model itself could fit well enough on the target function even without this feature.

Another feature added using the external dataset is the distance between two airports. This is a feature that gives valuable information about a route and the passenger behaviour. Passengers are more likely to book earlier longer flights, long flights are not that often and consistent, which may lead to very different characteristics. We are not aware of how this directly impacts the number of passengers but since it includes valuable and related information we calculate it and introduce it as a feature. In order to calculate it we used the external dataset where the latitude and longitude of every airport was stored. Then for every flight, the great circle distance was calculated in nautical miles using the law of cosines:

$$angle = acos(sin(Latitude_1) * sin(Latitude_2) + cos(Latitude_1) \\ * cos(Latitude_2) * cos(Longitude_1 - Longitude_2)) \quad (1)$$

where the subscript indicates the respective airport coordinate and the fact that each degree on a great circle of Earth is 60 nautical miles:

$$distance = 60.0 * degrees(angle) \quad (2)$$

Other features that were included was the season and the number of enplanements of the departure and arrival airport. The first is pretty straight-forward since on high season the number of travellers may have a different pattern that cannot be captured by the week or month directly, like sudden bursts. The season was encoded in three categories very high (Mid December into the first week of January), high (Early January to mid April & mid June to mid August) and normal. The information about the season was collected from the tripadvisor forums. The feature was included but gave a negligible boost. The number of enplanements is an average of the passengers that boarded a plane per airport, collected from wikipedia data and joined offline with the airport information in the external data csv. This feature gives valuable information about the average expected passengers from a source and therefore the significance of the airport. A closely related feature, the population of the cities, was not included, because after careful examination of the data it seems that many airports serve as a hub for flights and so the population of the cities is not directly relevant to the expected number of passengers.

Since the direction of the flight is also important an additional feature was added that represents the directed flight. All categorical features are represented using the one-hot encoding representation. Although some learning algorithms can directly deal with categorical non numeric features, one hot encoding or any label encoder helps to make the regressor invariant to the representation of the features. Last but not least, adding redundant features to learning algorithms could be harmful since they increase the dimensionality and they bias the model towards specific patterns. So the Pearson correlation was calculated between the features and the very highly correlated ones were excluded e.g std_wtd. As a last step we considered expanding the feature space by calculating the product between features but that didn't improve the score and increased the computational complexity significantly so it was not used.

3 Algorithms

3.1 Linear Regression

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. A linear regression line has an equation of the form $Y = a + bX$. For this project a simple linear model was implemented as well as the variants that implement regularization techniques

like Ridge and Lasso regression. The penalization and step parameters were tuned using a grid search and 3 fold cross validation for all variants.

3.2 Random Forests

The random forest algorithm is an ensemble method which uses many 'weak' classifiers and aggregates them together so as to make a more powerful one. The base classifier involved in Random Forests is the decision tree. A decision tree is built by cutting the space in different 'clusters' of elements that are as homogeneous as possible. The cut is made along one direction of the features space. The homogeneousness measurement can be made according to different criteria, such as the Gini diversity index, the Information Gain (based on Shannon 's entropy concept in information theory), etc.

The biggest advantage of random forests is that they avoid overfitting by bootstrapping a method similar with bagging. The main difference is that random forests use a modified tree learning algorithm that selects a random subset of the features at each candidate split in learning process. If p is numbers of features, we will use $\sqrt{p}$ features at each split. Random forest create B new trees with this bootstrapping method and train each tree imposing some limits to the maximum depth. All B trees predict the outcome for each original observation and at the end, the random forest prediction is only a simple majority vote. The downside of this method with respect to the original decision trees is that we lose the white box (visual aspect) of a unique decision tree.

For this project we exhaustively grid searched the number of estimators and the maximum depth of the tree as well as other more trivial (in regards to the effect on the result) parameters in order to get the best result.

3.3 Gradient Boosted trees

The gradient boosting algorithm like random forest, combines weak learners into a single strong learner, in an iterative fashion. It starts with a weak learner F_m and improves on it by constructing a new model that adds an estimator h to provide a better model $F_{m+1}(x) = F_m(x) + h(x)$. The estimator h is a function calculated by fitting h to the residual $y - F_m$ or in general the negative gradients of the squared error loss function.

For this project we exhaustively grid searched the number of estimators and the maximum depth of the tree as well as the learning rate and the subsampling option parameters in order to get the best result. The final model which is the one that gave us the best result overall used many but very shallow trees. Reducing the learning rate also boosted significantly the performance of the algorithm.

3.4 MLP

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f(\cdot) : R^m \rightarrow R^o$ by training on a dataset, where m is the num-

ber of dimensions for input and o is the the number of dimensions for output. Although it can work very well as a function approximator the MLP needs a lot of tuning and careful design. For this project we tried the MLP approach using the 0.18dev version of sklearn but the results were not on par with the other approaches neither in terms of effectiveness nor efficiency. The RMSE result was around 0.8 using a two hidden layer network of 100 neurons. This is by no means near the best setting for MLP and thus the result is not reported on the final comparative table. However it demonstrates the difficulty of implementation of feedforward neural nets, even when using a platform like sklearn which minimizes the programming and tuning effort of machine learning implementations.

3.5 Support Vector Machines

A support vector machine constructs a hyperplane or set of hyperplanes in a high dimensional space. The idea is to find the hyperplane that separates the data the better, in maximizing the distance between data points of the different classes, and then classify according to the part of the space where the test data point falls. In the same way as with classification approach there is motivation to seek and optimize the generalization bounds given for regression. When the data is not linearly separable, the idea is to use kernels, so as to 'project 'data to a higher dimensional space where the data is separable, by replacing the natural scalar product x . y by phi(x) . phi(y) = k(x,y) where k is a kernel operator (positive semi definite symmetric).

For this project we grid searched but not exhaustively the penalty of the error parameter C, the gamma parameter and tried the linear and rbf kernels.

4 Evaluation & Discussion

In this section we present the test results obtained from the algorithms previously presented and we also discuss them in order to extract useful conclusions

Algorithm	RMSE
Lasso	0.64
Ridge	0.54
SVR	0.80
Ridge-Per Route	0.41
RF	0.36
GB	0.33

During development we tried another approach as well: Instead of fitting directly the model using all features, a model per Route was trained and tested using the remaining features. Intuitively this could perform better and for the

linear regression and its variants it did, like the Ridge per Route which is reported on the table. In order to build a model for every route a Dictionary with the mappings of routes to models was kept consistently during the training and test phases. Nevertheless the model did not manage to outperform the ensemble methods trained with all features together. This along with the fact that many boosted trees are required in order to get the best result leads us to the conclusion that the model that best predicts the target function is highly complex and non-linear and simple linear models fail to capture it correctly. The reduction of the learning rate to 0.2 also led the gradient boosted method to a superior result although the combination of the subsampling and reduction of the learning rate didn't help. The gradient boosted approach resulted to 0.27, using the train and test sets of the submission site, which may be due to using larger training sets compared to the one that was provided offline.

5 Conclusion

To conclude in this project several approaches and feature combinations were tried in order to achieve the best result. Some methods and ideas could be better implemented or more thoroughly searched and this could lead to a significant improvement of the result. However I mostly tried to see different algorithms, their limits and their space and time complexity in practice. Furthermore the whole external data process was quite enlightening about parts of the job that data scientists have to manage besides the machine learning part. Gradient boosted trees gave us the best result and so we used it for the final submission. However the huge effort behind this project can by no means be summarized by a single number.