**Predicting funding for school projects**

Cristina Mac Gregor Vanegas

Machine Learning for Public Policy

Due: May 21, 2018

The present analysis uses a dataset available in Kaggle that describes weather a project will get fully funded or not.[[1]](#footnote-1) The two tables used (*outcome* and *project*) from the database correspond to information about projects proposed by professors in school requesting materials to enhance the education of their students. We restrict the information used to observations that got funded between the start of 2011 and the end of 2014. Our objective is to be able to predict whether a project will get fully funded or not, which is captured by a Boolean variable which by definition can only take two values: True or False. [[2]](#footnote-2)

In order to understand which models (and with which features), help us predict this variable optimally, we conduct an analysis with several parts. First we do an exploratory analysis of the variables available in the dataset. Then we split our data into training and testing sets, clean variables, and build features on each set. Lastly, we perform iterations in order to try out every model that can be built with the parameters specified in the small\_grid,[[3]](#footnote-3) and for each we calculate: precision, recall, accuracy, specificity, f1, and the area under the precision-recall curve.

On a first stage, I use auxiliary functions to perform a very simple exploratory analysis. Given that the dependent variable is a Boolean, I rely mostly on grouped by descriptions of the potential variables to be used as features than on scatter plots, and most importantly on the Spearman coefficient of correlation’s table.[[4]](#footnote-4) With this table, we see that variables that are more correlated with being fully funded are:

* Total price including optional support
* Charter school Boolean
* Proportion of great messages
* Amount of teachers that referred the project
* Amount of non teachers that referred the project
* One non teacher-acquired donor gave more than $100 (Bool)
* Donation from thoughtful donor.[[5]](#footnote-5)

The high correlation means, in other words, that they seem to provide information about the probability of getting fully funded (or at least more information relative to the other variables available). A more precise analysis through regressions and feature importance analysis through Random Forests can be done but in order to test that the loop constructed is working for testing all the models, I start by working with these specified variables.

After deciding which variables we would like to include in the models that we will build, I chose a time split for the data based on the amount of variables that we have corresponding to each time. On a first attempt to try all the models with different parameters, I aimed to have 80% of the observations on a training set, and 20% on the test set as an initial split.[[6]](#footnote-6) I run a loop that, given a range of parameters, computes all the possible combinations for each of the models. Moreover, besides evaluating the models, we evaluate the prediction made by a naïve model as a baseline, to compare our best resulting models against it. The models that I test are a Random Forest Classifier, Logistic Regression, Gradient Boosting Classifier, a Decision Tree Classifier, a and a KNN Classifier.

I am evaluating the following scores:

* Precision - Corresponds to the accuracy over the cases predicted to be positive. A high precision means that the proportion of cases that are actually true is high compared to those that have been predicted to be true, which in other words means that we are making correct decisions when classifying something as positive.
* Recall – true positive rate corresponds to being correct with our prediction, when the true value of the instance is positive
* F1 – The harmonic mean of precision and recall measures.
* AUC PR – is the area under the curve generated by testing recall and precision under different thresholds, and gives us a comprehensive idea of, overall, and regardless of the threshold used. A higher AUC corresponds to a better model.

We calculate each at different levels of probability, which means that when we make a prediction about a case, we decide if we are going to classify its value (that is between 0 and 1, where 1 is more likely to be true) as true or false, according to a threshold. The higher the threshold, the stricter we are being in allowing something to be classified as true, which implies a high precision, but usually corresponds to a lower measure of recall, because we might classify as false instances that in reality are true. In this case I care about both cases. Instead of using the *fully\_funded* variable as our target variable, we instead use *not\_fully\_funded*, meaning that we will assign a value of true to those that we predict won’t be funded. With this, we can use precision and recall metrics more intuitively.

Precision and recall are terribly important for the prediction we want to make. We want to be sure that we are allocating the resources we have to the people who need it the most, hence precision ensures that we are doing so. Recall ensures that we are targeting all of those who need the help. Given that we care about both metrics, and that both change as a result of the threshold that we choose, F1 metric and the area under the precision-recall curves are two variables that allow us to evaluate the overall efficiency of the model.

For an initial stage of the analysis, I restrict the frame in order to use only observations from 2012, due to time restrictions for testing the whole grid. Moreover, I try three different time splits, keeping test sets of .5, .6 and .7, respectively. For this cases, we find these results:

*Overall performance: Precision vs. Recall*

We start by analyzing the results for annex model\_2012-07-31.csv, which is a file done after a 50% cutoff (aiming to get 50% of the data in the training set and 50% in the test set).

*Decision Trees, Random Forests, Gradient Boosting and Linear regression.*

We find that in general, the numbers for each model regardless of the parameters specified are pretty consistent at any threshold for all precision, recall and f1m with very small changes along the results. This happens in particular with all the different 36 models we constructed for Decision Trees, the 10 for Logarithmic Regression, and 16 for Random Forests, where we can only see changes between models at the centesimal level for precision and recall.

As for Gradient Booster, we do see a big difference in performance when we minimize the learning rate given any subsample and maximum depth specified. We see that as we decrease this parameter from .5 to .1 or .001, our measurement for precision takes a hit at any given threshold level, while recall increases. Intuitively, this is because Gradient Boosting, after classifying all the observations, it attempts to re-classify by giving more weight to the ones that have been miss-classified, in order to make a more accurate prediction for them. The learning rate parameter indicates how much to shrink the contribution of each additional model created. When we shrink this parameter, we see that recall increases, meaning we make less mistakes in classifying those that are actually positive (which is our main objective), but we make less precise predictions with a low threshold.[[7]](#footnote-7) Moreover, it is expected that Gradient Boosting performs very good since its recursive approach of re-applying the model to better predict, results in a weighted average with better predictions. As we’ve covered in class, Gradient Boosting always performs better than doing one simple model.

*KNN*

With KNN it is worth noting that the time it takes to perform is very high, and that this time grows as the number of KNN calculated increases. Moreover, we see a homogeneity in the results for each threshold, with the number of neighbors being the most determinant parameter for our data, noting that precision is highest when the number of KNN is lower. However, here we notice again the trade-off between precision and recall given that recall increases as we increase KNN. The harmonic mean of both is highest with less neighbors, but our computation returns an area under the curve of 0. By analyzing the other models ran for KNN, we do see that area under the curve is greater when we use more neighbors in our analysis, so if we don’t aim at maximizing precision at a given percent, these specifications seem better. In particular, it is maximized with 50 nearest neighbors, a uniform type of weighting and a kd\_tree algorithm.

Of all the models conducted, the one that maximizes the area under the curve is Gradient Boosting, even though, with those parameters (learning rate = .001) it is the one that gives us the lower precision at the top percent. This underlines a big trade off decision to be made, about whether we care about maximizing the area under the curve estimation, or we care about the performance at a given threshold. An important factor to decide how to decide which model to use is to take into consideration is how much resources can be invested in helping the projects. If little resources are available, we might want to use precision at small percentages as the important measure evaluating the model that is more adequate for us, and thus use a Gradient Booster with a learning rate of 0.5. If we not only care about those that we can target at a specific level, but we would like to target more, and make sure to be reaching out to everyone that needs the help (which means not misclassifying as *not in need* those who are), we should decide better with an AUC performance.

*Time horizon*

As we increase the amount of data that we include in our training model and decrease the one we have in our training set, we see trends in the results we obtain. When we move from moving from 50% to 60% training data, we find that:

* We see an initial increase in the AUC estimations for most models, that is less clear when we move from 60 to 70.
* We see an upward trend for precision but that does not hold for all estimations.
* We don’t really see a particular trend for recall nor f1.
* We do see that, if we rank models for performance according to their precision at the top level, the way we order them changes:
  + For the first split, we found that after GB, Random Forests and Logistic Regression performed better, but on the second split, we no longer find Logistic Regression here, and on the third one we see that Decision Trees also make their way to the top as opposed to the other splits.
  + We know that Decision Trees are volatile, so small changes in the data can greatly change the predictions made and thus the metrics estimated.
  + A closer look shows that the precision at the top percent for these models is considerably similar, which can explain the changes in the rankings.

1. https://www.kaggle.com/c/kdd-cup-2014-predicting-excitement-at-donors-choose/data [↑](#footnote-ref-1)
2. Since our target is identifying who *doesn’t* get funding, we could have a dependent variable tagged as **no funding*,*** which we could as dependent variable. For this analysis, I predict those that get funding, which in terms of the interpretation of results means we want to intervene in those predicted as negative. [↑](#footnote-ref-2)
3. Taken from Rayid Ghani’s magic loop available at: https://github.com/rayidghani/magicloops/blob/master/magicloop.py [↑](#footnote-ref-3)
4. A first cleaning step that needed to be done at this stage was converting Boolean values into 0-1 integers instead of the strings ‘f’ and ‘t’ that we find as a default on both tables. [↑](#footnote-ref-4)
5. A closer look shows that using some of these variables is not desirable, because some of them seem to be directly related to the result variable (*fully funded)*, or to have been collected at the same time, and it would yield unpractical results for real-life predictions of the outcome variable. Given limited time to run the complete loop and taking these variables out, we are maintaining them for this exercise. [↑](#footnote-ref-5)
6. After doing this split, I perform transformation in order to transform the variables I chose as predictors as dummy features. Continuous variables are discretized based on quartiles, and then dummy variables are made for pertaining to each quartile; Categorical variables are also converted into dummy variables, one for each possible category. The test is transformed following the exact transformations that were performed on the training data. [↑](#footnote-ref-6)
7. Another curious observation is that we also can appreciate that there are no changes for the gradient boosting estimation given a learning rate, regardless of how we change the other parameters [↑](#footnote-ref-7)