Predicting projects on donorschoose that will not get fully funded within 60 days of posted

by Felipe Alamos

Problem

This project consists in developing machine learning models to predict if a project on donorschoose will not get fully funded within 60 days of posting.

Data used

The data used, published by donorschoose.org, is a list of projects and their characteristics. Each row is a project, and columns contain a variety of information related to the project (id, school, location, subject, students reached, etc), as well as the date of posted and date of funded. This last two columns are used to calculate how long it took for a project to get funded, and ultimately determine if that time was longer or not than 60 days. The data spans from Jan 1, 2012 to Dec 31, 2013.

Features generated

We generate a total of 109 features to make our predictions. Many of them are binary features generated from categorical data. Some examples are the state, whether the school of the project is urban or rural, the focus of the project, the poverty level, the grade level.
We also include some numerical features such as number of students reached with the project, and the total price of the project. Discrete values of these numerical features are included, to take into account the existence of significant outliers in these variables.

An aggregated feature is also considered. We calculate the amount of projects that has been founded 10 days before the day a project was posted. We believe this could be a good predictor of the possibility of a project not being funded in the next 60 days (it could be the case that if many projects have been funded lately, then the chances of our project not being funded soon are low).

Lastly, the month and year when the project was posted is included as feature. It might be the case that these variables also help to predict and have some correlation with the possibility of being funded (maybe some resources are more liberated in certain periods of time over others).

Test/train used

We generate 3 different validation/test sets, considering a rolling window of 6 months. This means that we have the following 3 train test sets:

- Train from 1st January 2012 to June 30th 2013, test from July 1st 2013 to December 31 2013
- Train from 1st January 2012 to December 31th 2012, test from January 1st 2013 to June 30th 2013
- Train from 1st January 2012 to June 30th 2012, test from July 1st 2012 to December 31st 2012

Models and metrics chosen

We use the following classification methods to make predictions: Decision Trees, Logistic Regression, Random Forest, Extra Trees, K Nearest Neighbors, Naive Bayes, Bagging, Ada Boosting, Gradient Boosting. We use a variety of different parameters for each of them.

Metrics considered are precision, recall and f1 at 1,2,5,10,20,30 and 50%. We also consider area under the roc curve as metric. As baseline we consider precision calculated at 100%.

Performance of different models and recommendations

Choosing the best model depends on the metrics that are more interesting to the audience. This decision will depend on the policy goal and context. Some models are better at precision, while others at auc-roc, for example. In addition, some models behave great for certain train/test sets but not for all, while other models may not show excellent results in any train/test set, but perform relatively good for all of the different train/test sets. A more conservative audience could prefer a model that behaves good throughout all different train/test splits. A risk taker audience that believes that the test set that is closer to the present time is the most reliable one, would rather choose a model that performs the best under that closest test set, regardless of the performance in other train/test sets.

We now present the best models, for the different train/test sets, and different metrics. We also present a plot of how the different models perform over time.

Best models for precision at 5%

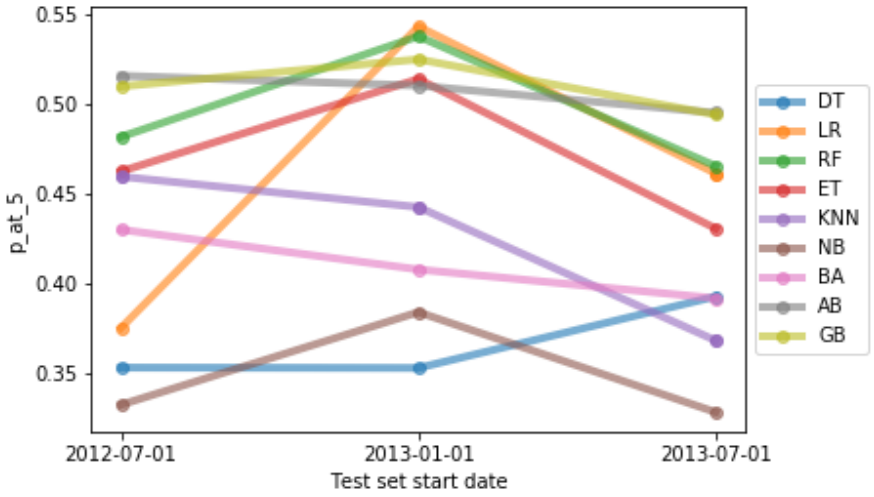
If someone wants to identify 5% of posted projects that are at highest risk of not getting fully funded to intervene with, he/she should focus on the precision at 5% metric. According to this metric, an AdaBoost model, with parameters 100 estimators and SAMME algorithm, performs the best for the test set starting in July 1, 2013. Its precision at 5% would be 0.49, this means that almost half of the predictions of projects that will not be funded in 60 days would be correct. We can also observe that this model performs relatively well throughout the different test sets starting points, so it's a reliable model.

We can also observe in the plot that a GradientBoosting model also performs relatively well throughout all test sets, with precision at 5% very close to AdaBoost. Random Forest model also performs relatively well, but presents a little bit more variation between test sets. Interestingly, Logistic Regression performs very well for the January 1st 2013 test set, but not for the others (particularly bad for the July 1st 2012 test set).

Best models according to precision at 5% for the different test sets

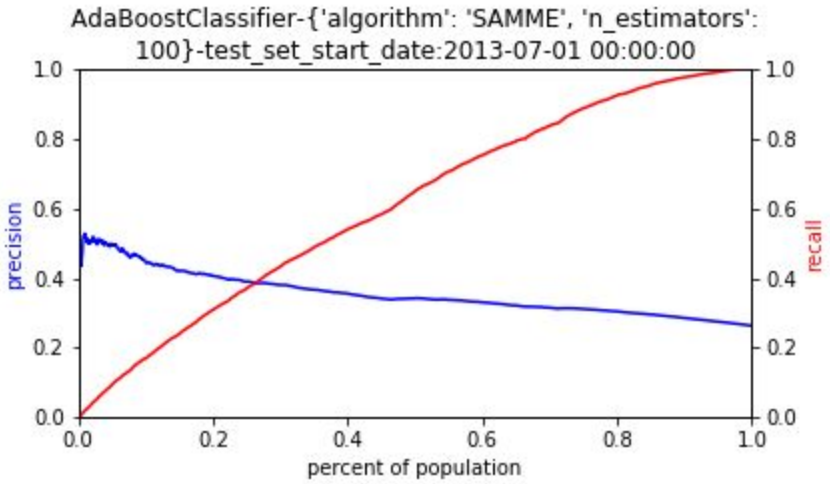
model_name	parameters	test_set_start_date	baseline	p_at_5	r_at_5	f1_at_5
AB	{'algorithm': 'SAMME', 'n_estimators': 100}	2013-07-01	0.263209	0.494888	0.093962	0.157938
LR	{'penalty': 'l1', 'C': 0.1}	2013-01-01	0.295413	0.542962	0.091865	0.157143
AB	{'algorithm': 'SAMME.R', 'n_estimators': 100}	2012-07-01	0.257396	0.515510	0.100029	0.167547

Models performance on precision at 5% through time



For the selected and recommended model (AdaBoost) we present the precision-recall curve. This curve tells us, for a given percent of population we want to intervene, what the precision is (of all the projects we predicted not to be funded in 60 days, what fraction was effectively not funded), and what the recall is (of all projects that were effectively not funded in 60 days, what fraction our model was able to correctly identify). We can observe that precision is relatively stable throughout the different percentages of population intervened. It's over 0.4 when intervening over the top 20%, but then it keeps relatively stable (decreasing moderately from approximately 0.4 to 0.25) when increasing the percent of population intervened.

Precision recall curve for AdaBoost model chosen



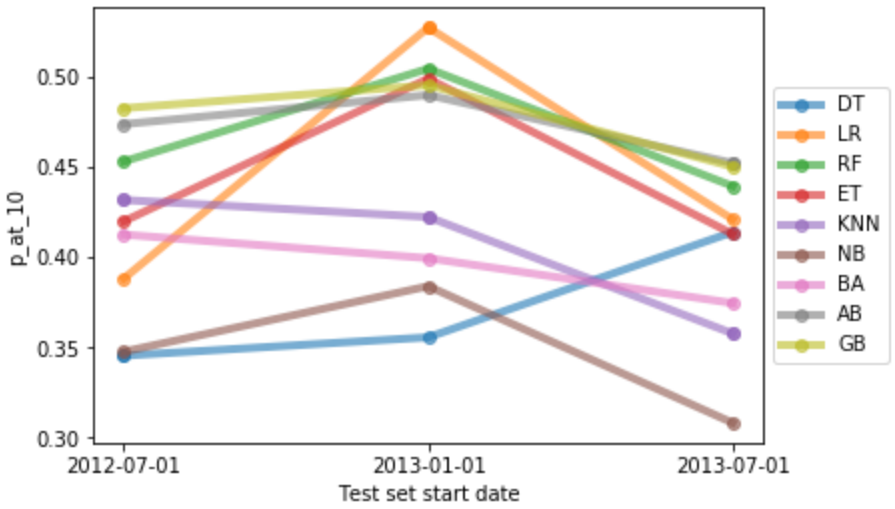
Best models for precision at 10%

A very similar pattern occurs for precision at 10% metric. AdaBoost is also the best model for the July 1st, 2013 test set, followed by GradientBoosting. Both perform relatively stable throughout time. Random Forest also performs relatively good.

Best models according to precision at 10% for the different test sets

model_name	parameters	test_set_start_date	baseline	p_at_10	r_at_10	f1_at_10
AB	{'algorithm': 'SAMME.R', 'n_estimators': 10}	2013-07-01	0.263209	0.452223	0.171811	0.249015
AB	{'algorithm': 'SAMME.R', 'n_estimators': 100}	2013-07-01	0.263209	0.452223	0.171811	0.249015
LR	{'penalty': 'l1', 'C': 1}	2013-01-01	0.295413	0.527422	0.178472	0.266697
LR	{'penalty': 'l1', 'C': 10}	2013-01-01	0.295413	0.527422	0.178472	0.266697
GB	{'subsample': 1.0, 'n_estimators': 100, 'learn...	2012-07-01	0.257396	0.482657	0.187446	0.270025

Models performance on precision at 10% through time



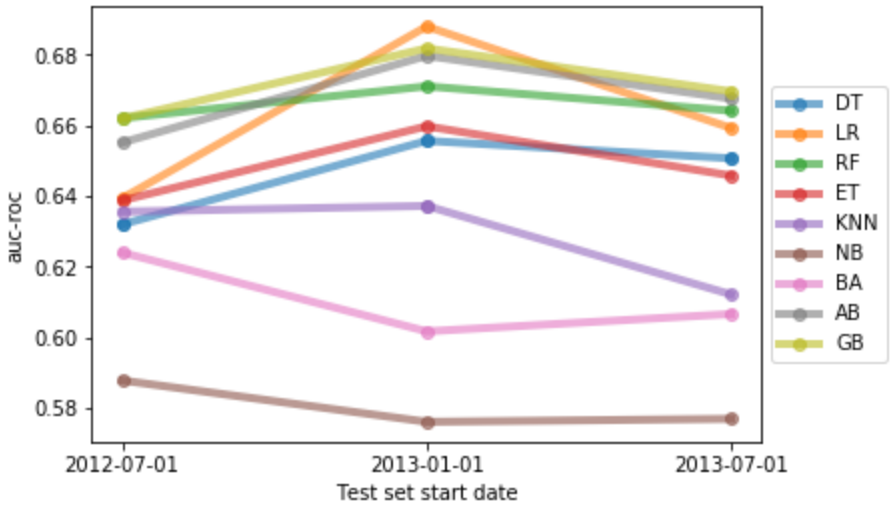
Best models for precision at auc-roc

Considering the area under de roc curve, the GradientBoosting model, with parameters {'subsample': 1.0, 'n_estimators': 100, 'learning_rate': 0.1}, is the best one for the July 1st 2013 test set, with a value of approximately 0.7. It is followed by the AdaBoosting model. This metric, and hence the GradientBoosting model, should be chosen under a scenario when the percentage of population for intervention is not defined/unclear.

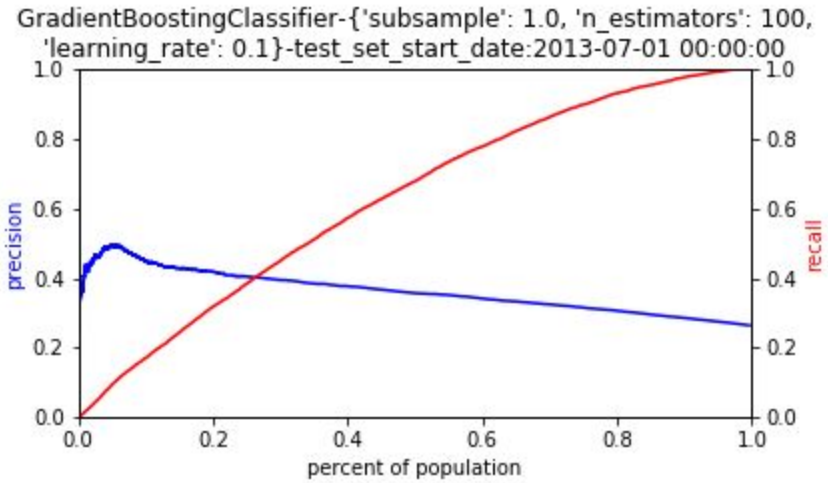
Best models according to auc-roc for the different test sets

model_name	parameters	test_set_start_date	baseline	auc-roc
GB	{'subsample': 1.0, 'n_estimators': 100, 'learn...	2013-07-01	0.263209	0.669584
LR	{'penalty': 'l2', 'C': 0.1}	2013-01-01	0.295413	0.687961
RF	{'max_features': 'sqrt', 'n_estimators': 100, ...	2012-07-01	0.257396	0.661983

Models performance on auc-roc through time



Precision recall curve for GradientBoosting model that performs the best according to auc-roc



Conclusion

If someone wants to identify 5% of posted projects that are at highest risk of not getting fully funded, we recommend using an AdaBoost model, with parameters 100 estimators and SAMME algorithm. For this 5%, the models predictions of which projects will not get funded will be correct for half of them. If increasing the percentage of population intervened, the performance of the model will not change significantly. The model is also a reliable in the sense that its performance does not change too much throughout time.