report

May 30, 2019

1 Report of prediction of projects not fully funded

Dear Donnor org:

In response to your petition to help you decide which schools to include in your pilot training program, we send you this report. We tries to create a model that can help you predict which models have a low probability of being fully funded in 60 days. To did this, we estimated a group a classification models that we will briefly describe. The report includes a brief description of the methodology, followed by the results we achieved and concludes with a recomendation of the next steps to follow.

1.1 Methodology

Using the 124,976 projects you send us, we saw that 29% of them did not achieve complete funding within 60 days of being posted. Using all the information about each project, we built seven different classification models that identify the most important characteristics of those projects that were not completelly funded. Each of the model has a different approach, but all have in common that the algorithm, ant not the researcher, identifies the most important variables. In particullar, we fitted the following models: K-nearest neighbors, decision tree, linear regression, support vector machines, random forest, gradient boosting and bagging.

We tried to simmulate the decision process you will face when predicting the projects that need the most help. This is, you will have information about the projects the moment they are posted, and you will have to decide if intervene on them based on their probability of being fully funded. For this reason, we divided the two years of information that you gave us in four semesters. Our models were evaluated three times, one for the second semester, using information from the first semester to build the model, one for the third semester, using information from the first and second semester, and one for the fourth semester, using all the previous information.

1.2 Results

To evaluate our models, we compared the projects that we predicted were not going to be fully funded against their real outcome. We built three meassures: Precision, equivalent to the percentage of projects that were in fact not fully funded, out of the total we predicted. Recall: percentage of projects that we correctly predicted that were not going to achieve the objective, out all the projects that did not make it. AUC ROC: Overall meassures that lets us compare how our capable are our models of deciding which projects will fail with a complete random meassure. Of this three meassures, we consider precision to be the most important because you want to maximize out of the 5% of projects that you will help, the number of project that will actually need it.

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

        prediction_grid = pd.read_csv('results_complete.csv')
```

1.3 Results in average for models

As the following table presents, in average the best model we estimated was random forestsin terms of precision. The average precision of this methodology was 51%, meaning that our of those projects predicted to not achieve the goal in 60 days, 51% actually did not. Following logistic regression had a precision of 50%, and in last place decision trees with no precision.

In terms of recall, the distribution is the same. The reason is that we are fixing the percentage of projects on 5%, so then maximizing precision is equivalent to maximizing recall. In terms of the AUC meassure, the best model was also random forests, with an score of 0.67. This score reflects that the models were actually capable of make some classification, but it is not very much greater to the 0.5 AUC that a complete random model would have.

```
In [5]: prediction_grid[prediction_grid['top_k'] == 0.05].groupby('model').agg({'precision': np
                                                                               'recall': np.me
                                                                               'AUC ROC': np.m
            .sort_values('precision', ascending = False)
```

```
Out[5]:                precision    recall   AUC ROC
        model
        random_forest   0.511016  0.089838  0.676480
        logistic_reg    0.501901  0.088252  0.670266
        svm             0.488009  0.085823  0.652777
        gradient_boost  0.448366  0.078571  0.647269
        KNN             0.425126  0.052143  0.595894
        bagging         0.424903  0.073806  0.628656
        decision_tree   0.000000  0.000000  0.562048
```

1.4 Results disagregated by specification

As the following table shows, one doest not see great variations in the results of random forests accross the different parameters specified. Wheather we use Gini or entropy as criterion to split, or we set the maximum depth of our trees to 15 or 10, or we vary the number of trees between 80 and 150, the precision at 5% remains around 0.55

```
In [32]: pd.set_option('max_colwidth', 800)
         prediction_grid.loc[prediction_grid['top_k'] == 0.05, ['model', 'parameters', 'precisi
             .sort_values('precision', ascending = False).head(10)
```

```
Out[32]:             model  \
         706  random_forest
         716  random_forest
         711  random_forest
         806  random_forest
```

```
801  random_forest
326            svm
296   logistic_reg
331            svm
686  random_forest
301   logistic_reg

                                                                      parameters  \
706   {'criterion': 'entropy', 'max_depth': 15, 'n_estimators': 80, 'seed': 1234}
716  {'criterion': 'entropy', 'max_depth': 15, 'n_estimators': 150, 'seed': 1234}
711  {'criterion': 'entropy', 'max_depth': 15, 'n_estimators': 100, 'seed': 1234}
806     {'criterion': 'gini', 'max_depth': 10, 'n_estimators': 150, 'seed': 1234}
801     {'criterion': 'gini', 'max_depth': 10, 'n_estimators': 100, 'seed': 1234}
326                                                   {'C': 0.01, 'seed': 1234}
296               {'C': 1, 'penalty': 'l2', 'fit_intercept': True, 'seed': 1234}
331                                                    {'C': 0.1, 'seed': 1234}
686     {'criterion': 'gini', 'max_depth': 15, 'n_estimators': 150, 'seed': 1234}
301              {'C': 1, 'penalty': 'l2', 'fit_intercept': False, 'seed': 1234}

     precision
706   0.559843
716   0.556693
711   0.556693
806   0.553943
801   0.552681
326   0.551181
296   0.550394
331   0.550394
686   0.550394
301   0.549606
```

1.5 Results disagregated by cross validation

We trained and tested each of our models three times. The first time assuming we only had information of the first semester of 2012, the second time information of the whole 2012, and the thirs time with information for 2012 and the first semester of 2013. The results show that the accuracy of our models increased from the first to the secodn training but decreased again for the thirs training. Random forests was consistently the mos accurate model at 5%, with a very small exception for the first training.

In [34]:
```
prediction_grid.groupby(['cross_k', 'model']).agg({'precision': np.mean, 'recall': np
```

Out[34]:
```
                        precision    recall   AUC ROC
cross_k model
3       random_forest    0.498621  0.218542  0.692079
        logistic_reg     0.486230  0.214398  0.683370
        svm              0.475416  0.209788  0.671590
        gradient_boost   0.444144  0.203073  0.664213
```

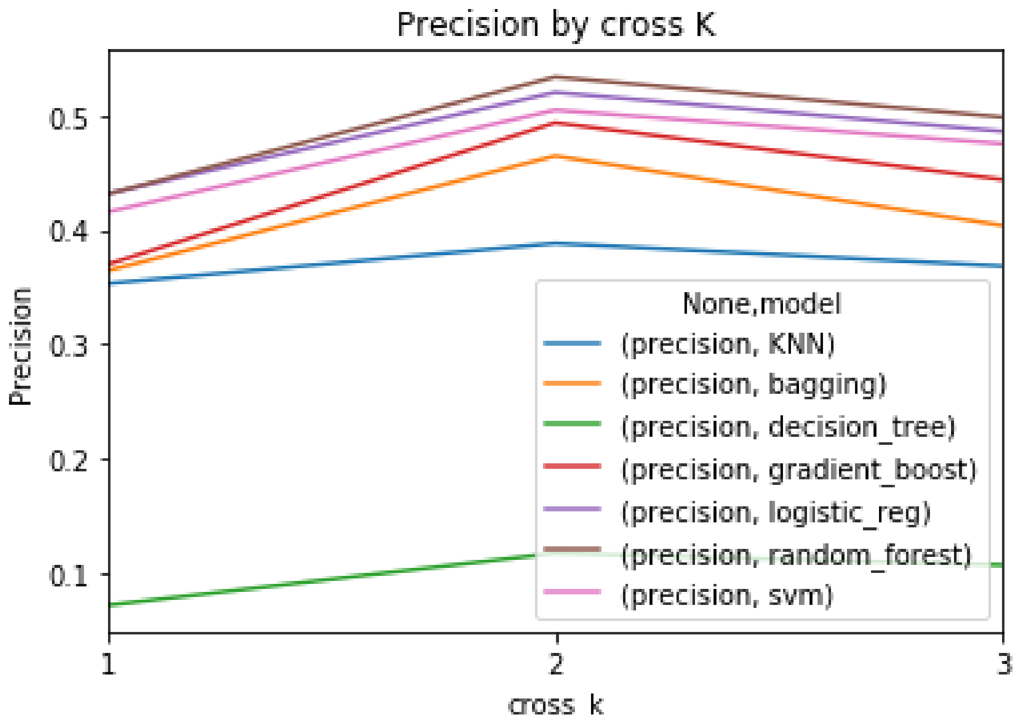
	bagging	0.404086	0.186534	0.634322
	KNN	0.368543	0.149503	0.605172
	decision_tree	0.107586	0.097085	0.569208
2	random_forest	0.533716	0.205788	0.675435
	logistic_reg	0.520248	0.204145	0.671292
	svm	0.504736	0.197036	0.648494
	gradient_boost	0.493661	0.196193	0.651702
	bagging	0.464664	0.186298	0.632339
	KNN	0.388367	0.141621	0.594299
	decision_tree	0.117822	0.095795	0.563289
1	logistic_reg	0.431884	0.205487	0.656135
	random_forest	0.431628	0.206438	0.661927
	svm	0.415959	0.198527	0.638246
	gradient_boost	0.370237	0.183895	0.625893
	bagging	0.364673	0.181204	0.619306
	KNN	0.353277	0.137219	0.588211
	decision_tree	0.072440	0.074016	0.553647

The following is an example of the precision scores of each of the models for each of the thresholds analyzed.

1.6 Graph by cross_k

The folowing graph shows how the porformance of the models vary by each cross validation. Is demonstrates the increase from the first to the second train and how it decreases again.

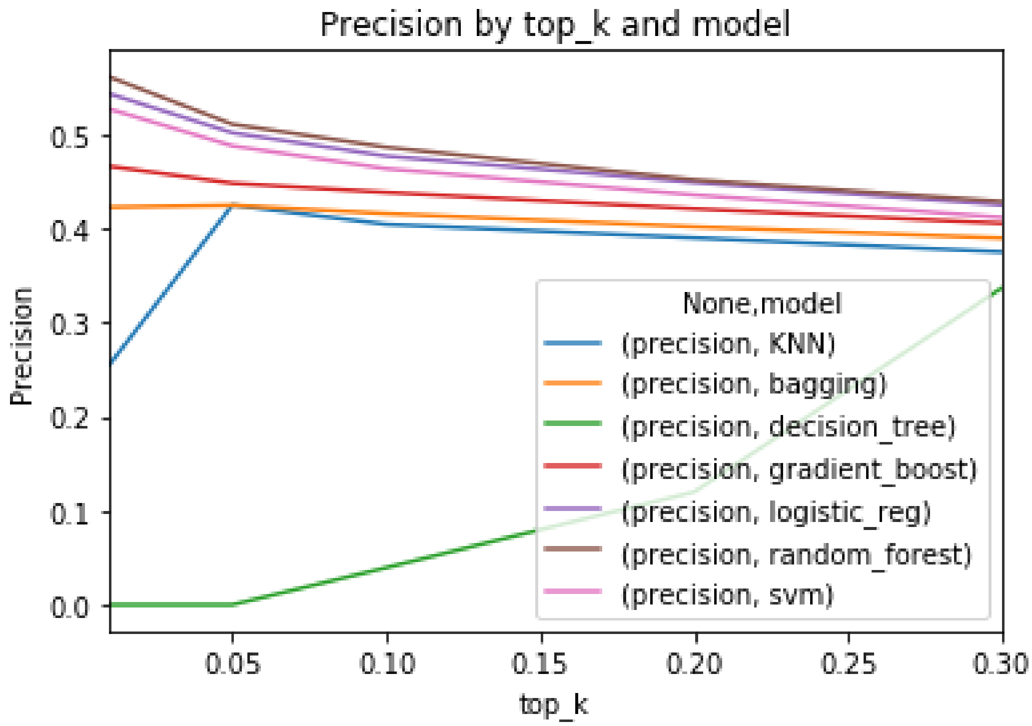
```
In [13]: prediction_grid.groupby(['cross_k', 'model']).agg({'precision': np.mean}).unstack().pl
         plt.title('Precision by cross K')
         plt.ylabel('Precision')
         plt.xticks([1, 2, 3])
         plt.show()
```



1.7 Graph by top-k

As the following graph shows, Random Forests are consistently the best models for the different percentages of population, so we can be sure it was not an excemption

```
In [10]: prediction_grid.groupby(['top_k', 'model']).agg({'precision': np.mean}).unstack().plo
         plt.title('Precision by top_k and model')
         plt.ylabel('Precision')
         plt.show()
```



1.8 Recommendation

We recommend to use the Random Forests model to identify the 5% of the projects that most probably wont be fully funded in 60 days.