CAPP 30254: Machine Learning for Public Policy
Assignment 3
Rachel Ker

Predicting projects on DonorsChoose that will not be fully funded within 60 days of posting

This report uses data from Kaggle about projects on DonorsChoose from 2012 to 2013 to predict project that will not be fully funded within 60 days of posting. 360 classifier models were built to model this prediction in total, including k-nearest neighbors, decision trees, support vector machines, logistic regressions, and random forest, boosting and bagging classifiers. Table 1 in Appendix shows the different parameters used for the classifiers build.

Validation Methodology

These classifiers were built only on the training set data and then validated on the testing set. The data was split into 3 training and testing sets by time period with the number of observations in each set specified below:

	Training Set	Training set size	Testing Set	Testing set size
1	Jan 2012 - Jun 2012	26,386	Jul 2012 – Dec 2012	32,838
2	Jan 2012 – Dec 2012	59,224	Jan 2013 – Jun 2013	21,585
3	Jan 2012 – Jun 2013	80,809	Jun 2013 – Dec 2013	44,167

Variables Used

The following are the variables used to build the classifier models, before discretization and generation of dummy variables:

- State the school is in
- Whether the school is in urban, rural or suburban area
- Whether the school is a charter school
- Whether the school is a magnet
- Teacher's prefix (Ms, Mrs, Mr)
- Main subject for which project materials are intended
- Main subject area for which project materials are intended
- Secondary subject for which project materials are intended
- Secondary subject area for which project materials are intended
- Type of resource requested by a project
- School's poverty level (measured by percentage of free and reduced lunch)
- Grade level for which project materials are intended
- Project cost including optional tip
- Number of students impacted by a project if funded
- Whether a project was eligible for a 50% off offer by a corporate partner

In this analysis, missing values were naively replaced with their mode since the variables are categorical variables and the descriptive statistics of the other variables were not sufficient to give more information about the value that should be imputed. Since our training and testing data are Training set data were imputed with the mode of that of the training set only, while the testing set was imputed with the

mode of the entire dataset. The total number of missing observations for each variable is given in Table 2 in Appendix.

Model Performance

The metrics we evaluated the models on were precision and recall at 1%, 2%, 5%, 10%, 20%, 30%, 50% thresholds and the area under the curve. All models were also compared to the baseline metrics when the model simply just predicts all observations as positives. Results of the best models for each metric are presented in Table 3 in the Appendix.

Overall, the linear classifier models (i.e. logistic regression and support vector machines) and tree ensemble classifiers did well in most of the metrics across all three training sets. All of the best models for each metric also did better than the baseline.

The best models however changed over time. In the first training set, the logistic regression models achieved the highest scores for most of the metrics. In the second training, while logistic regression models continue to do well, random forest classifier did the best for precision and recall at lower thresholds of 10% and below, the support vector machine did the best at precision and recall at higher thresholds of 20% and 50%, while the extra trees ensemble model did the best at precision and recall at 30%. Logistic regression continued to be the classifier with the highest area under the curve. In the third training set, the gradient boosting model did the best for precision and recall at lower thresholds of 10% and below, random forest and extra trees classifier did better for precision and recall at the higher thresholds.

It is also worthy to note that the precision (proportion of actual projects that will not be fully funded in 60 days of those predicted to be not fully funded within 60 days) for the best model increased from training set 1 to 2 to 3, indicating that the models performing best in the later training sets improved much more than the logistic regression model with more training data. This can be seen evidently from the precision-recall threshold curves in Figures 1 and 2 below.

Figure 1: Precision-recall on threshold curves for Gradient Boosting Classifier (max_depth=5, learning_rate=0.01, n_estimators=100, subsample=0.1) over the 3 training set – (left) training set 1, (center) training set 2, (right) training set 3

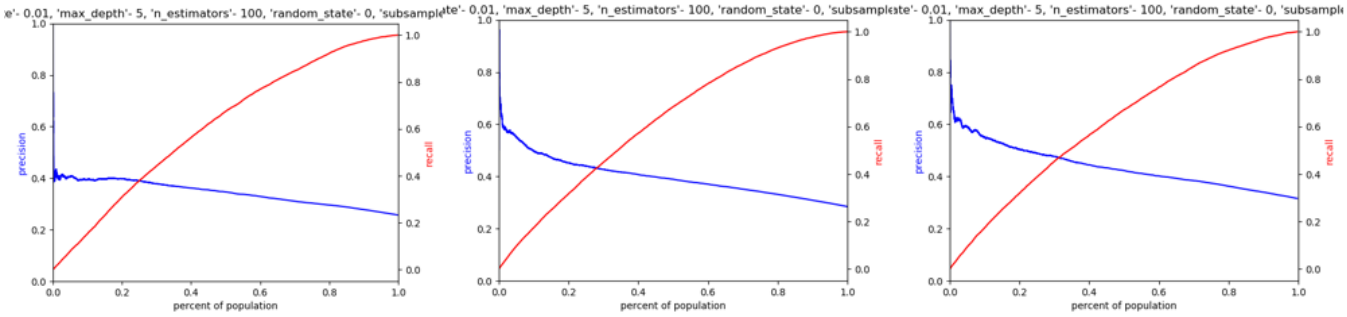
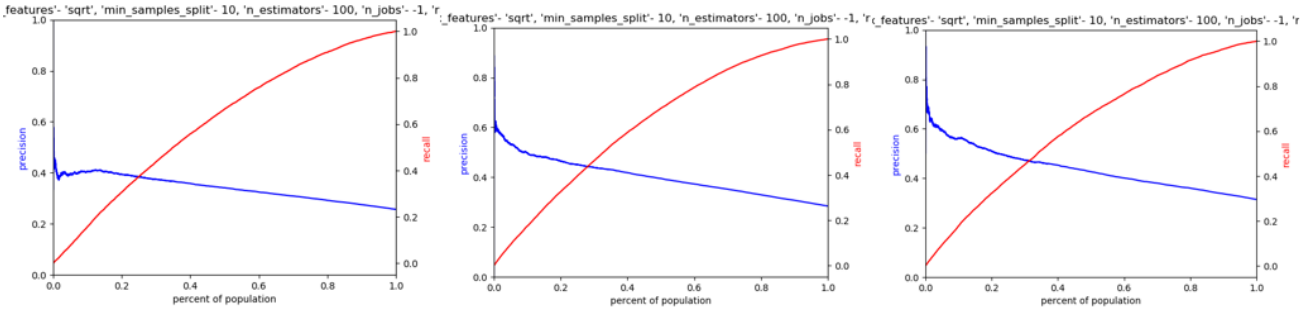


Figure 2: Precision-recall on threshold curves for Random Forest Classifier (max_depth=20, max_features='sqrt', min_samples_split=10, n_estimators=100) over the 3 training set – (left) training set 1, (center) training set 2, (right) training set 3



To recommend a model to identify 5% of the posted projects to intervene would depend on goal of the model. If we are looking to maximize recall or precision, I would recommend using either the **Random Forest (RF) Classifier**[1] or the **Gradient Boosting (GB) Classifier**[2] for deployment and field testing.

- Both the models have high precision of 59.4% (RF) and 55.3% (GB) at the threshold of 5%. This means that in the top 5% of the projects predicted to not be fully funded within 60 days of posting 59.4% (RF) and 55.3% (GB) of the projects are actually not funded within 60 days of posting.
- Both these models also have high recall of 9.4% (RF) and 9.7% (GB) at the threshold of 5%. This means that in the top 5% of projects predicted to not be fully funded within 60 days, 9.4% (RF) and 9.7% (GB) of the all projects that are actually did not get funded within 60 days were identified by this model.

The choice between the two models depends on whether precision or recall is of a higher priority in the goal. If the cost of expanding resources on a project that would actually be fully funded in 60 days but is predicted to not be fully funded is calculated to be higher than the cost of not detecting a project that would not be fully funded in 30 days (i.e. maximize for precision), the random forest classifier of the specified parameters should be chosen. If it is the reverse (cost of not detecting a project that would not be fully funded is higher so the goal is to maximize recall), then gradient boosting classifier of the specified parameters should be chosen.

However, the classifiers above are not consistently the best in precision and recall in all 3 training and testing set. Hence if we are looking for the most stable model with high recall and precision, I would instead recommend using the **Logistic Regression Classifier**[3] for deployment and field testing because this model with the specified parameters remains constantly in the top 15 of precision and recall for all 3 training and test sets.

- This model has a precision of 43-58% at the 5% threshold, meaning that in the top 5% of projects predicted to not be fully funded within 60 days, 43-58% of the projects are actually not funded within the 60 days.
- It also has a recall of 8.3-9.3% at the 5% threshold, meaning that 8.3-9.3% of the all of projects actually not funded within 60 days were identified among the top 5% of projects predicted to not be fully funded.

[1]Parameters: max_depth=20, max_features='sqrt', min_samples_split=10, n_estimators=100

[2]Parameters: max_depth=5, learning_rate=0.01, n_estimators=100, subsample=0.1

[3]Parameters: C=0.01, penalty='l2'

Appendix

Table 1: Parameters used for the classifiers

	K-nearest neighbors Classifier	Decision Tree Classifier	Support Vector Machine	Logistic Regression	Random Forest Classifier	Extra Trees Classifier	Gradient Boosting Classifier	AdaBoost Classifier	Bagging Classifier
n_neighbors	25, 50								
weights	uniform, distance								
algorithm	auto, ball_tree, kd_tree							SAMME, SAMME.R	
criterion		gini, entropy				gini, entropy			
max_depth		5, 20			5, 20	5, 20	5, 20		
max_features		None, sqrt, log2			sqrt, log2	sqrt, log2			
min_samples_split		2, 10			2, 10	2, 10			
c			0.01, 0.1, 1, 10	0.01, 0.1, 1, 10					
penalty				l1, l2					
n_estimators					10, 100	10, 100	10, 100	1, 10, 100	10, 100
n_jobs					-1	-1			-1
learning_rate							0.001, 0.01		
subsample							0.1, 0.5		

Appendix

Table 2: Table of number of missing observations for each variable in the data

	index	missing count
0	projectid	0
1	teacher_acctid	0
2	schoolid	0
3	school_ncesid	9233
4	school_latitude	0
5	school_longitude	0
6	school_city	0
7	school_state	0
8	school_metro	15224
9	school_district	172
10	school_county	0
11	school_charter	0
12	school_magnet	0
13	teacher_prefix	0
14	primary_focus_subject	15
15	primary_focus_area	15
16	secondary_focus_subject	40556
17	secondary_focus_area	40556
18	resource_type	17
19	poverty_level	0
20	grade_level	3
21	total_price_including_optional_support	0
22	students_reached	59
23	eligible_double_your_impact_match	0
24	date_posted	0
25	datefullyfunded	0
26	60daysafterpost	0
27	notfullyfundedin60days	0

Appendix

Table 3: Best models by metric and train-test set

Train -Test Set	Metric	Best Model	Parameters	Score
1	Baseline			0.256928
	Precision at 1%	LR	{‘C’: 0.01, ‘penalty’: ‘l1’, , 'random_state': 0}	0.50304
	Precision at 2%	GB, LR	{'learning_rate': 0.001, 'max_depth': 5, 'n_estimators': 10, 'random_state': 0, 'subsample': 0.5}", {'C': 0.1, 'penalty': 'l2', , 'random_state': 0}	0.460366
	Precision at 5%	ET	{'criterion': 'entropy', 'max_depth': 5, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 10, 'n_jobs': -1, , 'random_state': 0}	0.431444
	Precision at 10%	LR	{‘C’: 0.01, ‘penalty’: ‘l2’, , 'random_state': 0}	0.414255
	Precision at 20%	RF	{'max_depth': 5, 'max_features': 'sqrt', 'min_samples_split': 2, 'n_estimators': 100, 'n_jobs': -1, , 'random_state': 0}	0.400944
	Precision at 30%	LR	{‘C’: 0.01, ‘penalty’: ‘l1’, , 'random_state': 0}	0.38514
	Precision at 50%	LR, LR	{'C': 0.1, 'penalty': 'l1', 'random_state': 0}, {'C': 1, 'penalty': 'l1', 'random_state': 0}	0.348011
	Recall at 1%	LR	{'C': 0.1, 'penalty': 'l1', 'random_state': 0}	0.019557
	Recall at 2%	GB, LR	{'learning_rate': 0.001, 'max_depth': 5, 'n_estimators': 10, 'random_state': 0, 'subsample': 0.5}, {'C': 0.1, 'penalty': 'l2', 'random_state': 0}	0.035795
	Recall at 5%	ET	{'criterion': 'entropy', 'max_depth': 5, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 10, 'n_jobs': -1, 'random_state': 0}	0.083916
	Recall at 10%	LR	{'C': 0.01, 'penalty': 'l2', 'random_state': 0}	0.161195
	Recall at 20%	RF	{'max_depth': 5, 'max_features': 'sqrt', 'min_samples_split': 2, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.312078
	Recall at 30%	LR	{‘C’: 0.01, ‘penalty’: ‘l1’, , 'random_state': 0}	0.447789
	Recall at 50%	LR, LR	{'C': 0.1, 'penalty': 'l1', 'random_state': 0}, {'C': 1, 'penalty': 'l1', 'random_state': 0}	0.677255
	AUC ROC	LR	{'C': 0.1, 'penalty': 'l1', 'random_state': 0}	0.66056

Appendix

2	Baseline			0.31941
	Precision at 1%	RF	{'max_depth': 20, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.688372
	Precision at 2%	RF	{'max_depth': 20, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.635731
	Precision at 5%	RF	{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.594069
	Precision at 10%	RF, RF, ET	{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}, {'criterion': 'gini', 'max_depth': 20, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.562095
	Precision at 20%	SVM	{'C': 0.01, 'random_state': 0}	0.512393
	Precision at 30%	ET	{'criterion': 'gini', 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.479228
	Precision at 50%	SVM	{'C': 0.01, 'random_state': 0}	0.42791
	Recall at 1%	RF	{'max_depth': 20, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.21771
	Recall at 2%	RF	{'max_depth': 20, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.040306
	Recall at 5%	RF	{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.094292
	Recall at 10%	RF, RF, ET	{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}, {'max_depth': 20, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}, {'criterion': 'gini', 'max_depth': 20, 'max_features': 'log2', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.178435
	Recall at 20%	SVM	{'C': 0.01, 'random_state': 0}	0.3259
	Recall at 30%	ET	{'criterion': 'gini', 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.456458
	Recall at 50%	SVM	{'C': 0.01, 'random_state': 0}	0.679317
	AUC ROC	LR	{'C': 0.1, 'penalty': 'l1', 'random_state': 0}	0.682162
3	Baseline			0.284647
	Precision at 1%	GB	{'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.5}	0.619048
	Precision at 2%	GB	{'learning_rate': 0.001, 'max_depth': 5, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.1}	0.612684

Appendix

	Precision at 5%	GB	{'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.1}	0.552536
	Precision at 10%	GB, GB	{'learning_rate': 0.001, 'max_depth': 20, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.1}, {'learning_rate': 0.01, 'max_depth': 20, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.1}	0.502038
	Precision at 20%	RF	{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.464735
	Precision at 30%	RF, ET	{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}, {'criterion': 'gini', 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.439094
	Precision at 50%	ET	{'criterion': 'gini', 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.3951
	Recall at 1%	GB	{'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.5}	0.021715
	Recall at 2%	GB	{'learning_rate': 0.001, 'max_depth': 5, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.1}	0.043032
	Recall at 5%	GB	{'learning_rate': 0.01, 'max_depth': 5, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.1}	0.097041
	Recall at 10%	GB, GB	{'learning_rate': 0.001, 'max_depth': 20, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.1}, {'learning_rate': 0.01, 'max_depth': 20, 'n_estimators': 100, 'random_state': 0, 'subsample': 0.1}	0.176344
	Recall at 20%	RF	{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.326519
	Recall at 30%	RF, ET	{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}, {'criterion': 'gini', 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.462774
	Recall at 50%	ET	{'criterion': 'gini', 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.694003
	AUC ROC	RF	{'max_depth': 20, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 100, 'n_jobs': -1, 'random_state': 0}	0.683204