KDD Cup 2014: Predicting Excitement at DonorChoose.org

HARRIS, Jay (20362789)
LIU, Tao Marvin (20362959)

1 Summary

Our final model is an ensemble of two models: A gradient boosting machine using extra trees for feature selection and a linear SVM trained with stochastic gradient descent. We then used a weighted average of the predictions in order to combine the two models. The model makes use of several extra features we extract in a preprocessing stage.

2 Feature Engineering

We used a combination of the categorical features of projects.csv and historical donation features derived from donations.csv for our first model, and the project essays from essays.csv for our second model.

2.1 Project Categorical Features

We binarized the 25 categorical features from projects.csv by replacing each value with the frequency that the value occurs in the column. Originally, we used assign each instance of a categorical feature a number, but this was not desirable because it implied an ordering of the different categories. After looking into several options, including one-hot encoding, and exploring the Kaggle forums, we decided to go with the count frequency encoding as it was used in several benchmark scripts and mentioned in several posts.

2.2 Project Historical Features

We made use of several historical features which extracted by merging the projects and donations csv files together. This allowed us to extract information about teachers, schools, grades and subjects in relation to the amount of funding projects related to them have had in the past.

The historical features used in our model are as follows

1. Total donation by teacher
2. Number of donations by teacher
3. Average donation by teacher
4. Number of exciting projects supported per teacher
5. Total donations by school
6. Number of donations by school
7. Average donation by school
8. Number of exciting projects per school
9. Total donations by subject
10. Number of donations by subject
11. Average donation by subject
12. Number of exciting projects per subject
13. Total donation by grade level
14. Number of donations by grade level
15. Average donation by grade level
16. Number of exciting projects per grade

We decided on these features because we felt there was likely to be a correlation between these features that might not be readily apparent to our model. For example, a particular teacher might be very good at picking out exciting projects, or bringing in funding to a project, while a particular school might be in a richer area, bringing in a far greater number of donors. Similarly, it was noted that particular subjects and grade levels might be more likely to be exciting (or to receive funding).

2.3 Text Features

We added the word counts of the titles, short descriptions, need statements, and essays as features in our first model, as we felt that there may be some sort of correlation between those numbers and the outcome.
For the second model, we generated a TF-IDF (term frequency – inverse document frequency) matrix for the essays using the TfidfVectorizer module of scikit-learn. We used a list of common English stop-words to filter out words that were likely to be uncorrelated to the outcome, and considered both unigrams and bigrams in order to capture some detail about the ordering of the words. Using a TF-IDF matrix allowed us to find which words were more likely to be important (having a high term frequency in its own document and low term frequency across all the documents) and train a model based on this matrix.

3 Models

For this project, our model has gone through three iterations. A simple model, a text model and an ensemble model.

3.1 Gradient Boosting Machine+ Extra Trees

This model was our first attempt at making predictions on the data set and did not make use of any feature engineering. The model simply encoded all the features found in the projects.csv file and fed them into a gradient boosting machine. We used 100 decision trees, each with a maximum depth of 7, in our classifier. We chose to use a gradient boosting machine because of their simplicity and the fact that it is well-known to be a effective off-the-shelf algorithm used in many Kaggle competitions.
The performance of this model was negatively impacted when we added all of the historical features and word counts we extracted. We figured that this was due to the fact that some of our extra features were superfluous and just making our model more complicated than necessary. To remedy this, we used scikit-learn's ExtraTreeClassifier to compute feature importances and discard features that were unimportant.

After making use of the extra tree classifier, this model was vastly improved, with the predictions becoming more accurate on the validation set. Upon uploading this model's predictions to Kaggle, it scored an accuracy 0.57613.

3.2 Linear SVM on TF-IDF Vectors

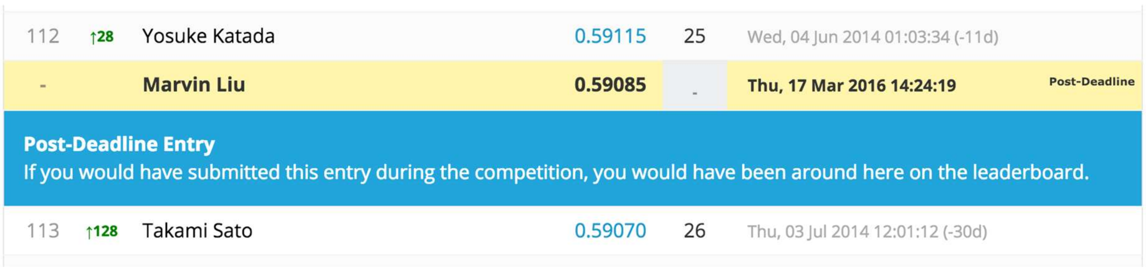
After vectorizing the essays into TF-IDF vectors, we trained a simple linear SVM classifier on those vectors to see if the outcome was exciting or not. To do so, we used a Stochastic Gradient Descent (SGD) classifier, which is simply a linear SVM approximated via stochastic gradient descent. We used l2-regularization to prevent overfitting and a logarithmic loss function to more heavily penalize classifiers that were confident about an incorrect prediction. This resulted in a score of 0.56924 on our Kaggle submission.

3.3 Ensemble of Previous Two Models

The ensemble model made use of the best features of the two original models. We decided to test taking the average of the two classifiers and see how that model performed when compared to the other two models. This improved performance to a degree (accuracy 0.58352) on Kaggle. However, we noticed that the predictions made by our gradient boosted model typically outperformed those of our text based model, on both the validation set and the test on Kaggle, so we attempted several variations of weighted average. The final version of our ensemble model weighs the prediction of the gradient boosted model as twice as heavily as that of the text model.

$$weigted\ average = \frac{(2 \times gbm + tm)}{3}$$

Using this model, the final version of our program predicts whether a program will be exciting with 0.59085 accuracy, placing us around 113th on the leaderboard.



112	↑28	Yosuke Katada	0.59115	25	Wed, 04 Jun 2014 01:03:34 (-11d)	
-		**Marvin Liu**	**0.59085**	-	**Thu, 17 Mar 2016 14:24:19**	**Post-Deadline**

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

113	↑128	Takami Sato	0.59070	26	Thu, 03 Jul 2014 12:01:12 (-30d)

4 Model Evaluation

We evaluated our models by scoring them against a validation set. For our validation set, we chose to use projects in 2013. We chose to use the most recent projects as validation rather than randomly choosing data because we had data ordered by time and we wanted to use older data to train a model and evaluate them on more recent data, since this is more similar to how it was tested on Kaggle.

5 Running the Code

There are several steps to running the code, as there is some preprocessing that needs to be done to extract our historical, categorical and text features. These are covered in the readme but are briefly summarized again here for convenience.

a. Run categorical_feature_extractor.py
b. Run historical_feature_extractor.py
c. Run text_feature_extractor.py
d. Finally, run the model of your choice. We used ensemble_model.py in our final submission, which is essentially a combination of the other two models (gbt_model.py and text_model.py).

6 Additional Comments

We encountered a multitude of problems in attempting to mine data on our laptops, which were far from powerful enough for the task. In particular, we regularly encountered memory exceptions while trying to load some of the larger csv files, and were forced to write several scripts for dealing with this.

We also decided not to include data before 2010, upon reading articles by some of top performers in the original contest, as they argued that this data had some odd quirks and did not truly reflect the rest of the data set.

7 Bibliography

Feature Selection. (n.d.). Retrieved from SciKit Learn: http://scikit-learn.org/stable/modules/feature_selection.html

Strategies to Encode Categorical Variables. (n.d.). Retrieved from https://www.kaggle.com/c/caterpillar-tube-pricing/forums/t/15748/strategies-to-encode-categorical-variables-with-many-categories

Using Gradient Boosted Trees to Predict Bike Sharing Demand. (n.d.). Retrieved from http://blog.dato.com/using-gradient-boosted-trees-to-predict-bike-sharing-demand