CS 4780 Final Project Propsal

1. Introduction

Our project focuses on a Kaggle dataset from Donorschoose.org, a site that allows teachers raise money for their classroom projects online, that gives an in-depth look into projects posted on the platform for the past 15 years. DonorsChoose differs from from other crowdfunding platforms in that teachers cannot raise money in excess of their goal, so once they hit their number the campaign is done. This also means that each campaign is all or nothing for the teacher, since if they don?t hit their goal the funders get their money back, and the teacher can?t set their goal slightly lower than what they wanted raise in order to try to lock in a minimum donation while hoping the momentum of their project will bring in enough money to fund it. Thus the problem we are trying to solve is what factors a teacher can alter to maximize their chance of getting funded.

Our basic approach was to use a combination of supervised machine learning techniques and statistical analysis to derive what factors have the biggest impact in a project getting funded. We tried LIST ALL THINGS WE TRIED to try to get interesting results and then did some research into the data by trying to find trends in excel. We also cleaned the data, did cross validation to tune our parameters for the learning algorithms we ran, and tried learning from a skewed dataset where we had an equal number of successful and failed projects. The work we did differs from the related work we researched because they were more focused on predicting the chance for a project at different intervals of time and based mostly on trends in social media and amount of money raised.

2. Problem Definition and Methods

2.1. Task Definition

The problem we?re addressing is what factors, that a teacher has control over, are most important to getting a project funded. We?re hopeful the results of our research can help guide teachers better on how to create projects for Donorschoose.org and lead to more projects being successfully funded. Some of the questions we hope to answer are when is the optimal time for a teacher to launch a project, what subject areas should they focus on, and how much money they should ask for.

2.2. Algorithms and Methods

The first main algorithm we use are Decision Trees and by extension Random Forests. A decision tree is a series of rules in the form of a tree. To classify a new instance, beginning at the root node of the tree, a particular attribute is examined, and depending on the result the the instance is passed off to one of the next level nodes. This continues until the instance reaches a terminal node, classifying it as the value of that terminal node. Trees are created using the TDIDT method of recursively creating nodes based on maximizing information gain. Early stopping is the preferred method to prevent overfitting. Different parameters into the TDIDT method include max tree depth and cost factor (similar to the -j parameter in an SVM).

Random Forests involve the creation of many Decision Trees using the above process. A forest classifies an instance i based on the classification made by a made by a majority of its trees on i. Each tree is created using the same TDIDT algorithm, but notably on a randomly chosen subset of the training data, and only permitting itself to split on a given subset of attributes. By doing this, noise in either outlier instances or unimportant attributes is statistically less likely to impact the final decision of the forest algorithm. Each Forest involved an odd number of trees to not have to break ties in an arbitrary fashion. Parameters for random forest creation include number of trees, number of samples (as a percent of the training set) given to each TDIDT running, and percent of attributes each TDIDT running was allowed to use to split. The overfitting parameters of max depth and cost factor were dropped in the Random Forest algorithm, because one of the main strengths of the Random Forest approach of averaging the responses of many decision trees is that individual overfitting to noise becomes much less of an issue.

//SVM EXPLANATION AND BRIEF USAGE

3. Experimental Evaluation

3.1. Methodology

Our main hope was to learn accurate predictors - rules able to correctly classify both positive and negative results. This was an important goal because, unlike many other problems, simply identifying positive instances at the cost of many false positives is not optimal. Knowing which projects are unlikely to get funded is equally important as knowing which projects are likely to be funded. Thus we determined statistical significance by OMG RAPHIE SAVE ME I HAVE NO IDEA WHAT I?M TALKING ABOUT.

Decision Trees and Random Forests pertain to this by the information they contain in their structure. The nodes at the top of the tree (or at the top of a majority of trees for forests) represent the attributes of a project instance that best distinguish funded projects from non-funded projects. By learning a statistically accurate decision tree or forest of trees, we could examine the makeup of those trees and make conclusions on their structures, identifying key project attributes.

//SVM STUFF AGAIN! Something about learning to find the value of vector w, what it?s attributes say. Also could just add to the above paragraph instead.

Because this is a Kaggle project, Kaggle provided the training and testing data. The provided data is actual project and funding data collected from DonorsChoose.org from 2000 to END DATE?. We selected a subset of 100,000 clean instances from the provided data to learn on, and partitioned that subset into ten 10,000 instance sections for use as training, validating, or testing. One of the first issues we ran into is that approximately 70% of projects on DonorsChoose.org get funded. Thus any learning algorithm must beat the naive approach of classifyng all instances as positive (funded), and achieving an accuracy of approximately 70%. While we did run tests on this data, we also wanted to see how our approaches would fare against an even set of data (50% positive), so we created sets of even ?fifty-fifty? data by randomly selecting and pairing equal numbers of positive and negative examples for each set of 10,000 instances.

For our methodology, we ran a full train-validate-test framework. For each algorithm, we selected train, validate, and test sets from the ten divided sections. From there, we trained the algorithm with all combinations of arguments on the train set to create a set of hypotheses. Each hypothesis was preliminarily tested on the validation set. The hypothesis with the highest accuracy on the validation set was outputted and tested on the test set to achieve final metrics. From this final test, we collected accuracy, precision, recall, FOne, and a full set of (Actual Label, Classified Label) tuples of instances in the test set. In the results section we examine the effect of different argument lists on the performance metrics to determine if there is any correlation between input and output metrics, and what the optimal argument input is for each algorithm.

3.2. Results

3.3. Discussion

4. Related Work

In our background readings we found that many people were doing post-mortem analysis of crowdfunding campaigns and looking for metrics to predict success after the project has already started. Vincent Etter, Matthias Grossglauser, Patrick Thiran (http://goo.gl/gWXrtf) looked into modelling the amount of money as a HMM and using KNN to find past projects at a similar state, tracking social media activity around a specific campaign, and graphing the other projects that the backers participated in. They found that the best way of predicting success at a given interval of time was to train a SVM that pulled from the results of modelling the donations and the social media activity. Professor Mollick (http://goo.gl/Pnyu8) tried to understand crowd funding through an analytical perspective and found that the strength of a person?s social network had a major effect in how successful the project was.

Our work differs from these papers because we?re focusing on what the creators can do to maximize their chance of getting funded as opposed to predicting success once a project has started. We are also using different mechanics to solve our problem than this related work since we?re using machine learning algorithms to extract insights from the data instead of using the data as a basis for future prediction. We?re confident that this approach will yield more meaningful results to the crowdfunding community since we?ll be able to deliver actionable insights instead of statistical significance.

5. Future Work

Although we were able to achieve significant improvements at classifying data from an even distribution of positive and negative examples, our best results run on the original data set do not represent much of an improvement over the naive classification approach. We have already exhausted many variants on decision trees and multiple kernals for SVM so the next logical step is to include more data in our analysis. Perhaps the most significant data we did not include in our learning algorithms is a one paragraph blurb that teachers submit along with project proposals. Perhaps running some NLP or sentiment analysis on these paragraphs and including this as a parameter in our learning algorithms would give a significant improvement on our error rate.

6. Conclusion

7. Schedule

- 25th October: Process and organize data
- 4th November: Begin classification
- 8th November: Compile results, begin comparison of different results
- 14th November: Compile models, begin comparison of different models
- 18th November: Model comparison, hypothesis testing
- 24th November: Begin writing poster and report.
- 4th December: Poster presentation.
- 10th December: Final project report (and code) due.