DonorsChoose.org:Application Screening

Srinithish Kandagadla[1]*, Rohit Bapat[2],Swarnima Sowani[3]

Abstract
Donor's Choose is an organization which helps the teachers who are in need of funding for implementation of the projects at their own school. Being such a great resourceful organization it gets many applications for the same. Hence a screening process is a must, however this requires human intervention at every level. We intend to reduce this manual process by incorporating machine learning techniques to predict whether a application gets accepted or not. Initial analysis to meet this objective started with exploring all the features for each application. As a part of EDA, we visualized certain features and their relationship with the acceptance of application. To prepare the data for model, feature engineering was done. We used techniques like Vectorization to represent words as features and train them on model.

Keywords
EDA — Vectorization — Donor'sChoose

[1,2,3]*Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*

Contents

Introduction 2
1 Background 2
2 Exploratory Data Analysis for Kaggle Data Set 2
2.1 Skewed Target Variables . . . 2
2.2 Student Teacher Prefix . . . 2
2.3 Project Grade Category . . . 2
2.4 Month Wise Analysis . . . 3
2.5 Weekday wise Analysis . . . 3
2.6 State wise Analysis . . . 3
2.7 Subject Categories and sub Categories Analysis 3
3 Feature Engineering for Kaggle Data Set 3
3.1 Essay Aggregation . . . 3
3.2 Word Count Calculation . . . 3
3.3 Word Cloud Formation . . . 3
3.4 Categories and sub-category . . . 4
3.5 Date and Timestamp Splits . . . 4
3.6 Teacher prefix . . . 4
3.7 Essay text preprocessing . . . 4
3.8 Tf-Idf Vectorizer . . . 4
3.9 Text Pipeline . . . 5
3.10 Sparse Matrix formation . . . 5
4 Questions-Essay Review Generation 5
4.1 Use of raw essay data . . . 5
4.2 Essay Review Classes . . . 5
4.3 Effect of Essay Reviews . . . 5
4.4 Skewness of Applications . . . 6
4.5 Effect of essay words on target variables . . . 6
5 Questions-Combining essays and resource data 6
5.1 Combining essays . . . 6
5.2 Feature Engineering on Resource Data . . . 6
5.3 Joining to application data . . . 6
6 Questions-Working on Donors Data 6
6.1 Exploratory Data Analysis for Donor's Data . . . 6
School State vs Donors State • Donor State Distribution • School State Distribution • Grade Category Distribution • Dropped Unimportant features
7 Models and Methodology 6
7.1 Application Screening . . . 6
7.2 Question: Donor State prediction . . . 7
Random Forest Classifier • Light GBM
7.3 Question: Essay and Resource data inclusion . . 7
7.4 Question: Essay Reviews . . . 8
Subsubsection
8 Experiments and Results 8
8.1 Question: Essay Review: Score Calculation . . . 8
8.2 Question:Essay Reviews . . . 8
8.3 Storing of preprocessed files . . . 8
9 Summary and Conclusions 8
9.1 Importance of textual data . . . 8
9.2 Inclusion of resource data . . . 8
9.3 State prediction based on project and donation data 9
9.4 Rank on Kaggle Leaderboard . . . 9
Acknowledgments 9
10 References 9

Introduction

DonorsChoose.org gives the opportunity to public school teachers to apply for materials and funds for student projects. Due to huge response, the number of applications to be screened is always increasing. This application screening requires human intervention. After going through the competition on Kaggle[1] we were interested in finding a way to automate this screening by applying machine learning. We chose this project as it had a mix of two interrelated domains Machine Learning,Natural Language Processing. Implementation of this project can also extend to other applications like Cover Letter filtering,Essay grading etc. This project will also help us learn nuances of NLP. Under the scope of this project we also graded this essay on certain values of polarity and subjectivity. On initial analysis we also found that the application data is highly skewed in favor of accepted applications. We explored the ways like up sampling and Down sampling to get an unbiased data.

1. Background

This was a Kaggle competition and a real world scenario. There have been numerous submissions to Kaggle implemented with various models. This is classic classification problem where in we predict whether an project application will be accepted or not. However, it is mainly based on the text features. As a part of background work we went through some submissions on Kaggle.com[1]. However we found most of the kernels were only considering superficial features for textual data like word counts and sentence counts. Submissions which used the textual data only combined the essays without text pre-processing.We got the information about the up sampling from all these submissions as all of them had identified the skewness initially. The skewness is in the ratio of 80%-20% in favor of accepted applications. Based on this background exploration we decided to prioritize work based on three milestones.

1. Work on reducing non-textual features to put more emphasis on textual features.
2. Reducing the textual features to unique occurrences and base words.
3. Identifying the decision making and important features from the newly generated features.

2. Exploratory Data Analysis for Kaggle Data Set

2.1 Skewed Target Variables

Insights From this diagram we could find that the percentage of accepted application was very high than the applications being rejected. Finding the factors which are introducing this skewness would help us find the important features for the target variable. Refer Fig 2:

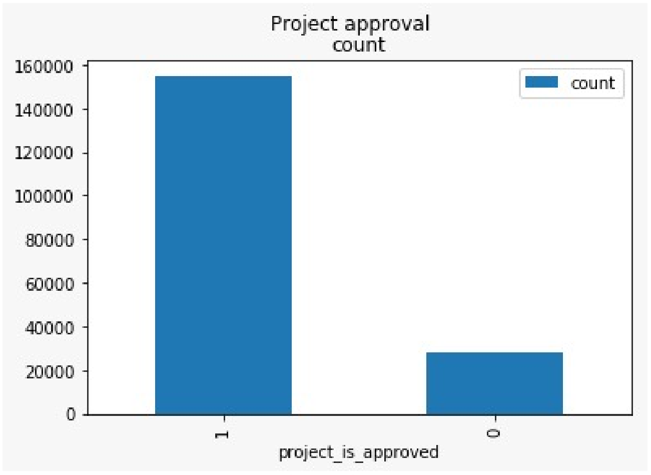

Figure 1. Skewed Acceptance

2.2 Student Teacher Prefix

Insights We plotted number of applications accepted or rejected based on the teacher prefix. The graph shows that most of the submissions were done by female faculties i.e the count for Mrs. and Ms. is substantially higher than Mr. prefix. We looked in to these anomalies and created a new feature which will be described in feature

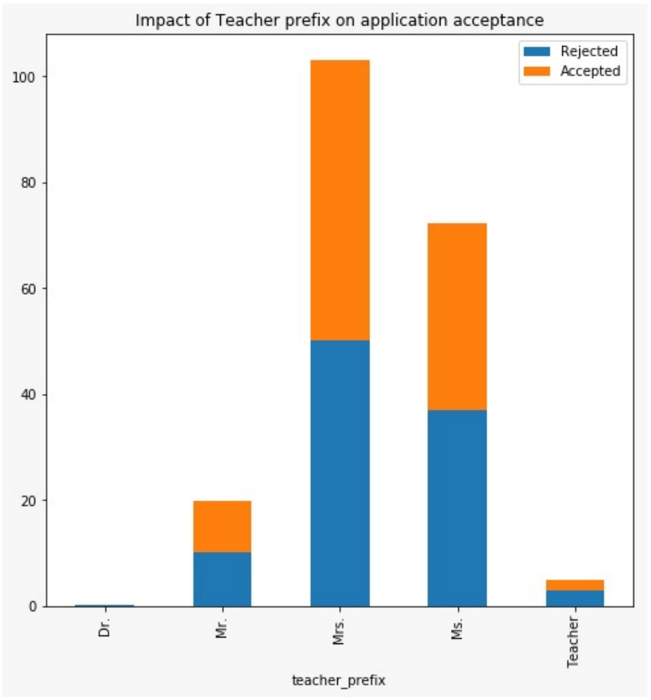

Figure 2. Teacher prefix Acceptance rate

2.3 Project Grade Category

Insights We had total four grade categories. We could see the distribution of application decisions based on grade category. Grade category describes the grade for which the project needs to be sponsored. We considered this grade category as ordinal as there was a natural ordering to the grade categories.

2.4 Month Wise Analysis

Insights The graph shows count of the submitted applications over months. From this graph, we can find the percentage of accepted applications each month.

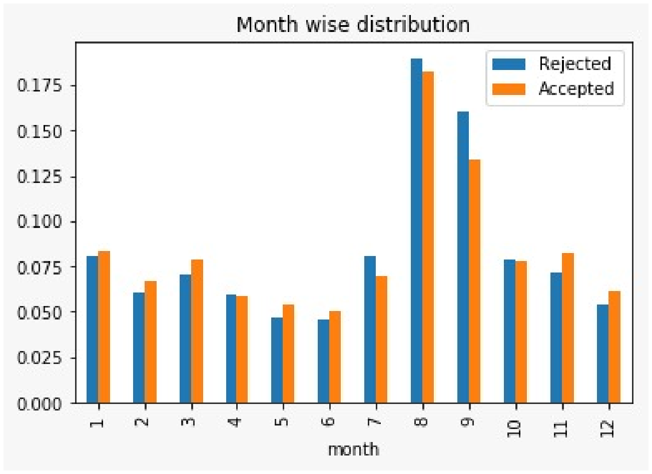

Figure 3. Monthwise Distribution

2.5 Weekday wise Analysis

Insights After analyzing the month wise data we looked at the count of applications submitted on each day of the week. We tried to find if submission done at start of the week impacts on the final decision.

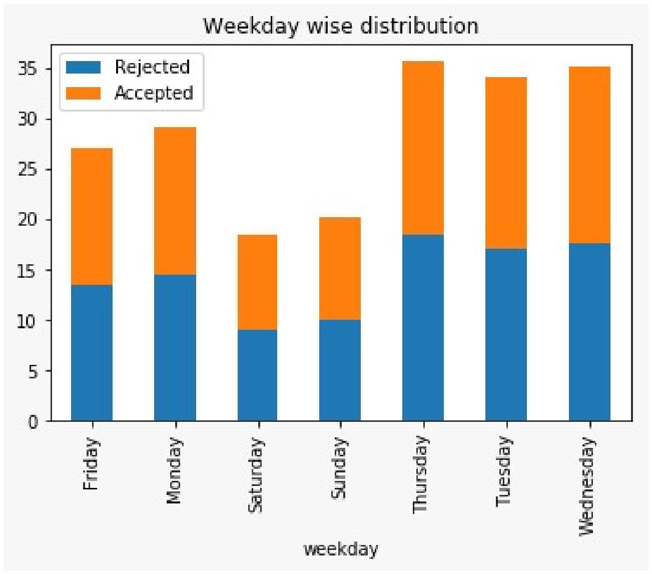

Figure 4. Weekday Analysis

2.6 State wise Analysis

Insights We analyzed the state feature and its effect on the target variable. The location from where the application was submitted had a specific trend. Many applications were from state of California.

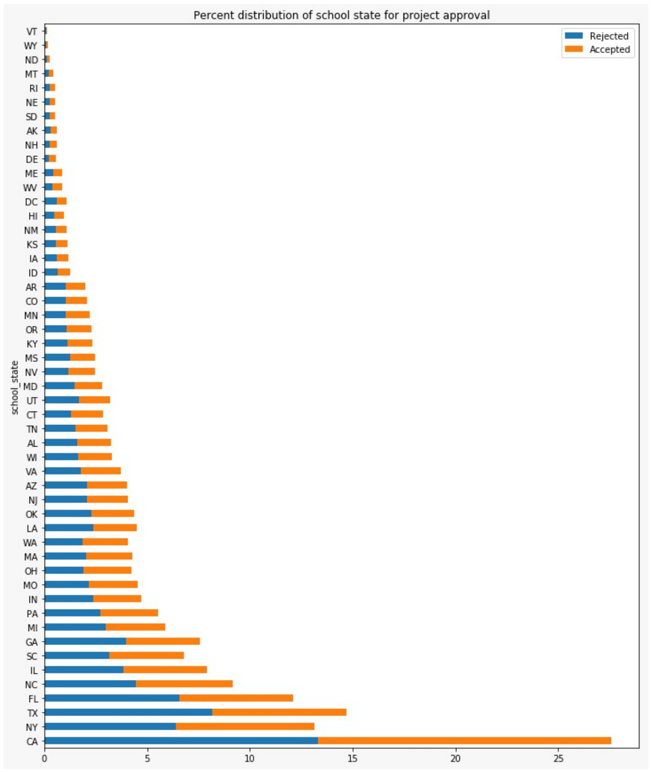

Figure 5. Statewise Analysis

2.7 Subject Categories and sub Categories Analysis

Insights Each application was classified into one of the categories and sub categories. The plot shows distribution of the accepted and rejected applications based on category or sub category.

3. Feature Engineering for Kaggle Data Set

3.1 Essay Aggregation

Combined As there were irregularities in the number of essay submissions we combined all the essays under one field of essays for further text processing.

3.2 Word Count Calculation

Calculated After combining the data of essays we conducted a word count calculation. We did this introduce the combined essay length as a feature to predict outcome.

3.3 Word Cloud Formation

Calculated We tokenized each word reduced to its base form and removed any specific punctuations. After getting the base words we formed a word cloud which gives most frequently words like 'student','book' etc used in application. We found that words like 'material' have the most negative impact on the decision.

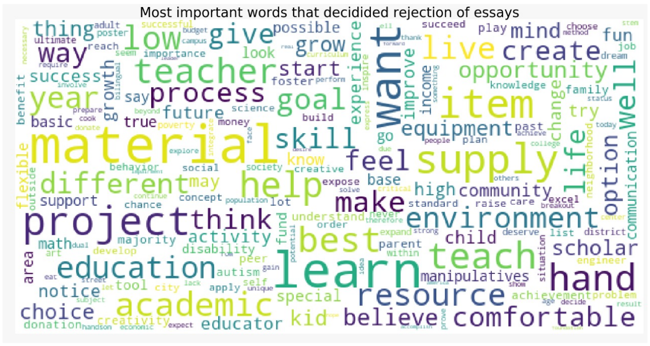

Figure 6. Word Cloud for Rejected Application

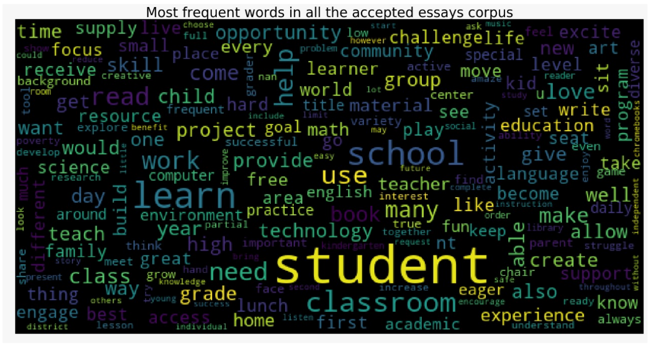

Figure 8. Word Cloud for Accept frequency

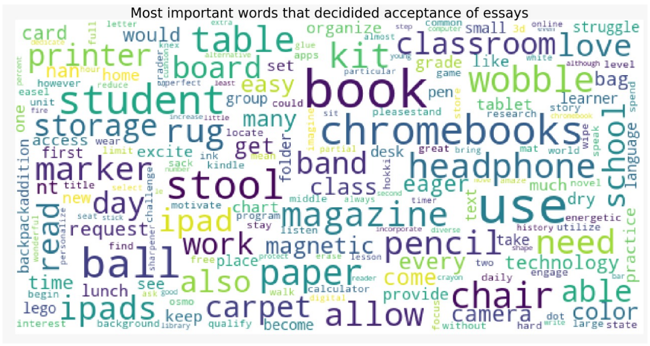

Figure 7. Word Cloud for Accepted Application

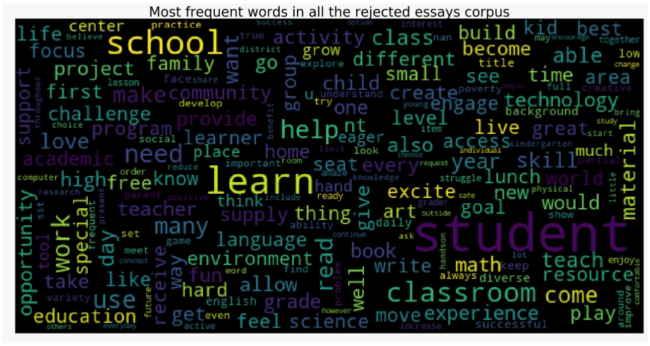

Figure 9. Word Cloud for Reject frequency

3.4 Categories and sub-category

Calculated There were a lot of unique combinations of categories and subcategories for each submission. After analyzing these categories we found that most of the categories can be split to form few single distinct categories. Additionally the sub-categories were combinations of the main categories. As a part of feature engineering we separated these sub categories and encoded these names as new features with dummy values. This significantly reduced the categories from 51 to just 9 and subcategories from 407 to 30.

3.5 Date and Timestamp Splits

Splitted We extracted the day,month and time from feature related to date of application submission and also made a continuous epoch representation of time stamp. This could be used as a continuous feature.

3.6 Teacher prefix

Skewed The EDA showed that the number of Mrs. and Ms. prefix was very high hence we decided to aggregate these teacher prefix and form gender based columns. For gender neutral features like Dr. and Teacher as the count was very low we grouped them in the same ratio as that of Male Female count.

3.7 Essay text preprocessing

Combined The number of essay submissions changed from 4 to 2 over time. We combined all these essays for each submission and dropped the other essay columns. Tokenizing of each sentence in essay was done. We reduced the words to their root form such that we would have lesser unique words across the data there by reducing the dimensions. We used WordNet's Lemmatizer to reduce words to root form based on its part of speech.We used stemmer to stem the words to further reduce the unique words. We explored Lancaster and Porter Stemmer to check its effect on the essay text. However after using stemmer many important words got reduced to meaningless root forms. For example, 'students' was stemmed to 'stud', so we decided to drop stemming and just use lemmatization as we wanted all important words as features. We used inflect engine to convert the numerical data in alphabets to reduce the number of features. However these numerical features did not come up in the feature importances. Hence we decided to drop this processing.

3.8 Tf-Idf Vectorizer

Frequency We used the TfIdf vectorizer to generate the term frequencies and inverse document frequencies to score the words according to their importance across the whole data. The TF-IDF is the term frequency times inverse document frequency. We used this vectorizer to scale down the impact of words which occur very frequently in our essays thus rendering them comparatively unimportant. We found a better description of Tf Idf Vectorizer on tf-idf.com

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus di-

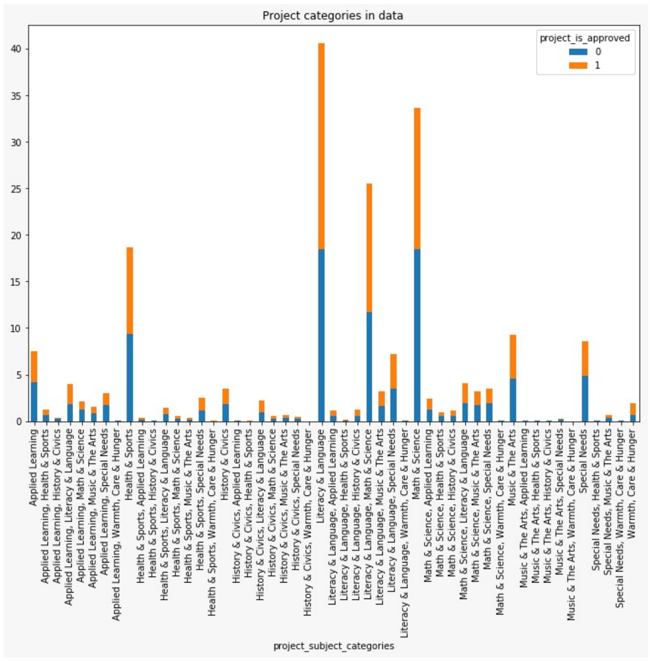

Figure 10. Categories

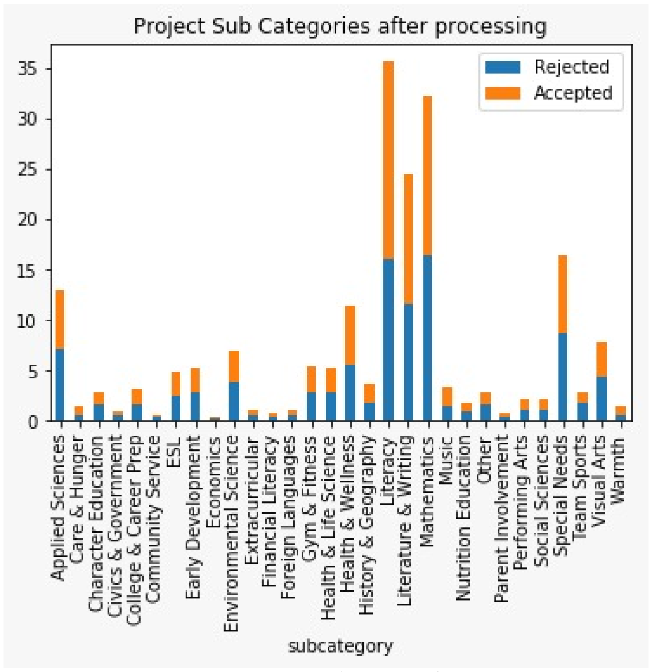

Figure 11. SubCategories

vided by the number of documents where the specific term appears.

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as ”is”, ”of”, and ”that”, may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

$IDF(t) = log_e(Totalnumberofdocuments/ Numberofdocumentswithtermtinit)$.

3.9 Text Pipeline

Text preprocessing We sent the word list of essays to our text pipeline to perform the following steps:

1. Converted all words into lower case.
2. Removed anything that is non alphabetical,non numerical or not a space.
3. Check that the word is not in stopword list.
4. Lemmatized each word with WordNet Lemmatizer.
5. Use of synsets to reduce complex words to root forms.
6. Include this processed word and send it back to data frame.

3.10 Sparse Matrix formation

Sparse We passed the combined essays, project titles and project summary descriptions to the text pipeline. We then vectorized each of these texts and built the vocabulary on training data. We combined all these dummy coded essays,titles and summaries to form a new sparse matrix for training.

Test Data After working on train data we used the same vocabulary used by training vectorizer to get tf-idf values over the same set of words.This ensured new words from test data were dropped and dimensions of train data and test data in terms of features remained same.

4. Questions-Essay Review Generation

4.1 Use of raw essay data

We included this step before dummy coding the values of combined essays and sending it to data frame. By using the text polarity and text subjectivity functions from textBlob package we found a score for each entry and added them as separate features. We used these features to identify thresholds.

4.2 Essay Review Classes

We explored some kernels from Kaggle[1] to identify the limiting conditions for essay to be rejected. Based on these conditions we came up with 4 categories ’Poor Polarity’,’Poor Subjectivity’,’Essay Below Average’ and ’Acceptable Essay’. m[2]

4.3 Effect of Essay Reviews

Based on the thresholds identified we split the essay data based on the target variable and essay review classes mentioned above.

4.4 Skewness of Applications

Considering only the text data as input data the overall acceptable essays are pretty high in both the sets of applications. However, the ratio of poor subjectivity to acceptable essay in rejected application set was slightly higher than ratio in accepted application set. This tells us the review score could have a very slight effect on the target variable.

4.5 Effect of essay words on target variables

Considering the vectorized words as important features we trained a Light GBM Model only on sparse data generated from the vectorizer. The AUC scores for the rounds ranged from 67% and increased to 73% till last round. This AUC score helped us identify that the essay words also play important role in deciding acceptance of application.

5. Questions-Combining essays and resource data

5.1 Combining essays

As of May 2016, the number of essays to be submitted reduced from 4 to 2. We observed normal trend that the length of the essay increased after the count reduced to 2. We combined these essays to form a single text entry.

5.2 Feature Engineering on Resource Data

After getting the data from Kaggle we decided to combine the resource data with application data. The overall cost of resources for the project can have an effect on target variable. We multiplied the quantity and price of the resource to get the total price for a particular project. We combined and added up all these total prices and made it unique for a particular project.

5.3 Joining to application data

After calculating the total price, we dropped the resource descriptions. We joined the data from resources to both training and test application data based on the project ID.

6. Questions-Working on Donors Data

6.1 Exploratory Data Analysis for Donor's Data

6.1.1 School State vs Donors State

State Data We had two sources of data from the Donors Choose website donation data and project data. The project data had school state while the donation data had the state as a feature in it. We plot these two features to check for count of applications from school state and donors state. We used a stacked graph to represent the donation states for a particular school state.

6.1.2 Donor State Distribution

Most Frequent State From the donor state distribution we found the count of applications in every state. We observed that California was highest grossing donation state.

6.1.3 School State Distribution

Most Frequent State From this feature we observed that the California state had the most number of accepted applications. However this balances out the fact that the donation rate is also higher in California.

6.1.4 Grade Category Distribution

Grade Distribution We had 4 categories for grade. We checked the distribution of projects accepted for donation based on grade categories. This gave us insights about acceptance rate based on grade category and location state of submission.

6.1.5 Dropped Unimportant features

We only considered the important features from the Dontation and Project datasets. We identified the important features based on the common features from Kaggle project data. We included all the location relevant features from the project data. We dropped location related features from the donation data as it introduced overfitting of training data. We dropped certain features like school latitude, school longitude,school country, school charter, year round and so on.

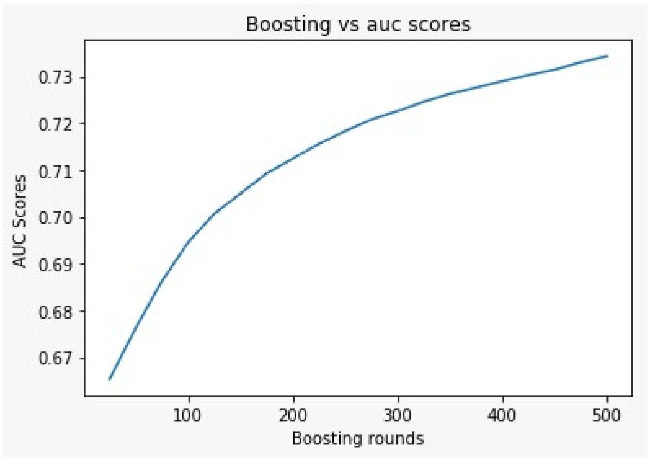

Figure 12. Light GBM Boosting

Reference to Figure 15.

7. Models and Methodology

We observed that the data was skewed for the acceptance rate in favor of accepted applications. We first decided to tackle the Kaggle problem statement to predict the acceptance decisions for each test entry. Later we tackled the questions put forth by us with respect to essay reviews, combination of essay & resource data addition, prediction of donor state based on the project submitted.

7.1 Application Screening

After the initial feature initial extensive feature engineering on the train data we developed a vocabulary for essay data,summary data and title data. This increased the number of features of training data to 200,000. As an preliminary step we considered only the textual data which consisted of:

1. Combined and tokenized essay data.
2. Tokenized summary data.
3. Tokenized title data.

Initially we started with a Random Forest Model to predict the score of testing data for which we got an accuracy for 84%. However these results looked strange considering skewness in data. By considering imbalanced data, we calculated the AUC score which was effectively a Kaggle submission requirement. We decided to train a baseline Ligh GBM model to extract important features from these words. Based on this decision we trained all three textual data features as sparse matrix. This model was boosted for around 3000 rounds before it stopped for AUC score of 0.73.
We decided to call feature importance from this trained model to extract top 10000 decision making words.
We trained the same model again with these important features to magnify the value of importances. We decided to take up top 2000 words influencing the decisions. The approximate distribution was as follows:

1. Around 1200 words from essay data.
2. Around 500 words for essay summary.
3. Around 300 words for essay title.

After this filtering criteria, we combined the textual data with the non text features like state,date,grade category. We trained a final Light GBM Model with the textual and non textual data. After calculating the AUC scores for test data we got a Kaggle public score of 0.76498 and private score of 0.75518. However we had still not combined the resource details of each project from the resources dataset.

7.2 Question: Donor State prediction

We intended to predict state from a which donation can come in based on the project application details. On initial analysis of donation and resource data we found that the project ID in the project data were masked. As a result we could not map these project IDs to the Kaggle data set. We decided to predict the donor state based on the 2 datasets polled from DonorsChoose.org[4] open data website.
We trained 2 models for donor state prediction Random Forest Classifier and Light GBM Model.
We tweaked the parameters to include a multiclass output of target variable. There were a total of 60 unique states for donation data.

7.2.1 Random Forest Classifier

Based on random forest classifier we got a score of 53% for the validation data. As this target variable was out of scope for the Kaggle project we decided to split the train data as 75% and 25% for train and test respectively.

7.2.2 Light GBM

In order to increase the accuracy of the data we decided to train the same data on Ligh GBM model. After changing the parameters for multi class problem we got a better accuracy of 55.8%. We identified the important features in this data set like school zip, number of donors for a state.,date posted in milliseconds,date completed,date expiration and so on. Including donation zip code in the training data was not a viable option as many zip codes were missing and it had direct effect om the target class of state.

7.3 Question: Essay and Resource data inclusion

After initial feature engineering on the resource data as mentioned in the initial proposal we mapped the total resource cost to both training and test data. Exploring the kaggle kernels help us identify that resource detail addition affected the AUC score. We followed the same steps for textual data preprocessing. For the non textual data we just used the new train data frame with added details about resource data. With the same parameters we got the AUC score for the test data. We submitted these results to Kaggle and got the highest score in our attempts as [3]

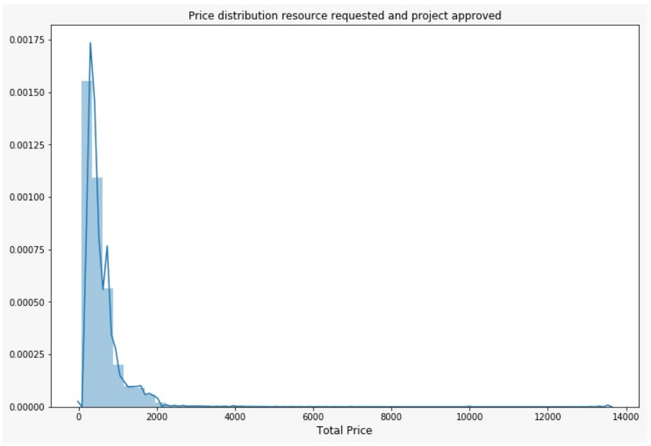

Figure 13. Price Distribution

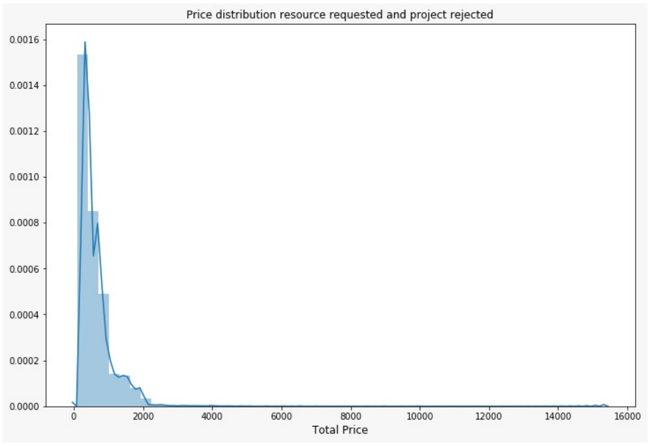

Figure 14. Price Distribution for Rejected Applications

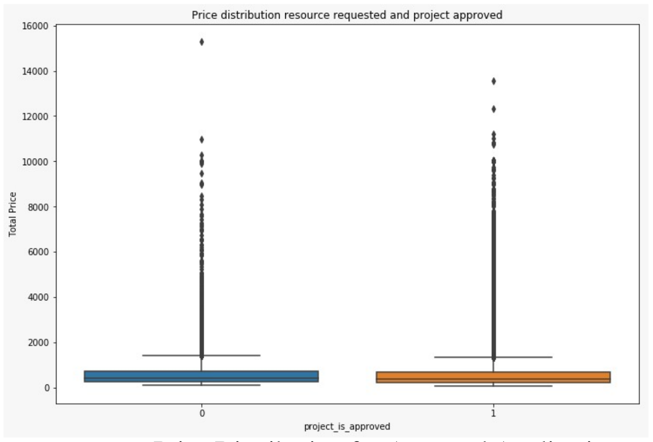

Figure 15. Price Distribution for Accepted Application

1. Public Score: 0.76614 .
2. Private Score: 0.75808 .

7.4 Question: Essay Reviews

We first identified that essays also play an important role in deciding whether the application is accepted or not. We first started treating raw data based on text polarity and subjectivity from the textBlob package. We cam up with thresholds for the calculated polarity and subjectivity.
Based on this preliminary analysis we decided to use an unsupervised model of K means clustering which we have described in the experiments section.
We again used Light GBM model[3] with a multi class parameter to group this data based on the essay classes defined initially. As the data is skewed the number of essays with acceptable essays was very high. Still we got around 2000 with a poor subjectivity and polarity scores based on tweaking of thresholds. For this model we first had a train test split of 80% and 20% for training and test data respectively. We got a AUC score for this model around 0.79.

7.4.1 Subsubsection

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Word Definition

Concept Explanation

Idea Text

- First item in a list
- Second item in a list
- Third item in a list

8. Experiments and Results

8.1 Question: Essay Review: Score Calculation

As we had a target variable and feature importance of words in an essay we decided to take in frequency counts of a particular word. We tagged this word as positive or negative based on frequency count. We tried to predict a score for each essay by multiplying the feature importance value, decision (1: positive word -1:negative word). However this score could not give us specific thresholds for essay grading.

8.2 Question:Essay Reviews

Based on essay content we wanted to generate a very generic review about it as a feedback for application. We first decided, as the original problem was binary a review just based on 2 classes would be sufficient. However on further study as the data was skewed, we thought of exploiting why a particular application is getting rejected with respect to essay. Hence we decided to have 2 categories each for accepted and rejected essay based on content. On execution of K means clustering model even though we got sufficient data in all 4 clusters there were no thresholds defined to rate one cluster over another. A split based on accepted and rejected was not useful. Hence we decided to calculate the polarity and subjectivity scores for each essay. Now based on these thresholds we executed the Light GBM model with 4 categories as mentioned in Models and Methodology section.

8.3 Storing of preprocessed files

The text pipeline for essay tokenization was taking a lot of time as there were around 200,000 rows in training and 78,000 in test data. Each row had textual content of title,summary and 2/4 essays based on date of submission. Hence we first decided to store the dense matrix file with .npz extension for every new preprocessing calls. However we decided to also include the vectorization and sparse matrix generation in the same phase and stored all the files in a single run of text preprocessing. We set a flag of text preprocessing which can be changed to 'False' if files need to be modified.

9. Summary and Conclusions

9.1 Importance of textual data

After training these models separately, once only on textual data and then on textual and non-textual data the AUC score was from 0.73 to 0.76. This shows processing of textual features was enough to get satisfactory AUC score. This rendered essay,title and summary as important attributes.

9.2 Inclusion of resource data

After including resource data and combining with the train dataframe the AUC score increased by approximately 0.05. This shows resource price aslo has an effect on the target variable.

9.3 State prediction based on project and donation data

For the state prediction based on Donor data we got a accuracy score of 55.8%. We think that had their been more data or a link between the project data and Kaggle data we could have got a better accuracy. Other features from our train dataframe could have contributed to state prediction.
For essay review prediction, even with good AUC score as the data was skewed towards acceptance, essays from rejected categories also got a good rating. We think in such cases the other non textual factors affected the final decision.

9.4 Rank on Kaggle Leaderboard

After submitting result file out of approximately 600 participants we got a rand of around 269. The highest Accuracy of Kernel was 82% compared to us which is 76%

Acknowledgments

We would like to thank Prof. Gregory Rawlins for giving us the opportunity to work and help us understand the concepts which we have used in DonorsChoose.org Application Screening project.
We would also like to thank Manoj Joshi for assisting us and mentoring whenever we faced any issues in our project.
We thank Zeeshan Ali Sayyed and Christopher Torres for suggesting us improvements in our initial project proposal and during all the phases of project.
We would thank IU infrastructure for Big Red II (https://kb.iu.edu/). It made computation easier for us.
We also used the Kaggle Competition Kernels to run and submit our results of AUC scores on the competition page.
Kaggle Kernels Page

10. References

[1]*KaggleKernelsPage*

[2]*EssayScoringKernel*

[3]*LGBMKernelImplementation*

[4]*DonorsChooseData*