Capstone Report

Udacity Machine Learning Engineer Nanodegree

DonorsChoose.org Application Screening

Aaron Penne

2018-07-07

Table of Contents

1 Definition 3
 1.1 Project Overview 3
 1.2 Problem Statement 3
 1.3 Metrics 4
2 Analysis 5
 2.1 Data Overview 5
 2.2 Data Exploration 5
  2.2.1 “Resources” Dataset 5
  2.2.2 “Training” and “Test” Datasets 6
 2.3 Exploratory Visualization 8
 2.4 Algorithms and Techniques 10
 2.5 Benchmark 11
3 Methodology 11
 3.1 Data Preprocessing 11
  3.1.1 Numerical Feature Engineering 11
  3.1.2 Handling Text 12
 3.2 Implementation 13
 3.3 Refinement 15
4 Results 15
 4.1 Model Evaluation and Validation 15
 4.2 Justification 16
5 Conclusion 16
 5.1 Free-Form Visualization 16
 5.2 Reflection 16
 5.3 Improvement 17
6 References 18

1 Definition

1.1 Project Overview

Teachers in low income schools often find the school is unable to provide their students with adequate resources. This leads some teachers to purchase items with their own money to meet the students' needs. To help solve this issue, Charles Best created the website DonorsChoose.org to allow philanthropic individuals the opportunity to selectively donate to various teacher projects. Teachers submit their applications to DonorsChoose.org, and if they are accepted then the project goes public and people may donate to it.

This charity has served as a Kickstarter for small public-school needs and provides a direct way to make an impact on young people's lives. DonorsChoose.org has received top industry awards every year since 2005 and has fulfilled over 600,000 classroom projects. This impact can be made even greater by leveraging advanced technologies to actively utilize the datasets on hand.

DonorsChoose.org implements a screening process to ensure a high-quality level of projects are listed on their site. This means hundreds of applications need to be read in detail by volunteers. This takes a lot of time and resources for the organization, and also adds delay to the teachers waiting for approval. The problem to be solved is a classification one, should the DonorsChoose.org accept or reject a given application. By automating a large segment of this process, applications that have an obvious classification can be accepted/rejected, while the more nuanced applications can be read more in depth by a volunteer.

1.2 Problem Statement

This is a supervised learning classification problem where the model takes text and numerical values as input features and outputs the probability of being in the correct class. There are two possible classes, therefore this is a binary classification problem. Initially the data will be run through a baseline predictor that matches the percent split in the classes. This gives a baseline to use for model comparison.

The next step will be to run the data through a random forest classifier. The sample size is about 200k records, as such a stochastic gradient descent classifier is a good fit for this problem. To ensure the right model is selected several models will be tested, and various combinations of them will be used in an ensemble voting method. The initial models tested with k fold cross validation will be logistic regression, stochastic gradient descent, support vector machines, and naïve Bayes.

The dataset is compiled from historical project applications that were submitted to DonorsChoose.org. These are presented on Kaggle [2] in the form of comma separated variable files (CSV). These will be used as inputs to create a predictive classifier. In order to solve this problem, several steps must be taken: The data must be imported, cleaned, examined, and joined to other inputs. Various algorithms must be selected and cross validated to identify which model fits best, perhaps combining models into a customized ensemble. The chosen algorithm(s) must be selected and trained on the historical data. Finally, testing data will be run through the model(s) to create predictions which will be scored by Kaggle.

There are three data files available: training application data, test application data, and data on the resources requested. The application data contains some metadata about the application and the essays. The resources data includes one-line item per resource requested and may have multiple line items for a single application. Initially it seems that there will be predictive information in the resources categorical data and numerical data. It also is possible that the categorical and numerical data in the application metadata will be valuable for prediction. The essays and other text information will likely contain the most information gain, as there will be a deep and wide breadth of words used by the nationwide applicants.

These initial observations lead me to believe a two-pronged approach will be needed from a high-level perspective. One will be model(s) based on the categorical and numerical data. The other will be model(s) based on the text essays. Several algorithms will be tested using cross validation of the test data to find the optimal model.

1.3 Metrics

The basic metrics for analyzing the performance of a classifier are counts of: True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These numbers are eponymous counts of, for example, the number times a prediction was positive when the actual label was positive. These are valuable metrics, but it is difficult to identify superior models by comparing 4 values for each model [3]. TP, TN, FP, and FN are typically combined into two metrics called precision and recall, defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

While these are useful in their own right, two metrics are still difficult to use when comparing multiple models. Therefore, they are combined into a single metric known as the F_1 score, defined as the harmonic mean of precision and recall

$$\text{F}_1 = \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Another way to use the TP and FP values is to use a receiver operating characteristic (ROC) curve. This metric originated in radar detection to score a radar operator's ability to correctly identify a target. Nowadays it is used to characterize algorithmic classification instead of human radar operators, but the principle is the same. ROC curves plot the FP rate (FPR) against the TP rate (TPR) for all decision thresholds. This form of plot is convenient to use as a metric because it is consistent across different models and methods [6].

A typical summary metric, which will be used for model selection and performance rating is the area under the ROC curve (AUC). This score essentially is the probability that a given prediction will be correct in predicting a positive result. For example, if the AUC score is 0.85, then there is an 85% chance that a positive prediction is correct.

Accuracy is a poor choice of a metric, although it is intuitive. When imbalanced datasets are used, accuracy gives a skewed result. Accuracy is intuitively based on a 50-50 random result being the baseline, but that is not the case with imbalanced datasets. This is where the benefit of

AUC comes into play, the metric has the same meaning whether the dataset is imbalanced or not. As this dataset is imbalanced, the AUC metric will be the primary method of rating.

2 Analysis

2.1 Data Overview

Each request contains metadata such as the school's state, class grade, teacher ID, categories, submission datetime, etc. The primary content of each request is essays filled out by the teacher, with 2-4 essays per request. Another resources dataset is provided that details the description, quantity, and price of each item in the applications. This resources dataset is tied back to the applications dataset using a unique id for each application.

The training set has 182k records and the testing set has 78k records. There are two class labels in this dataset, "accepted" and "rejected". An interesting break in the data is that applications before a certain date had 4 questions to be answered. The latter applications had 2 questions. It will take some experimentation to determine how to properly handle this discrepancy, more information on this is in Section 2.2.2.

Many of the attributes are numerical in nature. These will be the simplest to model. Many attributes are also categorical, meaning they can be converted to numerical values and modelled in similar ways. The bulk of the potentially interesting data, however, is the essays written by the teachers. These provide both a challenge and an opportunity, as there is quite a bit of information encoded in these texts.

2.2 Data Exploration

2.2.1 "Resources" Dataset

Each application has a list of items requested where one application may have many requested resources. Further, the quantity of each resource varies. For each resource item the dataset contains the quantity and price of each item. Pivoting the data is a quick way to summarize the numerical values by summation, then multiplying the total unit cost and quantity to get a total cost for each application. Some questions that arise from a dive into the raw data lend themselves to answers via feature engineering.

- Do cheap or expensive applications tend to get approved?
    - Feature created: Highest/Lowest price of items (max/min of price)
    - Feature created: Total cost of request (sum of price*quantity for each application)
- Do applications with few or many items tend to get approved?
    - Feature created: Total number of items requested (sum of quantity)
    - Feature created: Total number of unique items requested (count of items per application)
- Does the type of item request affect approval?
    - Feature created: Aggregation of all words from item descriptions (string join of description)

The possibility of information being gleaned from the list of descriptions means that text analysis will need to be used. This was obvious from the data descriptions but here it becomes apparent.

2.2.2 "Training" and "Test" Datasets

The training and test datasets have the same features except the training data has class labels identified.

Notes on first pass of features

- id – unique id of the project application used to tie in resources.csv data
- teacher_id – ID of the teacher submitting the application can be disregarded because counts of submissions per teacher captured in other feature, teacher_number_of_previously_posted_projects
- teacher_prefix – title of the teacher's name (Ms., Mr., etc.) may be useful once encoded
- school_state – US state of the teacher's school may be useful once encoded
- project_submitted_datetime – application submission timestamp
- project_grade_category – school grade levels (PreK-2, 3-5, 6-8, and 9-12) may be useful once encoded
- project_subject_categories – category of the project (e.g., "Music & The Arts") but several records have 2 entries, needs to be split into 2 new columns. May be useful once encoded.
- project_subject_subcategories – sub-category of the project (e.g., "Visual Arts") but several records have 2 entries, needs to be split into 2 new columns. May be useful once encoded.
- project_title – title of the project in raw text
- project_essay_1 – Long form essay
- project_essay_2 – Long form essay
- project_essay_3 – Long form essay for applications before May 17, 2016
- project_essay_4 – Long form essay for applications before May 17, 2016
- project_resource_summary – summary of the resources needed for the project
- teacher_number_of_previously_posted_projects – number of previously posted applications by the submitting teacher
- project_is_approved – whether DonorsChoose application was accepted (0="rejected", 1="accepted") present in the training data only

There is a caveat with the long form essays. Only 3.5% of the applications contain essays 3 and 4. According to DonorsChoose.org the prompts changed after May 17, 2016. Prior to this the four prompts were:

- project_essay_1: "Introduce us to your classroom"
- project_essay_2: "Tell us more about your students"
- project_essay_3: "Describe how your students will use the materials you're requesting"
- project_essay_4: "Close by sharing why your project will make a difference"

This poses a problem, as it is typically a good idea to remove features that have a large number of nulls. However, this is not desired in this case because there is a large amount of information in the long form essays. Upon closer inspection, they may be aggregated with the other two essays. The two new prompts appear to be an aggregation of the four old prompts. Here are the prompts after May 17, 2016:

- project_essay_1: "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

- project_essay_2: "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

Looking through the data confirms the initial hypothesis that a two-pronged approach will likely yield the largest amount of information gain from this data. Further questions answered with feature engineering approaches are described below.

- Do the categorical features independently have an effect on approval?
- Does the date of submission affect approval?
    - Feature created: Approval rates by year, month, day of year, day of week, hour
- Do new or experienced submitters get approved more?
- Does the title affect approval?
    - Feature created: Number of words in project_title
    - Feature created: Sentiment of project_title
- Do the essays affect approval?
    - Feature created: Number of words in each essay

For further insight, below is an overview of the data in the training dataset as well as full text examples from dataset:

Feature Name	**Value**	**Type**
id	p226941	String (key)
teacher_id	103cc1667cf9361bf1c58c8425e76e95	String (key)
teacher_prefix	Mrs.	String (Categorical)
school_state	CA	String (Categorical)
project_submitted_datetime	9/5/16 19:28	Datetime
project_grade_category	Grades PreK-2	String (Categorical)
project_subject_categories	Literacy & Language, Math & Science	String (Categorical)
project_subject_subcategories	Literacy, Mathematics	String (Categorical)
project_title	Technology Boost!	String
project_essay_1	<See Essay 1 below>	String
project_essay_2	With our new iPads, my students will…	String
project_essay_3	Null	String
project_essay_4	Null	String
project_resource_summary	My students need a projector and…	String
teacher_number_of_previously_posted_projects	1	Integer
project_is_approved	1	Integer

Essay 1:

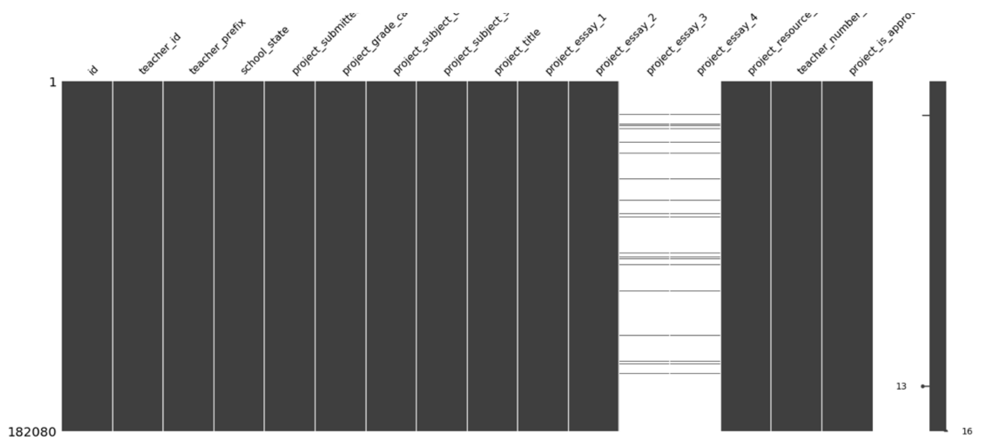
> *My children come to school everyday with the same expectations as any other children; they are eager to learn, excited about new discoveries, and want to feel like they have a place in this world. My challenge is to meet their expectations regardless of the fact that many of them face extremely difficult economic situations at home. \r\n\r\nAll children deserve access to educational tools, regardless of their family's economic standing! Most of my students come from working class families that are trying their best, but are unable to provide what many take for granted. The only exposure most of my students have to technology is at school, and our aging classroom equipment is barely limping along.*

Note that the essays are stored as a single string, with escaped characters such as "\n" present. This means some string cleaning will be needed when reading this data in. More on the approach to processing this data can be seen in section 3.1.2.

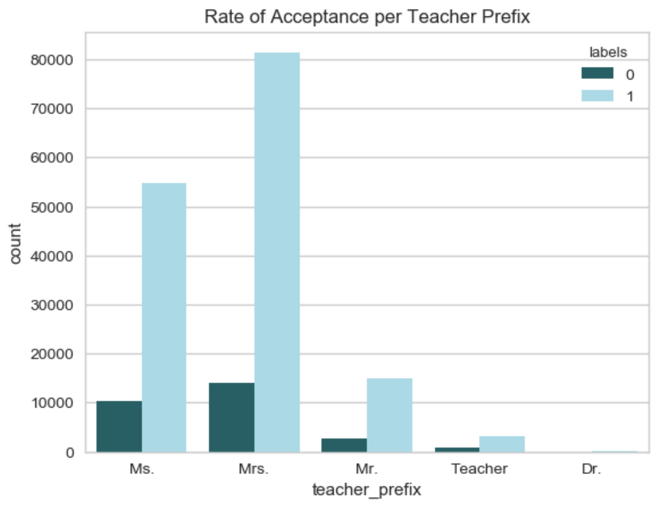
The project_subject_categories and project_subject_subcategories features are categorical, but many records contain multiple categories. For example, a project_subject_categories value may be a single entry such as "Applied Learning", or it may have two entries such as "Music & The Arts, Health & Sports". It will likely be a good idea to split these cases into multiple features. Using a string split it is possible to verify that the maximum number of categories in either feature is two. This anomaly initially points to splitting these features into 4 features. It is also intuitive that the subjects will give a good result if aggregated into a single feature.

2.3 Exploratory Visualization

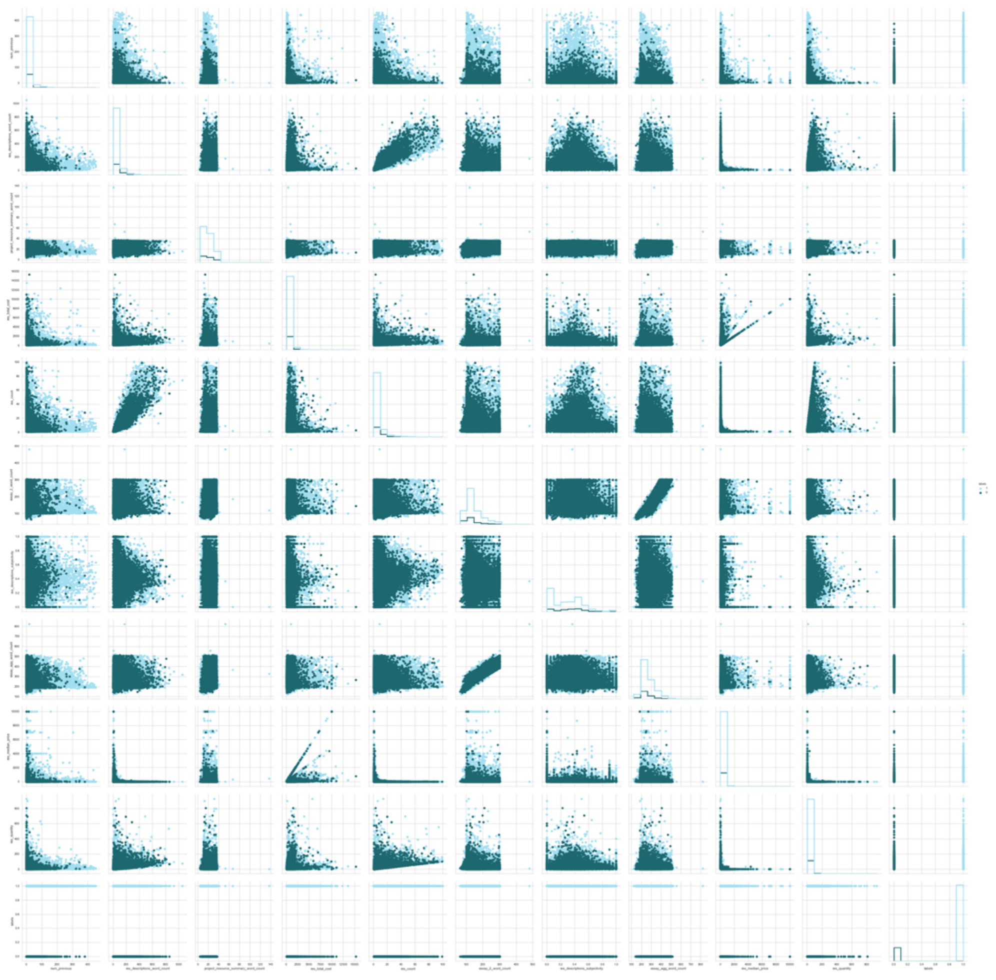
Exploratory Data Analysis (EDA) is frequently augmented by visualizations [4]. This gives an intuitive understanding of the data beyond summary statistics [1]. Most of the dataset contains categorical data, so the standard summary statistics of mean, quartiles, and outliers does not really apply. The most important factor here are what values are missing. This chart shows a high-level view of the entire training dataset, with a single horizontal line representing a filled data point. The white space present represents null values or NaNs.



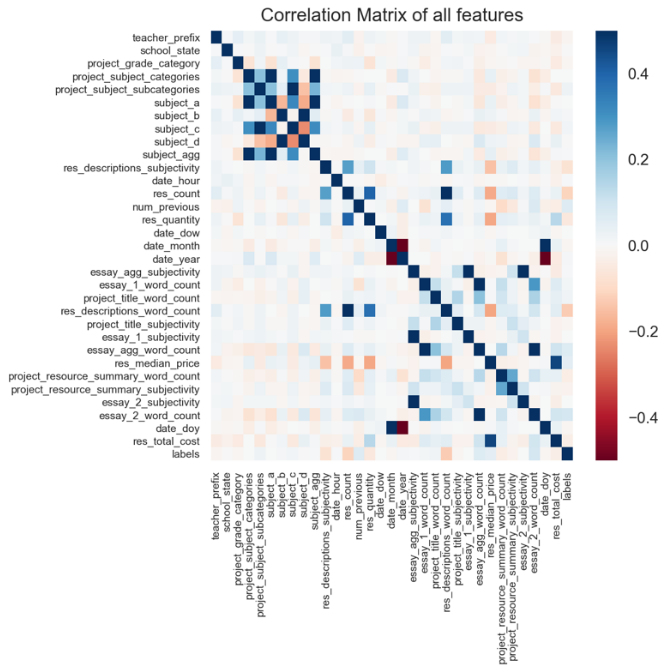
It is helpful when visualizing to dive into one feature in particular, or to do so repeatedly with all features. For this project, the following histogram was created for each feature to help determine if a particular state, etc. had a higher chance of getting accepted. This chart shows the counts of the teacher_prefix categories for accepted and rejected applications. While it is clear that more females submit, there is only a slight difference in the calculated rate of acceptance between Mr. (84.2%) and Mrs.+Ms. (84.9%).



After doing some feature engineering and creating numerical values from text and encodings it is possible to probe the numerical data. Pairwise scatterplots are a great way to get a feel for numerical data quickly. They allow correlations to be spotted between variables as well as with the class labels. In this plot the right column and bottom row represent the class labels. The diagonal contains univariate histograms colored by the class value. The upper and lower off diagonal sections of the matrix are composed of bivariate scatter plots of the respective features.



Unfortunately, this kind of plot is only useful for ‘eyeballing’ the data. To properly understand the inter feature correlation, a correlation matrix must be computed. These are also excellent to visualize as below.



This dataset is difficult, as it is clear that no features are highly correlated with the class labels. Only two features had a correlation magnitude higher than 0.3, res_description_word_count and res_count. This is disappointing as it means our algorithms will have to work harder to predict correct values and that more work will need to be done with the text data to extract useable information.

2.4 Algorithms and Techniques

There are three classifiers tested for use with both the numerical/categorical data and the text data. They are stochastic gradient descent, logistic regression, and support vector machine. The text data is converted to numerical features called term frequency inverse document frequency (TF-IDF).

Stochastic gradient descent (SGD) is performed by calculating the gradient of the current position, moving towards the largest negative gradient, and iterating that process until a minimum is found. SGD is sensitive to feature scaling, so it is appropriate to scale all feature values to a range such as [0,1] before training/testing the model. Logistic regression (LR) is actually a classifier in this case, where a sigmoid curve is applied to give the probability of a given point being in one of two binary classes. Overfitting is an issue with this classifier, as with the others, if there are too many features present. Support vector machines (SVM) operate by finding a plane that is furthest from the nearest point from either class label.

There are some justifications for choosing these models. The problem is binary classification, and these three models perform well on that problem. SGD works particularly well for large sample sizes (greater than 100,000) which this dataset has. LR provides intuitive probabilities for the binary classification, which is the expected output for this problem. SVM excels at separating highly dimensional data, and this dataset has dozens of features.

The dataset will be manipulated to fit into these models. All categorical features will be encoded to a one-hot matrix, where each possible category gets its own column filled with zeros but is one where the original feature matches that column. Numerical data will be scaled to a given range, such as [0,1]. Text data will be normalized, tokenized, vectorized, then counted to give numerical TF-IDF values.

TF-IDF is a method used to weight words which occur frequently in a document, tempered by the amount that a word appears in all the documents. This protects the values from being affected more strongly by words that appear more often in general. The term frequency is simply a count of times the word appears in a given document, but this can be skewed as some documents are much longer than others. Therefore, it is good practice to divide the term frequency by the document length. This term frequency is multiplied by the log of the quotient of the total number of documents divided by the number of documents with the particular term in it. This offsets the weight if it appears to frequently in all documents. By converting the text to numerical data, the normal classifiers are able to fit and predict on this information in the same manner as the rest of the dataset.

To get the best of the text and numerical data, these will be predicted on separately and the results will be a custom ensemble approach. The probabilities of each class are averaged to produce a single voted set of probabilities for submission to Kaggle.

2.5 Benchmark

The benchmark to use depends on the class balance, and from the training data it can be shown that the class balance is 84.76% approved and 15.24% denied. This means the zero rule for this problem is a baseline random model that randomly approve applications 84.76% of the time. Using this dummy classifier with kfold cross validation, the AUC scores range tightly around 50%. This means that if the predictive model performs well above 50% then it is doing well. If it performs under then there is a problem.

3 Methodology

3.1 Data Preprocessing

The data is read in, cleaned, and the data types are converted to the appropriate types. For example, all escaped characters in text strings are removed, and integers are converted from strings to int32. There are several steps of the basic data cleaning and feature engineering outlined in the next sections.

3.1.1 Numerical Feature Engineering

The questions posed in section 2.2 are answered here with feature engineering. This is the method of taking the few features provided in the original dataset and combining them or manipulating them to create new features.

The first of these involves the resources data. There are many lines per one application in the application data, so the resources data needs to be reduced to one line per application in order to be joined. First the quantity and unit price of each item are multiplied to get a total cost of each requested item. Then the resources related to a single application are grouped, the total cost is summed, and the text descriptions are joined into a single feature separated by commas. This new table can easily be joined to the application data table on the application ID.

The timestamps are further broken down to meaningful parts, such as day of week, day of year, month, hour of day, and year. This allows insight into when successful applications are submitted.

3.1.2 Handling Text

The resources dataset had some text data which may be useful. It contained one description for each record which was akin to a description that would exist on a purchase order, such as this example:

Dixon Ticonderoga Wood-Cased #2 HB Pencils, Box of 96, Yellow (13872)

For applications that had multiple resources requested, the descriptions were aggregated into a single record, separated by a comma. The new feature collecting all descriptions is title res_descriptions. This would allow the possibility of predicting on this row although not much information was expected to be gained from this feature. The applications dataset contained six text features that had potential for natural language processing (NLP). They are project_title, project_essay_1-4, project_resource_summary, and res_descriptions.

According to the explanation in section XYZ, records with four essays were aggregated into two. The first and second essays were combined into essay_1 while the third and fourth were combined into essay_2. If the record had 2 essays they were mapped directly, the first to essay_1 and the second to essay_2. The essays are also aggregated into a single feature, essay_agg, along with the project_title to make a single blob of the entire submitted essay text. It is possible for this aggregated essay to yield good separation between records and possibly have high correlation with the class labels.

After the pivoted resources dataset is joined to the applications dataset it is possible to separate out the text data and perform NLP actions on it. First all NaN or null values in the text data are filled with ‘’, meaning an empty string. Then all escaped characters such as “\n” or “\r” are removed. Next all surrounding whitespace is stripped from each value of each feature. The project_subject_categories and project_subject_subcategories features are split into 4 new features: subject_a, subject_b, subject_c, and subject_d. Perhaps these features will yield more information if concatenated into a single feature, so they are joined in a new feature subject_agg.

Word count is extracted by tokenizing the strings in each essay and measuring the length of the resulting list. This is performed for all text features.

Subjectivity, polarity, and sentiment are computed on the non-normalized text. These produce numerical results which have meaningful ranges. The results are joined to the numerical data. The TextBlob library is able to perform these calculations directly using pretrained vocabularies and the VADER method [4].

Text normalization is a critical component of the NLP approach. The normalization section takes the longest time to run, even longer than training models. For each text value, the following steps are taken. All characters are converted to lowercase. All stopwords from the Natural Language Toolkit (NLTK) are removed, as well as any punctuation. Words are stemmed, split into single word members of a list, then joined with a single space between all words. Stemming is a process where words are reduced to their base form, allowing for more universal matching and less unique variants of a given word. For example, the words “reading”, “reader”, “readers”, and “read” may all be reduced to the root “read”. Some stemmed words are not actual words themselves (such as “forgiv”) but that is irrelevant as far as models are concerned. These normalized values are created in new features and given the suffix “_norm” to preserve the original text features in addition to the new normalized features. This allows flexibility when evaluating models.

3.2 Implementation

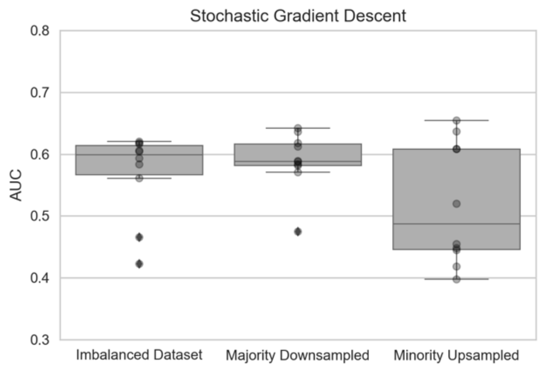
The entire project was completed using Python 3 and basic libraries including pandas, numpy, scikit-learn, matplotlib, seaborn, and os. The data was processed as described in the above sections. The model selection process involved trying several models with varied dataset resampling methods with varied model parameters. The optimal models were selected and trained on the preprocessed data. For this problem, the labeled training set and unlabeled test dataset were already split. The training process involved initializing a classifier to the optimal parameters, setting a random state initial value for repeatable results, then fitting the model with data and the corresponding labels. The labels were split from the test data during the preprocessing stage. The random forest classifier was selected for modeling the numerical data, while the linear SVM was selected for modeling the text based features. Once these models are trained, the predict method was called on the unlabeled test data to generate classification predictions. The probability of each record being in either class is given as the ultimate output for submission to Kaggle.

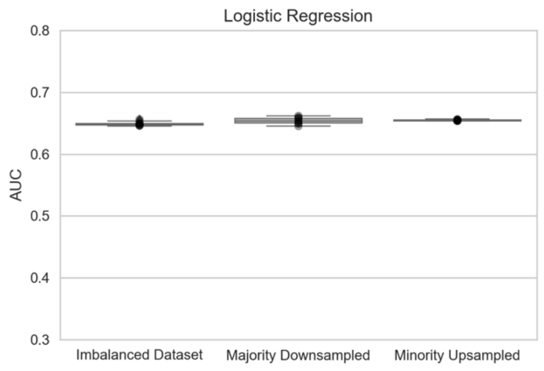
Cross validation is critical in order to properly gauge a model’s performance on training data. Since the training data is all we have, it must be split into training and test chunks to measure performance. It is better to do split and test multiple times with different chunks of data to get more accurate results. This is known as k-fold cross validation, where k is the number of times the split is performed. The dataset had a large class imbalance, with 84% of applications being approved. This means that a more refined method of cross validation is needed to respect the class imbalance. That method is stratified cross validation, where a shuffled sampling of the data is taken, respecting the class proportions. This was done for each model and each dataset and was repeated to compute more accurate scores. Instead of using 10 folds, this approach used 5 folds repeated twice. This allows for larger chunks of data to be used with less overlap, but still has the effect of larger score sampling leading to accurate results

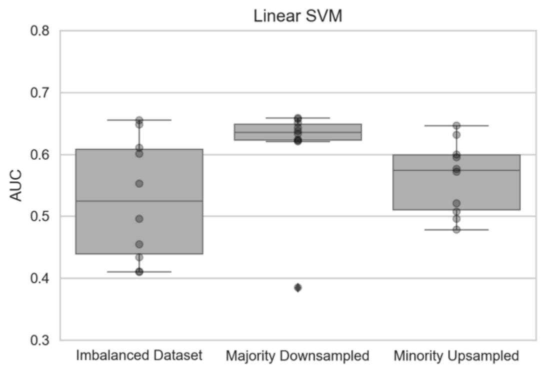
Imbalanced datasets need to be handled appropriately before being used in models, otherwise the model is at risk of having skewed results. There are many methods to deal with imbalanced dataset, but the most straightforward is resampling the data. In this project the minority class was upsampled to match the number of samples of the majority class. Also, in different data frames, the majority class was downsampled to match the number of samples in the minority class.

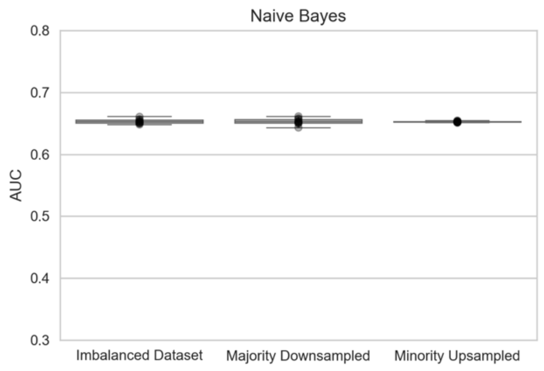
In order to properly select a model, several were tested. These were SGD, LR, SVM, and naïve Bayes (NB). Each classifier was tested with each dataset sampling scheme to identify both the best sampling method and the best algorithm. The results of these trials are in the table and charts below.

Classifier	Dataset Balance	AUC (median)	STD of AUC	Runtime (sec)
SGD	Imabalanced	0.599	0.065	1.22
SGD	Downsampled	0.588	0.045	0.27
SGD	Upsampled	0.487	0.094	2.18
Logistic Reg	Imabalanced	0.649	0.003	5.92
Logistic Reg	Downsampled	0.654	0.005	0.98
Logistic Reg	Upsampled	0.654	0.001	5.61
Linear SVM	Imabalanced	0.524	0.093	150.37
Linear SVM	Downsampled	0.635	0.077	38.75
Linear SVM	Upsampled	0.574	0.055	304.92
Naïve Bayes	Imabalanced	0.653	0.003	0.67
Naïve Bayes	Downsampled	0.652	0.005	0.15
Naïve Bayes	Upsampled	0.652	0.001	0.97









One caveat with the SVM was that the probabilities are not computed directly. A wrapper class was needed to perform this. The results, along with intuition point to logistic regression with a minority upsampled dataset as the model to use for this particular dataset. This makes sense as LR is the most direct model tested. As a result of these tests, SVM is selected for use with the text based features.

An interesting result occurred when submitting probability predictions to Kaggle, however. During the first pass of this dataset a random forest was used on the numerical data. After updating the code to test the four chosen models, and selecting logistic regression, the Kaggle score decreased. This is a good example of how cross validation with AUC score is not always an accurate predictor of performance, it only tells you how well the test data performed.

Several iterations of classifiers were used and submitted to try to improv the Kaggle score, but ultimately the best combination was random forest on the numerical data with support vector machines on the TF-IDF vectorized/normalized text data. All of the functions were implemented using the scikit-learn library. SGD classifiers used the log loss function to allow for probability of class prediction, and the random forest was limited to a max depth of 7 with 11 trees. All classifiers and cross validation was initialized with a random state to ensure the results were reproducible.

3.3 Refinement

Each classifier used from scikit-learn had hyperparameters adjusted to maximize the cross validation AUC scores. Some of these were the maximum number of iterations for the random tree classifier, the ngram size of the TF-IDF vectorizer, and the constraints of the repeated stratified k fold cross validation scorer.

The grid search method is powerful for finding the optimal family of models in a given problem. Lists of values for various parameters are created, then iterated to score each combination of those parameters. For the random forest classifier, the maximum tree depth (max_depth) used the values 3, 5, 7, 10, and infinity. Simultaneously, the number of trees used in the forest (n_estimators) was changed to 3, 5, 7, and 10. The optimal combination of these parameters turned out to be 10 estimators and infinite layers. The parameters for SVM were also set to the default of squared hinge loss. This sort of tuning is critical to identify improvement opportunities for various models.

The best result, however, was achieved with the initial conditions described in the previous section, and further refinements did not improve the score. In fact, some refinements made the AUC score significantly worse. Regardless, it is important to adjust the parameters on sensible ranges to achieve optimal results, and this was done with the parameters mentioned.

4 Results

4.1 Model Evaluation and Validation

The final model combined a random forest on numerical data with stochastic gradient descent on the TF-IDF text features. The output of each model is the probability of being in either the accepted or rejected class. For example, one test record may have a 5% probability of being rejected and a 95% chance of being accepted. This form allows for more nuanced scoring of the data, and is required for submission to Kaggle. The scores used are area under the curve, the same as used in this project's cross validation step.

The parameters used for the final models were determined using the grid search methodology described in section 3.3. For the random forest model the parameters were all set to default with a random state initialized. The default number of estimators is 10, the maximum depth is infinite, and the minimum samples to split is 2. For the SVM the default estimators are used as well, with a squared hinge loss function.

The top 70 scores on Kaggle for this problem achieved an AUC of approximately 82%. The winner of the competition achieved 82.8%. The results from the model in this report produced an AUC of 75.7%. This is a very satisfactory result, as the highest AUC achieved during cross validation model selection was only 65.4%.

The small changes in results from different random state initializations was negligible, and this model could well be used in a real world application. It is straightforward and simple to debug allowing for wide application. The model generalized well to the test data and provided a metric in the same range as anticipated results, if not higher.

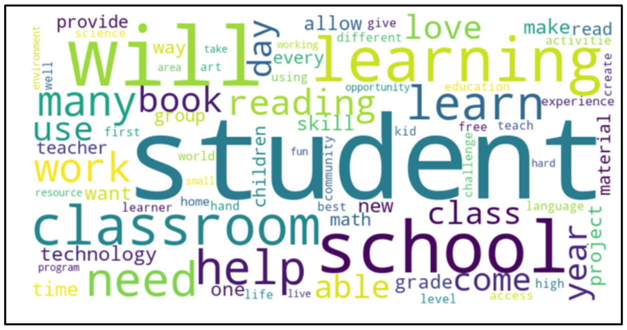
4.2 Justification

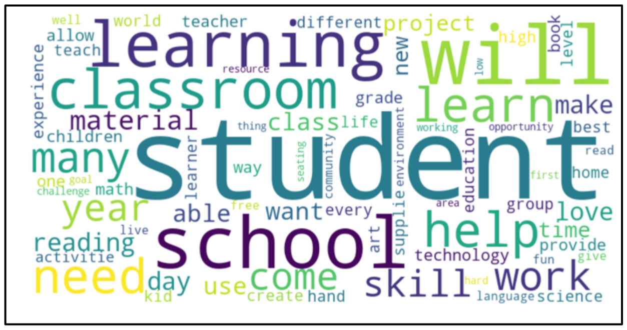
The final model score as provided by Kaggle was 0.757. The baseline model achieved a 0.5, so this is a marked improvement. The results were satisfactory and solved the problem enough that basic classification can be performed on the applications before volunteers must read them. The fact that humans are still in the loop makes the threshold less critical than in say a self-driving car problem. As long as a large amount of the obviously classifiable applications are sorted by machine, DonorsChoose achieves its goal.

5 Conclusion

5.1 Free-Form Visualization

Word clouds are a visual representation of word frequency. These are a favorite visualization among NLP practitioners, as they give a quick sense of what the underlying text contains. The first word cloud below is for the essay_agg features where the application was accepted. The second word cloud is for the applications that were rejected.





While these are interesting visualizations, it is clear that the primary words occur with approximately the same frequency in both classes. Words like "student", "classroom", and "learning" appear with similar weight. This highlights the primary difficulty with the classification problem using this dataset. The text contents of the applications are very similar, which makes extracting accurate predictions difficult. Words like "skill" appear to be present in different weights in the two classes, and further analysis may show that the numerical frequency is indeed greater in one over the other, allowing for some meaningful feature to be used there.

The non-normalized text was used to produce the word clouds displayed, but word clouds were also created for the normalized text (lower cased, stemmed, no stop words, etc.). The results were similar, with the same words appearing in both word clouds.

5.2 Reflection

In conclusion the results using aggressive feature engineering and the straightforward application of well-known classifiers gave a satisfying result. The test data was scored on Kaggle and resulted in a 0.757 on the leaderboard. This is not bad considering the top score achieved 0.828 with a highly customized ensemble method containing several stacked algorithms. The direct approaches in this report show that great things are possible with machine learning if it is applied in the right way.

The end-to-end summary of this project has many pieces. The data was collected, analyzed in its raw form, processed, cleaned, summarized, and combined. Several new features were extracted, and categorical data was converted to a one-hot representation. The imbalanced dataset was resampled in two ways to balance the class representation. Several classification algorithms were selected, tested, and had parameters tweaked to determine the optimal algorithms to solve this problem. The selected classifiers with optimal parameters were fit to the processed data and corresponding labels. Predictions were made using the unlabeled test data, and probability of class membership was produced as an output. This probability listing was uploaded to Kaggle, and the AUC score was returned for that dataset. This was iterated until the best score was received.

Some challenges with this project involved cleaning the text features. There are several nuances to the way each teacher writes their essays, titles, and resource descriptions yet these need to be normalized somewhat for a model to make use of them. There is a balance involved with text normalization, if too much is done then nuance will be lost, possibly diluting the deltas between the essays. For example, lowercasing everything and removing punctuation leaves the primary text intact, but stemming and selecting only n-grams change the data and the interpretation of it.

5.3 Improvement

As always with machine learning problems, improvements can continuously be made. One area that stands to receive large information gain with small improvements is the text analysis. An option for improvement is to use neural networks via Keras with a TensorFlow backend on the text data, as that sort of complex approach is well suited to the text classification problem.

Future steps for this project involve tagging the parts of speech as noun, adjective, etc. to refine the text modeling. It seems completely possible to increase the performance of the basic predictors by refining the input text itself. More nuanced sentiment analysis could be possible if only certain word types were used. It would be interesting to also do topic modelling, find the strongest topics in each class, and only perform sentiment analysis on the sentences surrounding those terms. There is always room for improvement, which is what makes these kinds of problems exciting.

6 References

[1] Anscombe, F. J. (1973). "Graphs in Statistical Analysis". American Statistician. 27 (1): 17–21. doi:10.1080/00031305.1973.10478966. JSTOR 2682899.

[2] DonorsChoose.org. (2018 April) DonorsChoose.org Application Screening. Retrieved 2018 June from https://www.kaggle.com/c/donorschoose-application-screening/data

[3] Fatourechi, R. K. Ward, S. G. Mason, J. Huggins, A. Schlögl and G. E. Birch, "Comparison of Evaluation Metrics in Classification Applications with Imbalanced Datasets," 2008 Seventh International Conference on Machine Learning and Applications, San Diego, CA, 2008, pp. 777-782.

[4] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[5] Tukey, John Wilder (1977). Exploratory Data Analysis. Addison-Wesley.

[6] Zweig MH, Campbell G (1993) Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. Clinical Chemistry 39:561-577.