DonorsChoose.org Application Screening

Team Name: Spartans
Sannisth Soni, Divyank Shukla, Manasi Vinay Thakur
(github link)

Chapter.1 Introduction

Motivation

DonorsChoose.org gives financial grants to classrooms belonging to public schools for much needed equipments and infrastructure needs of students till K12 level. They have hundreds and thousands of applications each year and they have been reviewing each of them manually. Currently, they need large number of volunteers to review the applications and decide if they should get the financial grant.

As per the statistics, it is expected that the Donors.org is going to receive 500,000 project proposals.As a result there are three main problems which need to be solved.

1. How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly as possible
2. How to increase the consistency of project vetting across different volunteers to improve the experience of teachers
3. How to focus volunteer time on the applications that need the most assistance

Objective

In the project, DonorsChoose.org challenges the Kaggle machine learning community to build a machine learning model that provide accurate prediction of the probability of the approval of the project. This machine learning model will help more teachers get funded more quickly, and with less cost to DonorsChoose.org, allowing them to channel even more funding directly to classrooms across the country.
Our main objective is to predict whether or not a DonorsChoose.org project proposal submitted by the teacher from various schools will be approved, using the text of project descriptions as well as additional metadata. This includes building a machine learning model using the data mining and machine learning methods and classification algorithms, resulting in accurate prediction which will help the organization better decision. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

Chapter 2

System Design & Implementation details

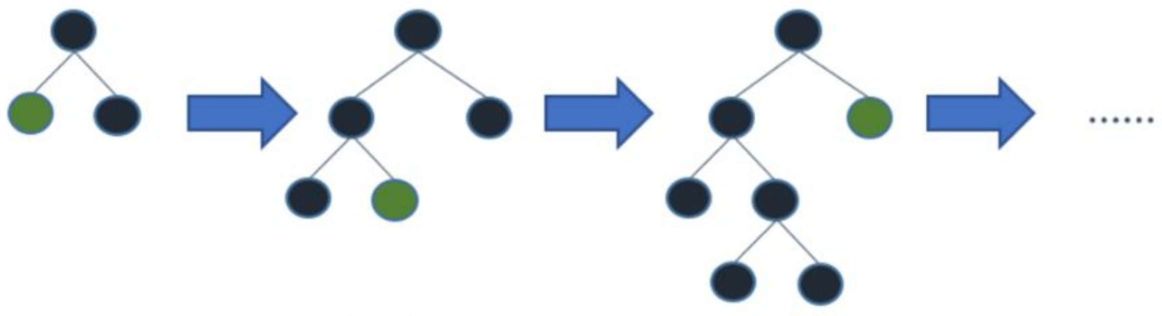
Algorithms Considered:

- **Logistic Regression**

The main reason we considered a regression algorithm is that it is aimed at improving the relationship between variables that is iteratively refined using a measure of error in the predictions made by the model. This means that the model is continuously improved based on growing dataset. We decided to choose linear regression algorithm but we found out that the predictions from linear regression can be any real number which is not what we want.
So we considered to Logistic regression algorithm. This worked in our favour because it did exactly what we wanted. Logistic regression algorithm gives us probability of classification of output. This means it doesn't classify the output whether it will occur or not but with what probabilities will it occur. This allows to also set a threshold above which we can classify the output as successful. Hence, it gives us more control over the output.
Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function.

- **LightGBM**

Light GBM is a gradient boosting framework that uses tree based learning algorithm. Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.


Leaf-wise tree growth

The size of data is increasing day by day and it is becoming difficult for traditional data science algorithms to give faster results. Light GBM is prefixed as 'Light' because of its high speed. Light GBM can handle the large size of data and takes lower memory to run. Another reason of why Light GBM is popular is because it focuses on accuracy of results. LGBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development.
Light GBM is sensitive to overfitting and can easily overfit small data. There is no threshold on the number of rows but it is recommended only for data with 10,000+ rows which suits our requirements.

- **XGBoost**

Unlike CatBoost or LGBM, XGBoost cannot handle categorical features by itself, it only accepts numerical values similar to Random Forest. Therefore one has to perform various encodings like label encoding, mean encoding or one-hot encoding before supplying categorical data to XGBoost. Its name stands for eXtreme Gradient Boosting. The implementation of XGBoost offers several advanced features for model tuning, computing environments and algorithm enhancement. It is capable of performing the three main forms of gradient boosting (Gradient Boosting (GB), Stochastic GB and Regularized GB) and it is robust enough to support fine tuning and addition of regularization parameters

Technologies & Tools

- Jupyter Notebook
- Numpy
- Scikit-learn
- Pandas
- Python 3

System Design Architecture

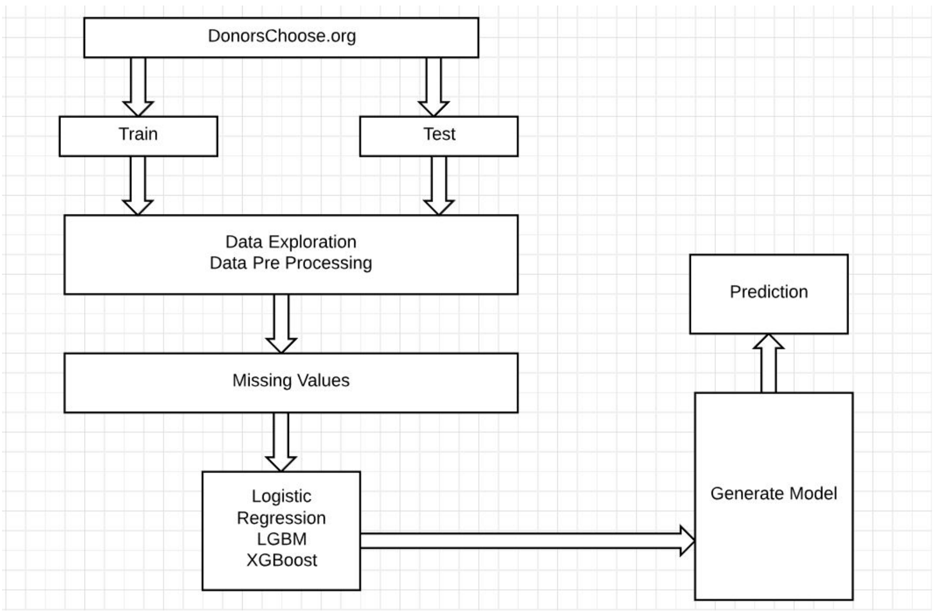


Figure 1: System Architecture

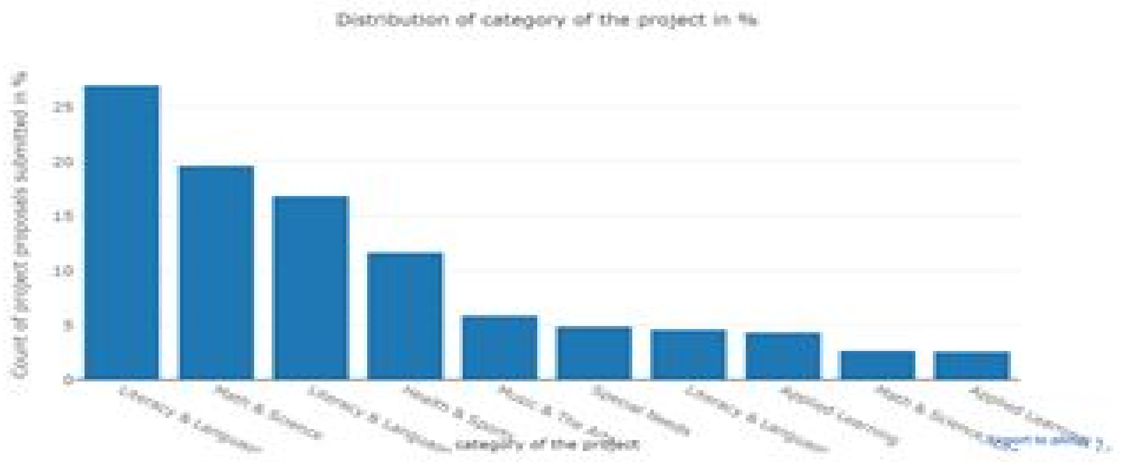
- First, we did data exploration which helped us to get clear insights into our entire dataset

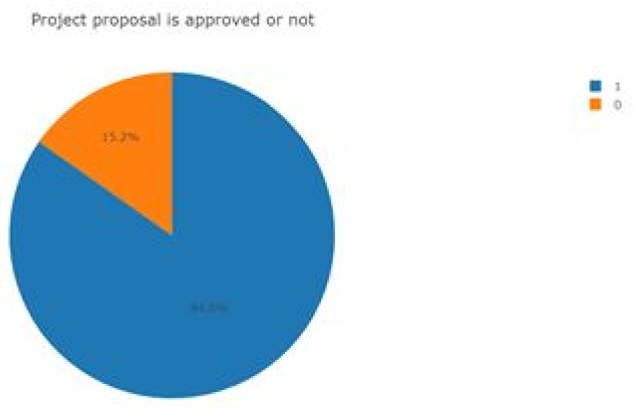
- In prex`processing we have removed so unwanted columns as they might not be good contributors towards making accurate predictions

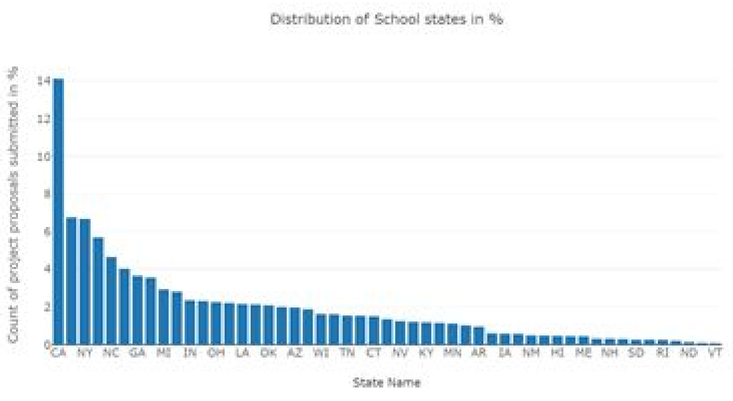
We have used three different classification techniques for making our classification model

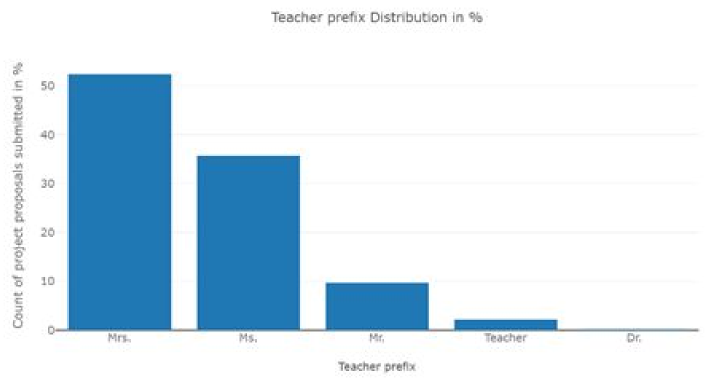
1. Logistic Regression
2. Light GBM
3. XGBoost

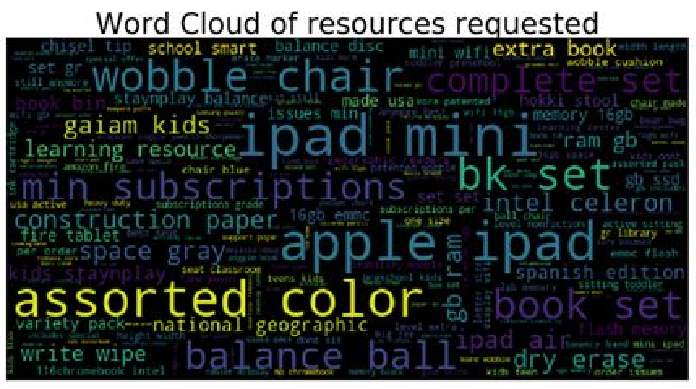
Data Visualization

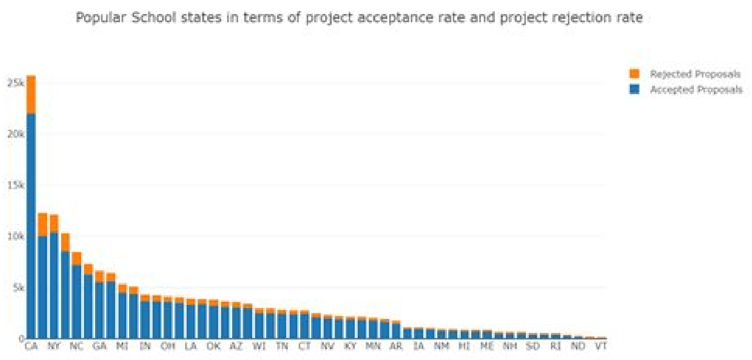












Chapter 3
Experimentation / Proof of concept evaluation

Dataset Used:

Name	DonorsChoose.org Application Screening
Source	https://www.kaggle.com/c/donorschoose-application-screening/data
Type of Data	Comma Separated Values (CSV) files
Size of Data	Train.csv (336.3 MB) Test.csv (143.8 MB) Resources (127.5 MB)

# of instances	Train.csv (182080) Test.csv (78035) Resources.csv (1554258)

Preprocessing Steps:

1. The many instances of resources.csv were joined by the Id and the total of each recurring instance was calculated and later joined with test and train dataset.

```
#Added a new column for total price per row
train_resource['value'] = train_resource['quantity'] * train_resource['price']
```

```
train_resource_grouped = train_resource.groupby(['id'], as_index=False)[['value']].sum()
test_resource_grouped = train_resource.groupby(['id'], as_index=False)[['value']].sum()
```

```
#Train & Test dataset - resource part added
train_joined = pd.merge(train,train_resource_grouped,on='id')
test_joined = pd.merge(test,test_resource_grouped,on='id')
```

2. Replaces missing essay 3 and 4 with empty string and then concatenating them with essay 1 and 2 respectively. Effectively leaving us with 2 essays for each row.

```
#Replacing NaN with empty string
#Addding essay 1 & 3
#Adding essay 2 & 4
#Effectively every row will have 2 essay columns
train_joined['projectessay_1_3'] = train_joined['project_essay_1']+
train_joined['project_essay_3'].replace(np.nan, '', regex=True)

train_joined['projectessay_2_4'] = train_joined['project_essay_2']+
train_joined['project_essay_4'].replace(np.nan, '', regex=True)

test_joined['projectessay_1_3'] = test_joined['project_essay_1']+
test_joined['project_essay_3'].replace(np.nan, '', regex=True)

test_joined['projectessay_2_4'] = test_joined['project_essay_2']+
test_joined['project_essay_4'].replace(np.nan, '', regex=True)
```

3. Dividing dataset into columns with text, without text and categories and treating them differently.

```
#Classifying acc to the type of cols
categorical_columns = ['teacher_prefix','school_state', 'project_grade_category',
                       'project_subject_categories', 'project_subject_subcategories']
non_cat_columns = ["project_submitted_datetime",
                   "teacher_number_of_previously_posted_projects","value"]
text_columns = ["project_title","project_resource_summary",
                "projectessay_1_3","projectessay_2_4"]
```

```
# y (aka target var) is train['project_is_approved']
train_cat = train_joined[categorical_columns]
train_non_cat = train_joined[non_cat_columns]
train_text = train_joined[text_columns]
test_cat = test_joined[categorical_columns]
test_non_cat = test_joined[non_cat_columns]
test_text = test_joined[text_columns]
y = train_joined['project_is_approved']
```

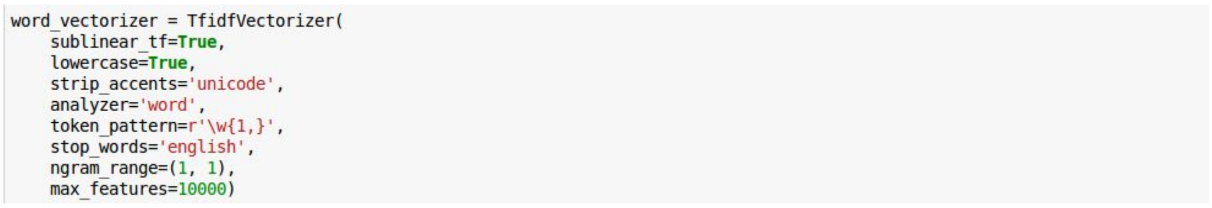
4. Getting dummy/numerical values for categorical values.

```
train_cat = pd.get_dummies(train_cat)
test_cat = pd.get_dummies(test_cat)
```

5. Doing feature engineering : extracting month and year from time of submission

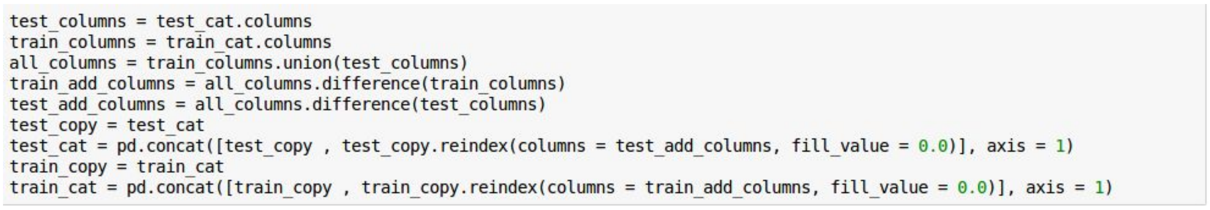
```
#Convert column to datetime column
train_non_cat["project_submitted_datetime"] = pd.to_datetime(train_non_cat["project_submitted_datetime"])
test_non_cat["project_submitted_datetime"] = pd.to_datetime(test_non_cat["project_submitted_datetime"])
```

6. Converting the text columns into CSR matrices by TfidVectorizer



```
word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    lowercase=True,
    strip_accents='unicode',
    analyzer='word',
    token_pattern=r'\w{1,}',
    stop_words='english',
    ngram_range=(1, 1),
    max_features=10000)
```

7. Finally columns are merged and made ready to fit into prediction algorithms



```
test_columns = test_cat.columns
train_columns = train_cat.columns
all_columns = train_columns.union(test_columns)
train_add_columns = all_columns.difference(train_columns)
test_add_columns = all_columns.difference(test_columns)
test_copy = test_cat
test_cat = pd.concat([test_copy , test_copy.reindex(columns = test_add_columns, fill_value = 0.0)], axis = 1)
train_copy = train_cat
train_cat = pd.concat([train_copy , train_copy.reindex(columns = train_add_columns, fill_value = 0.0)], axis = 1)
```

Methodology Followed:

Size of Trainset : 182080 * 16
Size of Testset : 78035 * 15

A predictive model was made using the training dataset and tested for testing dataset. The training dataset has information of only if the project was approved or not. The predictive model returned the probability of approval given other features.

Comparison:

Algorithm	**Time Taken on local machine**
LightGBM	**30 min 24s**
LogisticRegression	**1 min 25s**
XGBoost	**10 min 43 s**
SGDClassifier	**2.15 s**
RandomForestClassifier	**3.42 s**

Analysis:

- LightGBM worked the best from among the algorithms selected for this project.
- RandomForestClassifier gave the least accuracy from among the algorithms selected for this project.

- Theoretically, XGBoost runs slower than lightGBM but gives same accuracy, but in our case XGBoost, is ran with default parameters but lightGBM is ran with custom parameters.
- SGDClassifier took the least time and gave better accuracy than RandomForestClassifier.

Chapter. 4
Discussion & Conclusion

Decisions Made

- We decided merge some of the columns in our dataset to handle the missing values and drop the the original columns
- We converted the columns which contained categorical values into dummy columns by using the pandas get_dummies function and then dropped the original columns
- We decided to divide the dataset into three parts categorical, non-categorical and text columns after the preprocessing

Difficulties Faced:

- We were unsure about how to handle essays submitted by the schools along with proposal.
- Unlike reviews or comments, we cannot classify essays into positive or negative and hence the NLP libraries that we encountered weren't useful.
- The dataset is highly imbalanced as there are more instances of approved proposals as compared to refused ones.
- The train data didn't have probability, which was supposed to be predicted by the model and hence we couldn't find accuracy locally.

Things that worked Well

- Data preprocessing and Data exploration gave us much better idea and helped us in understanding the data more
- Converting the categorical value into dummy values helped us in generating better results
- Dividing the dataset into three categories helped us in process the data efficiently

Things that didn't work well

- Our first naive assumption was long essays imply more sincere proposals and hence we replaced essays text with it's length and it didn't give us good results.
- VaderSentimentAnalysis gave us a positive, negative and neutral score of essay which we replaced our essays with and results were bad.

Conclusion

- For the data we had, LightGBM gave the best results.

- Feature Engineering new features based on understanding of data during exploration

Chapter 5
Project Plan / Task Distribution

Task	Responsibility
Problem statement selection	**All**
EDA	**All**
Research on algorithms	**All**
Data preprocessing	**Manasi , Sannisth**
Logistic Regression	**Divyank, Manasi**
LightGBM	**Sannisth, Divyank**
Random Forest	**Manasi, Divyank**
SGDCLassifier	**Sannisth, Manasi**
XGBoost	**Divyank, Sannisth**
Documentation	**All**

References:

- http://www.numpy.org/devdocs/reference/
- https://pandas.pydata.org/pandas-docs/stable/
- https://stackoverflow.com/questions/32565829/simple-way-to-measure-cell-execution-time-in-ipython-notebook
- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.groupby.html
- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.merge.html
- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.replace.html

- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.drop.html
- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html
- http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
- http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
- https://docs.scipy.org/doc/numpy-1.14.0/reference/generated/numpy.hstack.html
- https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html
- http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
- http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://github.com/Microsoft/LightGBM
- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_csv.html
- https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf