BUDT758T Final Group Project

Group 9: Arjan Singh, Christopher Iacono, Eunisa Lu, Keerthana Pramudi Suresh, Marco Sesay

Index

Executive Summary

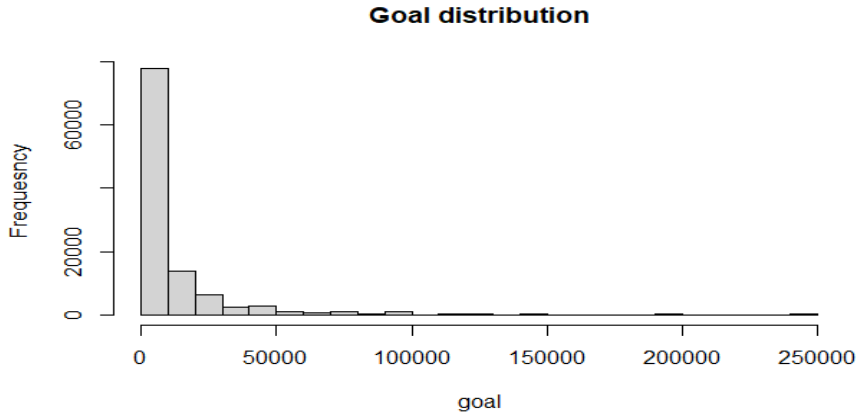
Kickstarter.com is a popular crowdfunding website used by individuals and organizations in need of financial support. By creating a custom page with an outline of their project and funding target, fundraisers are able to gather the support that they need to to achieve their goals. Kickstarter differs from other crowdfunding platforms due to its "all or nothing" funding policy. In short, only projects that meet or surpass their target are able to keep all of the money raised. If a project fails to reach its target, the creators are unable to withdraw any donations that went towards their goal.

The goal of this project was to predict one of three target variables (success, big hit, or backers count) based on a Kickstarter project dataset. The kickstarter dataset used in this project is composed of roughly 100,000 projects with deadlines ranging from 2009 - 2014. By utilizing various data mining and analysis techniques from BUDT758T, our group was able to predict the success of a project on Kickstarter.com.

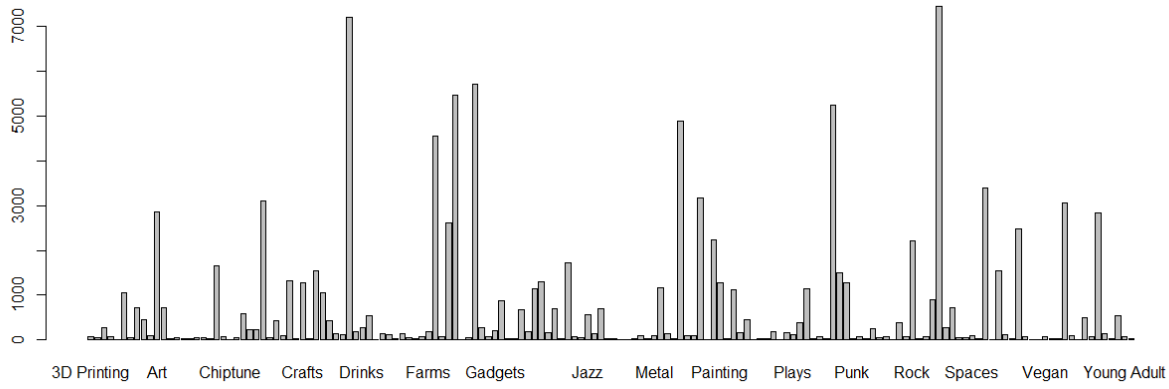
After testing various predictive models and performing feature engineering, our group determined that the XGBoost model was the most accurate when trying to predict the success of a Kickstarter project. From a financial perspective, the XGBoost is the ideal model due its high accuracy when predicting the success of a project. By selecting variables such as duration, goal size, launch year, creation month, state, tag frequency, and project picture, we were able to predict the success rate of a project on Kickstarter with a 76% accuracy on the test set. Adjusting the variables listed above, we can help fundraisers improve their chances of being successful and reaching their target goal.

Data Exploration

The kickstarter dataset includes 58 columns with different data types ranging from numeric, double, character, data time as well long unstructured texts and lists. Starting with numerical/double columns like *goal* for example, we found that the data was highly right skewed with high outlier values.



Though it is not clear from this histogram, the interquartile range falls between 2000 and 14000. This seems to be the situation for multiple other numerical/double data.
We fitted a logistic regression model with just the numerical and categorical data to get an understanding of which variables are better correlated with our target variable - *success*.
Columns such as *goal, category_parent, category_name, location_type*, data on different aspects of *blurb* text, demographics on creator or projects, etc.
Furthermore, some factor variables had over 100 levels. An image snippet of the frequency counts for some of these can be found below.



Most with a frequency of over 1000 add the most value to the correlation summary extracted from a linear/logistic regression models.

Feature Engineering

Many columns were removed to do our initial data exploration so as to fit a regression model error free. In order to glean useful information from the remaining columns, preprocessing of the data is required.

Below is a table of all the final parameters used for the training and testing.

Parameter Name	Description
Existing Parameters	
goal	Fundraising goal in US $
location_type	Project location type
region	Project location region

category_parent	Main project category
category_name	Project subcategory
numfaces_project	Estimated number of faces in main project picture
numfaces_creator	Estimated number of faces in creator profile picture
male_project	Estimated number of male faces in main project picture
male_creator	Estimated number of male faces in creator profile picture
female_project	Estimated number of female faces in main profile picture
female_creator	Estimated number of female faces in creator profile picture
smiling_project	Estimated probability of at least one smiling face in main project picture (0-100)
minage_project	Estimated minimum age of faces in main project picture
maxage_creator	Estimated maximum age of faces in creator profile picture
isTextPic	Whether the main project picture is text or not
isLogoPic	Whether the main project picture is a logo or not
isCalendarPic	Whether the main project picture is a calendar or not
isDiagramPic	Whether the main project picture is a diagram or not
isShapePic	Whether the main project picture is a shape or not
num_words	Word count of the full project description text
avg_wordlengths	Average word length of the full project description text (# of characters)
contains_youtube	Whether the description contains a link to a YouTube video
sentence_counter	Number of sentences in the full project description text
avgsentencelength	Average length of sentences in the full project description text (number of words)
avgsyls	Average number of syllables per word in the full project description text
grade_level	Estimated Flesch Kincaid grade level of the description text
afinn_pos	Number of words in the description with an AFINN sentiment score of >= +3
afinn_neg	Number of words in the description with an AFINN sentiment score of <= -3

ADV	Count of adverbs in the description
NOUN	Count of nouns in the description
ADP	Count of adpositions in the description
PRT	Count of particles in the description
DET	Count of determiners in the description
PRON	Count of pronounds in the description
VERB	Count of verbs in the description
NUM	Count of numerals in the description
CONJ	Count of coordinating conjunctions in the description
ADJ	Count of adjectives in the description
New Parameters	
reward_sum	Sum of all rewards, in the reward_amounts list
reward_min	Minimum reward, in the reward_amounts list
reward_max	Maximum reward, in the reward_amounts list
reward_count	Count of all rewards, in the reward_amounts list
Top_Tag	First Tag from the tags field
creator_gender	Gender added from babynames.csv external data
date_diff	Difference in days between the launched_at and deadline date fields
date_diff_bin	Binned date_diff based on quartiles
goalsize	binning goals based on quartiles
launch_month	Month extracted from launched_at
launch_year	Year extracted from launched_at
deadline_month	Month extracted from deadline
deadline_year	Year extracted from deadline
created_month	Month extracted from createded_at

created_year	Year extracted from created_at
state	State extracted from location_slug
city_name	City extracted from location_slug
dow_avg	Average stock activity from external data dow10y_new.csv
unemployment_rates	unemployment rates by month from external data unemployment.csv
help	One of top 5 vocabs extracted from blurb
new	One of top 5 vocabs extracted from blurb
film	One of top 5 vocabs extracted from blurb
will	One of top 5 vocabs extracted from blurb
music	One of top 5 vocabs extracted from blurb

Date columns

Columns such as *created_at, deadline and launched_at* were string fields that needed to be converted to a date data type. After this conversion, one can extract month, year and perhaps take the difference between the launched_at and deadline and bin these differences to see if projects do better when under a shorter duration.

Extracting City and State information

The dataset came with a column that had both city and state details in it. However since the two values were merger as one long string, we used regular expressions and the string substitution functions like *gsub()* to extract and clean the values into two separate columns.

Handling Tags

We were given a column of auto generated tags for each project. However it is unknown whether the tags are generated based on the *blurb, captions* columns or some other point of reference. We extracted the first and top most tag from a list of tags assuming they were in the order of significance. Another split function with regular expressions extracted the desired tag while we dropped the original list.

Bag of Words

We tried extracting data from the blurb field. By lowercasing, tokenizing, removing numbers, punctuation, stopwords and finally stemming the tokens we are able to create repository of words. This repository can be pruned and converted into a Document Term Matrix. We extracted the top 5 most occurring terms and attached them to the kickstarter dataset.

Reward amounts

The *reward_amounts* column was a list of numbers in the form of strings. After separating the components of the list by the function *strsplit()* using `,` as split basis, a *type.convert()* function changed the data type to numeric/double and finally extract information usnig functions like *mean(), length(), sum(), min(), max()*.

Reduction of factor levels and binning

The models we used have restrictions on the number of levels a categorical variable can have. For this reason as well to remove those levels with less frequency counts, we bin those under a certain threshold. This is good way to handle outliers. As in the case of the *goal* column, binning based on ntile reduces the range and gives focus to frequency.

NA Engineering

A good number of columns had NAs and information that was not meaningful, this was especially so for categorical columns. To remove them one can either remove the entire row or impute with appropriate values. Removal of rows leads to loss of information in other columns. For this reason, we decided to impute values like ‘None’ for categorical nulls, 0 for nulls in rewards.

External Datasets

Adding external datasets is one way to improve models accuracy significantly. In our case, the external datasets needed considerable amount of cleaning - firstly, pivoting the data to a format that fits with the existing dataset, handing NAs, splitting and cleaning of cells to extract required information. Our Model improved but not by much. One must have relevant field knowledge to select appropriate external sources.

Dummifying Predictor variables

For certain models, like XGboost, trees and most importantly Neural Networks, input data must be of the form 1s and 0s, that is binary numerical values. Though these values are taken by other models like logistic regression, we have experienced that it takes more processing

to get through all the features generated by categorical values. (If you have many factor levels in your categorical columns).

Normalization

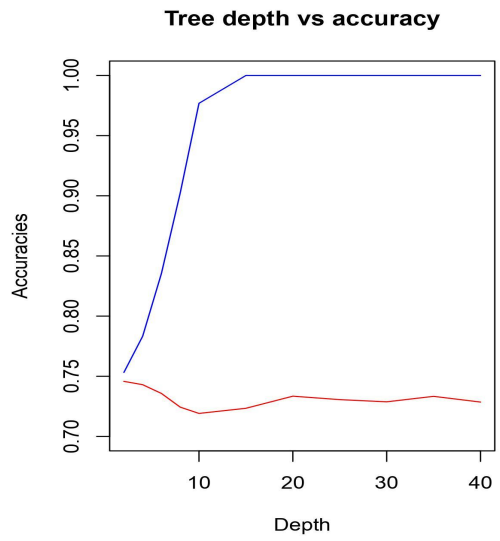
Since our data range was not uniform, we had to make a choice between normalization and standardization for some models. Since the data does not follow a gaussian distribution and we were using a classification Neural Net, we decided to go ahead with normalization. The specific method we chose was min-max normalization as we wanted our data to range from 0 to 1 with no negatives.

Validation and Model Evaluation Process

In our project, we analyzed various models such as logistic regression, XGBoost, Random Forest, Lasso, Tree, Ridge and MLP Neural networks. When compared between each other, different modeling procedures have different performance on the same data. While each model has its own set of benefits, we selected the XGBoost due to its high accuracy when compared to other models. Our model evaluation process was improved by performing cross validation. In cross validation, we split the training data into non overlapping folds, used each fold as the test set, and took an average of the generalization performance across folds. By performing cross validation, we were able to reduce the reliance of one test set.

Fitting Curve

Here is a fitting curve on max depth vs accuracy for the XGBoost model. Maximum depth specifies the number of splits the user requires at each level. This is a means of pruning the tree and one can observe the extreme case of over-fitting as the splits increase. With increasing depth, the training data starts to over fit (blue). The lower values for max-depth seem to have higher accuracies than for higher ones.



Model Specification & Comparison

Before settling for XGBoost, we tried our hand at different models from Regression, ensemble methods like Random Forests, regression tuning models like Lasso and Ridge, Neural Networks using Keras and the XGBoost Tree model.

Selection Methodology

While regression models are good to understand the importance of different variables, they lack tuning parameters and one cannot be too sure whether the model is over-fitting or under-fitting. This is where Lasso and Ridge can make a difference by introducing rugularization by adding a penalty to the loss function. These also come with cross-validations built into them for a more balanced accuracy. However there is much left to be desired. On all three we achieved a validation accuracy of 72% following more feature engineering after the final test prediction submissions.

Four our next model we were considering decision trees. However, the model has tighter restrictions on the number of factor levels allowed for a categorical predictor. This pushed us to try the Ensemble random forest method. This model also offers options to parameter tune our model easily give you better accuracies. It is also a fact that Random Forest is a superior model compared to the decision tree, as it makes multiple trees from random data points and takes a collective decision on the accuracy. This method leaves nothing unsampled if the user so wishes to, hence reducing reliance on one tree. Ensemble methods gave a slight improvement.

If one wanted to improve model accuracies further, they would prefer the boosting method where a classifier is run, accuracy is recorded, misclassified data get upweighted, repeats

previous steps again and the model after running multiple times, gives a weighted sum of classifiers. This models boosted our accuracy by at least 2%. We achieved a final accuracy of 74.5%

Of special mention here are Neural Networks. Though we got superior accuracy on the smaller training and testing set provided, by cutting down some features, we were not able to improve the accuracy as efficiently as XGBoost did, for a larger feature set. The reason for this could be that neural networks need extensive parameter tuning and the layer depth needs to be optimal. Handling missing values can also be difficult in NN since there are no defaults to take care of them. We created a 4 layer deep model with input/hidden/output config of (353,2000,500,100,1). We also utilized layer dropout and GPU processing using Nvidia CUDA to train the model faster. However, overfitting was still a problem and due restrictions and time considerations, XGBoost was our final model.

Using Logistic Regression -
Accuracy Before External Data and Top 5 Frequent Words

```
    Null deviance: 92478  on 68193  degrees of freedom
Residual deviance: 73267  on 67886  degrees of freedom
AIC: 73883

Number of Fisher Scoring iterations: 10

> logistic_acc
[1] 0.7274345
>
```

Accuracy After External Data and Before Top 5 Frequent Words

```
    Null deviance: 92478  on 68193  degrees of freedom
Residual deviance: 73106  on 67870  degrees of freedom
AIC: 73754

Number of Fisher Scoring iterations: 10

> logistic_acc
[1] 0.7289742
>
```

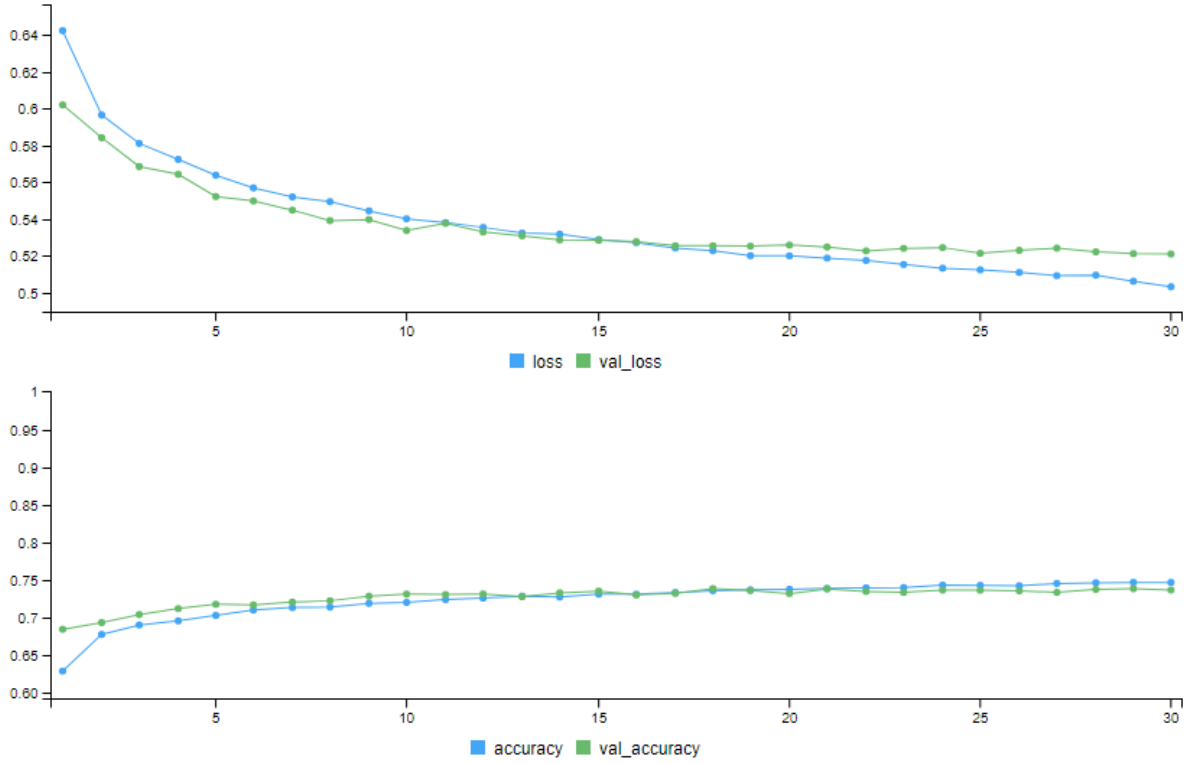
Accuracy After External Data and Top 5 Frequent Words

```
    Null deviance: 92478  on 68193  degrees of freedom
Residual deviance: 73004  on 67865  degrees of freedom
AIC: 73662

Number of Fisher Scoring iterations: 10

> logistic_acc
[1] 0.7291111
>
```

Training our neural network:



(Above) Loss on training set vs Validation set
(Below) Accuracy on training set vs Validation set

Conclusion

Key Takeaways

The most important take away for our team is that, no matter how great of a model one might have and the minute level of tuning one might have performed, if the feature engineering is not upto the mark, no amount of tuning parameters are going to improve your accuracy/RMSE score or in general your model performance.

It also makes a difference how one might handle their NAs. We had an external dataset on baby names which brought in the gender demographic parameter. However since the names on the kickstarter dataset had mixed information. Some were real names, others was an avatar name a user goes by. This introduced many NAs in the gender column. We set ‘Male’, assuming that most project creators might predominantly be this gender. This reduced our accuracy. On the

flip side, when we assigned a random string value instead such as - ‘None’, there was definitely an improvement in the overall performance.

Challenges

The real challenge in handling different models is understanding what type of inputs work with the estimator. Some models might need a Matrix, others might require dummies for categorical variables and some others are fine with handling categorical factors as they are. Error handling when running models can take up time, especially if the dataset is large. Multiple times, Rstudio shuts down or crashes which could perhaps be due to space constraints, CPU overload, RAM restrictions, etc.

References

Hyperparameter Tuning - https://towardsdatascience.com/hyperparameter-tuning-in-lasso-and-ridge-regressions-70a4b158ae6d
Different Exploration Methods - https://towardsdatascience.com/15-data-exploration-techniques-to-go-from-data-to-insights-93f66e6805df