Crowdfunding : Predicting Kickstarter Project Success

Authors

Siddharth Bhandari
19UCS080

Meet Kumar Jain
19UCS035

Vidhi Mittal
19UCC108

Under Guidance of
Dr. Bharavi Mishra,
Faculty - Machine Learning

Abstract

Investors usually find it difficult to predict or find the companies and startups that have high chances of success as it requires them to do manual calculation and analysis. crowdfunding has emerged as an alternative for this approach as disruptive innovation for financing a variety of new entrepreneurial ventures without standard financial intermediaries. The aim of the present study is to study the concept of Kickstarter, we predict whether a Kickstarter project will succeed or fail in achieving its fundraising goal using only information from project launch. Using various algorithms of Machine Learning, we evaluate the success rate of the startup, the accuracy and the precision of the project.
This is calculated from the given information such as fundraising goal, short description of the project, creator description.

Introduction

Kickstarter is an amazing platform that helps entrepreneurs launch campaigns, and "help bring creative projects to life".
In order to be successful i.e. have enough and required funding goals, a campaign must effectively convey its mission to its targeted backers or audience and try to get them to choose their project over others. Although anyone could broadcast their creative ideas on this website, not everyone manages to get enough pledge before the campaign deadline. We try to find out whether it is possible to predict the success or the failure of a campaign in the start by using the information present at the beginning of the campaign.
Crowdfunding has gained popularity over time, with an ever-increasing number of campaigns and investors participating. It was relatively popular from the start, and it has rapidly grown in prominence since then.

Relevant Literature

There have been several studies that leverage machine learning techniques to predict the success of a campaign. The project or we can say the website is of great help both for the creator as well as for the Kickstarter as such knowledge would enable creators to work more on their project and give sufficient and justified time to their project while for the Kickstarters, the platform will help in prioritizing the projects based on the accuracy and precision of the following (it is assumed that the accuracy lie in the range of 60 to 80 for a good project).

Dataset and Features

DATA

Projects on Kickstarter combine a variety of data types. There are almost 38 columns or we can say there are almost 38 entries for each dataset which includes fundraising goals, creator details, details of the project or startup, backstory, categories, photos, videos.

We use a well-maintained repository containing data for over 2,50,000 Kickstarter projects. The authors designed a robot to crawl Kickstarter each month and scrape the labeled HTML output into CSV files. This output contains all the information we use for this analysis, including the funding goal, project categories, and a short project blurb. Although the data is stored as a CSV, many of these features are stored as JSON strings, which we expand to obtain our desired project variables.

We attempt only to select information that would be available at the time of launch. However, since our data comes from a monthly snapshot, if a creator were to edit their project metadata after launch, we would not be able to detect this.

Luckily, many project aspects cannot be changed after launch, including the funding goal.

For our project, we are using only the dataset of the month of April for the year 2019 which includes more than 2,00,000 entries for the same.

CATEGORY ENCODING

Preprocess our features and dummy categorical variables before looking into the data further.

We did data preprocessing where we deleted duplicate or redundant columns and split time from its UNIX format to normal time zone format.

We also splitted the countries and states into their subcategories to understand the data more precisely.

We also divided all the projects into their parent categories where the parent category was further divided into subcategories which we named child categories which would make it easy for Kickstarters to see or understand and explore their categories.

As discussed above, each Kickstarter project is associated with a variety of categorical variables, including the project category (e.g. Board Games, Documentary, etc.) and the parent category (e.g. Gaming, Film, etc.), the creator's city and the country of origin as well as the currency of the donation.

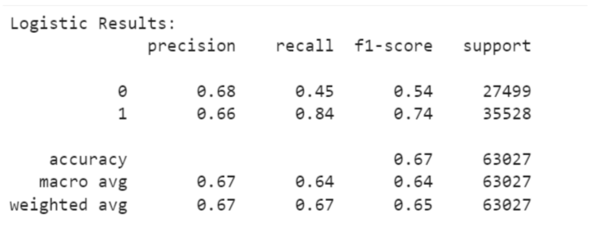
We divided our dataset into two categories in the ratio of 70:30. While the minority ratio is for testing purposes, the 70% of the dataset is used for training purposes under which 30% of the 70% which is technically 21% of the original dataset is for validation purposes.

Models

In machine learning, we can choose various models for analysis of the projects and calculate their accuracy and precision of being successful or failed.
We start the analysis with Logistic Regression, then we move on to Naive Bayes, followed by KNN and several other methods.

LOGISTIC REGRESSION

Logistic Regression uses a sigmoid function $S(x) = \frac{1}{1+e^{-x}}$ as the value should be in between 0 and 1.
We chose Logistic Regression to be our baseline model and split the data into 70% training, 30% of training data for validation and 30% for testing samples. After fitting the different train, validation and test samples into our baseline model,we got the results in below table:

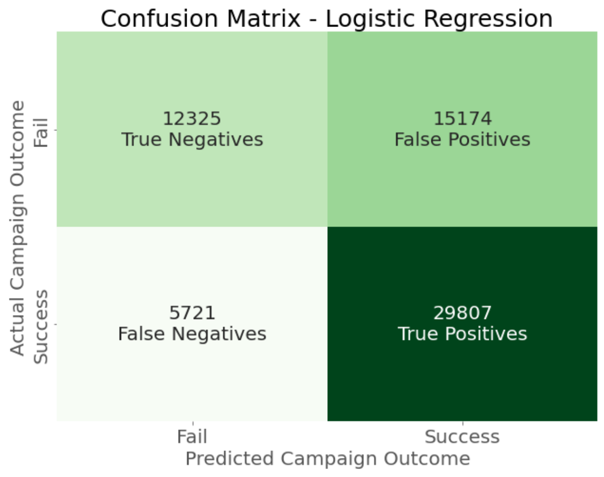


```
Logistic Results:
              precision    recall  f1-score   support

           0       0.68      0.45      0.54     27499
           1       0.66      0.84      0.74     35528

    accuracy                           0.67     63027
   macro avg       0.67      0.64      0.64     63027
weighted avg       0.67      0.67      0.65     63027
```

As the test recall is higher than the test precision, so there is some scope for raising our precision score with other models.



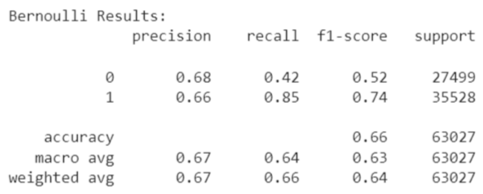
The model is making more false-positive predictions than false-negative predictions which means that there are more predictions on successful campaigns that are actually unsuccessful than successful campaigns that are actually successful.
So, we will try on the same process with some other models.

BERNOULLI NAIVE BAYES

Bernoulli Naive Bayes predicts membership probabilities for each class such as the probability that a given record or data point belongs to a particular class.
This can be expressed mathematically as:

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

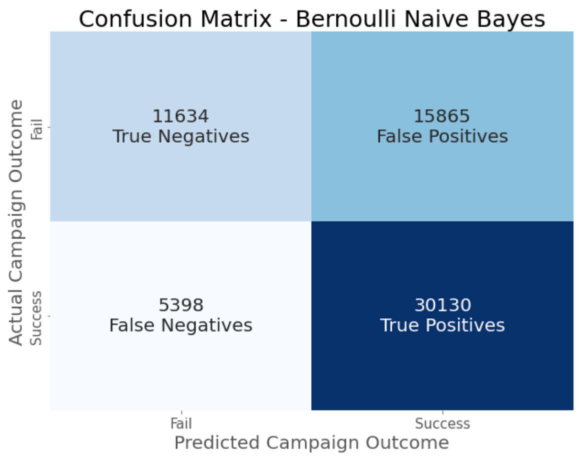


```
Bernoulli Results:
              precision    recall  f1-score   support

           0       0.68      0.42      0.52     27499
           1       0.66      0.85      0.74     35528

    accuracy                           0.66     63027
   macro avg       0.67      0.64      0.63     63027
weighted avg       0.67      0.66      0.64     63027
```

As the test recall is higher than the test precision, so there is some scope for raising our precision score with other models.

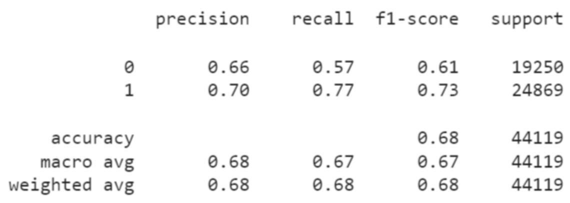


The model is making more false-positive predictions than false-negative predictions which means that there are more predictions on successful campaigns that are actually unsuccessful than successful campaigns that are actually successful.

So, we will try the same process with some other models.

K-NEAREST NEIGHBOR

The KNN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. After fitting the different train, validation and test samples into our KNN model,we got the results in below table:

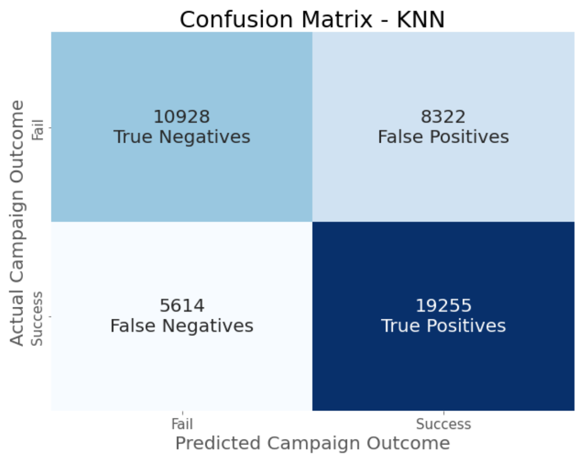


```
              precision    recall  f1-score   support

           0       0.66      0.57      0.61     19250
           1       0.70      0.77      0.73     24869

    accuracy                           0.68     44119
   macro avg       0.68      0.67      0.67     44119
weighted avg       0.68      0.68      0.68     44119
```

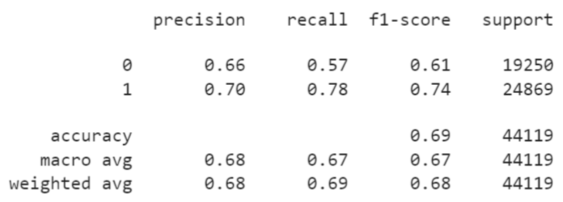
The following table is our confusion matrix for the KNN algorithm :



RANDOM FOREST

Random Forest builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

Precision Recall Table :

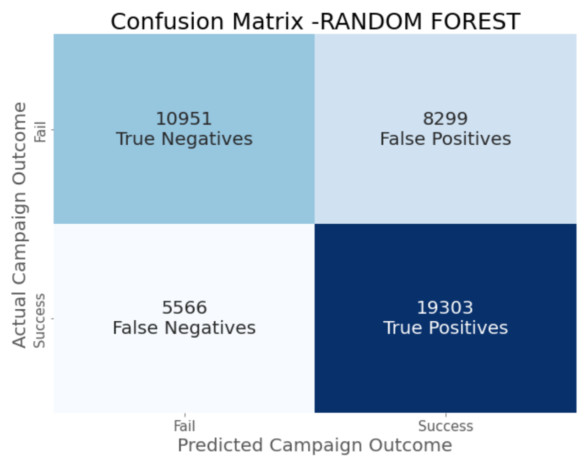


```
              precision    recall  f1-score   support

           0       0.66      0.57      0.61     19250
           1       0.70      0.78      0.74     24869

    accuracy                           0.69     44119
   macro avg       0.68      0.67      0.67     44119
weighted avg       0.68      0.69      0.68     44119
```

Confusion Matrix :



XGBOOST

XGBoost stands for Extreme Gradient Boosting, is a distributed decision tree which provides parallel tree boosting are created in sequential form.

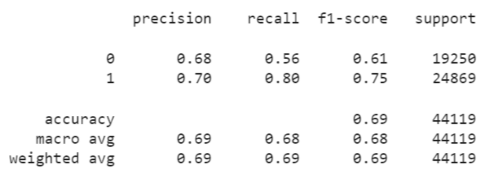
XGBoost, in the form of equation, can be expressed as :

$$obj^{(t)} = \sum_{j=1}^{T}[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T$$

where,

$$G_j = \sum_{i \in I_j} g_i$$
$$H_j = \sum_{i \in I_j} h_i$$

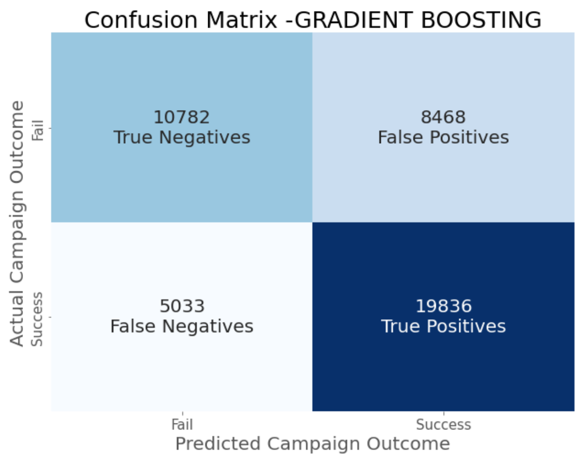
After applying XGBoost on our dataset, we get to a point where the below two images gives us our precision-recall table and the confusion matrix for the same.



```
              precision    recall  f1-score   support

           0       0.68      0.56      0.61     19250
           1       0.70      0.80      0.75     24869

    accuracy                           0.69     44119
   macro avg       0.69      0.68      0.68     44119
weighted avg       0.69      0.69      0.69     44119
```



Results and Discussion

As the studies and analysis from the above algorithms shows, from our dataset, we have calculated the accuracy and the precision for all the models where Gradient Boosting, also known as XGBoost model performs the best among all algorithms, while the remaining other models have fairly similar performance, as depicted in table below :

Model	LR	NB	KNN	RF	XGBoost
F-1	0.74	0.74	0.73	0.74	0.75
Recall	0.84	0.85	0.77	0.78	0.80
Precision	0.66	0.66	0.70	0.70	0.70
Accuracy	0.67	0.66	0.68	0.69	0.69

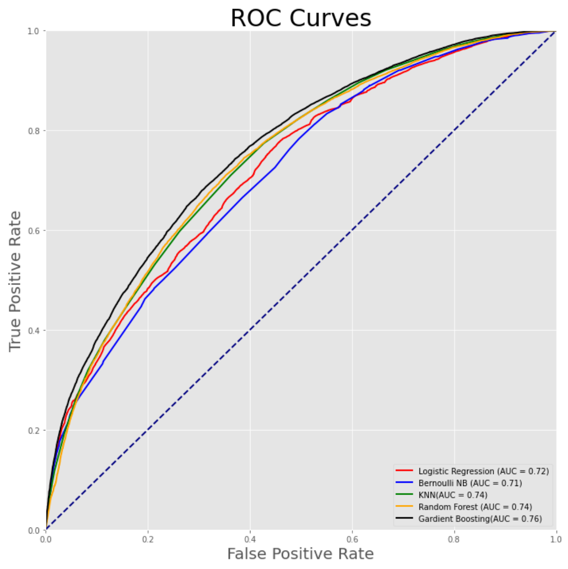
On study of confusion matrix, we come across various terms such as Precision, F-measure, Recall which are defined below :

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

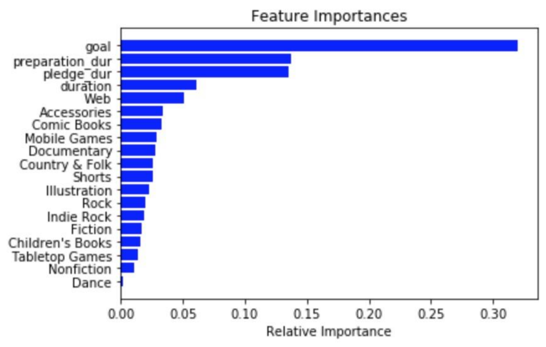
$$\textit{F - measure} = \frac{2*\text{Recall}*\text{Precision}}{\text{Recall} + \text{Precision}}$$

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

Another measure of study is the ROC Curve which gives us the plot between the following two parameters : True Positive Rate and False Positive Rate.



It was also studied that goal and time-related features are the most important features in predicting a campaign's success. This finding makes sense because the success of your campaign is decided by whether the amount of pledge meets your goal, and whether you get the amount of pledge within the campaign deadline.



Limitations

While running the Support Vector machine (SVM) algorithm, we almost waited for half an hour but the algorithm was still under process and took time.

Reading some papers and resources, we came across a line which states that SVM is not supported for large data sets and we decided to skip that algorithm.

The main problem with preprocessing is that we have to manually do the feature selection, explained in the category encoding section, which requires a lot of manpower when it comes to millions and trillions of datasets.

While running the XGBoost algorithm, we saw its performance in predicting different features was different for each subcategory of same parent category.

Conclusion

According to our results section, we would advise campaigners to:

-> Set a small goal that fits the project scope

-> Keep the campaign duration short

-> Consider the category carefully

in order to increase their chances of launching successful campaigns!

Out of all the algorithms that we studied, we found XGBoost to be the best among all. We usually give preference to Precision, followed by other factors such as Accuracy and Recall. Precision and Accuracy were same among some of the algorithms, but XGBoost got an edge in case of Recall parameter.

Link for the Project and its Files

https://github.com/slundberg/shap

https://drive.google.com/drive/folders/1Y62JNikSS_pu5UQM0PEJYUJ5rjcKiNuK?usp=sharing

References

- https://www.geeksforgeeks.org/libraries-in-python/
- https://ai.stackexchange.com/questions/7202/why-does-training-an-svm-take-so-long-how-can-i-speed-it-up
- https://en.wikipedia.org/wiki/Random_forest#:~:text=Random%20forests%20or%20random%20decision,class%20selected%20by%20most%20trees.
- https://www.datacamp.com/tutorial/xgboost-in-python
- https://stanford.edu/~kartiks2/kickstarter.pdf
- https://github.com/slundberg/shap