Ji Ho Lee

Predicting Success of Kickstarter Projects

1. Introduction

With the widespread use of Internet and technology over the past two decades, crowdfunding has become a successful source of financial backing for many project owners. Kickstarter is an online crowdfunding platform where the main focus is to "help bring creative projects to life". As of 2019, Kickstarter has collected over $4.6 billion for more than 445,000 projects in categories ranging from music, video games, stage shows and technology. Kickstarter allows unique projects and personal dreams to come to fruition.

However, this does not mean all Kickstarter projects are successful. In fact, **over 60% of the projects have failed to achieve their funding goal**, with only about 35% of projects achieving full funding. This project aims to utilize the past data of over 370,000 Kickstarter projects to **effectively predict which projects will be successfully funded**.

2. Audience

- Project creators would be able to streamline their postings by understanding what features of a project have the highest potential for funding. It would also give project creators a more realistic idea of what their funding goal amount should be and what they can expect based on previous projects.
- Project backers would benefit from this project as they would have a better understanding of how successful a Kickstarter project would be. Although users are automatically refunded if a project is canceled or fails to meet its deadline, this project would decrease the amount of users that got their hopes up for the success of a project.
- From the perspective of Kickstarter, this would help the company become more efficient in vetting the hundreds of projects that are submitted daily to Kickstarter. By providing a prediction of success, it gives the employees another source of information to more accurately judge whether a project should be accepted by Kickstarter.

2. Data

The dataset used for this project was found on Kaggle and **contains around 378,000 past Kickstarter projects** ranging from 2009 to 2018. The dataset contains 16 columns containing information such as the name, category, launch date and deadline, funding goal, amount pledged, and state of the project.

Thankfully, the dataset did not have too many issues; only minor cleaning was necessary. For example, almost 4000 projects had 'N,0"' listed as its country of origin. This

was simply fixed by extrapolating the information from the original currency the project was posted in. The state column consisted of ‘failed’, ‘successful’, ‘undefined’, ‘live’, ‘suspended’, and ‘canceled’. Although a project could’ve been canceled for many reasons, it still means that the project’s funding goal was not met, so I categorized ‘canceled’ projects as ‘failed’. I dropped the rest of the projects that were neither ‘failed’ nor ‘successful’ as these projects only accounted for 2.1% of the whole dataset. The ‘state’ column is what the models will be predicting.

Before the data could be used for modeling, the data had to be pre-processed in order to accurately predict the outcome of the projects. All numerical features, except for the ‘state’, were heavily right skewed. Because of this, **extreme outliers were removed** by dropping all cases where there was at least one feature that had a value greater or less than three standard deviations from the mean. This only slightly fixed the skewness of the data and so **all features were log scaled** to produce a more normal distribution. Finally, all numerical columns were scaled using scikit-learn’s StandardScaler and all categorical columns were one-hot encoded

During the preprocessing step, I also **dropped columns that contained redundant or irrelevant data**. These columns included ‘category’, ‘currency’, ‘deadline’, and ‘launched’; the information from these columns were present in some form in different columns that were left in for the modeling. Removing these redundant columns prevented the models from suffering the curse of dimensionality.

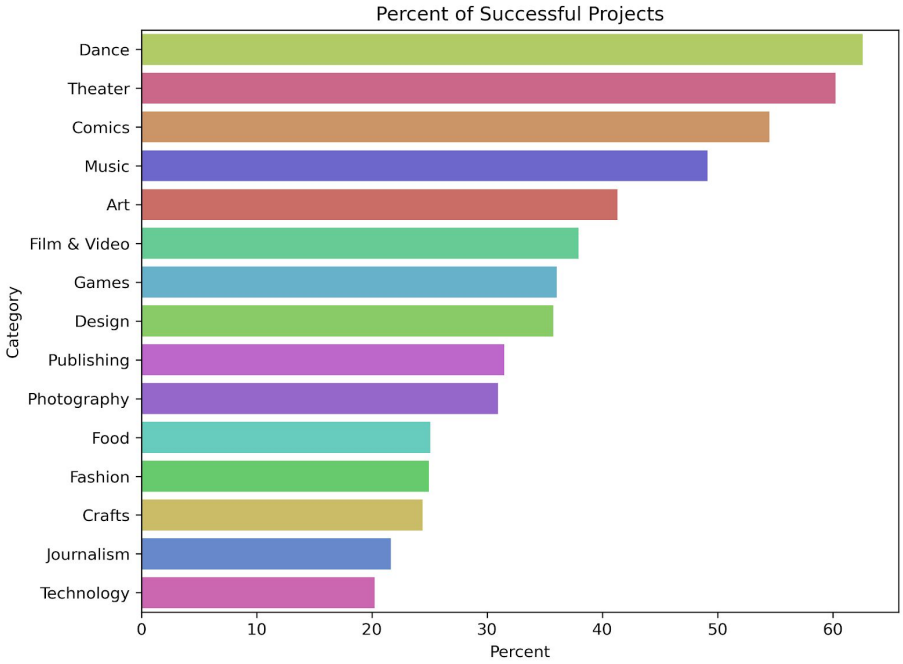
While modeling, I found that the models were predicting with almost perfect accuracy and this seemed too good to be true. I investigated the features’ importances and found that ‘backers’ and ‘pledged’ were significantly more important than any other feature. This made sense as the state of funding would be directly correlated to the number of backers and the amount pledged to the project. To improve the generalizability of the models, I dropped both of these columns.

3. Exploratory Data Analysis

With this dataset, there are a lot of initial questions that could be asked. Which project categories are the most common? Which ones get the most funding and which ones get the highest percentage funded relative to their goal? Are there any correlations between the features?

While exploring the dataset, I found that **Film and Video projects are the most common** with 63,583 projects (16.8%) in the dataset, and Dance being the least common at 3,768 projects (~1%). However, dance projects have the highest percent of projects successfully funded at slightly over 60% while **projects in the Technology category have the lowest percentage of successful funding** (20.3%). The low success rate for projects in the Technology category could be attributed to the high funding goal amount. Projects in the Technology category, on average, set the funding goal amount to around $120,000 with a median goal of $20,000, while the average for all projects is around $4,800 and the median is

$5500. The significantly large goal amount could definitely be an explanation as to why the Technology category has a low success rate.

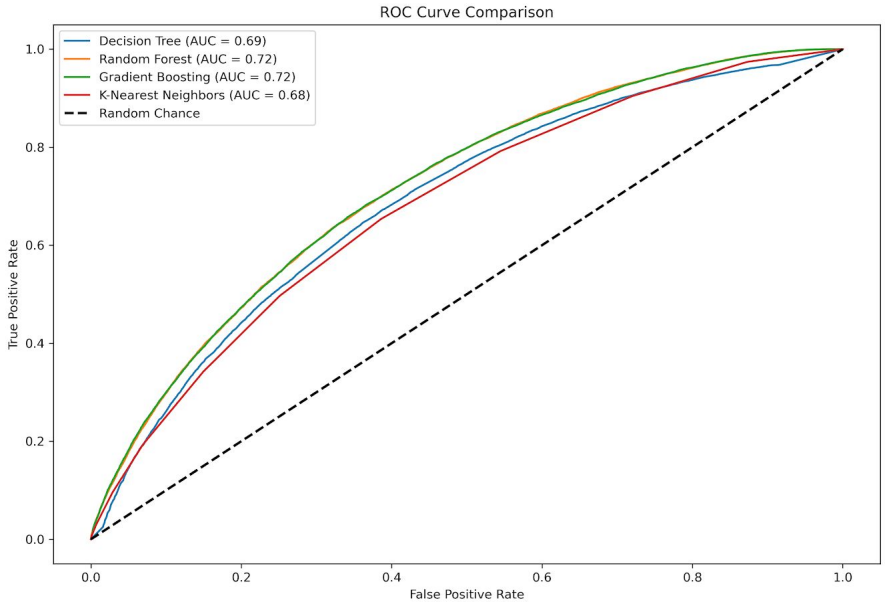


4. Modeling

To predict the success of Kickstarter projects, I chose to test four different models: Decision Tree classifier, Random Forest classifier, Gradient Boosting classifier, and K-Nearest Neighbors classifier. These **models were mainly evaluated based on their precision scores** as I wanted to minimize the number of predicted false positive errors. False positive errors could mislead users and project owners into thinking a project will be successful, and companies would waste resources to host a project that wouldn't be successful when those resources could've been used for another project.

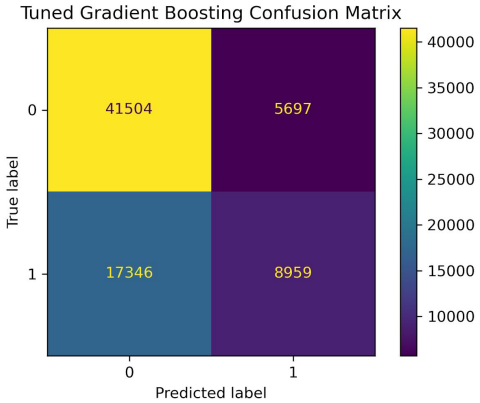
I implemented a Naive Classifier to understand the baseline prediction accuracy using the dataset. The Naive Classifier predicted all test cases to be successful regardless of the features. This is useful in understanding that a 99% accuracy does not necessarily translate to a great model. For example, if 99% of cases in a dataset were class 1, naively predicting all cases to be 1 would provide an accuracy score of 99%. This paints an inaccurate picture as this model would've failed to correctly predict the relevant 1% of cases. On this dataset, the Naive Classifier scored 0.5328 for its precision and 0.6586 for its accuracy, meaning there is only a slight case of class imbalance.

The Decision Tree, Random Forest, and Gradient Boosting classifiers were **all hyperparameter tuned** using RandomizedSearchCV for each of the models' most important parameters; while the K-Nearest Neighbors classifier was tuned for the K value using the Elbow method.



From the precision scores and ROC curves shown above, I decided to **select the Gradient Boosting classifier as the final model**. The Gradient Boosting classifier had the highest precision score, returning the least amount of false positive errors, and also had the best ROC curve (closest to the top left corner of the graph).

	Accuracy	**Balanced Accuracy**	**Precision**	**Recall**
Decision Tree	0.6758	0.6079	0.5730	0.3692
Random Forest	0.872	0.6128	0.6091	0.3512
Gradient Boosting	0.6865	0.6099	0.6113	0.3406
K-NN	0.6686	0.5963	0.5605	0.3419



5. Future Improvements

More work can be done to improve on these results. One improvement that could be made would be in feature engineering. More time could be spent coming up with **creative and innovative ways to create new features** from the existing features. Also, implementing NLP and analyzing sentiment and other features on the name of the projects could provide extra data to better predict the success of these projects. Another source of improvement could be made in hyperparameter tuning. Due to the lack of computing power, only a couple important hyperparameters were tuned for each model. **With more computing power and time, more hyperparameters can be tuned, further optimizing the results.**