William Chirciu
CSC 540 Final Project
Predicting Kickstarter Success at Launch

Abstract

Objective: According to Kickstarter, around 63% of the projects on their platform are a failure. Failed projects have a cost: the creator loses all that time and money spent into promoting their project. The purpose of this research was to build a model- using the full Kickstarter repository - that would be able to predict Kickstarter project success or failure solely with details available at launch and to figure out which of these details were most important. These solutions could then potentially be applied in a way to minimize the disparity between the number of failed and successful projects.

Methodology: Exploratory data analysis was the first step in identifying which features in the data were most important. Looking at the distributions of failed/successful projects across the different categorical variables (country, categories, currency) and the descriptives of different numerical variables (duration, goal amount), predictions were made on which features would be most important. Following this, the data was fit to 5 different models. These were Decision Tree, Random Forest, Gradient Boost, Ada Boost, and Neural Network. After many iterations of hyperparameter tuning and feature selection, several competitive models were discovered.

Results: The decision tree was the base estimator and predicted with 67% accuracy and had the fastest run times between 5 and 7 seconds. The gradient boost model had the best accuracy at 69%. Random Forest had 68% accuracy with the second-best run times. Both Neural Network and Ada Boost underperformed with 66% and 64% accuracy respectively with extensive run times. Because our data was slightly imbalanced with 40% successful projects and 60% failed projects, the results needed to be reevaluated on a balanced dataset. SMOTE was used to balance the data and the models were re-computed and re-evaluated on an imbalanced validation set. We obtained similar accuracies as our original models with the AUC scores decreasing by around 0.1, which scales proportionally with the 10% rebalance. Therefore, our initial results were valid.

Conclusion: The best model was the Gradient Boost with all the features as it had the highest accuracy and AUC scores. Exploratory data analysis and multiple feature selection methods pointed to the project goal, campaign duration, and project category as being the most important details in determining project state at launch. With more static features (reward levels, product descriptions, staff picks) and computing power, both in the realm of memory and parallelization, a better model could be found and applied in the real world.

1. Introduction

Kickstarter is a crowd-funding platform where entrepreneurs and creative people try to bring their ideas to fruition. They set a monetary goal to be met within a specified time-frame. Most often, contributions are made in small increments by consumer-investors. If the goal is met by the deadline, then the money pledged by these “backers” is given to the entrepreneurs to begin project development. Otherwise, no money changes hands and the project is classified as a failure.

As of January 1st, 2019, around 63% of all Kickstarter projects are a failure. There are many factors that could play a role in this: creators asking for too much money, the length of the campaign is too short, the proposed project has a weak first impression, and so on. As this is not an investment platform, no money is lost when a project fails. However, it is most likely the case that a lot of time and effort went into the project's campaign by both the creators and the backers; the creators as they are the ones maintaining the campaign and the backers as they are the ones that help promote the campaign through social media. Therefore, it is in every party's best interest to know the likelihood of a project turning on its head. The goal of this research was to be able to predict Kickstarter project success at launch and to identify the most important factors that affect this. The paper will discuss the process leading up to the optimal prediction model as well as potential applications. The next section details similar studies in predicting Kickstarter project states.

2. Literature Review

Several researchers have explored ways to predict project success/failure on the Kickstarter platform. As the disparity between the percentages of successful and failed projects continues to grow, it has become increasingly important to pursue this. One such study incorporated dynamic features (features unknown at launch) in a Markov Model to predict project success mid-campaign **[1].** Kickstarter has a web crawler that captures project information at different time segments. Using this data, they created a time series on the amount of money pledged and used it to create a Markov Model. They were able to predict a project's outcome with 86% accuracy after only 15% of a project's campaign duration. They combined this model with an SVM built from social features (Twitter and Facebook posts) and managed to have a prediction accuracy of 76% after only 4 hours into a project's campaign. Though this paper covers only static features, it is interesting to see the potential of Markov models in this domain.

A few students at Georgia Institute of Technology took text features a bit further and incorporated over 26 thousand popular phrases from a corpus of 9 million phrases into a logistic regression model **[2]**. These phrases were obtained from the descriptions of 46 thousand Kickstarter projects. Their initial approach using only 59 variables describing different aspects of the crowdfunding campaign, some of which will be discussed in this research, led to a model with a high accuracy of 83%. Adding in the text features, their accuracy was improved by an additional 15%, validating their hypothesis that phrases used in Kickstarter descriptions are significant and impact the choice people make when deciding whether to pledge or not.

Adebola Lamidi created several models, including Random Forest, to predict Kickstarter project success at launch using all projects from 2009 – 2011. **[3]**. The features he used were very similar to the ones used in this research, except for the number of reward levels and the project's associated continent. He achieved an accuracy of 66% with logistic regression, 63% with K-nearest neighbors, and 68% with random forest. On a related note, Brian McMahon performed a similar process in not only predicting Kickstarter success, but also the number of backers and how far into their campaign goal they will get **[4]**. The features he had available included staff picks and the length of the project description. Using XGBoost, an optimized version of gradient boost, he managed to achieve an accuracy of 77.5% with an AUC of 78%.

The top 4 most important features he selected were the goal amount, the length of the description, the campaign duration, and whether the project was a staff pick. Srishti Saha in her mission to predict the success of a Kickstarter campaign using all projects up to 2017, managed to come up with a Light Gradient Boosting model that resulted in 70% accuracy **[5]**. She managed to incorporate competition in her features by including the average goal and average project duration within each project category along with average success rates. Her top three features were the category, average success rate, and the campaign duration. Out of all the research previously mentioned, these ones are the most comparable to what is trying to be done in this paper. It will be shown if using even fewer features will have competitive results and if the most important features are similar.

3. Methodology

The Kickstarter data was collected from the Kickstarter repository and made available on Kaggle. There was a total of around 378k projects from 2009 - 2018. The available information for these projects included the project ID, name, main category, sub-category, the project country and currency, launch date and deadline, the project goal and pledged amount in the project currency and in USD, the number of backers, and the target variable 'state'. The project state had 6 possible values: {successful, failure, suspended, cancelled, live, undefined}. For the purposes of predicting project success or failure at launch, all projects that were suspended, cancelled, live, or labeled as undefined were removed. Additionally, all instances with missing values were removed, though there were only 213.

The categorical attributes were converted to numeric using Label Encoding. In other words, each category was assigned a number. Understanding the implications of label encoder, i.e. the models will see these values in an ordinal manner, One Hot Encoding was also tested. Because there are features with so many categories, the run times were increased severely, especially with the decision tree ensemble methods. Taking this into consideration as well as the change in performance, Label Encoding was the chosen approach.

Manual feature selection was done to remove features that were redundant or irrelevant. Name and project Id were obviously deemed as irrelevant as they are unique to each project and give no information. The amount of money pledged, and the number of backers were removed as well because they are dynamic features. This information is unknown at launch and is best used in a time series analysis as mentioned in the previous literature review. There were 2 features for the goal amount: goal in project currency and goal in USD. The goal amount in project currency was removed to preserve consistency because there are multiple "units" so to speak within this feature. Finally, launch date and deadline were combined into a single feature, campaign length, and removed from the data entirely. Post preprocessing, we are left with 331462 observations and 7 features including the target variable.

	Successful	Failure
# of Projects	133,851	197,611
Average Goal	$9,536	$63,189
Average Dur.	32 days	35 days
Average Pledged	$17,367	$1,133
# of Backers	264	16

Figure 1: Descriptive Statistics Across Failed/Successful Project

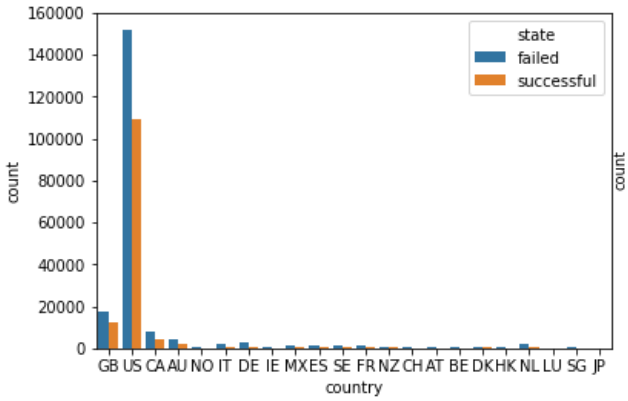

Figure 2: Distribution of Failed/Successful Projects on Countries

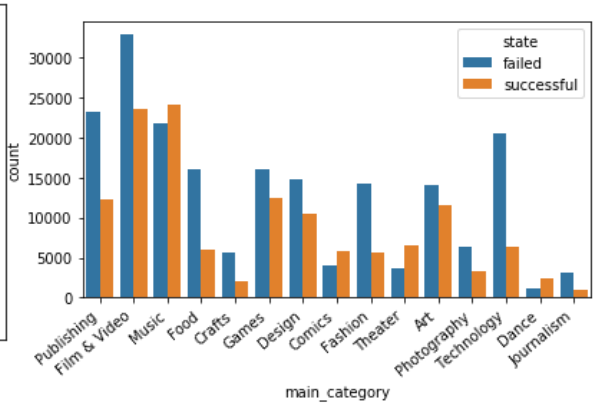

Figure 3: Dist. of Failed/Successful Projects on Main Category

Exploratory data analysis was done to get a better picture of the data. **Figure 1** shows the average values of each quantitative variable across the two project states. The most important thing to notice is the disparity in the average goal amounts between the classes: $9,536 for successful projects and $63,189 for failed projects. This coupled with the relatively small difference in the average campaign duration suggests that people are asking for way too much money in too little a time. From this, it was likely to assume that these two features in conjunction would be significant in model building. **Figure 2** reveals the distribution of failed and successful projects across the 22 different countries. Clearly, the United States holds by far the greatest number of projects. Because the distribution is so imbalanced, it was hypothesized that 'country' would not be among the more important features in the predictive model. Finally, in **Figure 3**, the relatively balanced distribution of failed and successful projects is shown across the 15 "umbrella" categories. The disparity in the distributions can be seen by looking at "Music" and "Publishing" as they contain a similar number of projects. Projects in "Music" have a 36% chance of failure and projects in "Publishing" have a 65% chance of failure. Therefore, it was hypothesized that category (umbrella or sub), would be a significant predictor.

After performing exploratory data analysis, it was time to go about crafting the predictive models. The simple decision tree was to be used as the base model as it is known for its quick-and-dirty approach to predicting the target variable. Four additional models were used to see whether the scores from the decision tree could be improved which include Random Forest, Gradient Boost, Ada Boost, and a Multi-layer Perceptron otherwise known as a Neural Network. The performance of a model was scored using accuracy, AUC, number of features, and run time

under 5-fold cross validation. Accuracy is the percent of correctly predicted instances. AUC describes how well a model distinguishes between classes and is particularly useful in binary classification problems. As for the number of features, it is important to minimize them without sacrificing the performance of the model. Therefore, this metric will be secondary to accuracy and AUC scores. Lastly, having models with minimal run times is ideal. This was taken into careful consideration as the dataset is very large.

Random Forest is an ensemble method that trains deep, independent decision trees with the bagging method and averages out their predictions. It is unique in that it introduces randomness into the model when growing the trees. When deciding on what feature to split on a node, it selects the most important feature from a random subset of features. As with any ensemble method, the number of estimators is an incredibly significant parameter. Taking into consideration the limits of the cpu involved and the run times, this parameter was adjusted between values of 10 and 200. To measure the quality of a split, the Gini criterion was used, though it is interchangeable with Entropy. Additionally, the minimum number of samples required to split an internal node was adjusted between ~0.001%-1% of the total number of observations. The minimum samples required at a leaf node was adjusted between values of 1 and 100.

Gradient Boost is another ensemble method that involves building weak learners in succession to form one strong predictor by minimizing some loss function. In this case, the weak learners were decision trees. The maximum depth and the minimum number of samples to split an internal node were considered to be the most important parameters when deciding on how weak to make the learners. Once again, the min_samples_split was adjusted between ~0.001%-1% of the total number of observations. Max depths between values of 5 and 10 were also tested. Learning rate was decreased proportionally to increasing the number of estimators to preserve the robustness of the model.

AdaBoost is yet another ensemble of shallow, weak learners. However, it builds off gradient boost in that it negatively weights samples that were classified incorrectly so the subsequent learners can focus more on these cases. The number of estimators and the learning rate are the only consideration here. The number of estimators was adjusted to values between 50 and 1000.

The Multi-layer Perceptron, or Neural Network, was the final model used to take advantage of finding hidden patterns in the data. Because the dataset is so large, the ‘Adam’ solver seemed like the best choice intuitively speaking. The activation functions tested were logistic sigmoid function and the rectified linear unit function. Between 1 and 2 hidden layers were used with node values between 1 and 100.

Feature selection played an important role in building the models. Initially, wrapper-based feature selection was used for each iteration, choosing the features based on the current running model. However, because not much of the feature space was being explored, a more standard approach to feature selection was taken using the Chi-squared method. This method returned a ranked list of the 6 features. In order of importance: **{usd_goal_real, campaign_len_days,category,country,currency,main_category**}. When building the predictive models, a different subset of features was used, starting with all 6. After each iteration, the lowest ranked feature was removed, until one predictor is left. The models were running on 6

subsets of features, the last subset being {usd_goal_real}. This allows for a more complete analysis of the feature space and consistency when comparing across the different models.

4. Results

Model	Accuracy	AUC	# Feat.
1	0.67	0.71	6
2	0.66	0.70	5
3	0.66	0.70	4
4	0.66	0.70	3
5 (- cat)	0.64	0.67	2
6	0.61	0.63	1

Figure 4: Decision Tree Performance

Model	Accuracy	AUC	# Feat.
1	0.68	0.73	6
2	0.67	0.72	5
3	0.67	0.72	4
4	0.67	0.72	3
5 (-cat)	0.64	0.67	2
6	0.61	0.63	1

Figure 5: Random Forest Performance

As shown in **Figure 4** the base decision tree model managed to obtain a peak accuracy of 0.67 and peak AUC of 0.71 using all 6 features. This mostly remained consistent until the 'Category' feature was dropped from the subset of features, resulting in a 3% decrease in both scores. The execution times were very appealing, running between 5 and 7 seconds. The models were run with the 'Gini' criterion using the 'Best' splitter.

Figure 5 depicts the performance of the Random Forest classifier. There is a peak accuracy of 0.68 and peak AUC of 0.73. Like the decision tree, there is a 3% dip in accuracy and a 5% dip in AUC when the 'Category' feature is dropped from the subset of features. For each iteration, 20 estimators were used in building the model as it minimized run times and had competitive results. The run times ended up being between 21 and 84 seconds.

Model	Accuracy	AUC	# Feat.
1	0.69	0.74	6
2	0.68	0.73	5
3	0.68	0.73	4
4	0.68	0.73	3
5 (-cat)	0.64	0.67	2
6	0.61	0.64	1

Figure 6: Gradient Boosting Performance

Model	Accuracy	AUC	# Feat.
1	0.64	0.68	6
2	0.64	0.68	5
3	0.64	0.68	4
4	0.64	0.67	3
5 (-cat)	0.63	0.66	2
6	0.61	0.63	1

Figure 7: Ada Boost Performance

Figure 6 depicts the performance of the Gradient Boosting classifier. There is a peak accuracy of 0.69 and peak AUC of 0.74 with the max number of features. This remains consistent once again until the 5th model where the accuracy drops by 4% and the AUC drops by 6%. 50 Estimators were used to minimize run times and retain competitive accuracy with respect to using a higher number of estimators. The optimal depth for the weak learners was 8. The executions took anywhere from 1 to 9 minutes.

Figure 7 shows the performance of the Ada Boost classifier. There is a peak accuracy of 0.64 and a peak AUC of 0.74 with all 6 features. These scores remain consistent until only 1 feature (usd_goal_real) remains where there is a steeper drop in performance. 100 estimators

were used with a learning rate of 0.1 to account for run times as well as maintaining competitive results. The run times were between 2 to 3 minutes.

Model	Accuracy	AUC	# Feat.
1	0.66	0.69	6
2	0.64	0.68	5
3	0.65	0.68	4
4	0.64	0.67	3
5 (-cat)	0.63	0.66	2
6	0.61	0.63	1

Figure 8: Neural Network Performance

Finally, **Figure 8** shows the performance of the Neural Network. The peak accuracy was 0.66 and the peak AUC was 0.69 at the max number of features. Once again, this model stays consistent through the subset of features until the 2nd to last feature 'currency' is removed, where we see the steepest drop in scores. The optimal performance was found using 2 hidden layers with node values of 10. The run times took between 97 and 357 seconds. An important thing to note is that peak accuracy was obtained when the data was normalized. The rest of the models showed no effect when normalizing the data.

5. Discussion

The Gradient Boosting model had consistently better performance than the rest of the models in terms of accuracy as well as AUC across most of the feature subsets. The decision tree had by far the best run times and the Random Forest was kind of the jack of all trades in that it had the second-best scores and second-best run times. Neural Networks and Ada Boost were both underperforming and had very long run times. Determining the optimal model was a bit tricky. As it is important to minimize the features and run times for a long-term model, sacrificing even 1% of performance is detrimental to the purposes of this project. Minimizing the number of features is ideal when peak performance is preserved. However, as the goal of this project was to maximize the predictive accuracy, taking the model with the max accuracy and max AUC was preferred.

One thing that must be taken into consideration is the impact of the project state distributions on the peak score, which in this case is 69% accuracy. For this data set, approximately 40% of projects were successful and 60% were failures. When you think about it, this model appears to be only doing 9% better than if you were to just guess that a project was a failure. To make sure that our models were not biased towards failed projects, the data was rebalanced with the SMOTE oversampling method to obtain a 50/50 distribution.

Model	Accuracy	AUC
Decision Tree	65.1%	64.7%
Random Forest	66.7%	66.4%
Gradient Boost	66.9%	66.4%
Ada Boost	61.8%	62.7%
Neural Network	63.6%	63.1%

Figure 9: Model Performances with Respect to Balanced Dataset

Figure 9 shows the performance metrics for each model with the classes balanced using all the features. The models were tested on a hold-out unbalanced validation set. The results were consistent with our initial approach for the most part. There is a slight decline in accuracy across the board of about 2%. As for the AUC, they decreased by about 10%, which is proportional to the rebalancing of the majority class from 60% to 50%. It was therefore concluded that we retain the "ladder" so to speak between the distribution of the target variable and the prediction accuracy. The models were not biased towards the majority class and our original results were still valid.

6. Conclusion

Though it was a close call, Gradient Boosting was the optimal model for predicting Kickstarter success and failure due to its higher accuracy and AUC score. The potential for higher scores is there however. Because of the limitations of the cpu and the size of the dataset, grid search was not feasible. SVM was also attempted on multiple kernels. However, the run times were abysmal, lasting for more than 5 hours. With enough computational resources it might be possible to boost the scores for the Gradient Boosting model and explore Support Vector Machines more thoroughly.

Another limitation was the lack of features. With 6 features we were able to predict with 69% accuracy. It was seen in several studies that features such as the number of reward levels, staff picks, the lengths of the descriptions, and average success rates had significant impacts on their models **[3][4][5]**. Using the minimal amount of data available in this dataset, there was an attempt to see if competitive results could be obtained. Though the results obtained in **[3]** and **[5]** were incredibly similar, the XGBoost model in **[4]** with 77.5% accuracy was the clear winner. It looks like the product description and 'staff pick' information would have been worthy additions to our feature list.

The most important features were 'usd_goal_real', 'campaign_len_days', and 'category'. This is evidenced by our feature selection process as well as our performance results. The chi-squared method as mentioned previously ranked these scores the highest. However, this was not the only feature selection used. Wrapper-based feature selection was also used, with each classifier as the wrapper. This selection method always chose at least these three features. Looking at the performance, the accuracy and AUC scores always took a steeper drop in random forest, gradient boost, and decision tree models when removing the Category feature. The scores remained consistent up until that third feature was removed. Therefore, our initial assumptions

derived from the exploratory data analysis were correct and anyone looking to start a campaign should pay special attention to these attributes.

7. Future Work

Solving the disparity in the number of failed and successful projects on Crowdfunding platforms is an issue that has been tackled many times over. Using a small representation of the full Kickstarter repository, a robust Gradient Boost Model was able to predict project state with 69% accuracy on "Day Zero". This can have many applications. The one that comes to mind is a third-party application designed to assist potential entrepreneurs or creators in setting up their campaigns. Obviously, this is not in Kickstarter's best interest as they don't want people to start second guessing projects that could potentially be successful, as they get a 5% cut. With a computer application built on a model such as the one proposed in this paper, a project creator can have their parameters "criticized". For example, if someone were to enter a $10,000 campaign goal in 1 week, the application will run these parameters through the model and spit out the likelihood of success, or in addition recommend parameter adjustments.

Another application could involve a feedback study on project descriptions. So random participants are selected to comb through all the details of several projects and record whether they would pledge. From this, a text-based analysis could be done to figure out what made the descriptions so appealing (or unappealing) to potential backers. This is similar to what was done at GIT **[2]**, but this time we would have actual human feedback instead of trying to determine significant phrases based on numbers alone. So, supplementing this with the model proposed, or others, could help further boost predictive accuracy. As the number of creative ideas continue to expand, it is up to these crowdfunding platforms to support them. With all this data constantly being gathered, it should eventually become easy for people to assess their situation prior to and during the launch of their campaigns.

8. References

[1] Etter, Vincent, Matthias Grossglauser, and Patrick Thiran. "Launch hard or go home!: predicting the success of kickstarter campaigns." *Proceedings of the first ACM conference on Online social networks*. ACM, 2013.

[2] Mitra, Tanushree, and Eric Gilbert. "The language that gets people to give: Phrases that predict success on kickstarter." *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 2014.

[3] Predicting the success of Kickstarter campaigns (2017)
Adebola Lamidi - https://towardsdatascience.com/predicting-the-success-of-kickstarter-campaigns-3f4a976419b9

[4] Predicting Kickstarter Campaign Success (2018)
Brian McMahon - https://medium.com/@cipher813/predicting-kickstarter-campaign-success-a9cf1f81e09

[5] Will your Kickstarter Project be successful? | A simple analysis to help you predict better!!! (2018) Srishti Saha - https://blog.goodaudience.com/kickstarter-projects-prediction-of-state-steps-for-a-beginner-analysis-f4630a50b7fe