Predicting Outcome of Kickstarter Projects

Ensemble Model with Text Exploration

Problem Statement

When Creators are seeking for ways to bring their project to life, they have several means of achieving this. Some projects are easy to bring to life based on stuff that creators already have, while others require a bit of financial assistance. For ideas that need help, they can go through a financer such as a publisher, for writers, or a studio, for game developers. However, not all creators want to go that route. For those, there is the crowdfunding path where their project can come to life with the help of other people who want to give money to see their creations.

Kickstarter's platform allows crowdfunding to easily happen. A Creator can post a project proposal, describing the idea, the path to the idea, and how the community can help. The community of crowdfunders can give money to the Creator to see their project come to life.

However, the Creator will only receive funding from the community if the project is 'fully backed', i.e. the monetary goal amount set by the Creator was reached by the community. Kickstarter states that the success of projects is only 37.06%. While not all ideas will, or should, be successfully funded, there are ways for Creators to improve their proposal, or "campaign", to reach their funding goal.

A prediction model can help both Kickstarter and the Creator by providing a way to increase the success rate and help Creators optimize their campaign. Campaigns can be edited to follow patterns of success of previous successful campaigns. They can also be optimized to release on certain days to reach a wide amount of audience members.

Kickstarter Data

Kickstarter, unfortunately, does not provide a public API and the community has limited access to their data. There are organizations, such as Web Robots, who, as a side project, scrape sites and provide the data they gathered to the public. I will be using the Kickstarter dataset provided by Web Robots to analyze successful Kickstarter campaigns. Web Robots provides their files in JSON or CSV with updates every month.

The initial kickstarter dataset is provided by webrobots.io from their free web scraping data projects. The dataset is pulled from the 2019-05-16 set and contains 210k rows with 37 columns. The data is split into 56 csv files.

Initial Cleaning

For the most part, the data set is generally clean. However there still needs to be some cleaning done. We'll start with combining the csv files into one single file. I'll be using the glob module to find all the file names for reading into a dataframe using pandas. From there, I'll be using pd.concat to attach all the DataFrames together into one large data frame.

The first data point to clean would be the timestamps and the format that they're stored in. According to the webrobots.io website, the time format is in unix, or epoch time. Using the time module, we're able to easily convert the epoch timestamps stored on the 'created_at', 'deadline', 'launched_at', and 'state_changed_at' columns. We store the value into two separate columns, "x_date" and "x_time" where x is one of the original columns. By storing time and date into separate columns, we'll be able to do easier analysis later on when we're exploring the data.

Our next points to clean is the columns stored in JSON formats since the web scrape project pulled the information in JSON. To do this, we simply use the json module to read the string in as a dict and pull information that way. Since there are several columns that store format in this manner, we'll be using a dict to store column:category information. There are also some columns that contain multiple categories that we want to pull so this way will also address that. We'll then use a for loop over key-value pairs to pull information in an easy manner.

Doing a bit of simple analysis on the data set, there are several columns that contain a large number of nulls compared to our 210k rows. From our exploration, the columns 'friends', 'is_backing', 'is_starred', and 'permissions' look to be related to a certain account. It provides information that is not relevant to the overall analysis.

Additionally, there are other columns that, while informational, provide little value to an analysis. For example, there is the currency_symbol which shows which symbol the currency uses. This would be helpful but we already have a currency column which provides more detail than the symbol. For the "$", this would relate to USD, CAD, and AUS. The symbol is not able to tell us this information. Other columns such as "photo" also fall under the unusable list and are therefore dropped.

Finally, there is an opportunity to pull more information from Kickstarter using the URLs provided by the dataset. To prepare for this, we'll be cleaning up the URLs once more by reading the JSON into a dict and parsing through the data that way.

Web Scraping

This is easily the most frustrating portion of the wrangling process. The script to scrape information itself was simple. Using the requests library and the BeautifulSoup library, I was able to pull the description from the HTML of the Kickstarter. However, when running this script on 210,088 rows, each iteration took about 2 seconds. This totalled to an expected time of 116 hours or almost 5 days. This quickly became unreasonable.

I knew right away that there could be optimizations made to speed this process up. I broke down my script into three parts: requesting the HTML, parsing the HTML, and creating the text from the HTML.

I ended up comparing between two libraries: urllib and request. The urllib library ended up being way faster on several tests when pulling the HTML of the page. It also ended up being more consistent with what it would pull. The requests library would sometimes be unable to pull information from a page. As for the cause, I'm still unsure but I do know that it was worth the switch. This ended up making the script run way faster and averaged less than 1 second per iteration.

When parsing the html, I compared BeautifulSoup and the selectolax libraries. From my testing, the selectolax library ended up being faster by a few ms but with the benefit of BeautifulSoup being able to parse multiple tags, the minimal speed increase of selectolax was not enough to overcome BeautifulSoup. Additionally, I also ended up parsing the image and video tags to count how many images and videos are in each description. Time was not saved in this section.

For creating the description text, there was an optimization issue since each p tag was separated in our HTML parsing. To overcome this, I used list comprehension when joining text.

Overall, the time taken for the script was reduced in half.

To run this script on a personal computer is taxing. There runs the risk of the computer turning off, the laptop going into sleep mode, and a multitude of other issues. To overcome this, I searched for ways to run this on a cloud server. Google Cloud Platform became the answer for this. This will allow the script to run without worry of any computer outages or errors.

Using GCP's free trial, I spun up a Linux based compute engine to simply run my script. The overall time ran for 2.5 days, with error checking for each row to ensure that the script does not crash.

Further Cleaning

Once the script finished, there were additional steps to clean since the data was not always pulled in the nicest way. To parse through html, the parser script stored information into a comma delimited list resulting in one additional column. The cleaning process addressed this issue by moving to individual columns. Further cleaning also included removing of rows that resulted in errors during the parsing process. Since there were only 184 rows, and those rows were consistently returning errors, it made sense to remove them. Additional processes included removing NaNs and creating new columns such as 'percent goal' or 'video usage'.

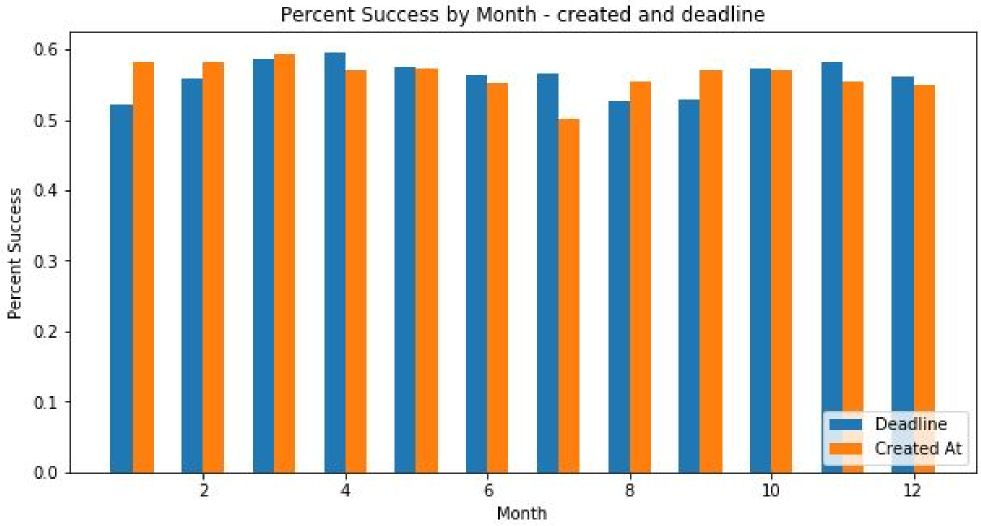
Past the cleaning, I realized that after certain procedures such as crafting the 'percent goal' or moving all monetary values to USD that other columns were not necessary for the analytics portion. This resulted in a new cleaning process where I dropped several of the tables that were not used for analytics such as the web urls or the original goal in JPY or GBP. I also made a final column to calculate the median of the rewards since that would be useful for analytics.

Exploratory Data Analysis

Diving deep into the data set, we can find several points that may lead to a solution for our problem.

Projects by Month

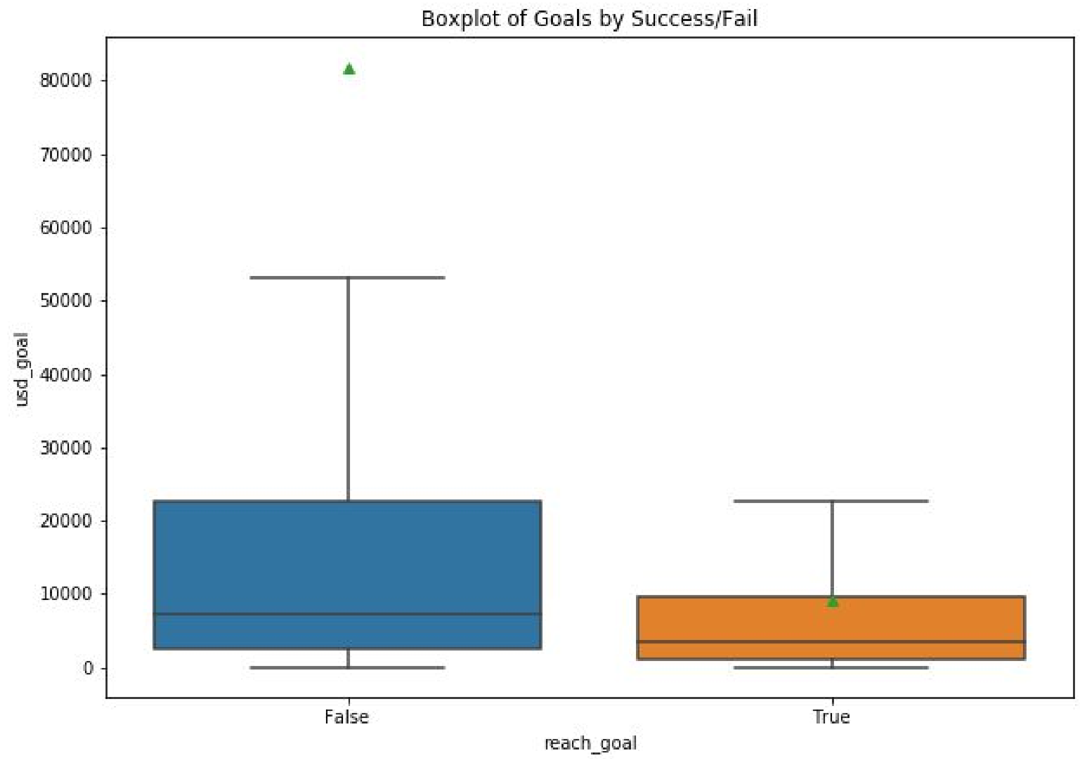
Kickstarter projects are being started no matter the time. Some months, namely January and December, have more projects stated during that time. Others have less projects starting at that time. However, frequency does not mean success.



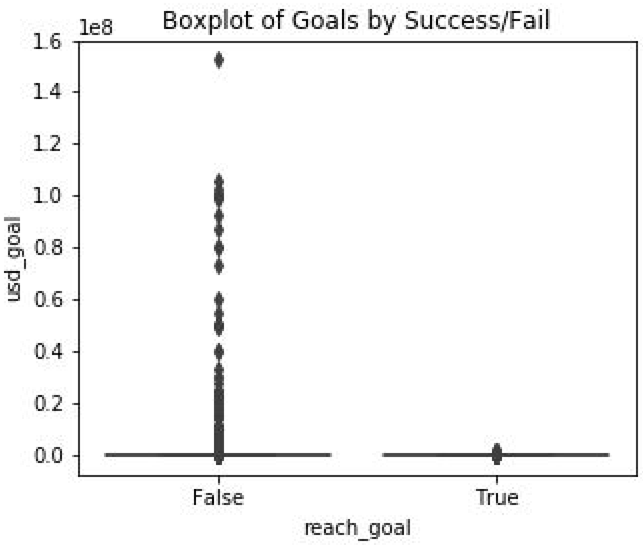
Taking a look at the success rate by month shows that projects started during March are the most successful while projects starter in July are the least. Deadline date also plays a large part in success rate where projects ending on April are the most successful.

Projects by Goal (USD)

Let's start by exploring the projects who have reached their goal and those who have not. A good comparison between them would be the Goal (USD) that each project has. Let's look at a boxplot comparing the two. For clarity, we've removed the outliers in the data.



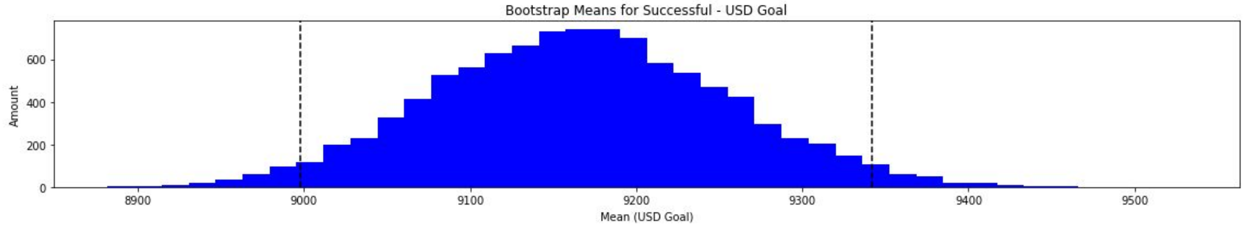
Here, we can see that the median Goal for successful projects is much less than projects that have failed. Intuitively this makes sense since smaller goals are easier to reach. Additionally, the range for projects that have succeeded is much smaller than the range for projects that have failed. If we take a quick look at the outliers, we can also see why the mean of failed projects is much higher (81,797.52) compared to successful projects (9,167.36).

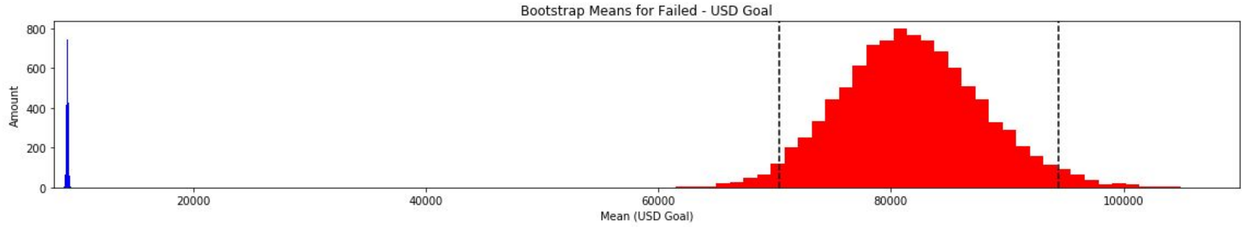


It’s clear that many of the projects that have incredibly high goals did not succeed, something definitely reasonable to expect. These are projects reaching 100 million.

Performing a t-test on the two groups, we get a value of -13.39 and a p-value of 6.49e-41. Since our p-value is close to 0, we can reasonably assume that there is a statistical difference between the goals of projects that fail and those that succeed.

Additionally, we can see this graphically in bootstrap replicates of the mean goal between successes and failures.

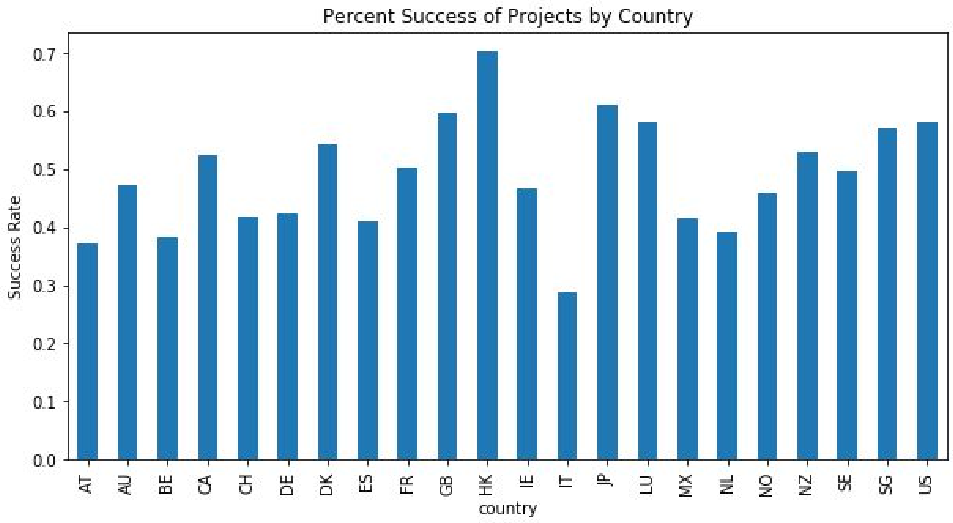




The first graph shows the set of bootstrap replicates for the mean of goals (usd) for successful projects. The second one shows the failed projects compared to the successful ones. We can see a large difference in their means. If we perform a hypothesis test and assume that their means are equal and no different, how likely are we to get the observed difference of means between successful and failed projects.

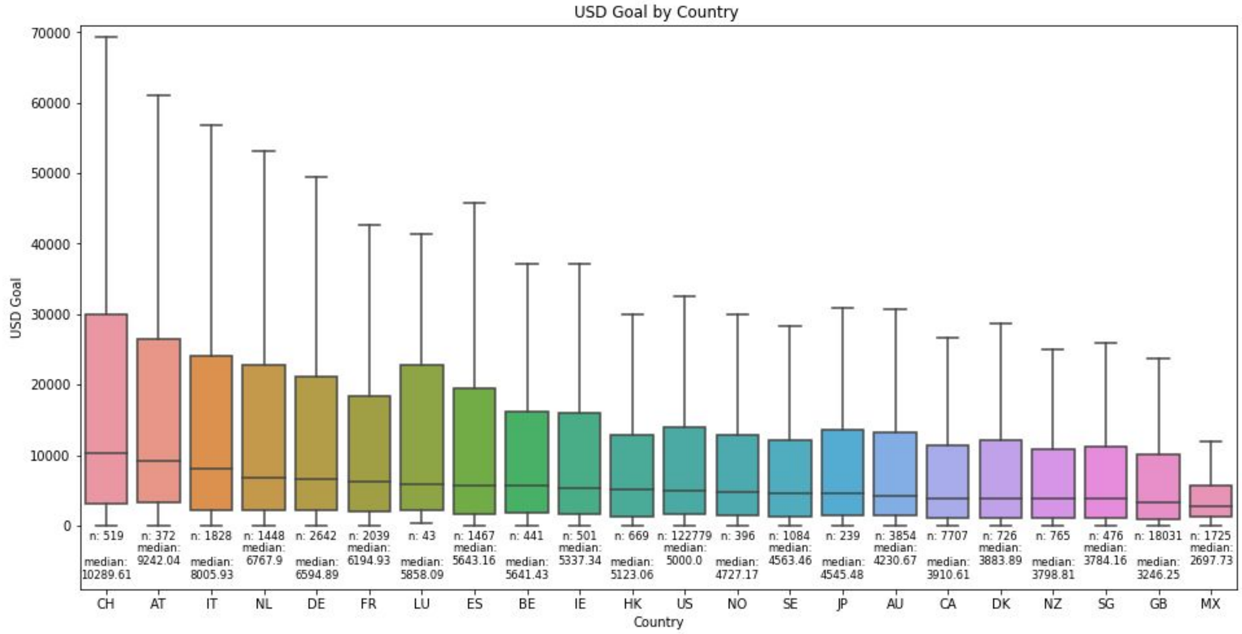
Projects by Country

Kickstarter is a global idea and so has projects being started all over the world. The main country where projects start out of is the United States where 72% of Kickstarter projects originated. However, does this necessarily mean that leads to a successful project?



By taking a quick look at the success rate per Country, we can see that the US is actually at around a 58% success rate. The highest one is Hong Kong with a success rate of 74.7%. The country with the lowest success rate is Italy with a success rate of 32.8%.

We can also explore countries in relation to their USD Goal.

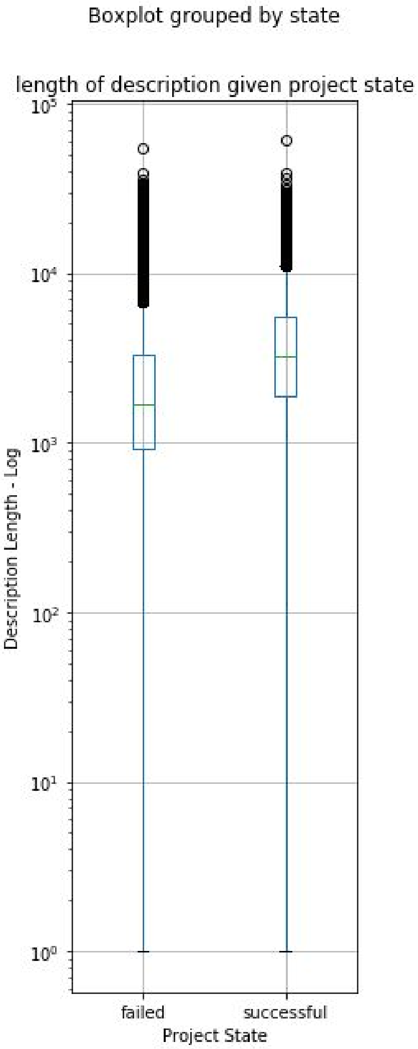


Here, we can see that China has the highest median goal followed by Austria then Italy. As the median goes down, we can also see the range decreasing.

Projects Descriptions - Length

Descriptions of projects are the best way to entice potential backers to support a Kickstarter. We can take a look at the lengths of descriptions and see if they determine success.

Here, projects that were successful have longer descriptions than projects that failed. The median description length for failed projects was 1564 characters compared to the 2782 characters for successful projects. Additionally, failed projects had a mean of 2434 while successful projects had a mean of 3773. Performing a t-test on the groups shows a p-value of 0.0. From this we can assume that there is a statistically significant difference between successful and failed projects.



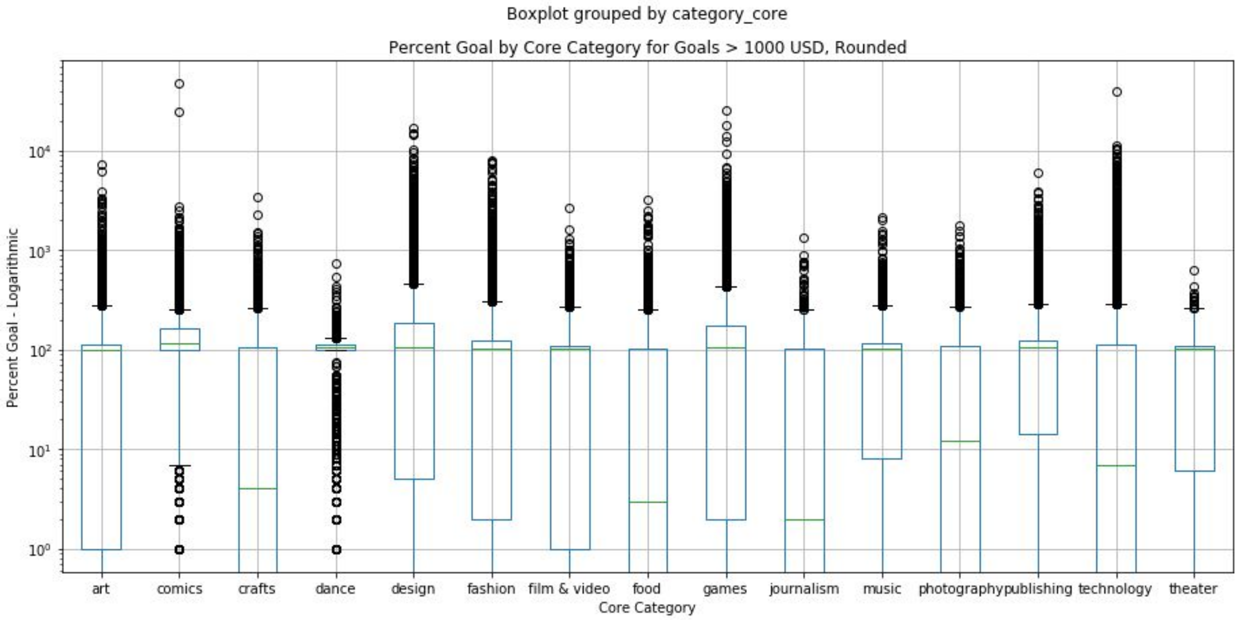
Projects by Category

Category - Percent Goal

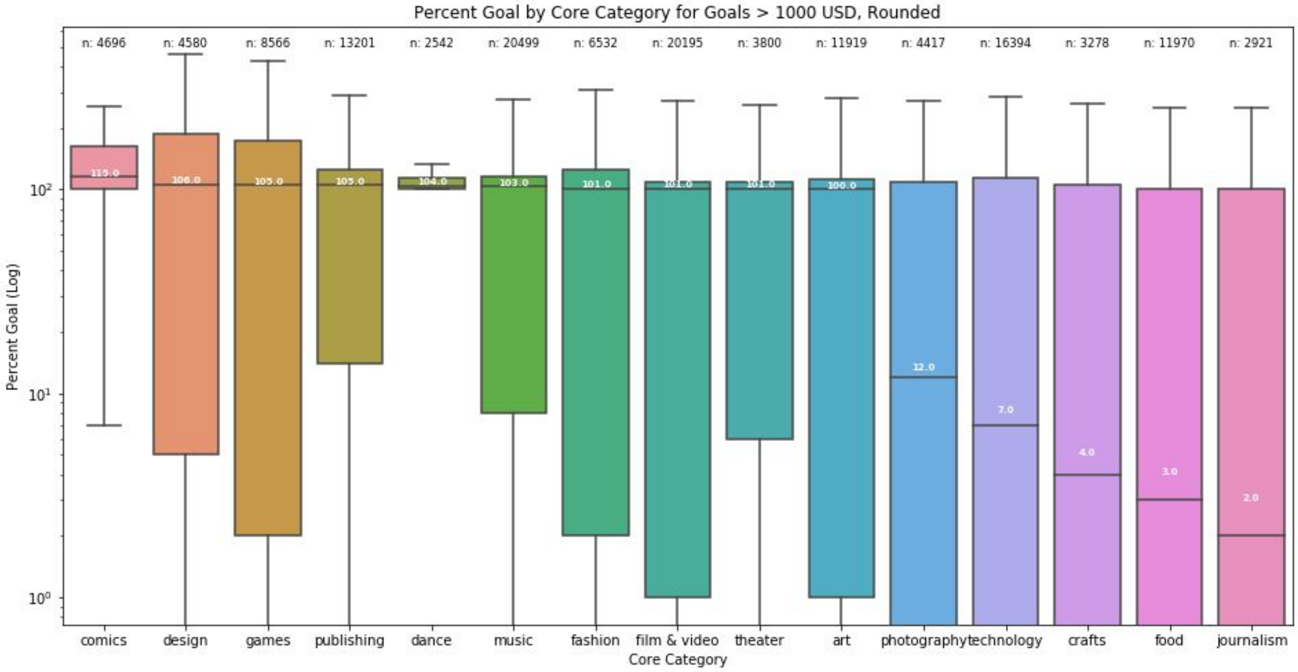
Each project on Kickstarter is broken down into one of 15 categories and further broken down into subcategories. These categories are selected by the creator and generally encompass what the project entails. These range from “art” to “food” to “technology”. Some may overlap such as “film & video” “dance” project. It’s up to the creators to distinguish where their project belongs.

Are certain categories more successful? Does the final product drive backers to the project? To start, we can look at the percent success of projects based on their category.

For most categories, it looks like their average percent to goal is 100, meaning they’ve hit their goal. For some, namely the crafts, food, journalism, photography, and technology categories, it looks like they’re more affected by other factors and bring their success rate down. Another interesting point is the dance category looks to center around 100% success rate. The highest ‘max’ between the categories look to be between design and games.



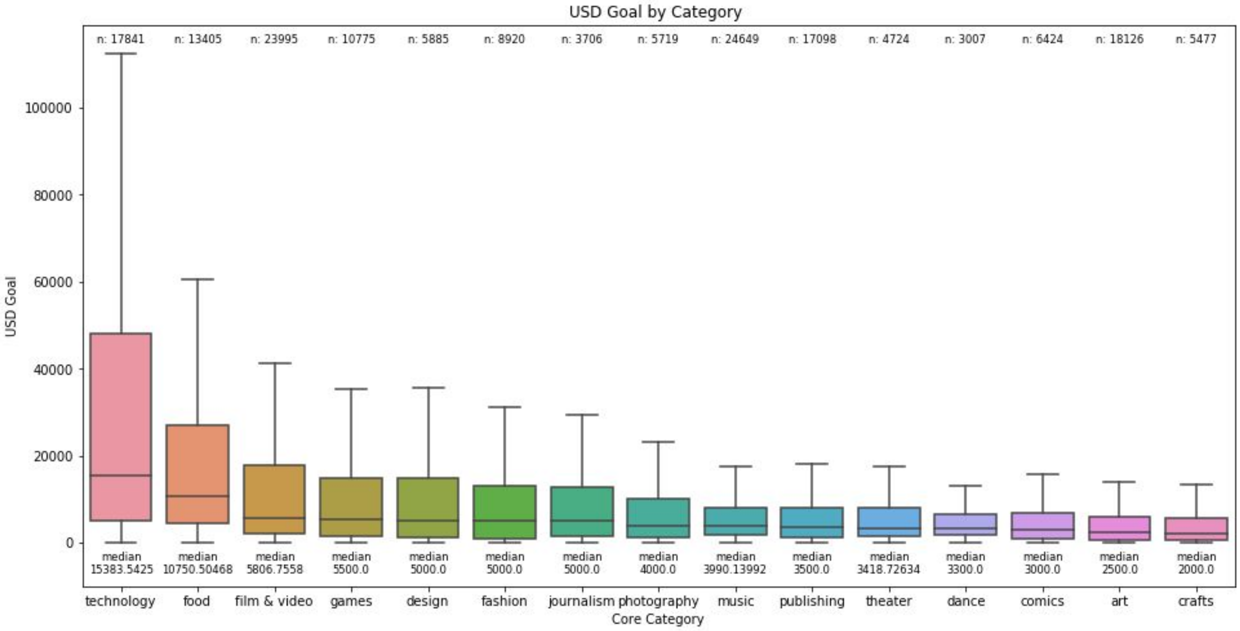
Let’s dive deeper into the categories.



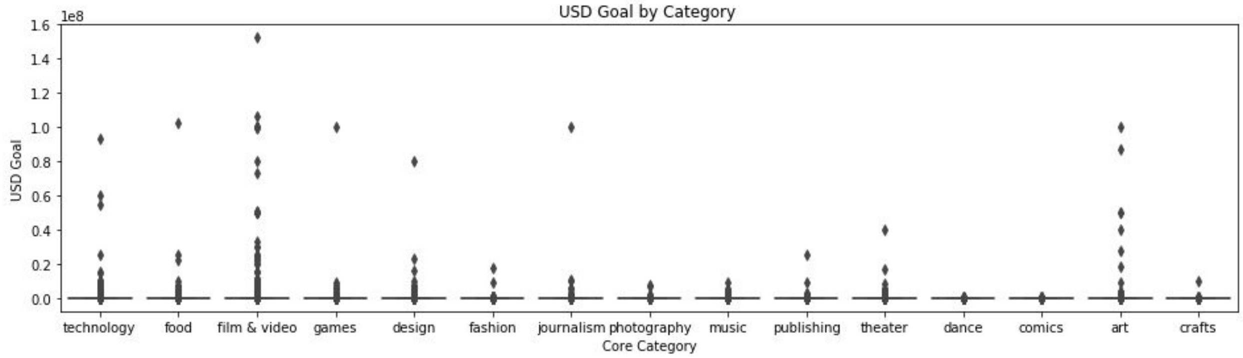
From this we can see that comics has the highest median success rate, followed by design and games. It's also interesting that these are some of the categories with the lower counts having smaller ranges compared to categories with higher counts with very large ranges.

Category - USD Goal

We'd also like to see how the categories reflect their USD Goals relative to other categories. We can do that with a boxplot.

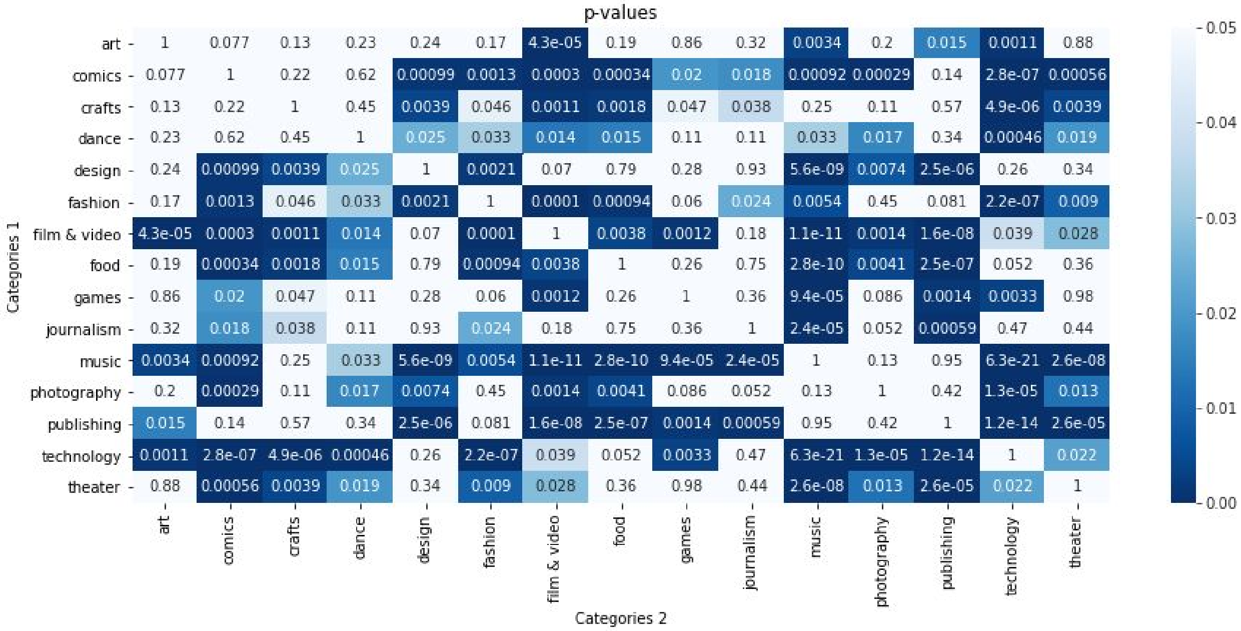


We can see that technology has the widest range and also the highest median goal. Other categories tend to be in the same range for their goals. We can also see that the size of each category fluctuates widely ranging from 24,000 to 3,000. A deep look into the categories would be to also look at their outliers and see how they affect each category.



It looks like Film & Video have the widest range of outliers. Technology also has a spread of outliers in the higher ranges. For a few categories such as food, games, and design, there look to be a cluster of data points with lower goals but have a few high reaching goals that may be affecting the mean. Art, similar to Technology, looks to also have a wide spread of data points in the higher goal range.

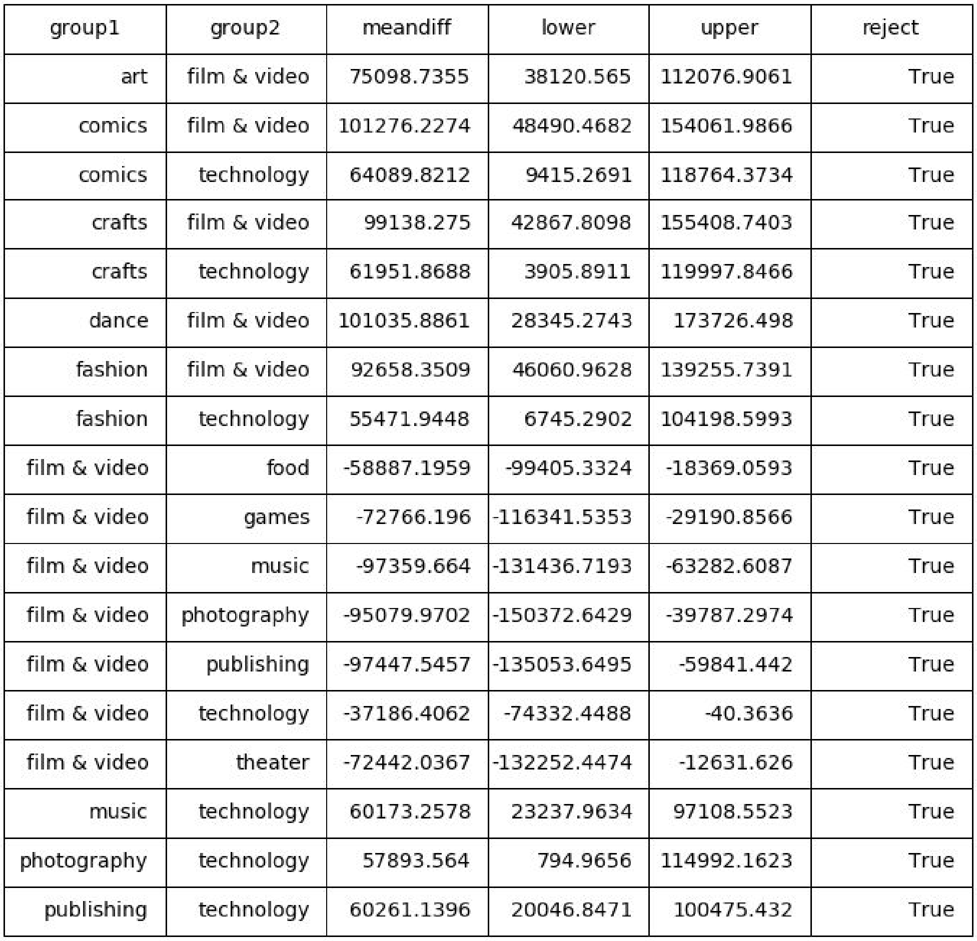
We'd also want to compare categories between each other to see whether or not the difference between them is significant in predicting success.



	art	comics	crafts	dance	design	fashion	film & video	food	games	journalism	music	photography	publishing	technology	theater
art	1	0.077	0.13	0.23	0.24	0.17	4.3e-05	0.19	0.86	0.32	0.0034	0.2	0.015	0.0011	0.88
comics	0.077	1	0.22	0.62	0.00099	0.0013	0.0003	0.00034	0.02	0.018	0.00092	0.00029	0.14	2.8e-07	0.00056
crafts	0.13	0.22	1	0.45	0.0039	0.046	0.0011	0.0018	0.047	0.038	0.25	0.11	0.57	4.9e-06	0.0039
dance	0.23	0.62	0.45	1	0.025	0.033	0.014	0.015	0.11	0.11	0.033	0.017	0.34	0.00046	0.019
design	0.24	0.00099	0.0039	0.025	1	0.0021	0.07	0.79	0.28	0.93	5.6e-09	0.0074	2.5e-06	0.26	0.34
fashion	0.17	0.0013	0.046	0.033	0.0021	1	0.0001	0.00094	0.06	0.024	0.0054	0.45	0.081	2.2e-07	0.009
film & video	4.3e-05	0.0003	0.0011	0.014	0.07	0.0001	1	0.0038	0.0012	0.18	1.1e-11	0.0014	1.6e-08	0.039	0.028
food	0.19	0.00034	0.0018	0.015	0.79	0.00094	0.0038	1	0.26	0.75	2.8e-10	0.0041	2.5e-07	0.052	0.36
games	0.86	0.02	0.047	0.11	0.28	0.06	0.0012	0.26	1	0.36	9.4e-05	0.086	0.0014	0.0033	0.98
journalism	0.32	0.018	0.038	0.11	0.93	0.024	0.18	0.75	0.36	1	2.4e-05	0.052	0.00059	0.47	0.44
music	0.0034	0.00092	0.25	0.033	5.6e-09	0.0054	1.1e-11	2.8e-10	9.4e-05	2.4e-05	1	0.13	0.95	6.3e-21	2.6e-08
photography	0.2	0.00029	0.11	0.017	0.0074	0.45	0.0014	0.0041	0.086	0.052	0.13	1	0.42	1.3e-05	0.013
publishing	0.015	0.14	0.57	0.34	2.5e-06	0.081	1.6e-08	2.5e-07	0.0014	0.00059	0.95	0.42	1	1.2e-14	2.6e-05
technology	0.0011	2.8e-07	4.9e-06	0.00046	0.26	2.2e-07	0.039	0.052	0.0033	0.47	6.3e-21	1.3e-05	1.2e-14	1	0.022
theater	0.88	0.00056	0.0039	0.019	0.34	0.009	0.028	0.36	0.98	0.44	2.6e-08	0.013	2.6e-05	0.022	1

Using a 5% threshold, we can reject the hypothesis that the categories have similar average usd goals. This shows that simply comparing successful and failed projects as a whole is not enough to tell a complete story and that we would need to pay attention to the category of the project. If we performed a one way ANOVA test on our USD Goals with a null hypothesis that mean usd goals between categories are the same, we'd 2.185e-26, well below an alpha of

0.05. Since we're rejecting the null hypothesis, we can further explore to determine which categories differ the most by performing a Tukey test.

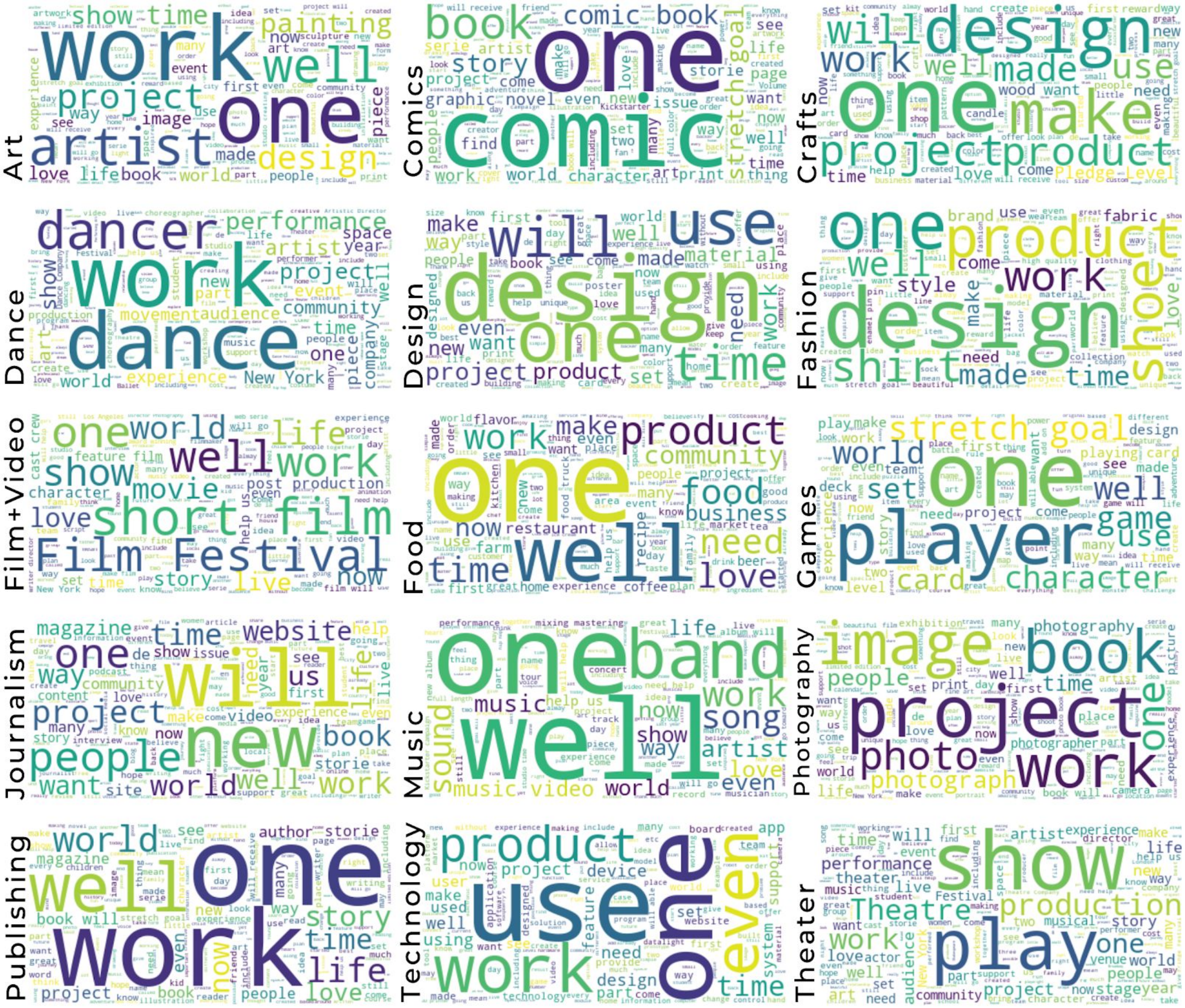


group1	group2	meandiff	lower	upper	reject
art	film & video	75098.7355	38120.565	112076.9061	True
comics	film & video	101276.2274	48490.4682	154061.9866	True
comics	technology	64089.8212	9415.2691	118764.3734	True
crafts	film & video	99138.275	42867.8098	155408.7403	True
crafts	technology	61951.8688	3905.8911	119997.8466	True
dance	film & video	101035.8861	28345.2743	173726.498	True
fashion	film & video	92658.3509	46060.9628	139255.7391	True
fashion	technology	55471.9448	6745.2902	104198.5993	True
film & video	food	-58887.1959	-99405.3324	-18369.0593	True
film & video	games	-72766.196	-116341.5353	-29190.8566	True
film & video	music	-97359.664	-131436.7193	-63282.6087	True
film & video	photography	-95079.9702	-150372.6429	-39787.2974	True
film & video	publishing	-97447.5457	-135053.6495	-59841.442	True
film & video	technology	-37186.4062	-74332.4488	-40.3636	True
film & video	theater	-72442.0367	-132252.4474	-12631.626	True
music	technology	60173.2578	23237.9634	97108.5523	True
photography	technology	57893.564	794.9656	114992.1623	True
publishing	technology	60261.1396	20046.8471	100475.432	True

Taking the results of the Tukey test and looking at those that it rejects based on an alpha of 0.5, we can see that film & video looks to be the most different to other categories in terms of mean usd goals. This is followed by technology which make up the other comparisons in this list (group1 and group2 include either tech or film/video). Several factors could be playing into this but it's worth noting that technology and film & video are one of the highest size categories that also have one of the higher ranges of usd goals.

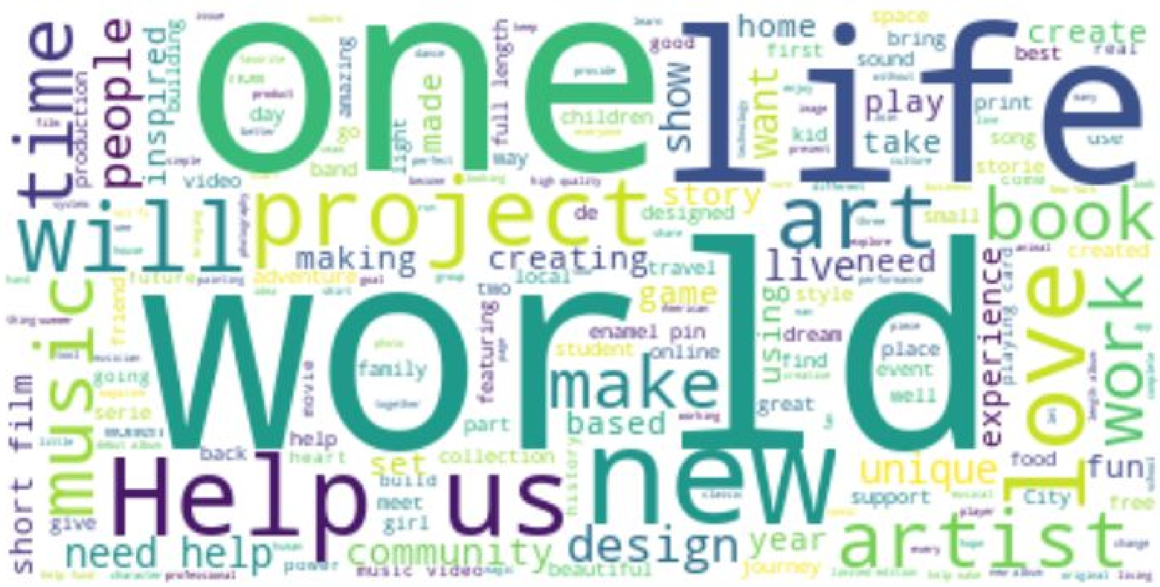
Projects on Word Choice

We can also take a look at the categories and their most used description terms.



From this we can see some key terms that each category uses and some terms that are shared across categories. Immediately, we can see the term “one” being shared across multiple categories, regardless of relation. Taking a look, it seems as if the descriptions use “one” to describe their product in a way it stands above others (i.e. “one of a kind”, or “one perfect brew”). There are also other instances where one is just a passing term not meant to have any meaning related to the product (i.e. “one day”, or “one last thing”).

Some categories that require creativity use the word “design” seen in the design, fashion, and art categories. Some words like “app” or “device” live solely in the Technology category and words like “recipe” and “farm” belong to the food category.



In addition to descriptions, Kickstarter projects also include blurbs as a way to entice people to click into their project.

Similar to the descriptions, “one” becomes a key term used in a lot of these blurbs. Again, it’s use varies but most use it to describe their project in the superlative (“one of the most”).

The blurbs also use the word “world” frequently. The usage generally falls into two categories, the world as a whole or the world of the product they want to enter. Things like “world premier” or “around the world” describes the first category while phrases like “word of wine”, “art world”, or “word of sugar” describe the second.

Additionally, we’d like to explore the pull that each word has towards a certain category. In other words, given a category, what words are most likely to relate to that category? We can do that by using Multinomial Naive Bayes.

ART: enamel | 0.93 pin | 0.92 sketchbook | 0.92 coloring | 0.91 78 | 0.90	COMICS: issue | 0.96 collected | 0.95 collection | 0.93 high | 0.93 face | 0.93	CRAFTS: plush | 0.91 enamel | 0.88 timeless | 0.84 giant | 0.82 orphan | 0.82
DANCE: work | 0.96 collaboration | 0.96 concert | 0.96 length | 0.95 premiere | 0.95	DESIGN: watch | 0.95 leather | 0.95 pen | 0.95 pocket | 0.95 bag | 0.94	FASHION: adventure | 0.90 wallet | 0.88 tote | 0.87 enamel | 0.87 anti | 0.86

FILM & VIDEO: portrait | 0.93 documentary | 0.93 stretch | 0.92 funeral | 0.88 refugee | 0.88	FOOD: keto | 0.75 bitter | 0.72 knife | 0.71 butcher | 0.70 iconic | 0.69	GAMES: 28mm | 0.97 novel | 0.95 miniature | 0.94 5e | 0.94 visual | 0.93
JOURNALISM: winning | 0.77 annual | 0.76 och | 0.76 award | 0.75 edition | 0.72	MUSIC: folk | 0.94 heading | 0.94 alt | 0.93 printing | 0.93 roll | 0.92	PHOTOGRAPHY : muse | 0.88 audience | 0.88 mature | 0.88 monograph | 0.86 contain | 0.84
PUBLISHING : letterpress | 0.93 mountain | 0.92 ocean | 0.91 coast | 0.91 picture | 0.91	TECHNOLOGY : oscilloscope | 0.85 cortex | 0.82 toothbrush | 0.79 ruler | 0.79 nixie | 0.77	THEATER : identity | 0.88 installation | 0.87 produced | 0.87 satire | 0.87 cycle | 0.86

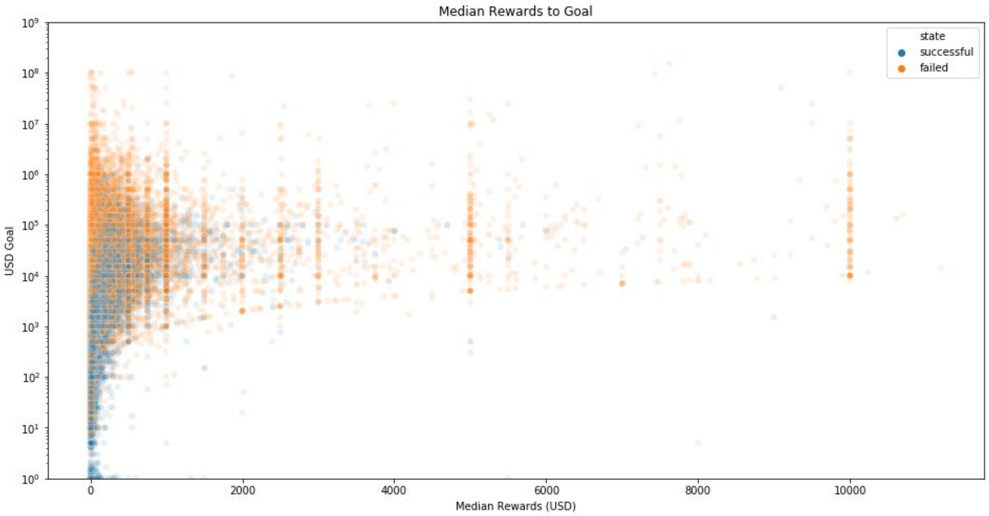
For the most part, most of the categories and their top 5 words make sense. They pretty much describe the category that they're in (i.e. the probability of “art” given that the word is “enamel” is 93%). It's surprising to see that not all categories have at least a few strongly predictive words (i.e. words giving probability > 90%). The category Food's most predictive word is “keto” at 75% and Journalism's most predictive word is “winning”.

Additionally, there are words that seem to appear in multiple categories such as “enamel” showing up in Art, Crafts, and Fashion. This makes sense since enamel can be used in all three categories.

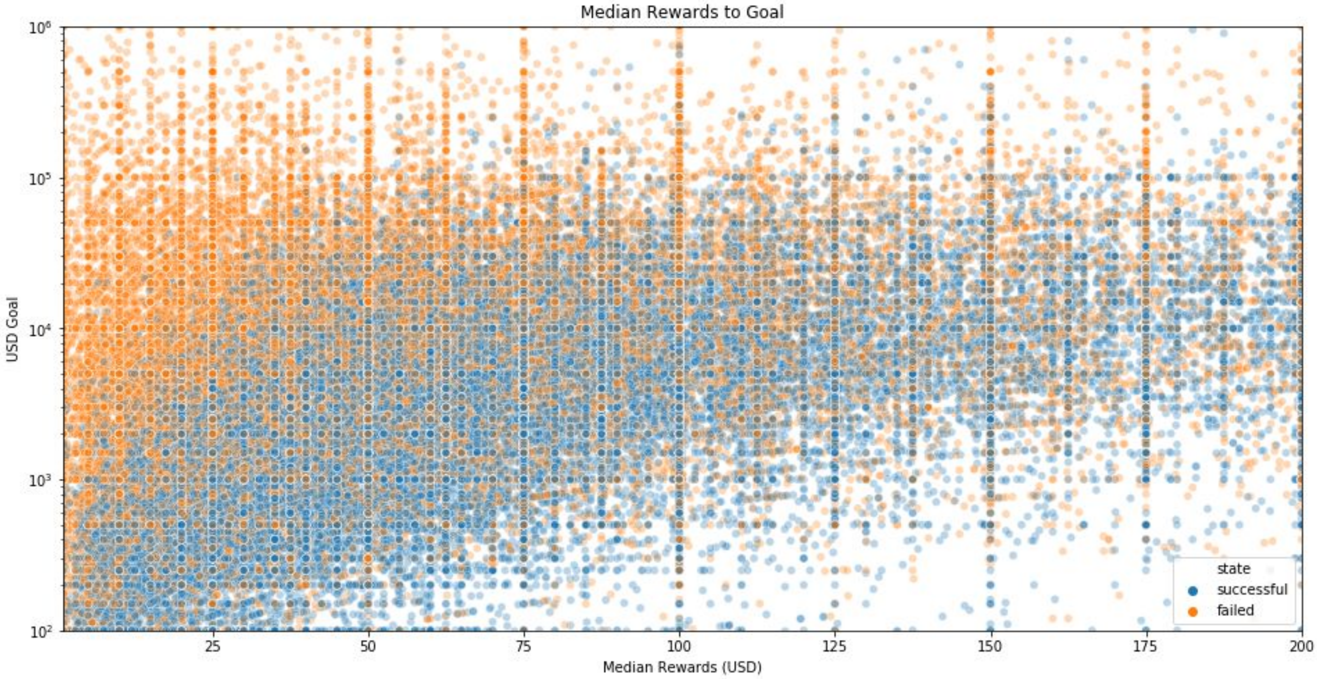
From this we can see trends that are successful in each category. For example, Games has “miniature” and “5e” as highly predictive words for success. 5e relates to the fifth edition of Dungeons & Dragons (D&D), a popular story driven rpg game. “Miniature” is sometimes associated with D&D as some players like to use them to help improve their gameplay. The following is so positive that these projects tend to do better.

Projects by Rewards

Let's explore how rewards for Kickstarters can affect their progress.

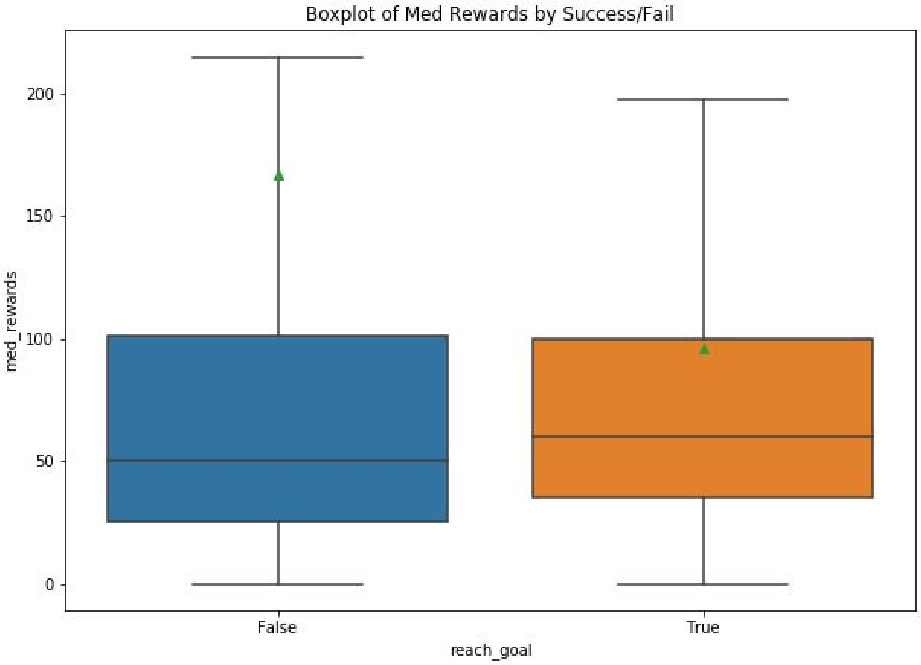


Here we've put the median reward price in relation to their goal. We can see that most Kickstarters that have a median reward of over 4000 usually fail. The median rewards over 200 actually encompass such a small amount of Kickstarters. Let's zoom in and see if we can focus on a core percent of projects.



Here, we're only looking at projects with a median reward less than 200. Right away, we can see that median rewards less than 25 and goals more than 10,000 usually fail. There's a nice curve of success vs fail as we increase in median rewards. It makes sense that if you have small reward prizes, it'll be hard to reach bigger goals. However, it's not all successful since there's still patches of failures between the successes if you're in the higher median v goal area.

We can also explore the differences between whether successful projects have a different range of median rewards compared to failed projects. Looking at their boxplots, we can see that for the most part, they look similar in their offerings.



The medians for failed and successful projects are 50 and 60, respectively. We can also see that their 25 and 75 quartiles are similar with successful projects have a slightly smaller range. What is noticeable is that their means (denoted in green triangles) are vastly different. This could be due to outliers for failed projects having median rewards higher than expected.

Performing a t-test on the groups, we get a p-value of 9.05e-269. From this, we can conclude that there exists a statistical difference between median rewards of successful projects and failed projects.

In-Depth Analysis of ML Models

The model creation process covered many different portions to create an acceptable algorithm to work with. From scoring, to feature selection, to tuning, each algorithm had their own different method. Here, we'll be exploring how the process lead to the most acceptable model and what future processes we could work with to improve the model. We will be exploring these models in several steps:

1. Base Model Exploration
2. Model Tuning
3. Model with Text Exploration
4. Thresholding

Before we begin, we need to determine our scoring system. This is reflected from our use case. The goal is to provide Creators the information needed to make their project successful. Simply predicting success does not give actionable insight since if their project is predicted to be successful, they would not feel the need to change something. If we predict failure, however, it will give the Creator more reason to look over their Kickstarter project once more to see what can be improved.

From this, we want to emphasize recalling all the failed projects. We could just use the recall itself but we still want to keep track of the projects that were predict successful but were actually failures. This would be done by using the precision of the model. A good balance is the fbeta score, a combination of the two scoring systems. Since we still want to emphasize recalling the failed projects, we'll be utilizing a beta higher than 1. For this project, it will be a beta of 1.1.

Base Model Exploration

To start, we'll be looking at the 'base' algorithms of several models, meaning we won't be tuning them. We'll be exploring Logistic Regression, Linear SVC, K Nearest Neighbors, and Random Forests.

The models performed as follows:

Model	Score
Logistic Regression	69.63%
Linear SVC	68.92%
KNN	62.34%
Random Forest	68.56%

It looks like the Logistic Regression is our best performing base model. With tuning, we know that all of these models can be improved. However, the Linear SVC was failing to converge. With increased iterations, up to 4000, it still failed. This could be that our dataset is not fit for a linear SVC. Additionally, the KNN prediction took 8min 46s. Since our dataset is large, this is not an optimal algorithm. Hence, we'll be focusing on Logistic Regression and Random Forests in our tuning.

Model Tuning

Logistic Regression and Random Forests have different parameters so we'll be tuning them differently. We'll be using a GridSearchCV on the Logistic Regression classifier since we're optimizing 'C'. For Random Forests, we'll be using the RandomizedSearchCV since we're tuning 7 hyperparameters with up to 1,980 combinations. RandomizedSearchCV allows us to hop from combinations to find the best one, rather than test all 1,980.

After, we take the tuned parameters through a k-fold cv to ensure our score.

The models returned:

Model	SearchCV Score	KFoldCV Score
Logistic Regression	69.60%	69.77%
Random Forest	72.84%	72.75%

From this, we can see that the tuned Random Forest classifier does the best between the two models. As such, we'll be using these parameters for our final classifier.

Modeling with Text Exploration

To improve our score, we'll be employing a stacking ensemble method. With this, we can utilize the text (the blurbs and descriptions) in our dataset. This can be done in two ways. We can explore the text of the dataset all together or explore it based on the categories since word usage by category differs.

The process of the text analysis is as follows:

1. Preprocess Text
2. Vectorize using TfIdfVectorizer
3. Classify using MultinomialNB
4. Add prediction to original dataset

Through this method, we'll end up with four additional columns: a classifier on all rows and a classifier per category, for both blurbs and descriptions. The multinomialNB classifier either uses the training set for the whole dataset or the training set for each category. With these new columns, we can add them as features to our dataset and utilize the prediction scores in our tuned Random Forest classifier. Here, we'll look at the differences between the "all" classifier and the "category" classifier.

Using our tuned Random Forest Classifier, the results are as follows:

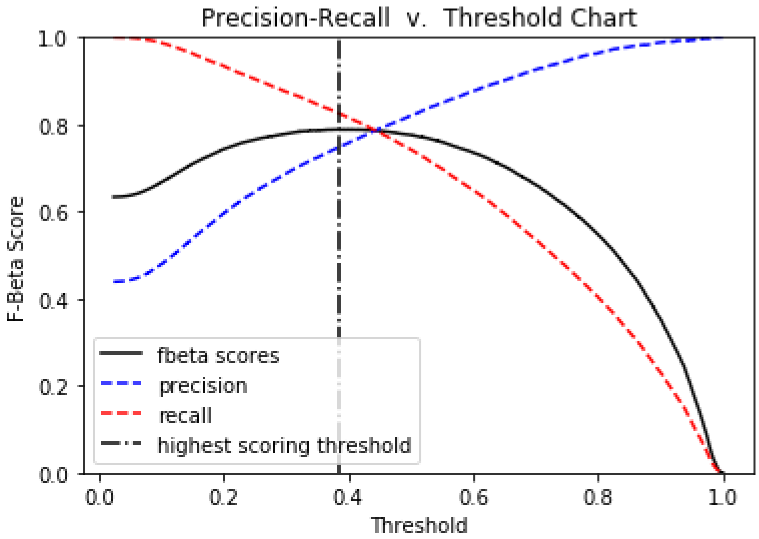
Features Selected	Training Score	Test Score
“All” column	94.27%	75.68%
“Category” column	94.81%	79.68%

The classifier utilizing the multinomialNB on categories does better than the multinomialNB on all the rows. This is easy to reason since categories are a large part in the success of a project.

Thresholding

Our classifier has a base threshold of 05, meaning that when a project is run through the model, it is given a probability of it failing and if it is equal to or higher than 0.5, it will fail. However, we are not sure if this is the correct place to threshold the model. If we want to maximize our fbeta score, there may be a better threshold to capture more of the Failed projects.

We can determine the best threshold by using the precision_recall_curve function and using the precision and recall to calculate the fbeta score (given our previous beta of 1.1). We find that our maximum fbeta score is 78.848% given a threshold of 0.3822. We can plot these points and see how they compare to each other.



We can apply this new threshold to our tuned Random Forest classifier.

Model	Score	KFoldCV Score
Random Forest	80.75%	80.56%

And looking at how this compared to the rest of the models, bolding the accepted model taken at each step:

Model	Score
Base KNN	62.34%
Base Random Forest	**68.56%**
Base Linear SVC	68.92%
Base Logistic Regression	69.63%
Tuned Logistic Regression	69.77%
Tuned Random Forest	**72.75%**
Tuned Random Forest - “All” Column	75.68%
Tuned Random Forest - “Category” Column	**79.68%**
Tuned Random Forest - “Category” Column - Threshold Adjusted	**80.56%**

Conclusion

The most optimal model in predicting failures is the tuned Random Forest classifier utilizing a stacking ensemble method from a ‘per category’ MultinomialNB classifier and an adjusted threshold. However, there can be improvements in the model overall. in the MultinomialNB layer of the ensemble, the preprocessing of the text can be improved. Non-english words were dropped, lemmatization could cause improper grouping, and proper nouns could be misclassified. Text preprocessing for this dataset could have been a paper on its own since it is such a diverse dataset.

Additionally, further analysis of the dates in a time series analysis could be performed to improve the model. Our features simply had the month and day the project started and finished. By including a more detailed dates analysis, it could pinpoint specific times to start or finish a project, along with how long a project should last.

As creators, there are many factors to consider before employing a successful Kickstarter project. First, the category in which your project exists can play a big factor. Some categories are more successful than others so there is an initial risk to factor in when creating projects in areas such as journalism or crafts.

The timing of the projects also play a big part. Starting projects in July is the least effective while finishing projects in January has the lowest success rate. Projects starting in March and projects finishing in April have the highest success rates for each. This does not mean to start a project in March and finish in April, this just shows that people are more willing to contribute to projects during these months.

The end price of the reward of a project, estimated from using the median of the rewards, also plays a big part. This is in relation to the goal that the project has. Projects with a median reward of $25 and a goal of $10,000 tend to fail. As the price of the reward increases, try to keep the overall goal from increasing too much so that the success chance improves.