Capstone Project 1 Kickstarter Projects Success Prediction Final Report

Zongkai Wu

1. The problem:

Kickstarter is a crowd-fund based website for creators to raise funds for their projects and in return provide their backers exclusive rewards and benefits. In this project, we are going to analyze previous kickstarter projects and provide predictions whether a new project will be successful or not.

The potential clients of this project include: kickstarter project creators, kickstarter backers and kickstarter company or similar crowd-funding companies.

For kickstarter project creators, by applying the most successful strategy from previous project creators, they will be more likely to meet the goal and raise as much funding as they needed to successfully finish their project.

For kickstarter backers, they could predict and analyze if a project they wanted to back are more likely to be successful or not, in return guide their decision of whether to back this project or not, and how much they should pledge.

For kickstarter company or similar crowd-funding companies, their revenue is directly decided by the success rate of their projects, since they take a 5% fee for every successful project. So for these companies, having a higher success rate will benefit both their revenue and attract more content creators to use their platform. Therefore, they can develop the most successful strategy to help increase the success rate on their platform. Focus their limited promotion resources on the projects that are most successful or need the help most to meet the goal.

2. Dataset:

The dataset we used in this project is from Web Robots, which uses a scraper robot to crawl all Kickstarter projects and collects data in CSV and Json formats. From March 2016 they run this data crawl once a month. In the project, we used the dataset from scrape date 2019-10-17, which is the most current dataset at the time this project started. Links to Web Robots kickstarter datasets: https://webrobots.io/kickstarter-datasets/

3. Initial findings:

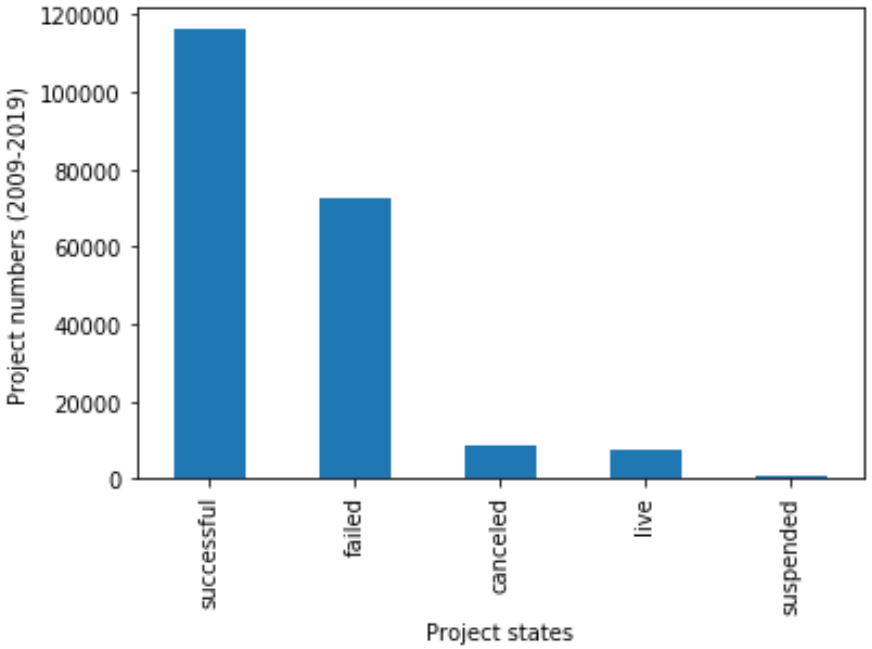

Figure 1. Project states distribution histogram

From the state distribution histogram, we can see there are more projects successful compared to failure, and canceled, suspended and live projects were much smaller compared to finished projects considering we included projects for five years, and most projects only last for 1 month or 2 month. Also there are 11k successful projects and 7k failure projects, which provide enough entries for our analysis and train models.

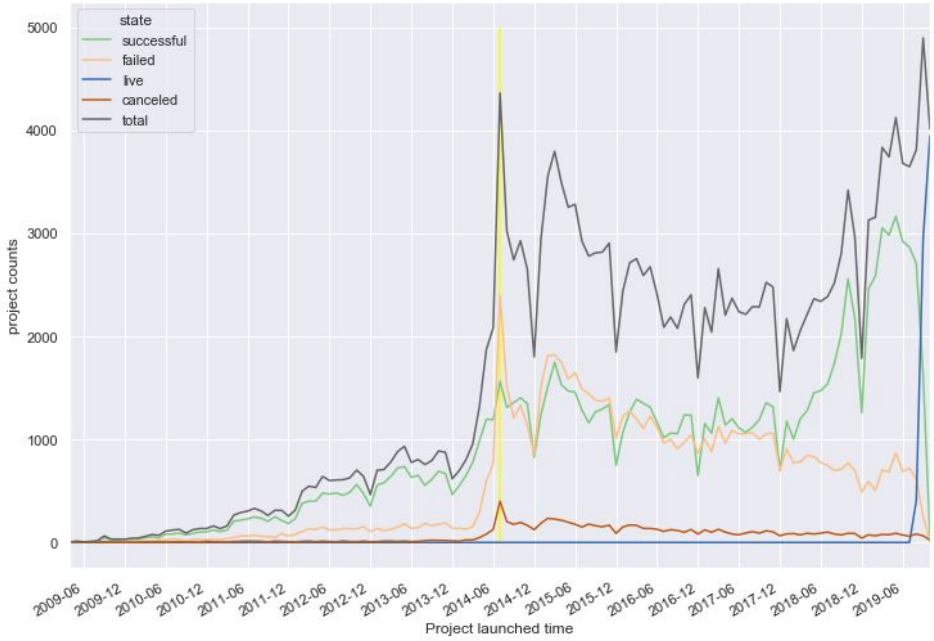

Figure 2. Time series graph with project counts with different states

From Figure 2, we notice a few interesting patterns for the project launch time series. 1) There is a clear project count increase around July of 2014 (labeled by yellow line). We couldn't find a record of what happened to the kickstarter website that month, but our suspicion is there was a large promotion that largely increased their traffic. It can also be seen that the successful project number and failed project number both showed a peak at this month, but it's more obvious for the failed case than successful. This indicates although the promotion at that month boost website traffic, it sacrifices success rate a bit at the same time. 2) There's a clear cycle pattern with project counts, each December launched project will show a clear drop, while at the start of year project numbers increase significantly, and peak around June or July. This pattern is clearer to total and successful project counts, and can still be seen in failed counts.

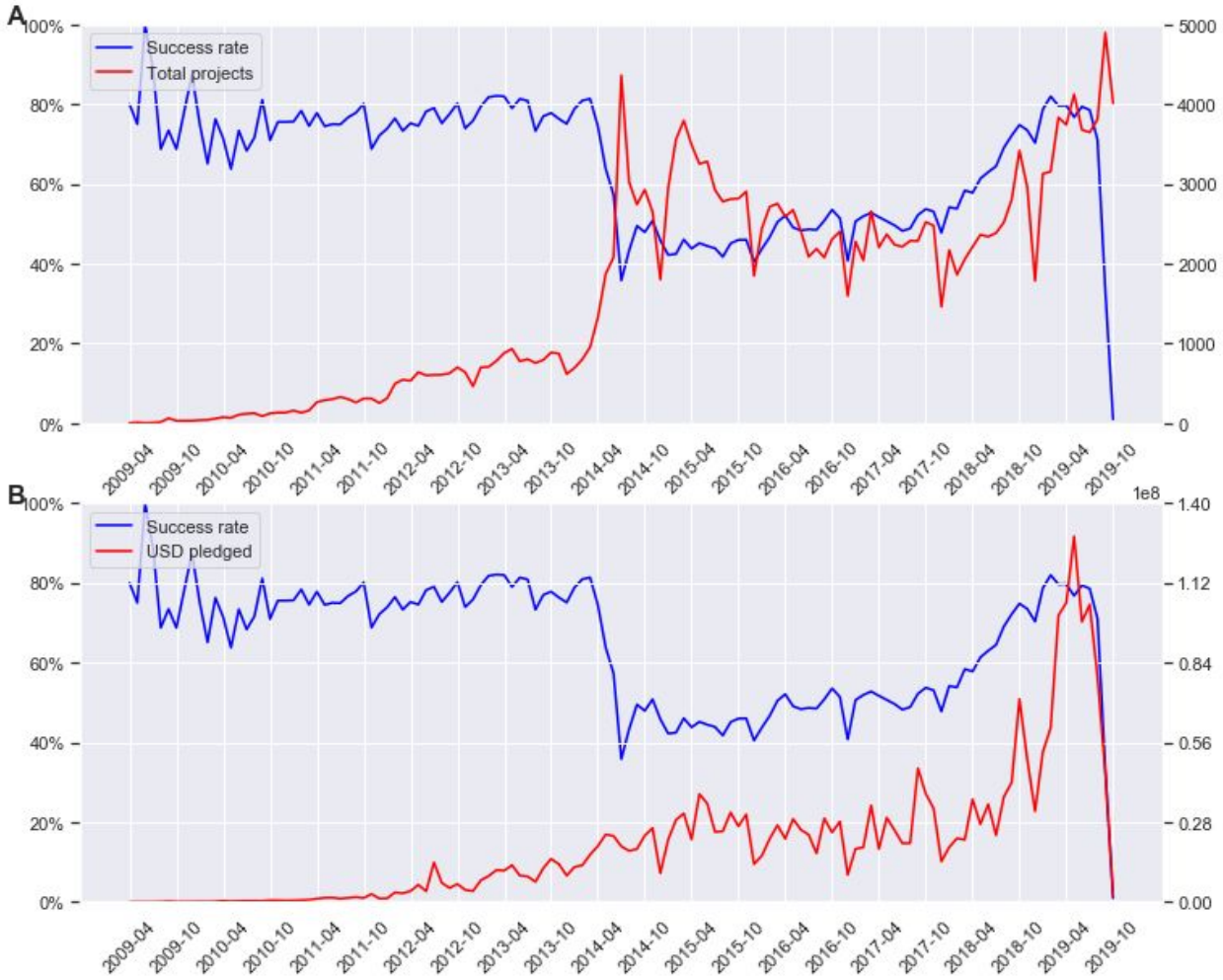

Figure 3. Time series A) Success rate compared to total project counts B) success rate compared to USD pledged

From Figure 3A, we got a similar conclusion from Figure 2, that during 2014-July, there's a clear increase in total project number, but decrease in project success rate, and success rate

gradually increase and get closer to previous level (~80%) in 2019. From Figure 3B, we can get an important message that although the success rate decreased a lot by 2014-July, the actual total USD pledged is actually increased due to the total project number increase. This value is more important for company revenue compared to success rate. Total USD pledged numbers remain relatively stable until 2018-Nov, which starts to see large increases again.

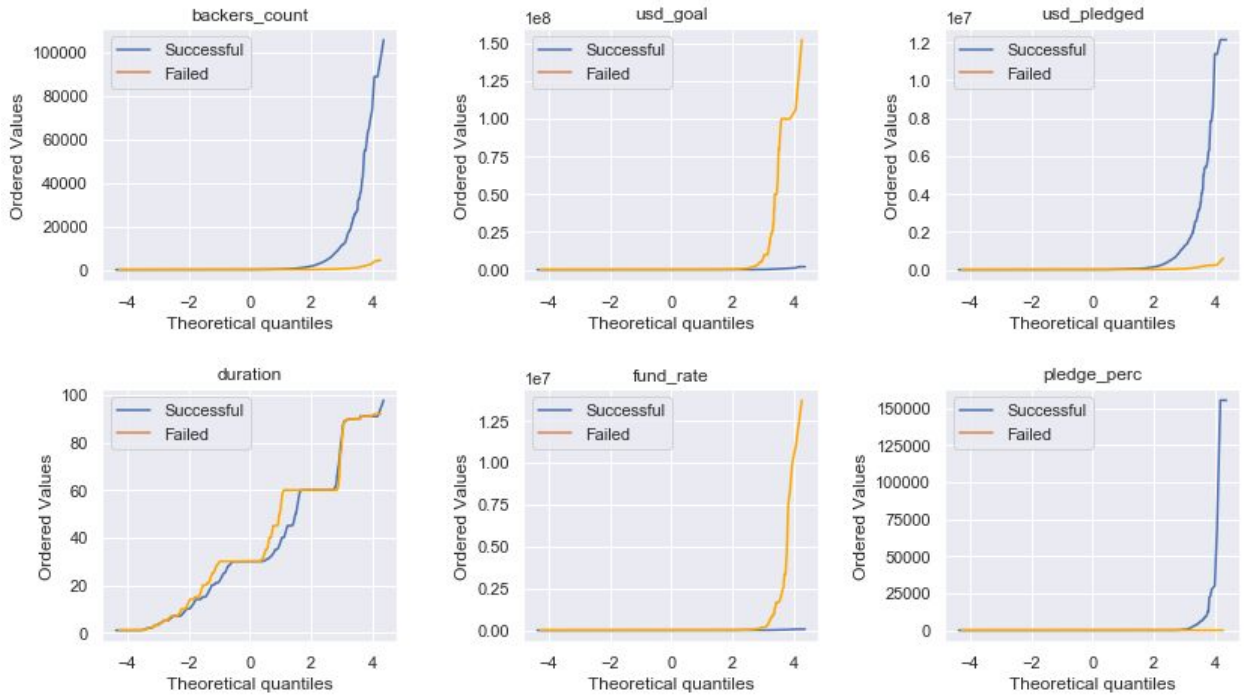

Figure 4. Probability plots for numeric columns

Quick histogram check for all numeric column distribution conditions. The result is not very clear because some columns have some far off outliers.I think use probability plots could show distribution clearer. Probability plot shows why for most numeric columns, histogram is not working.For most of the columns data, there's a very large range but most of the data (within 2 standard deviation, around 97%) are much small compared to the large data.Some interesting observation here is for fund rate and fund goal, failed projects tend to require higher amount and fund rate, which indicate it's more difficult for them to reach goal within target time. As for duration, there's no clear difference between successful and failed cases.

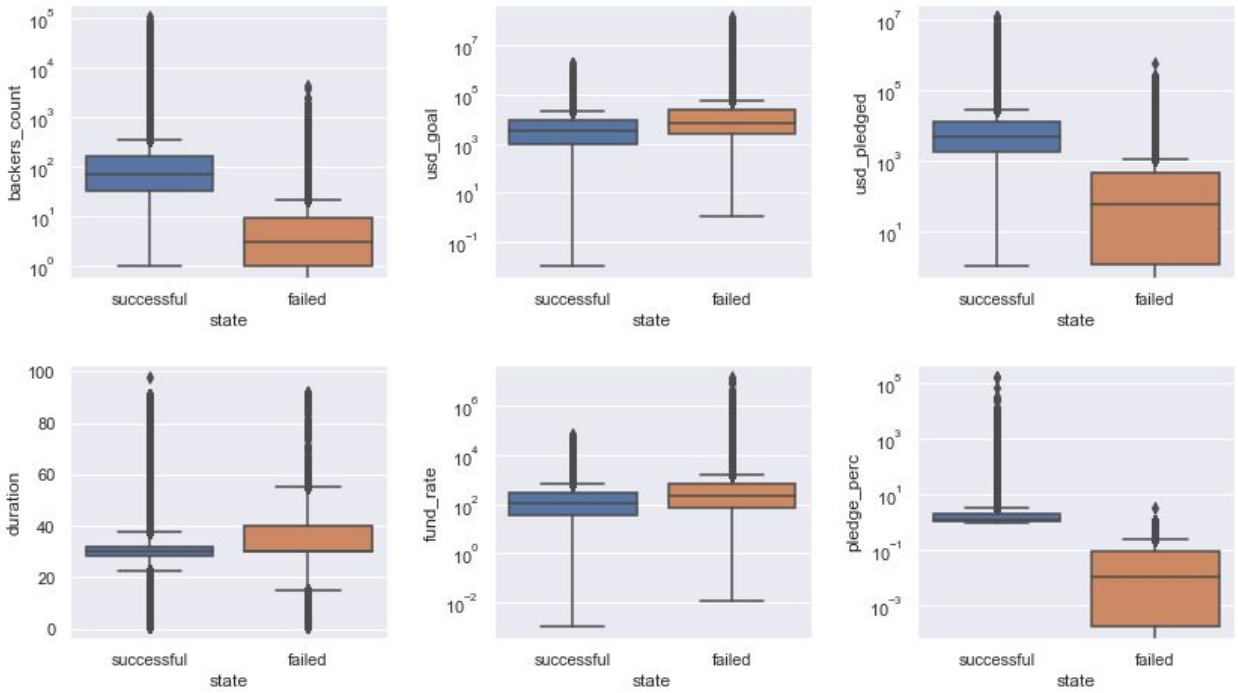

Figure 5. Box plots for numeric columns

Probability plot successfully shows the position of outliers and how far they can go for different states, but it doesn't provide a clear comparison due to the majority of the data being too small compared to the large outlier. So we use box plots, and take log scales for some data to show clear comparison. From these box plots, we can get several interesting conclusions, such as successful projects have more backers compared to failed one. Failed cases are likely to set higher goals, and interestingly also like to require longer time for funding. While projects with less fund request and shorter request time are more likely to be successful.

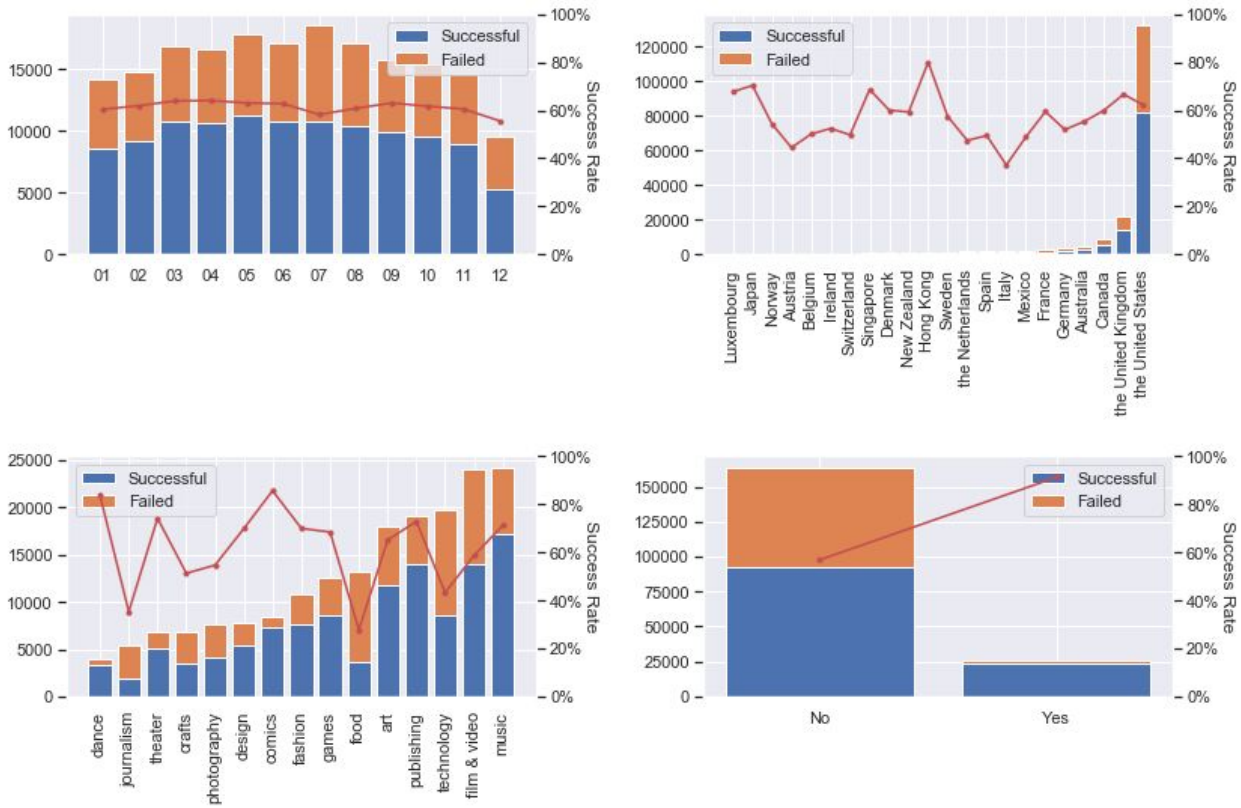

Figure 6. Stack bar plot for category columns

We use stacked bar plots to analyze category columns. From this figure we can see December has a smaller amount of projects launched compared to other months, while July has the most. Interestingly, the success rate was pretty stable but shows lowest at July and December, which are the months with most and least projects. For country distribution, most of the projects are from the US, second place is the UK, which only has around 1/6 of US project numbers. Third place is Canada, all other countries' project numbers are much smaller compared to the top 3.

4. Statistic analysis

Question 1: Is there a strong correlation between numeric columns of each project?

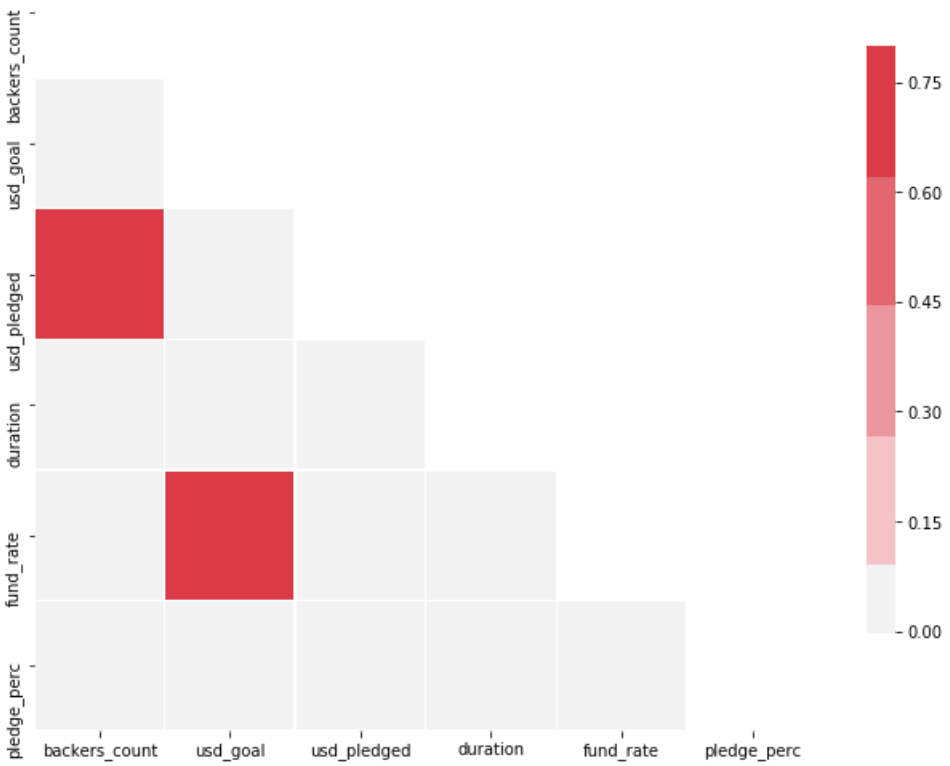

Figure 7. Correlation coefficient heat map for numeric columns

From the correlation coefficient heat map, we can see most numeric columns don't show strong correlation except total US dollar pledged with number of backers, and US dollar goal with fund rate, which is how much money need to be pledged per day to reach the goal. From our previous EDA session, we know the variation of duration is relatively small, so the higher the goal was setted, the higher the required fund rate will be, so this is easy to understand.

However, we are interested in analyzing if higher backers count really means a higher amount of money will be pledged, which we will use bootstrap statistic method to test in the next session.

Question 2. Does higher backer count lead to higher usd_pledged?

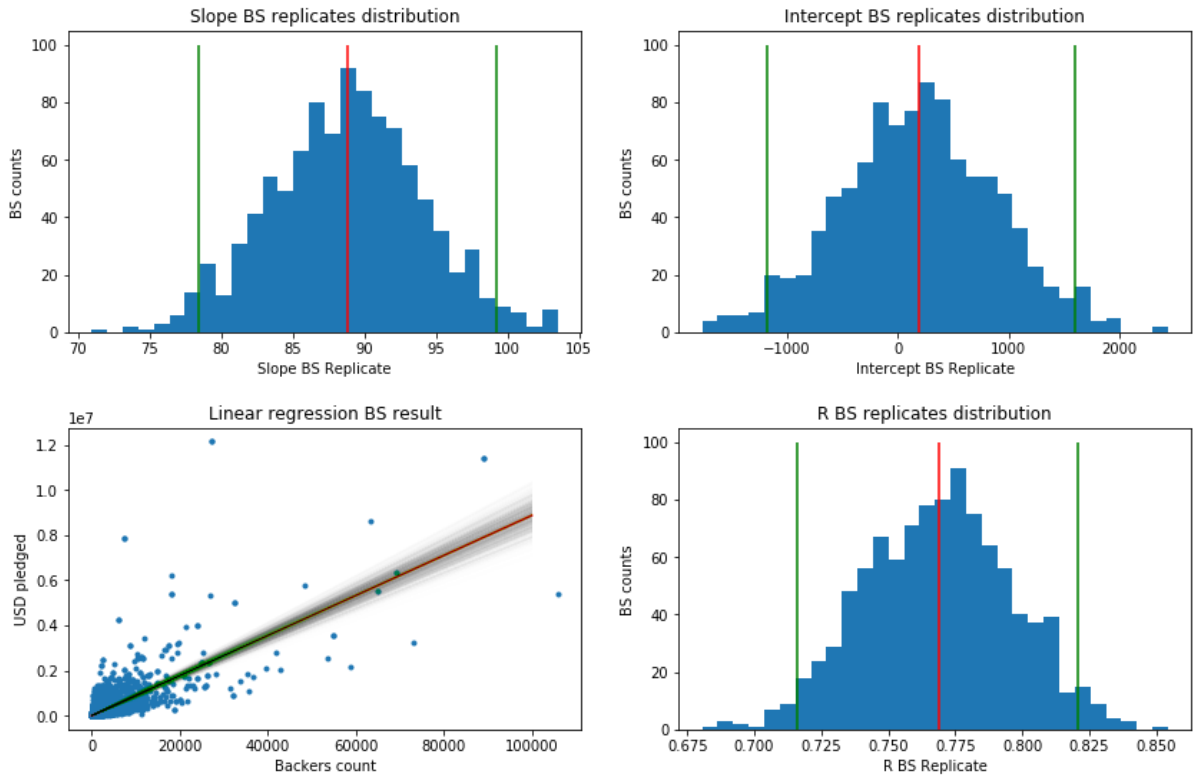

Figure 8. Bootstrap analysis of backers count and USD pledged linear regression and pearson coefficient

From the result we can see there's a clear linear relation between backers count and usd pledged amount. 95% confidence interval of pearson coefficient is around 0.71 to 0.82.

5. Prediction model

For prediction models, our goal is to predict whether a project can be successful or failure given the known condition of a project. For numerical factors, all information will be used separately as a different factor without interaction with each other, however, for categorical factors, we need to implant encoders to transform those information for machine learning classifier models to train.

Encoder:

I use two different methods to encode categorical features, one is one hot encoder, for which feature numbers increase as each unique category entries, so we only use 'country', 'main_cat' and 'loc_state' as features for this encoder. To keep more features, we use binary encoder from category_encoder, which allows us to keep more features while limiting the dimension of the dataset. Binary encoders perform ordinary encoders on categorical features first, then convert each number to binary code and separate into different columns.

Pick features for the training set. Numeric column we chose duration, usd_goal, year and month, we dropped backers counts as it can only be confirmed once the project is finished. For boolean columns we pick disable_communication, is_starrable and staff_pick. For categorical

columns, we will use two seperate encoders to encode categorical columns, one is one hot encoder and the other is binary encoder. For one hot encoder, due to its expanding dimension according to the number of unique combinations, we only pick country, main_cat and loc_state, which provide location and category information with relatively small unique entries as shown below. Those columns were stored in cat_col. For binary encoder, each categorical column gets ordinal encoded first, then convert the number to binary code and use each digit as feature, this way cancel out the ordinal features to avoid providing false ordinal information to the model. Those columns were stored in cat_col2.

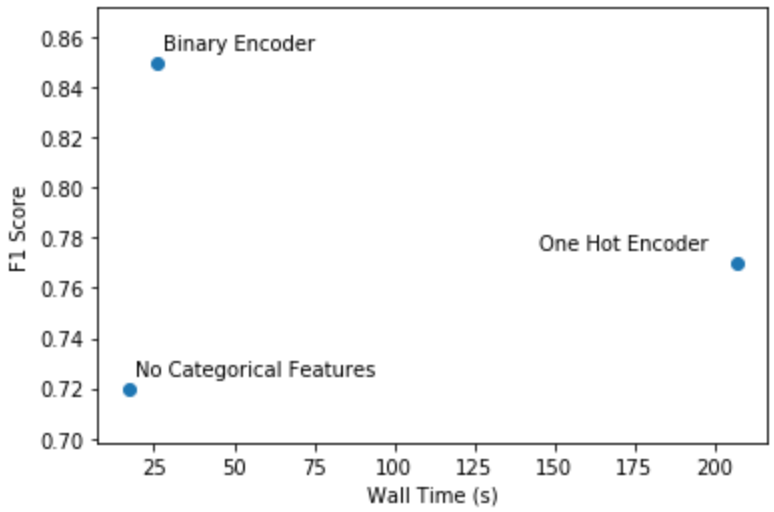

Figure 9. Encoder comparison

From the above result, we can clearly see that binary encoder shows much better F1 score than One hot encoder and without categorical features, while only taking slightly longer wall time compared to non-categorical features conditions. Using one hot encoder in this case takes the longest time with random forest, due to dimension is too high, while its F1 score is only slightly better than non-categorical condition, due to include less features compared to binary encoder condition. Therefore, for the rest of this project with model comparison, we use binary encoder dataset as our data. Also added cat_col_b's feature names to features list for future visualization use.

Models:

In this project, we decide to use 6 classifier models for test and comparison. Models are random forest, svm, knn, logistic regression, gradient boosting, naive bayes. We did hyper parameter tuning for all models and performed cross validation to check the result for all models. Results are stored in pandas dataframe results using the defined function store_result for all models.

Overall comparison:

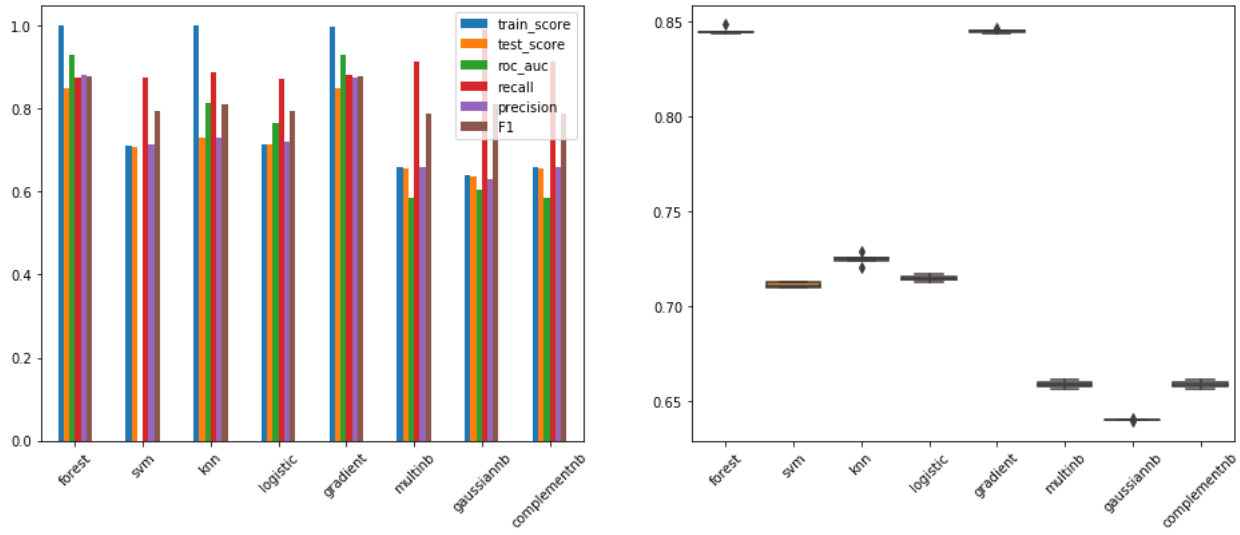

Figure 10. Model score comparison

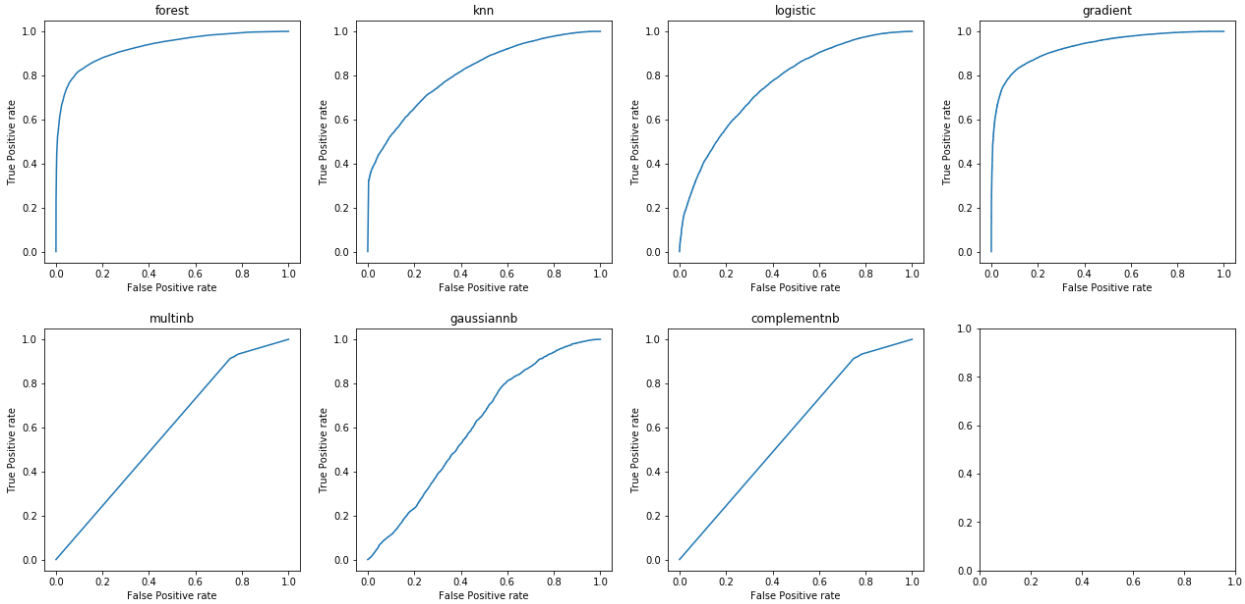

Figure 11. Model roc comparison

From the above graphs, we can see ensemble methods are the clear winner in terms of prediction ability as both Random forest and gradient boosting model shows accuracy of around 85%, much higher than any other models (75%) on testing dataset. Logistic regression and naive bayes methods show relative poor accuracy, but the prediction speed is much faster compared to ensemble methods. KNN and SVM predict results are also poor, but they take much longer to predict compared to logistic regression or naive bayes. Among the three naive bayes methods, multinomial and complement naive bayes show the exact same result, and both perform slightly better than gaussian naive bayes.

Based on these test results, we picked random forest as our model since it not only gave the highest accuracy score (slightly higher than gradient boosting), but also took much time compared to gradient boosting.

We performed hyperparameter tuning for all 6 models, details can be viewed in notebook: https://github.com/zokwu/Capstone1_kickstarter/blob/master/Machine_Learning.ipynb

Below we just go over parameter tuning process for random forest as an example

Parameters for tuning:

n_estimators: Default=100, higher could improve but cause more computing power\
min_samples_split: Default=2, higher could avoid overfit\
max_depth: Default=None, set to a certain number could avoid overfit\
max_leaf_nodes : Default=None, total number of nodes before stop\
max_features: Default=Auto=sqrt(n_features), more features could lead to overfit\
max_samples: Default=None, less samples per tree save computing power and avoid overfit\

Overall, all the default features seem set to provide the best predicted power, all the changes to hyper parameters are only to avoid overfitting or save computing power. Except n_estimators and max_features, which use more computing power or might increase overfit. Max_depth, max_leaf_nodes and min_samples_split both work to control the size of each tree but use different standards

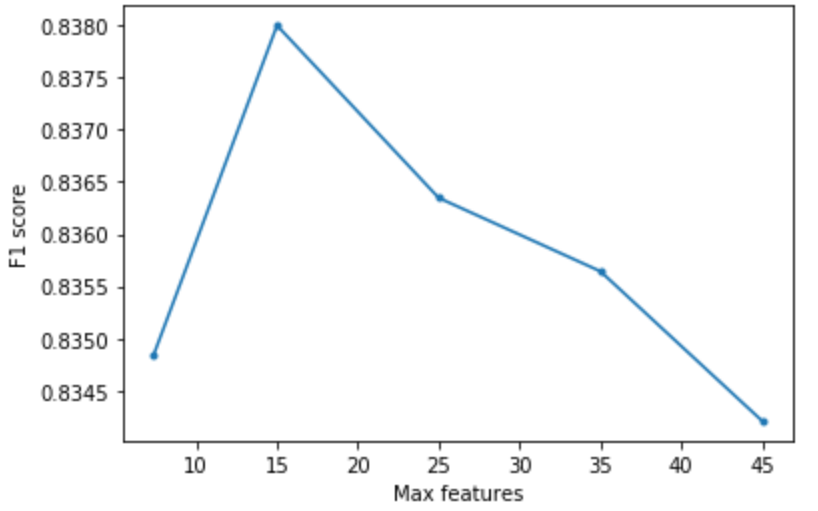

Figure 12. Random forest parameter tuning

From the above result, max_features = 15 shows better results. Check max_feature together with max_sample to optimize parameters.

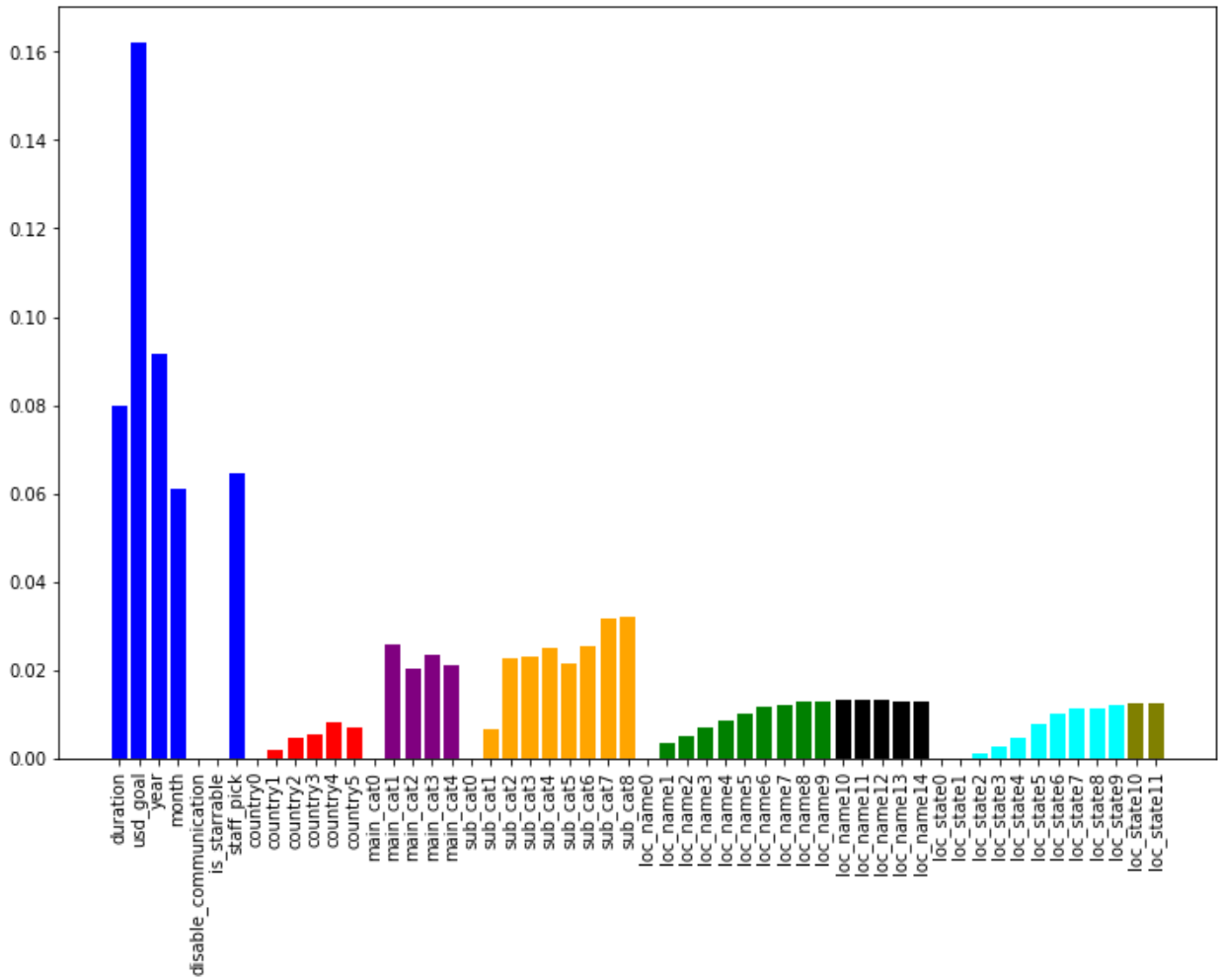

Figure 13. Factor importance for random forest

From the above graph we can see numerical features are the most important in predicting whether a project can be successful or not, especially duration, usd_goal and staff pick. Duration and usd_goal are money related, as if you allow longer time and a shorter amount of money to be considered successful, the chance of success will be much higher. As for staff pick, it's recommended by the website, which will increase the chance of projects to be viewed by customers.

6. Conclusion

By choosing to use a random forest classifier as our prediction model, we were able to get an accuracy score of 99.98% on training dataset and 85.0% on testing dataset. And area under curve score for roc is 93% and positive precision and recall both around 88%.