Kickstarter Final Project Report

Group 28
Ching-Hsiang Mao, Chi-Hao Tseng, Kai-Hsiang Lin, Shubham Arora

Executive Summary

Kickstarter is an American crowdfunding platform with a mission to “bring creative projects to life”. Over the years it has backed numerous projects in a variety of genres including films, music, comics, video games, technology, food etc. They have a unique model for determining the success of their projects which offers tangible returns on investment in exchange for their pledges for the people who back their projects.

When we tried to research more about Kickstarter, we found that it had a variety of projects for creators to choose from or to introduce in the market. A project is successful if the project meets the goal in the time set by the creators. Therefore, if we are able to come up with a prediction model that can help the creators predict the successfulness of a project, the model will be of a huge asset as it will give project creators more clarity on which type of product to work on before initializing a project.

Considering our prediction model, we added more than 50 new features to the original data and also took into consideration an external data source so as to improve the accuracy of the prediction. Besides helping creators make better decisions on which type of project to create, we also believed that our model could be a huge benefit for any business, organization, or even individual seeking to fund a new project at Kickstarter.com.

Our team decided to predict the “success” target variable and was able to achieve an accuracy around 80% on the unlabeled dataset by trying various models and features which are explained in detail in this report.

Feature Engineering - Variables Table

Variable Id	Clean/ Create	Original Variable	New Variable	Definition
1	Create	deadline, launch_at	launch_period	how long the project is launched
2	Create	launch_at	launch_on_weekend	Check whether a project is launched on weekend
3, 4, 5	Create & clean	create_at	create_year, create_month, create_date	Separate project create date into years and months → year as a number and month as a factor variable; Extract project create date and categorize it into three groups: early, mid, and late
6, 7, 8	Create & clean	launch_at	launch_year, launch_month, launch_date	Separate project launch date into years and months → year as a number and month as a factor variable; Extract project launch date and categorize it into three groups: early, mid, and late
9, 10, 11	Create & clean	deadline	deadline_year, deadline_month, deadline_date	Separate project deadline into years and months → year as a number and month as a factor variable; Extract project deadline date and categorize it into three groups: early, mid, and late
12, 13	Create	location_slug	location_state, location_place	extract the location state and place from the location slug and capitalized it
14	Clean & create	creator_name	creator_firstname	Extract only the first name of the project creator
15	Create	creator_firstname	creator_female_name	Introduce an external data that contains baby first name and his/her gender and assign a gender to the creator based on their first names
16	Create	goal	goal_quantile_rank	Rank project goals based on quantile
17	Create	goal_quantile_rank	big_project	If a project has the highest goal rank, consider it as a big project
18	Create	name	name_word_count	Count the number of words in the project name
19	Clean	isbwImg1		Convert the variable from a logical variable into factors and create a third group “Unknown” for NAs
20 - 24	Clean	isTextPic, isLogoPic, isCalendarPic, isDiagramPic, isShapePic		Convert the 0s into NO, 1s into YES, and NAs into UNKNOWN
25, 26	Clean	smiling_project, smiling_creator		If the probability > 100, transform to max = 100 and round it up to an integer
27	Create	reward_amounts	reward_amount_count	Count the number of reward donation options for each project

28, 29	Create	reward_amounts	min_reward_amount, max_reward_amount	Identify the minimum and maximum value of reward_amounts (min/max donation)
30	Create	tag_names	tag_count	Count the number of tags for each project
31	Create	category_parent	category_freq	Group the data by projects' parent category and count the number of projects for each category
32	Create	category_freq	is_popular_cat	Determine whether a project is within a popular category (if category_freq > average count for each category)
33, 34	Create	minage_project, minage_creator	minage_project_group, minage_creator_group	Group the estimated min age of faces in the main project and creator pictures into 7 groups (toddlers, grade schoolers, teen, young adults, mid-aged, elder, and unknown)
35, 36	Create	maxage_project, maxage_creator	Maxage_project_group, maxage_creator_group	Group the estimated max age of faces in the main project and creator pictures into 7 groups (toddlers, grade schoolers, teen, young adults, mid-aged, elder, and unknown)
37, 38	Clean	color_foreground, color_background		Create a new group for NAs: unknown
39	Create	location_type	location_type_freq	Group the data by location_type and count the number of projects for each type; reassign types by their frequencies (<100 - "Others")
40	Create	afinn_pos, affin_neg	descriptionPositive	Determine whether a project description is positive or negative by calculating the difference between the number of positive and negative words in the description
41	Clean	tag_names		Remove white spaces first then the symbol "|" between each tag so each tag can be considered as a single vocabulary for further text mining task
42-52	Create	ADV, NOUN, ADP, PRT, DET, PRON, VERB, NUM, CONJ, ADJ	ADV_percentage, NOUN_percentage, ADP_percentage, PRT_percentage, DET_percentage, PRON_percentage, VERB_percentage, NUM_percentage, CONJ_percentage, ADJ_percentage	Calculate and round up the percentage of each types of words in the full project description; replace NAs with 0

Analysis on Introducing External Data Source and Text Mining

Our team decided to introduce external data that contains babies' first names and their assigned gender so we could identify whether a creator was considered as a male, female, or unknown.

Moreover, the team decided to conduct text mining analysis on two columns: blurb (short project description) and tag_names (automatically generated description tags for the main project picture.) We believed that this would help us better categorize the projects based on the vocabularies that these two variables have and identify whether certain features of a project or its picture will make it successful.

For text mining tasks, we trained a ridge, lasso, random forest, and xgboost model with TFIDF values so that we could identify the top-15 most important vocabularies generated from each model and select the ones based on personal judgment or those that appeared multiple times across the lists for both variables.

Then, we conducted three different predictions with three different variables selections using logistic regression: 1. Without external data and text mining, 2. Only with external data, and 3. Only with text mining. We wanted to see whether incorporating these additional features would have an impact on the overall prediction performance of the model on the validation data. On three sets of features, our team did a 70/30 simple partition to create a training and validation data set and calculated the accuracies of the predictions on the validation sets.

	var_selection	result_acc
1	Original Features without Any Features Created from External Data and Text Mining	0.7092315
2	Original Features with Only Features Created from External Data	0.7098474
3	Original Features with Only Features Created from Text Mining	0.7117293

As we can see in the result table, it is obvious that text mining has a positive impact on the performance of our prediction model. On the other hand, the variables created from external data didn't seem to have too much influence on the overall performance. Nonetheless, incorporating features created from an external data source and text mining analysis both can increase the accuracy of the validation data set. Hence, we decided to keep these variables for the finalized features set.

Model Evaluation and Selection Methodology

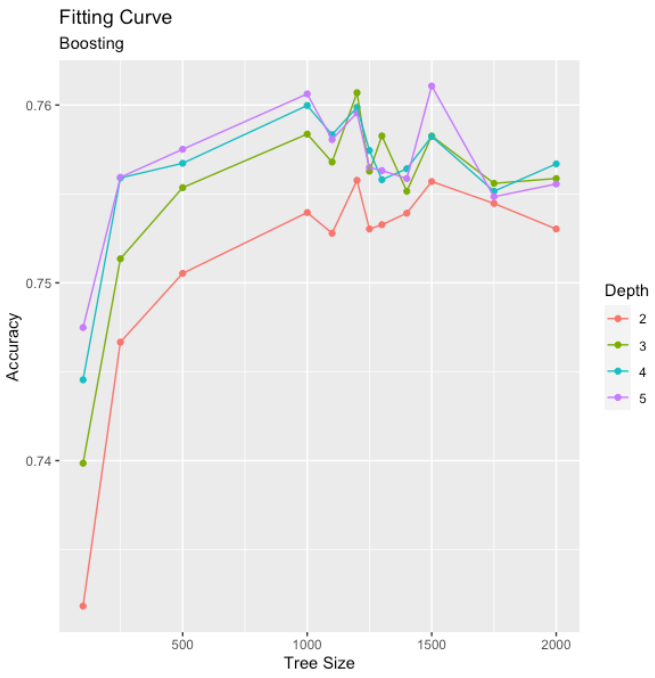
For model evaluation, we use cross-validation to compare between 7 different models, which is: Tree, Logistic Regression, Ridge, Lasso, Random Forest (ranger), Boosting (gbm), and Xgboost. In addition, we put each model into a function that takes training set, validation set, and several model specifications as the input and the accuracy as the output. That way, we could simply call the functions with the data and model specifications to get the accuracy of the model. It will also be easier for us to do cross-validation or grid search later on. Since Ridge and Lasso have their own cross-validation functionality included in the library, we manually compare the cross-validation performance for the rest with 5-folds. After getting 5 accuracy from each model, we calculate the average accuracy of the 5 models then compare them to the accuracy of Ridge and Lasso. The following table is the result of the comparison.

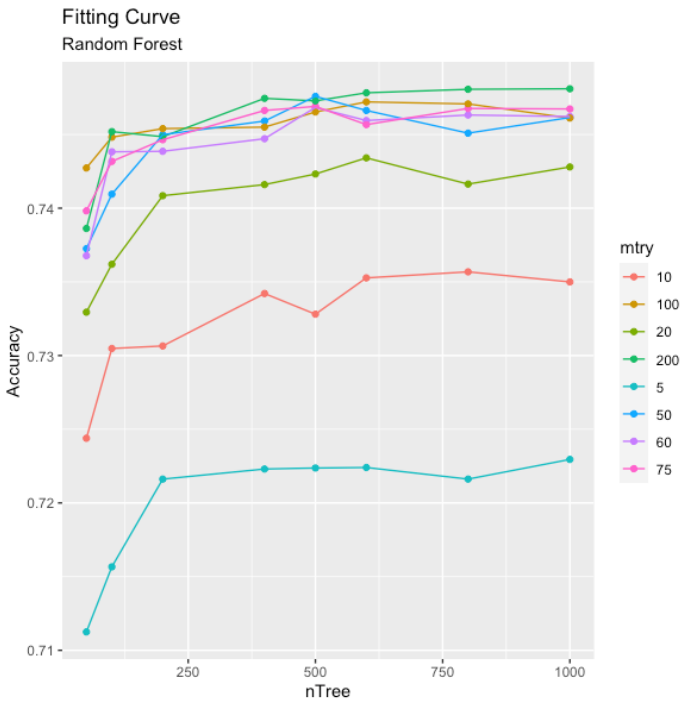
Model	Accuracy
Tree	0.7197187
Logistic	0.7099979
Ridge	0.7079655
Lasso	0.6995997
Random Forest	0.7288955
Xgboost	0.7478136
Boosting	0.7574831

As we can see from the table above, Random Forest, Xgboost, and Boosting have better performance than the other models. However, since Xgboost and Boosting are models using the same concept, we will only choose one of them. As a result, we decide to choose Random Forest and Boosting as the models we want to do more research and explore in.

Model Specification Comparison

Since we found the ensemble models perform better in our validation, we decided to choose gbm boosting and random forest for further grid search. Following graphs show the accuracy of these two models with their hyperparameters. We can see that the random forest model has the best performance, which is nearly 75%, with mtry at 200 and ntree at 1000. On the other hand, gbm boosting has its best accuracy, slightly above 76%, when depth is at 5 and the tree size is at 1500. Therefore, we decided to use gbm boosting as our prediction models for our most evaluations.





After several evaluations, we found out that the accuracy on unlabeled data barely increased. Hence, we thought assembling our predictions might be a good alternative to improve our accuracy. First, we chose the two evaluation results with the highest accuracy and identified the instances that had the same predictions on the target variable. The reason why we wanted to isolate these instances was that we found out these instances had higher accuracy, so we assumed that this portion of data was easier to predict. Therefore, we expanded our training data by combining these instances that we selected out from the unlabeled dataset with the whole labeled dataset and retrained a new model to predict those that had different predictions on the target variable, which we believed were harder to predict.

Surprisingly, it turned out that we were able to generate the best accuracy through this method, and the result was 81.76%. As a consequence, we chose this as our final predictions for the project.

Conclusion

We believed that by putting a great deal of time and effort on features engineering in the first place indeed yielded a preferable result , validating by the fact that our team was able to achieve the first place for the first three evaluations. This also helped us to have more time and energy to be spent on further model selection and specifications. Another task that we thought we did well was that we built all of the models into functions so that it was much easier and more efficient for us to generate results by simply calling and changing the arguments in the functions that we built.

However, after taking the first place for three consecutive evaluations, our biggest challenge was to find constructive ways to improve our model performances. We tried out various model selections and specifications, even techniques of conducting ensemble on the evaluation predictions, but the performances were not having steady improvements.

Therefore, what we thought could be done differently was that we should definitely go back to feature engineering when we were stuck on improving the performance by solely tuning the function's hyperparameters. We believed that having an open mindset would certainly assist us, and we should make extra effort on exploring the features instead of sticking with what we believed was right throughout the process. We might miss many possibilities and opportunities to take our model performance to the next level.

Our team would suggest that the students who just started this project spend some time exploring the data and think about what information may be useful before actually conducting the feature engineering and model building tasks. This thought process can definitely be a valuable source for ideas and inspirations once you encounter challenges.

Appendix - Contribution Table

Name	Main Contribution
Ching-Hsiang Mao	1. Feature engineering 2. Logistic Regression, KNN, tree, random forest, and xgboost model functions 3. Cross-validation on model selection 4. Model evaluation and selection methodology 5. Organizing and commenting on the final R scripts
Chi-Hao Tseng	1. Feature engineering 2. GBM boosting model function 3. Grid search on GBM boosting model 4. Ensembling on predictions 5. Model specification
Kai-Hsiang Lin	1. Feature engineering 2. Conduct text mining 3. Ridge and Lasso regression model functions 4. Feature engineering table 5. External data source and unstructured text mining performance analysis
Shubham Arora	1. Naive Bayes model function tryout 2. Executive summary