Kickstarter Prediction Project Report

Group 13

Christopher Garrett
Isha Jain
Shubhangkar Girish Jain
Parth Nanwani
Dhairya Parekh

Table of Contents

List of Tables

List of Figures

Executive Summary

With over 400,000 projects launched on Kickstarter a year, most of them tend to have a goal lower than they need, not to appear needy. While that may help them initially, they will need more money than that, so a successful Kickstarter isn't always what the fundraisers need. So, this brings attention to the fundraisers that achieve more than what they set as their goal. These fundraisers have a high chance of succeeding in their fundraiser, but by hitting big, they will have the money necessary to succeed in their project, something everyone wants to contribute.

While exploring all three target variables, they all have solid points to be the more important variable. Success has a case because there is a project that only needs what it asks for. They may have taken the time to plan for any unfortunate circumstances that may come about and cause money to be required for fixes or changes. The number of backers makes a strong case because this can tell how many people are interested in your project. The difference between getting five rich people to donate $50,000 vs. 500 people donating $50,000 can make a significant difference in how your project is perceived. Once you finish working on the project and enter the next phase of selling your product, this number will be significant. With them all having their pros, it became a pick-em situation with each variable helping somehow. These pros led to us selecting "Big Hit" to aid other success beyond the fundraising phase.

Once we decided that our target variable was "Big Hit" we used various modeling techniques to improve the performance altogether. In all, we tried six different models and out of all the models we could observe that the Lasso model has the highest performance. We used robust feature engineering and hyperparameter tuning to maximize the model's predictive accuracy. Apart from this we also used an external dataset and extracted relevant features to improve the performance.

Feature Engineering

Datasets

We combined the ks_training_x and ks_training_y using the left join to get one training data set to get a basic idea about the data.

Apart from these, we incorporated an external data source, the **baby_names dataset** that we previously used in class which contains around 1 million data points that have names and indicate whether it is a female or a male depending on the probability value. So we are using this to check if the creator is female or male. We categorized it into male and female using the female percentage, and we only selected the first name and the categorized column, which is 'female_name' from the data set. This data set helped us predict the gender of the creator which overall helped us increase the performance in the Lasso model by 0.006%. (Result table attached below)

Data Cleaning

First, we created a tokenizer function to do the text mining in which we removed the numbers, punctuations, and stop words. Then we created a function for data cleaning, which we used once on the training data and then on the test data.

Table 1 Data Cleaning

	Original Column Names	Derived Columns	Description
1.	region	-	Since the region has different categories we used as.factor to categorize the region column.
2.	creator_name	firstname	Renamed creator_name as firstname. Then just kept the first name and dropped the extras to just keep the first name in the column. Also converted all the names to Upper to perform the left join to merge the baby names dataset. Also to avoid the NAs after left join we mutate the 1's to F for females, 0's to M for Males, and NAs to U for Unknown. From this, we get the gender of the creator name using an external dataset.
3.	location_slug	state	Extracted the state details from location_slug as it has a lot of extra information like city names so we just extracted the state names from it and created a state column.
4.	deadline, launched_at	time_deadline_launch_days	Calculated the difference between deadlines and launched_at to get the duration of time after the project has been launched until the deadline and stored it in time_deadline_launch_days as numeric values
5.	deadline, created_at	time_deadline_create_days	Calculated the difference between deadlines and created_at to get the duration of time after the project has been created until the deadline and stored it in time_deadline_create_days as numeric values
6.	launched_at, created_at	time_launch_create_days	Calculated the difference between launched_at and created_at to get the duration of time after the project has been created until the launch date and stored it in time_launch_create_days as numeric values
7.	goal	1. goal_per_day_deadline_la	1. We divided the goal which is in numerical format with the number of days of the

		unch 2. Goal_per_day_deadline_create 3. goal_bin	deadline from the launch which we calculated earlier and stored in time_deadline_launch_days 2. Similarly, we divided the goal by the number of days of the deadline from the created which we calculated earlier and stored in time_deadline_create_days 3. Binned the goal range using 5 cuts with the n-tile function to categorize the goal
8.	deadline, launched_at	deadline_month, deadline_year, deadline_date, launched_at_month, launched_at_year, launched_at_date	Extracted the month, year, and date of the deadline, and launched_at onto separate columns to observe their significance independently
9.	created_at	created_at_month, created_at_year	Extracted the month and year of the deadline onto separate columns to observe their significance independently. Did not extract the date for created_at as it did not seem to be useful
10.	creator_id	-	Grouped the creator id to get the count of projects for each creator
11.	location_type	-	Grouped the location_type and counted the number of projects for each location_type and changed the location type with less than 200 projects to other.
12.	category_parent, category_name	-	We grouped the category_parent. Then we further grouped the category_name based on the category_parent. We also grouped any category_name which has less than 200 values and renamed it to another but kept the category parent name to understand to which parent it belongs
13.	numfaces_project, numfaces_creator, male_project, male_creator, female_project,	-	Since the majority of the data in these columns belong between 0-2 we grouped them, if the value is 0, 1, 2 keep it as is but anything above 2 will be changed to 3+

	female_creator		
14.	smiling_project, smiling_creator	-	Since this is a value for probability/percentages the values cannot be greater than 100. So we replaced all the values above 100 to 100
15.	maxage_project , minage_project, maxage_creator , minage_creator ,	diff_age_project, diff_age_creator	Calculated the difference between maxage_project and minage_project stored it in diff_age_project. Repeated the same to create diff_age_creator
16.	maxage_project , minage_project, maxage_creator , minage_creator ,		Each of these columns has a lot of variation but we observed a pattern that there are a similar number of projects for certain values which shows the importance of that value. So we binned those values and categorized them accordingly.
17.	isbwImg1	Null_col	isbwImg1 had a lot of null values so we created a Null_col where it shows whether the row contains null in the column or not using 1 and 0
18.	isbwImg1, color_foreground, color_background, isTextPic, isLogoPic, isCalendarPic, isDiagramPic, isShapePic		Replacing NA's. So basically we replaced all the NAs in all of these columns using text for example color_foreground NA values were replaced by 'No foreground' or NAs in some other columns with the text NULL so that we do not lose any data by dropping the NA and we do not have to deal with the NA's either.
19.	accent_color	accent_color_red, accent_color_green, accent_color_blue	Extracted whether the color has red, blue, or green and extracted it to separate columns as accent_color_red, accent_color_green, accent_color_blue after which we removed the NAs same as the variables above. They still had a lot of values so we categorized all the groups under the threshold values based on the

			number of levels to others.
20.	Num_words_bin, sentence_counter_bin, grade_level_bin	-	Grouped into bins using ntile function
21.	afinn_pos, afinn_neg	1. affin_pos_neg 2. afinn_pos_bin, afinn_neg_bin 3. afinn_pos_by_neg 4. afinn_neg_tot, afinn_pos_tot	1. Combined all positive negatives together in the single-column were 1 if positive 0 if negative 2. Since these columns have huge variance binned them to reduce the number of factors 3. Calculated the ratio of positive to negative 4. Calculated the intensity of positivity or negativity in a particular row as compared to the entire column
22.	reward_amounts	reward_amounts_avg, reward_amounts_count, reward_amounts_min, reward_amounts_max, reward_amounts_sd	Got rid of the NAs, then created a list of the comma-separated values and calculated its mean, maximum, minimum, count and standard deviation for each row.
23.	tag_names, captions, blurb	-	Split tag_names based on '|' and calculated the length. Also counted the number of words for captions and blurb by splitting on " ".
24.	ADV, NOUN, ADP, PRT, DET, PRON, VERB, NUM, CONJ, ADJ	xxx_per_word, xxx_per_sen, xxx_bin (xxx represents each of the attributes)	Each of the aforementioned attribute were first divided by the num_words if it wasn't a zero value and then divided by the sentence count to check its frequency per instance and per sentence. A bin was also created using ntile method

Validation and Model Evaluation Process

While we initially got up to 135 total instances in our data set, we decided to use 84 in our training data and 81 in the testing data.

Table 2 Validation & Model Evaluation Process

Training	Testing
Null_col, region, state, time_deadline_launch_days, time_deadline_create_days, goal_per_day_deadline_launch, goal_per_day_deadline_create, deadline_month, deadline_year, launched_at_month, launched_at_year, Created_at_month, created_at_year, goal, goal_bin, creator_repeat_count, location_type, category_parent, category_name, numfaces_project, numfaces_creator, male_project, male_creator, female_project, female_creator, smiling_project, smiling_creator, diff_age_project, diff_age_creator, minage_project, maxage_project, minage_creator, maxage_creator, isbwImg1, color_foreground, color_background, accent_color_red, accent_color_green, accent_color_blue, isTextPic, isLogoPic, isCalendarPic, isDiagramPic, isShapePic, num_words_bin, avg_wordlengths, contains_youtube, sentence_counter_bin, avgsentencelength, avgsyls, grade_level_bin, afinn_pos_bin, afinn_neg_bin, afin_pos_neg, reward_amounts_avg, reward_amounts_count, reward_amounts_min, reward_amounts_max, ADV_per_word, NOUN_per_word, ADP_per_word, PRT_per_word, DET_per_word, PRON_per_word, VERB_per_word, NUM_per_word, CONJ_per_word, ADJ_per_word, ADV_bin, NOUN_bin, ADP_bin, PRT_bin, DET_bin, PRON_bin, VERB_bin, NUM_bin, CONJ_bin, ADJ_bin, afinn_pos_tot, afinn_neg_tot, reward_amounts_sd, big_hit, backers_count, success	Null_col, region, state, time_deadline_launch_days, time_deadline_create_days, goal_per_day_deadline_launch, goal_per_day_deadline_create, deadline_month, deadline_year, launched_at_month, launched_at_year, Created_at_month, created_at_year, goal, goal_bin, creator_repeat_count, location_type, category_parent, category_name, numfaces_project, numfaces_creator, male_project, male_creator, female_project, female_creator, smiling_project, smiling_creator, diff_age_project, diff_age_creator, minage_project, maxage_project, minage_creator, maxage_creator, isbwImg1, color_foreground, color_background, accent_color_red, accent_color_green, accent_color_blue, isTextPic, isLogoPic, isCalendarPic, isDiagramPic, isShapePic, num_words_bin, avg_wordlengths, contains_youtube, sentence_counter_bin, avgsentencelength, avgsyls, grade_level_bin, afinn_pos_bin, afinn_neg_bin, afin_pos_neg, reward_amounts_avg, reward_amounts_count, reward_amounts_min, reward_amounts_max, ADV_per_word, NOUN_per_word, ADP_per_word, PRT_per_word, DET_per_word, PRON_per_word, VERB_per_word, NUM_per_word, CONJ_per_word, ADJ_per_word, ADV_bin, NOUN_bin, ADP_bin, PRT_bin, DET_bin, PRON_bin, VERB_bin, NUM_bin, CONJ_bin, ADJ_bin, afinn_pos_tot, afinn_neg_tot, reward_amounts_sd

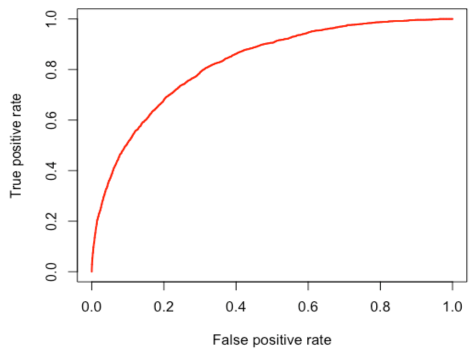
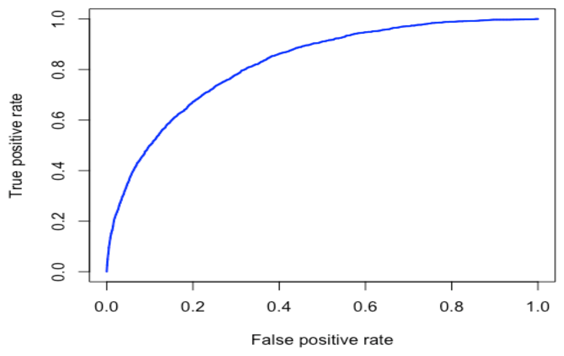
While validating the data, we used the cross-validation method known as the k-Fold. Using five-folds, we can get an accurate score by taking the average of all five data sections. Having it run five times avoids the model only receiving either good or bad data and being biased, allowing for a combined score of all five models to make one accurate AUC score for each model.

Model Specification Comparison

Table 3 Model Specification Comparison

Model	Performance (AUC)
Lasso	0.8271588
Ridge	0.8231465
Naive Bayes	0.7203537
GLM	0.8217661
LM	0.8165933
Random Forest	0.8165254

With our target variable being "Big Hit," we are looking to see how we can cater our model to fit best with the data that we have. We are looking for the highest AUC to display what model is performing the best. After testing the model and finding the AUC for the six models, we used data to aid our decision-making. As you can see in the table above, both the Lasso and the Ridge models had high AUC values. The Lasso model slightly outperformed the Ridge model.

TPR and FPR of Lasso Model	**TPR and FPR of Ridge Model**
 *Figure 1 AUC Curve for Lasso Model*	 *Figure 2 AUC Curve Ridge Model*

With these two models performing so close to each other, we decided to look at the True Positive Rate to strengthen our decision further. By plotting out the True Positive Rate and False Positive Rate, we can glimpse how the models perform. The two graphs are plotted identically, allowing us to now allow the slightly higher AUC performance from the Lasso Model to select this model as the best one.

Takeaways / Conclusions

As a group, we did an excellent job observing our position after receiving the scorings from each week. By not being stuck on one variable, we could rank high with "big_hit." If we had had more time to work on the project, we would have tried even more models for all three variables to rank higher. Some advice for other students who do this project next year would be to look back at past assignments and pull minor parts from them that you've learned throughout the course to bring your project together. Sometimes you're missing a piece that you already have but didn't think could help you.

Team Contribution

Table 4 Team Contribution

Member Name	Contribution
Christopher Garrett	Data cleaning, feature engineering, project report
Isha Jain	Data cleaning, feature engineering, Naive Bayes model, project report
Subhangkar Girish Jain	Data cleaning, feature engineering, GLM model, linear regression
Parth Nanwani	Data cleaning, feature engineering, ridge model, lasso model, GLM model, linear regression
Dhairya Parekh	Data cleaning, feature engineering, random forest model, Naive Bayes model