Intro to Data Science term project

Kickstarter analysis

Sebastian Brarda *SB5518*
Felipe Ducau *FND212*
Luisa E. Quispe Ortiz *LQO202*
Maria E. Villalobos Ponte *MVP291*

Instructor:
Brian D'Alessandro

December 2015

Contents

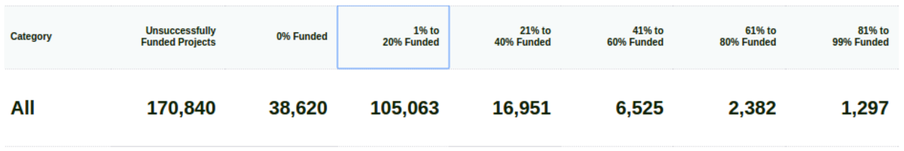
1 Business understanding

Kickstarter (KS) is one of the most famous crowdsourcing platforms for creative projects worldwide. Their mission is to help bringing creative projects to life. Proposals for thousands of potential projects have been uploaded to Kickstarter every year. Since their launch in 2009, 10 million people have backed a project, $2.1 billion has been pledged, and 98,223 projects have been successfully funded.

Figure 1: KickStarter Logo

In order to help project creators to find resources and support, KS provides a platform that lets users share their story and build a community of supporters. The idea is that supporters can contribute by giving a certain amount of money to reach the goal as well as feedback on the project's idea and planning. When creating a project in KS, users must specify a length for the collection period, which cannot be modified afterwards. If the project is not able to collect the money in that timeframe, it is considered as failed and money is returned to donors. While 14% of all the projects finished without even receiving a single pledge, 78% of projects that raised more than 20% of their goal were able to successfully accomplish the funding afterwards.

According to their analytics and insights web page, the current success rate for project funding is 36.72% and it varies depending on the category of the project (although many independent studies report higher success rates). Additionally, success seem to depend on the required amount of money as well. Most successfully funded projects raised less than $10,000, we can see the amount of successfully funded projects by goal ranges in the following chart.



Category	Unsuccessfully Funded Projects	0% Funded	1% to 20% Funded	21% to 40% Funded	41% to 60% Funded	61% to 80% Funded	81% to 99% Funded
All	170,840	38,620	105,063	16,951	6,525	2,382	1,297

Figure 2: Success Rate by range of funding- Source: Kickstarter Webpage

Having projects that are creative, trustworthy and well planned is in the best interest of KS. They make great effort into helping their users in all the stages of the project funding to ensure that their projects are successful. In their handbook they make recommendations for telling better stories by explaining the key elements you should take into consideration in the KS's project web page, like having an appealing video and images, a succinct description, a timeline or schedule, and a budget breakdown. These insights are based on what successful creators think makes a great project proposal.

We believe that a general increase in project funding success rate would be valuable both for the organization and its community. KS charges a fee of 5% to successfully-funded projects, so increasing their funding success rate would ultimately increase their revenue, but more than that it would add value to their support platform from which artists, musicians, filmmakers, designers, and other creators around the world benefit from.

Our proposal aims to increase the general funding success rate and KS's expected revenue by performing the following subtasks:

- Create and train a supervised data mining model that estimates the probability of success of a given project
- Estimate a set of possible changes to the project that would increase the probability of success.
- Deploy a recommendation system that can provide specific real-time advices to users based on the previous calculations.

If users happened to follow these suggestions, the probability of success for each project would increase and thus general success rate would increase as well. Recommendations would be based on features such as the goal amount, the rewards given to backers, conciseness and descriptiveness of the project proposal and the duration of the funding period. An example of recommendation could be *If you add a video your project, you will have 11% more chances of being successful.* In this case, an increase in the probability of success would be directly related to an increase in expected revenue for KS. It also exists the possibility of recommending to raise or decrease the money goal amount. In this case, the recommendation would be more complex, since decreasing the amount of money requested could increase the probability of success but still diminish expected revenue perceived by KS.

Current recommendations for project proposals in KS are based on domain knowledge and there is no automated way to assess the quality and likelihood of success of a given project based on a set of descriptive attributes. So the proposed system would provide both an automated assessment and recommendations to improve project proposals.

2 Data understanding

The main dataset for the project was found in the “Database Search tool” from the Fung Institute, UC Berkeley. This 74 Mb open dataset contains a significative amount of information from KS’s past projects from 2009 to 2014. The target variable in this case would be “state” which represents if a particular project succeeded or failed in collecting the required amount of money.

The dataset contains information about 105,785 projects, with 47.8% corresponding to successful projects and 52.2% to unsuccessful projects. Despite the fact that the success rate in our dataset is higher than the one reported by KS, after reading several independent studies we concluded that it is reasonably representative of the truth project population. The following attributes with valid entries were found in the database: projectid (int), title (string), category (string), goal (int), amount_pledged (int), state (string), start_date (date), state_changed_at (date), end_date (date), location (string), founder (int), description (string), url (string). “state” was the target variable for our model (successful or unsuccessful). For a detailed description of the dataset, please refer to “Appendix - Dataset Documentation”.

The first thing to notice is that title and description are text fields which require special treatment in order to add information, so some text mining techniques would be applicable in this case . It is also important to take into account that the variables amount_pledged (the amount of money collected), backers_count (number of users that backed the project) and state_changed_at (date) generate leakage, because they could only be obtained once the end date has already passed. It is also clear that if the amount of money collected was higher than the ”goal”, the project would be successful. In fact, after running our baseline model with this variable, the prediction accuracy was over 98%.

Another thing that we noticed is that the variable ”category” is actually the ”subcategory” of the project, and that parent categories exist in the KS’s webpage.

After carefully thinking about the feature engineering that we could perform on the data and launching our baseline model, we started thinking about other features that could be helpful. We took into account the domain knowledge we had gathered from KS’s recommendations on every stage of project funding which can be found in their Handbook. For instance, KS reports that 80% of their projects have video and project pages without one have a considerably lower success rate. Other elements they mention are the project’s description length and the number of images. Because of that, we decided to perform web scraping to go through each of the projects’ URL’s in the dataset and obtain further information about them. After several hours of running the script, we could gather the following features: num_rewards, min_pledge_reward, max_pledge_reward, num_images, has_video, len_description and full_description. KS allow users to offer rewards depending on the amount of money donated (i.e. if you donate $5, you will receive a book of ”my project”), that is what num_rewards refers to. min_pledge_reward and max_pledge reward are the minimum and maximum amounts of donation required for the different rewards. We thought that if something was given back to donors, it might be correlated to project success. num_images is the number of images attached to the KS project, and len_description is the number of characters of the full description (remember that our dataset has only a short version of the description). The full description text is in full_description which was not taken into account for modeling in this project but might be interesting to add this information in a future iteration. The variable has_video is self explanatory.

At this point we wanted to understand how the variables we had so far related to the target variable, so we calculated the information gain for each feature and the correlation sign to see if they were positively or negatively related to success.

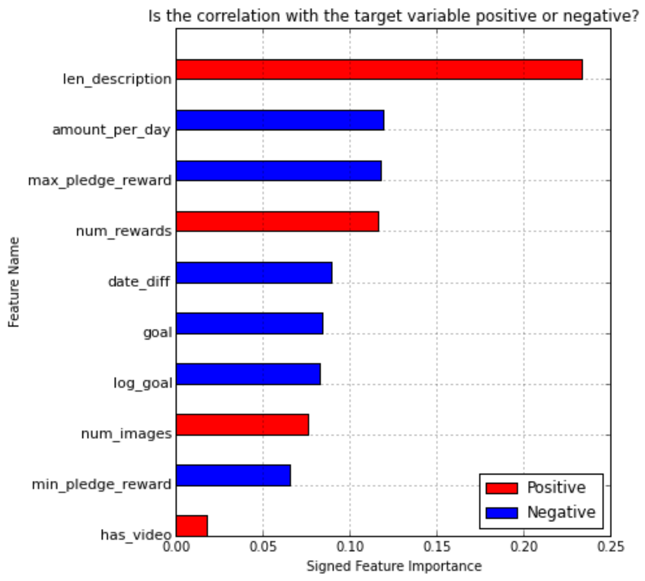


Figure 3: Feature importance

Something interesting we encountered in the way is a leakage that was not necessarily expected or easily detected. When the scraped variables were included, we could suddenly predict success on the test set with an AUC of 0.94, which seemed suspicious. We detected that num_images variable had a correlation level with success that seemed unlikely. In fact, the minimum number of images differed in both classes and also conditioned on having video or not. This was due to the html design of KS's web pages. The original minimum values for each case can be seen in Figure 4. We corrected this miscount on the number of images to solve the leakage.

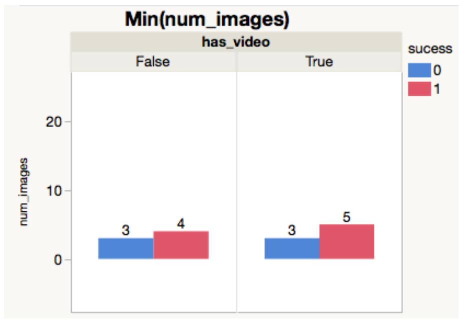


Figure 4: Leakage Variable: Number of images by success rate

3 Data Preparation

Please see "Appendix - Dataset Documentation" for more details on the features used.

- Data creation and integration steps
  - Created a scraping script to obtain *num_rewards, min_pledge_reward, max_pledge_reward, num_images, has_video, len_description* and *full_description* for each of the projects in the database.
  - Joined the new features to the original database.

- Data Cleaning and feature creation steps
  - Fixed several invalid "end of lines" in the description and title.
  - Cleaned a small number of NULL values.
  - Renamed the target variable from "state" to "success" to prevent confusions with the location variables.
  - Cleaned all entries with non-valid labels (i.e. there were some entries with values like "live" which not only were not useful for our predictions, but were also probably outdated)
  - Created the "parent_category" feature based on the category (sub category) column. As we said before, we found that the dataset contained only the sub categories, so we associated each subcategory to the corresponding parent category that we found on kickstarter.com.
  - Created the date_diff variable, that represents the amount of days the project was open for donations (difference between end_date and start_date)
  - Created the amount_per_day variable, which is the goal / date_diff (i.e. average amount of money required to collect per day to achieve the goal)

- Cleaned outliers for the "goal" variable (Projects that requested extreme amounts like \$0 or \$10.000.000). We cut down the top and bottom 1% of the records.
- Cleaned projects originated outside the US since there were only few of them and would have been problematic to take into account the exchange rate. Furthermore nowadays KS only allows US based projects in the platform.
- Created dummy variables for start_month and end_month (i.e. A specific project started in November, finished on January)
- Created dummies for parent categories, subcategories, and states (i.e. California).
- Created the log_goal feature (Logarithm of Goal)because the original feature was highly skewed.
- Dropped start_date, end_date, projectid(int), category(string), amount_pledged(int), location(string), founder(int), url(string), state_changed_at(date)
- With "title" and "description", we processed the text to create new features with TFIDF vectorizer from SKLearn, obtaining a sparse matrix. We then transformed the rest of the numeric features and dummies into a sparse matrix as well in order to create one big sparse matrix with all the features.

After this whole process we ended up with 79,108 instances, 48.1% of which were successful and 51.9% of unsuccessful projects (very similar to the original dataset). The total number of numeric and dummy variables was 150, without taking into consideration the text variables created after processing the title and description (539,748 new features).

4 Modeling

Our prediction goal was to have calibrated estimates of the probability of success. Any algorithm that allowed us to compute estimated probabilities was a candidate to solve our modeling problem. In this particular case, since KS is a tech company with relatively high technological understanding, we did not constraint our models based on interpretability. Decisions would be ultimately made by the system, so it was not necessary to make it specifically easy to understand by a human decision-maker. Moreover, since the purpose was to make recommendations in real time, it was critical that the algorithm could make improvement suggestions in a reasonable amount of time.

Because we were not trying to estimate a numerical quantity, linear regression was automatically discarded. Naïve Bayes was also discarded because it generally performs poorly in probability estimation (because of conditional independence assumption). We also discarded SVM for the enormously amount of time it takes to train in a cross-validation procedure, which was unfeasible for the scope of this project. We decided to move forward and try three different algorithms: Logistic Regression, Decision Tree and Random Forest. Given that all our models had different sets of hyper-parameters, we decided to use cross-validation to choose the best configuration for each algorithm using grid search.

- Logistic Regression: The advantages of logistic regression are that it performs well on high-dimensional data and it is fast to train and predict. Once trained the model is easy to store. The set of hyperparameters considered for cross validation were: Regularizations: L1 and L2, C: from 10^{-2} **to** 10^{5}
- Decision Tree: Trees are easy to understand and have good scalability properties. The algorithm considers linear decision boundaries, slicing the feature space into hyper-rectangles. We expected this simple model to perform well on our dataset given that it can build complex decision regions. We considered the following hyperparameters to tune during cross validation: minimum sample leaf: 20, 50, 100, 500, 1000, 2000, 5000
- Random Forests: The embedded algorithm of random forests is characterized for being robust. The "multiple independent expert voting" method is an important advantage in comparison to a single model approach. The considerably slower speed in making the probability estimation was something to evaluate. Hyperparameter: Number of trees(100, 200, 500)

5 Evaluation

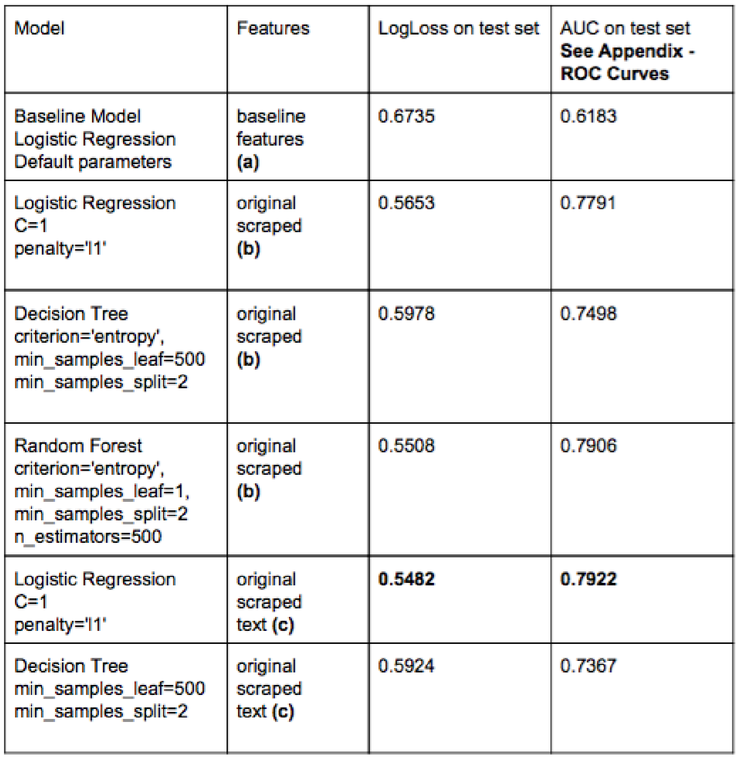
The model was meant to be used to predict probabilities in real time, so that each time a new project is created, we could estimate the probability of success and make specific recommendations on what to change in order to improve this probability (and by what %). Following this logic, the natural evaluation metric was LogLoss, because it is appropriate for class probability estimation problems.

As a second evaluation metric, we decided to use AUC. Despite the fact that we were not trying to do classification, a general metric that evaluates the predictive power of the model was certainly aligned with the goal and gave us a good understanding of the performance of the model given the set of the features.

We followed the procedure outlined below in order to decide which model to use for deployment:

- Dataset was splitted into a Training (85%) and a Test (15%) sets.
- We created 3 different versions of our database: a) Baseline features (see Appendix - Dataset Documentation) b) Full dataset without text features c) Full dataset with text features created from the description.

- Computed the baseline model: Logistic regression with no parameter optimization on dataset (a).
- Afterwards we conducted a 5-fold cross-validation grid search to select the optimal hyperparameters for each of the models for datasets (b) and (c). Given the learning curve (see Appendix - Learning Curve) a 5-fold cross-validation is appropriate. Even though it would have been better to run one cross-validation procedure to tune all the models altogether, we ran one cross-validation process for the dataset with text features and one without them. This was done because sparse matrices used for text features require a special treatment.
- From each cross-validation procedure and family of models, we decided to choose the setup with the lowest LogLoss.
- The chosen models for each cross-validation were then re-trained using the whole train dataset and compared based on their performance. We decided not to consider random forest for dataset (c) because the training time was extraordinarily high.



Model	Features	LogLoss on test set	AUC on test set **See Appendix - ROC Curves**
Baseline Model Logistic Regression Default parameters	baseline features **(a)**	0.6735	0.6183
Logistic Regression C=1 penalty='l1'	original scraped **(b)**	0.5653	0.7791
Decision Tree criterion='entropy', min_samples_leaf=500 min_samples_split=2	original scraped **(b)**	0.5978	0.7498
Random Forest criterion='entropy', min_samples_leaf=1, min_samples_split=2 n_estimators=500	original scraped **(b)**	0.5508	0.7906
Logistic Regression C=1 penalty='l1'	original scraped text **(c)**	**0.5482**	**0.7922**
Decision Tree min_samples_leaf=500 min_samples_split=2	original scraped text **(c)**	0.5924	0.7367

Figure 5: Model Evaluation Metrics

From the results obtained, we observe that Logistic Regression with dataset (c) outperforms Decision Tree and Logistic Regression on dataset (b), and outperforms Decision Tree on dataset (c). Prediction time with Random Forest was about 250 times higher than Logistic Regression. When evaluated with a potential recommendation algorithm (it will be explained later), the random forest model took 36 seconds to optimize a single project, while the same algorithm took 800 ms for the Logistic Regression model. For this reason we decided to discard Random Forest as a feasible option.

As a consequence of this analysis, we decided to proceed with Logistic Regression and the feature vector in dataset (c) as our final model for this project. It is worth noticing that from Logistic Regression family, the one that performs better is the less complex as well, therefore, the one standard error rule of thumb does not make sense in this particular case.

As a next step in the evaluation we built a calibration plot, which shows that probability estimations are close to the identity line. This is an important aspect, since the overall system evaluation is based on expected value analysis. (see Appendix - Calibration Plot)

5.1 Model Evaluation after deployment

In order to guarantee the performance of the model over time, it is important to monitor its predictive power using LogLoss and AUC on newly generated production data.

Given that the training time required to re-train the model is reasonably low, we suggest re-training at least once a week. As mentioned in the "Next Steps" section of this document, one of the important metrics to determine is the optimal time window to consider data instances as valid for training. The shorter this period is, the more flexibility to adapt to changes the model would have.

6 Deployment

The deployment proposal for the model is to launch an automated real-time recommendations system into the KS's web page. This system would include not only a data mining model, but also an optimization algorithm that runs on top of this model to generate recommendations. More specifically, a "recommendations step" would be added to the project creation flow after the user fills all the required fields. The optimization algorithm (simplified version of constrained stochastic gradient descend) would run over a subset of fields of the new project's data to generate these recommendations. Some variables of the project would be omitted from the recommendation system, since we assume that they are part of the nature of the project and are not supposed to be changed (for example, category). Let's say that we choose a subset of fields such as: goal (money requested), num_images, has_video and date_diff, num_rewards, len_description. The algorithm would calculate at which values of these variables our model achieves the maximum probability of succeeding. Afterwards, it would compute the difference between these optimal values against the values chosen by the user in the project creation flow. Lastly, it would make the recommendations and inform user about the associated potential probability increase. Some examples of recommendations would be:

- We have found that donors really like to see pictures about the projects. Increase the probability of reaching your goal in 7% by adding 2 more pictures to your project.
- Having an explanatory video attached to your project has proved to increase the chances of collecting your required money in 11%! Be creative and add a video to your project.
- Extend the collection period by one week to increment your chances of succeeding in 4%!
- If you could cut down some costs, it is important for you to know that if you requested 530 USD less, your chances of funding your project would be 13% higher!

6.1 Estimation of Potential Impact

In the Appendix - Estimation Potential Impact a simulation of a potential recommendation of the system is shown, indicating the uplift in the estimated probability of success (23.8%) along with the recommended values of the features.
In order to quantify the expected improvement on the projects that follow the recommendations we calculated the value that would have been generated if all the customers followed the recommendations as:

$$E(gain) = \sum p_{success\_modified} \cdot goal - \sum p_{success\_original} \cdot goal$$

The mean gain over all the projects in the test set was \$1356 (increase of 16.5% in mean probability of success). This number represents an upper bound for the expected gain. If we consider the case when 30% of the project creators followed the recommendation, we would get an average expected gain in raised funds by project of \$537. This represents an average increase in KS's revenue by project of \$26.85.

6.2 System Evaluation after Deployment

Before launching the system for the whole website, we suggest conducting an A/B test in order to gather data and evaluate potential positive and negative impacts that would allow risk minimization. This would not only help us determine if any step in the model/system was not accurately calibrated or taken into account, but also provide us with more data to optimize the system further.

If the project was to be deployed, we would expect an increase of success rate and KS's profit. To account for the proposal's impact on these variables it is important to measure if users are following the recommendations of the system and which recommendations, as well as storing these data which would be extremely valuable for future analyses. Comparing the success rate and average profit of projects that followed the recommendations vs. projects that did not follow the recommendations would give us an easy metric to evaluate the success of the system. With the total amount of new projects that follow the recommendations and the average revenue generated by these projects we could easily estimate the revenue uplift generated by the deployed system.

7 Next Steps

- Normalize data and evaluate SVM model. Due to time constraints it was not possible to do it for V1.0 (Model was taking several hours to train even in Courant servers) .
- Use the Full description and Title to create new text features.
- Try creating one model for every parent_category, or maybe even for every sub category. This might help reducing the Log Loss even further.
- Get more data. Both more instances and features. Some possible features/data that we think might add information are: Number of social followers, number of previous projects from the creator (and their success rate), information about budgets and schedule planning, information about the projects' own webpages, quality and length of the videos, etc.
- Determine which is the optimal time window for considering project instances as valid for training purposes (Bias-Variance tradeoff).

References

[1] Kickstarter, *Creator Handbook*, $https://www.kickstarter.com/help/handbook$

[2] Kickstarter, *Statistics*, $https://www.kickstarter.com/help/stats?ref = about_subnav$

[3] D' Alessandro B. , *Lecture Notes DSGA: 1001 Intro to Data Science*, New York University (Fall 2015)

[4] Greenberg M.,Pardo B.,et al *Crowdfunding support tools: predicting success and failure*, Extended Abstracts on Human Factors in Computing Syste, ACM (2013), pp 1815–1820

[5] Fawcett T. and Provost F., *Data Science for Business*, (2013)

Appendix

1. Initial Variables

Feature	Type	Description
projectid	int	Unique Project Identifier
state	string	"success" or "fail", it is our target variable.
title	string	String of text that corresponds to the title
category	string	it corresponds to the subcategory of the project.
goal	int	total amount of money requested
amount_pledged	int	total amount of money collected - It is clearly a leakage variables, since if amount_pledged $\geq$ goal, the project was successful.
start_date	date	date when the money collection started
end_date	date	date when the money collection was set to finish
state_changed_at	date	it is also a leaked variable, since if state_changed_at <end_date, it was a successful project
location	string	String representing location in the following pattern: 'City, State'
founder	int	unique identifier for the user that created the project
founder_date	date	date the founder created his user account
backers_count	int	number of people that donated money. It is clearly a leakage variable, since we do not know a-priori how many people will donate
description	string	String of text that corresponds to the short description of the project
url	string	Complete URL for the project in KS
Download_time, currency and currency_rate	None	These variables did not have Data, so were discarded as the vast majority of the instances were NULL

2. Final Variables

Feature	Type	Source	Description
goal	int	Original Dataset	total amount of money requested
log_goal	float	Feature creation	logarithm of goal
title	Scipy Sparse Matrix	Original Dataset	Created a TFIDF matrix with ngram_range=(1, 2), stop_words='english'
description	Scipy Sparse Matrix	Original Dataset	Created a TFIDF matrix with ngram_range=(1, 2), stop_words='english'
num_rewards	int	Scrapping	Number of different rewards offered to donors for different amounts donated
min_pledge_reward	int	Scrapping	Minimum donation required to receive a Reward
max_pledge_reward	int	Scrapping	Maximum donation required to receive a Reward
num_images	int	Scrapping	Number of images posted
has_video	Binary	Scrapping	1 = 'has video', 0 = 'no video'
len_description	int	Scrapping	Total number of characters characters in the Full description of the Project
success	binary	Original Dataset	Target Variable1 = 'success', 0 = 'fail'
date_diff	int	Feature Creation	Total number of days that the project collected money. (end_date-start_date)
amount_per_day	int	Feature Creation	Amount required to collect per day (goal/date_diff)
14 parent_category Dummies	binary	Feature Creation	14 Dummies for 15 Parent Categories1 = 'belongs to category', 0 = 'not in category'

Column	Type	Source	Description
50 - category Dummies	binary	Original Database	50 Dummies for 51 Sub Categories1 = 'belongs to category', 0 = 'not in category'
11 -start_month Dummies	binary	Feature Creation	11 Dummies for 12 start_months1 = 'started in this month', 0 = 'not started in this month
11 -end_month Dummies	binary	Feature Creation	11 Dummies for 12 end_months1 = 'ended in this month', 0 = 'not ended in this month
49 -Location Dummies	binary	Original Database	49 Dummies for 50 States1 = 'belongs to State', 0 = 'not belongs to State'

3. Calibration Plot

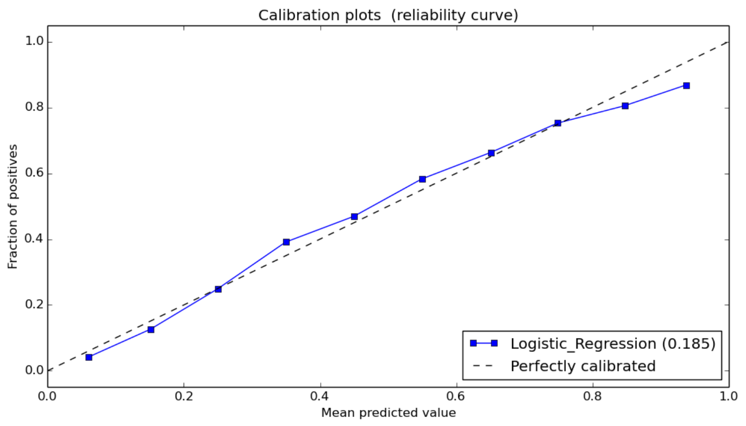


Figure 6: Calibration Plot

4. Learning Curve

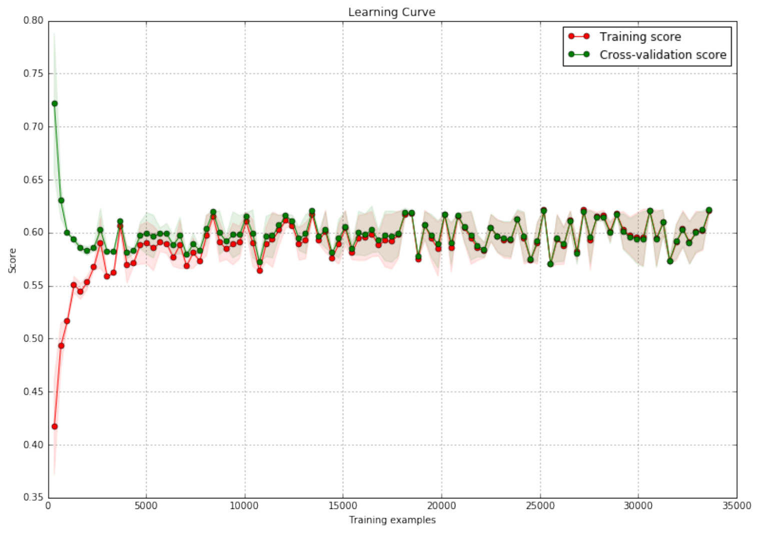


Figure 7: Learning Curve

5. ROC Curves

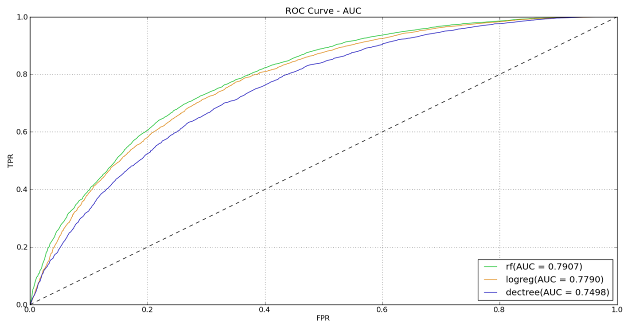


Figure 8: ROC curves, models without text

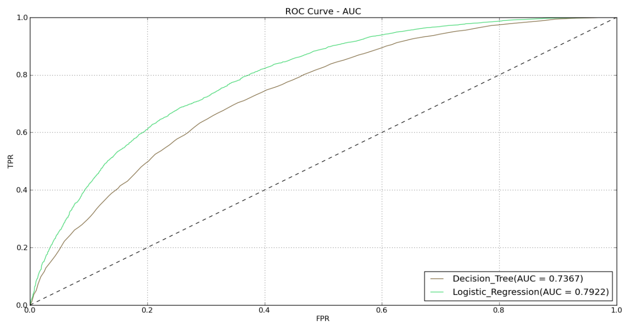


Figure 9: Rock curves, models with text

6. Estimation of Potential Impact

	Original Features	Proposed Features
Success probability estimation	0.609	0.847
goal	600	600
num_rewards	3	13
num_images	4	6
has_video	True	True
date_diff	30	30

7. Team Member Contributions Each member contributed equally in the project, since the database preparation to the deployment ideas.