Modelling Kickstarter Campaign Success Using Classification Methods

ITEC 4230 Project Report

Boris Chan
Christian Tobias
Salar Haddad

Table of Contents

Modelling Kickstarter Campaign Success Using Classification Methods

1. Introduction

Kickstarter is an online crowdfunding platform where creators can share and gather interest on a creative project they would like to launch. The creative projects can range from board games to innovative technology such as high-efficiency water saving shower heads. Crowdfunding is a method of raising capital through the general public to send these projects into production. People that fund Kickstarter projects are usually offered additional rewards (e.g. source code used in the project) or the finished product itself.

A crowdfunding campaign can make or break a business idea, so it is advantageous to investigate common attributes of successful Kickstarter projects. Discovering common patterns or trends from successful and failed projects will provide invaluable business insight on how to maximize the chances for a successful Kickstarter project or crowdfunding campaign.

1.1 Project goal/formalization

Our goal for this project is to analyze from clean and computable datasets and predict whether a Kickstarter project going to be a success or failure based on the features they have (e.g., project goal, duration, category which the project will belong to, etc.).

2. Dataset

The datasets that we worked with was sourced from the Kickstarter platform. The data reflects the results of funded Kickstarter projects. This 2018 dataset includes 375764 records and 15 feature variables.

Initially, all these features were taken with equal importance into the data preprocessing phase, but took note of the 159 different sub categories which may only aid model effectiveness in a minor way. On the other hand, having only 15 main categories should prove to have more impact on classification. The project deadline, monetary goal, date launched, pledged amount and number of backers held the most importance among the features input into the models as they are predicted to have the most effect on model performance.

3. Data Preprocessing

The tuples that had null values for the name and category attributes were dropped, because these tuples had incorrect data in their other attributes (e.g. having a date value in the 'goal' attribute). These tuples were most likely a result of an error when collecting the data.

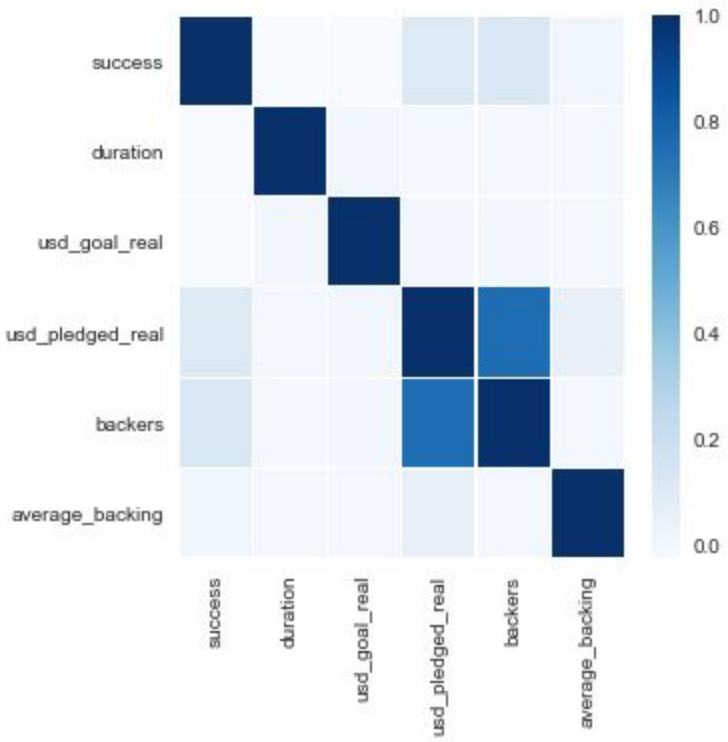
3.1 Feature engineering

The dataset contains an attribute named 'state', which holds the values: successful, failed, and others. At first, this seems like an appropriate attribute to be used as our class in our models. However, there are some values such as 'cancelled' or 'live' which do not describe success of failure. We therefore need to engineer a reliable binary attribute 'success' which only describes success (1) or failure (0) to be used as the class in our models. Within this dataset, success can be defined by the condition: pledged > goal. In other words, if the amount of money pledged is greater than the goal, then the project is considered a success, else it will be considered a failure.

3.2 Correlation analysis

Pearson's correlation coefficient is used to describe the linear correlation between two variables. We wanted to see if there were any attributes that were strongly correlated with each other. Figure 3.1 displays a heatmap representing the correlation coefficient between pairs of attributes.

Figure 3.1 - Heatmap of correlation coefficients.



The attributes 'backers' and 'usd_pledged_real' were highly correlated with each other. We need to consider why these two attributes are correlated. If the number of backers of a project is high, then what value would be pledged? It should make sense that if a project has a high number of backers, then the project should have a high amount pledged; and inversely if a project has a low number of backers, then the project should have a low amount pledged. It is unknown if the situation where a low number of backers results in a high amount pledged exists, however it is highly possible, where each individual backer pledges a greater amount than the mean pledge.

4. Approach

The models were implemented using Python with libraries to assist in data analysis such as Numpy, Pandas, Scikit-learn, etc. Scikit-learn provided the actual implementations for all the models listed in Section 5, as well as a splitting function to split our dataset into a training set and a testing set. Other modules helped perform t-tests and k-fold cross validation, which will be explored in Section 7.

5. Algorithms

5.1 Logistic Regression

Logistic regression is a classification algorithm that uses predictive analysis and is best implemented when the dependent variable is binary. This algorithm uses the logistic function to map a real number to a value between 0 and 1 asymptotically. In this fashion, logistic regression is used to find a relation between independent variables and one dependant binary variable in order to determine the object's probable classification. The training data is used to estimate the coefficients of the model by use of maximum-likelihood estimation. This makes the adjustments to the algorithm so that the probability of getting the results observed in the training data is maximized, assuming a normal distribution. The model can be defined with:

$$P = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

Where P is the probability that an observation β_0 is the constant and β_1 the slope defines the steepness of the curve.

5.2 K-nearest neighbours

KNN is a simple classification algorithm that sorts objects by evaluating similarity based on distance and majority voting. The algorithm is trained with records from the data set that maps labeled objects in *n*-dimensional space defined by *n* attributes. When an unlabeled object is to be classified, the distance to the labeled objects is measured and the closest *k* neighbors are found. These *k*

labeled objects could be from different classes and so the unlabeled object is assigned a class using majority voting.

5.3 Decision Trees (CART)

Decision tree induction is a method which utilizes a tree model of decisions where each branch represents an outcome of the test, and each leaf holds a class label. A trained decision tree will test a given tuple X (where the class is unknown) using its attributes and a path will be traced to a leaf node which holds the class prediction for X. The CART method produces a binary decision tree, which uses the Gini Index as its splitting rule [1, pp. 332-350]. This can be contrasted with the ID3 method which uses information gain as its attribute selection measure. The decision tree will be used as the base model for Random forests and AdaBoost.

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2$$

5.4 Random forests

Random forests are an ensemble learning method using classification. Each of the classifiers in the ensemble is a decision tree which results in a "forest". The individual decision trees are generated using a random selection of attributes at each node to determine the split. Each decision tree depends on the values of a random vector sampled independently with the same distribution for all trees in the forest. During classification, each tree votes and the most popular class is returned.

5.5 AdaBoost

AdaBoost is an ensemble learning method that uses multiple learners to create a model. The algorithm cyclically creates and tests weak learners, and updates the weights of training examples based on the results. Training examples that weak learners classify incorrectly are given a higher weight, and are more likely to appear in the training set of following test, and is therefore 'adaptive', leading up to the final model with the expectation to perform better each iteration. Our implementation of AdaBoost uses Decision Trees as its 'weak learner' or base classifier. The SAMME algorithm [7] will be used, which is very similar to the original AdaBoost algorithm except it puts more weight on misclassified data points, as well as combining weak learners a bit differently than AdaBoost.

6. Experimental Analysis and Results

The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier [1, p. 365], which can be denoted by:

$$accuracy = \frac{TP + TN}{P + N}$$

The accuracy results from an initial analysis using the dataset provides the following results in the Random forests and Logistic Regression model:

Table 5.1 - Accuracy scores from statistically significant correlation

Model	**Accuracy (5 s.f.)**
Random forests	0.99473
Logistic regression	0.99927

The accuracy scores show a near perfect score, however these scores are invalid. We argue that this is the result of multicollinearity. Multicollinearity is a state of high correlations among dependent variables, and is a type of disturbance in the data [8]. If it is present, the statistical inferences made about the data may not be reliable. This phenomenon generally occurs when the variables are highly correlated to each other.

The Pearson's correlation coefficient analysis performed in section 3.2 indicates that the two attributes 'backers' and 'usd_pledged_real' were highly correlated with each other. To address multicollinearity, we dropped these two columns entirely, then created new training and testing sets. The accuracy scores dropped to a reasonable level, as illustrated in the following table:

Table 5.2 - Initial accuracy scores from algorithms.

Model	**Accuracy (5 s.f.)**
Logistic Regression	0.71535
K-Nearest Neighbors	0.74174
Decision Tree	0.69198
Random Forests	0.79358
AdaBoost	0.80628

The results show that the AdaBoost model resulted in the highest accuracy score. It is important to note that the accuracy scores are not always the same. This is because the training and testing set are selected at runtime, therefore different samples are selected every time the code is executed

and hence affect the model. A technique called k-fold cross validation can be employed to find a mean accuracy score at a 95% confidence interval. K-fold cross validation analysis will be described in the evaluation.

Whilst accuracy isn't always the best measure to use when comparing models (this will be discussed further in Section 7), the accuracy of the Random Forests and AdaBoost models are higher than of the Decision Tree. This is to be expected, as Random Forests and AdaBoost use Decision Trees as their base classifiers.

7. Evaluation

7.1 Model selection

A confusion matrix can be used to evaluate the quality of a classifier. The matrix shows the true positives, true negatives, false positives, and false negatives. The measures used to assess a classifier predictive ability are: accuracy, recall, and precision. It should be noted that the accuracy measure might not be reliable when the main class of interest is in the minority [1, p. 386] Precision is a measure of exactness, whereas recall is a measure of completeness [1, p. 386]; these measures can be computed by the following formulas:

$$precision = \frac{TP}{TP + FP} \text{ , } recall = \frac{TP}{TP + FN} \text{ ,}$$

Tables 7.1 to 7.5 illustrate the confusion matrices for each of the models.

Table 7.1 - Logistic Regression

	Predicted 0	**Predicted 1**
Actual 0	44861	3387
Actual 1	17918	9567

Table 7.2 - K-Nearest Neighbours

	Predicted 0	**Predicted 1**
Actual 0	43495	4753
Actual 1	14699	12786

Table 7.3 - Decision Tree

	Predicted 0	**Predicted 1**
Actual 0	27297	21079
Actual 1	2248	25109

Table 7.4 - Random Forests

	Predicted 0	**Predicted 1**
Actual 0	41401	6847
Actual 1	8718	18767

Table 7.5 - AdaBoost

	Predicted 0	**Predicted 1**
Actual 0	43495	4753
Actual 1	14699	12786

Table 7.6 - Precision, recall and F-1 scores for models (5 s.f.)

Model	**Precision**	**Recall**	**F Score**
Logistic Regression	0.74577	0.32732	0.45496
K-Nearest Neighbors	0.72447	0.46542	0.56675
Decision Tree	0.54363	0.91782	0.68282
Random Forests	0.73048	0.68341	0.70616
AdaBoost	0.72389	0.75374	0.73851

Precision can be used as a measure for comparing models when the cost of false-positive is high. For predicting the success of Kickstarter projects, a false positive means that a project that is going to fail (actual negative), has been predicted to be successful (predicted positive). Similarly, recall can also be used as a measure when the cost of false-negative is high. If a Kickstarter project is going to be successful (actual positive), but is predicted to fail (predicted negative).

Given these scenarios, it can be argued that using precision is a better measure for comparing models, as the cost of predicting a false-positive is higher than the cost of predicting a false-negative. If a potential campaign backer uses a model that minimizes the amount of false-positives, then the backer is in a greater position to invest in a successful project. Table 7.5 illustrates that the logistic regression model offered the highest precision measure, whereas the decision tree model offered the highest recall measure.

What if we value both precision and recall? This is where usage of the f-score measure will be appropriate. The f-score measure is the harmonic mean of precision and recall, where it gives equal weight to precision and recall. When a model has perfect precision and recall, the f score would be 1, and when a model has the lowest precision and lowest recall, the f-score would be 0. The results above show that the AdaBoost model has the highest f-score [6].

7.2 K-fold cross validation

It is important to note that the accuracy scores are not always the same. This is because the training and testing set are selected at runtime, therefore different samples are selected every time the code is executed and hence affect the model. This is why k-fold cross validation is required.

In k-fold cross validation, the dataset is randomly partitioned into k subsets, $D_1, D_2, \ldots, D_k$. Training and testing is done k times. In iteration i, subset D_i is reserved as the test set, and the remaining subsets are used to train the model. For instance, when $i = 1$, $D_2, \ldots, D_k$ will be used to train the model and D_1 will be used as the testing set; then when $i = 2$, $D_1, D_3, \ldots, D_k$ will be used to train the model and D_2 will be used as the testing set, etc. [1, p. 370].

This paper will use stratified cross-validation, where the folds are stratified to ensure that each fold is a good representative of the whole. For example, in a binary classification problem where each class consists of 50% of the data, it is best to arrange the data so that every fold follows this distribution [4]. A study conducted by Kohavi concludes that using 10-fold stratified cross validation is the best method to use for model selection, in terms of of bias and variance when compared to regular cross-validation [5].

Table 7.7 - Sample mean accuracy for 10-fold cross validation (5 s.f.)

Model	**Accuracy**	**Error Rate**	**Margin of Error**
Logistic Regression	0.71827	0.28173	0.0012323
K-Nearest Neighbors	0.74311	0.25689	0.0010319
Decision Tree	0.69129	0.30871	0.0010781
Random Forests	0.79469	0.20531	0.0012592
AdaBoost	0.80563	0.19437	0.0010042

For a given model, the mean error rates calculated in the cross-validations may be considered as different independent samples from a probability distribution, where they follow a t-distribution with $k-1$ degrees of freedom (where $k=10$ from the 10-fold cross validation). We can conduct a test of statistical significance to know if the difference between 2 error rates is attributed to chance. Our hypothesis is that two models are the same (the difference in mean standard error is 0). If we reject this hypothesis (null hypothesis), then we can conclude that the difference between the two models is statistically significant, in which case we can select the model with the lower error rate [1, p. 372]. Let us compare AdaBoost and Random Forests (as they have the lowest error rates).

H_0 : AdaBoost has the same performance as Random Forests.
H_a : The models have a difference in performance.

$$t = \frac{err(M_1) - err(M_2)}{\sqrt{var(M_1 - M_2)/k}}$$

The t-test for the mean accuracies from the 10-fold cross validation of the AdaBoost and Random Forests resulted in a t-statistic of: 14.62702 and a p-value of 1.96296e-11. From the t-table shows that the critical value 2.262 (at $k-1$ degrees of freedom). Our t-statistic is greater than the critical value, and therefore we can reject the null hypothesis and can select the model with the lower error rate, that is the AdaBoost model.

7.3 ROC/AUC Evaluation

Another evaluation method is to analyse the Receiver operating characteristic (ROC) curve and the area under the curve(AUC).

ROC curve is a visualization to evaluate the trade-off between true positive rate (TPR) and false positive rate (FPR) calculated as TP/P (P as positive tuples) and FP/N (N as negative tuples). Naturally, one figure increases as the other decreases. The curve itself is an algorithm's predictions of object classification, and the shape it takes is dependent on the success of binary separation in this case.

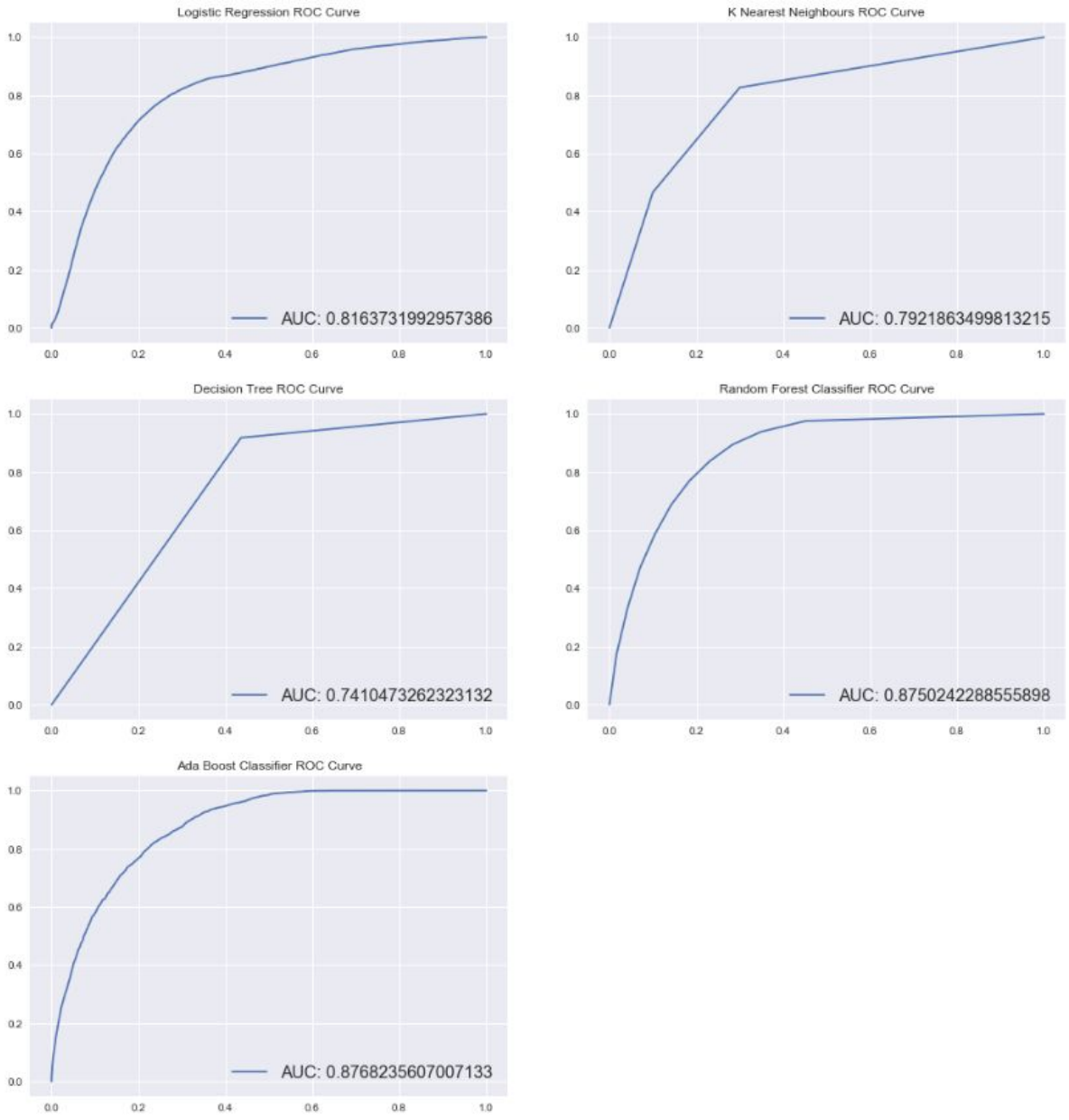
The graph organizes this data with TPR along the y axis and FPR along the x axis. To simplify the plotting procedure, ordered pairs are calculated by using a tuple's true label and probability score or threshold. The threshold divides the model's prediction in two with objects above the threshold being predicted as successful Kickstarter projects and below as unsuccessful. From these predictions, TPR and FPR are calculated by looking at the real classification of the objects in comparison to the predictions made. This procedure is carried out for threshold values ranging from 0-1 in order to create the ROC curve.

Knowing this, an ideal classification model will have an ROC curve that tends towards 1 on the y axis. This indicates a greater TPR and a lower FPR. Having a more linear ROC curve indicates a weaker degree of class separation. AUC is related to ROC and provides a similar assessment of accuracy displayed as a percentage. An AUC of 0.5 reflects a model has nearly no ability to distinguish different classes, while values closer to 1 show an ability to distinguish classes accurately. Figure 7.1 illustrates the ROC Curves while Table 7.8 displays the models and their AUC values.

Table 7.8 - AUC values

Model	**AUC**
Logistic Regression	0.81568
K-Nearest Neighbors	0.79241
Decision Tree	0.74104
Random Forests	0.87249
AdaBoost	0.87798

Figure 7.1 - ROC Curves



Analyzing the ROC curves of the classifiers, AdaBoost possesses the greatest ability to distinguish classes with Random Forest being a close second. This is further supported by the AUC which reveals that these two models are comparatively more accurate than the other models by a considerable margin. Conversely, the Decision tree performed with the least accuracy as reflected by its ROC curve and confusion matrix in a prior section. Classifying nearly half the negative tuples incorrectly caused the ROC curve to be less rounded and further from the top left corner of the graph.

8. Conclusion

In summary, our findings showed that the best model depends on the measure that is valued by the individual when looking for Kickstarter projects to invest in. Precision and Recall are popular measures that can be used depending on the situation, however we argued that for Kickstarter projects, the cost of false-positive is high, therefore the selected model should have a high precision score. The results in Section 7.6 show that the Logistic Regression model had the highest precision. If both precision and recall are important, then the model with the highest f-score should be considered, which for this paper was the AdaBoost model. Furthermore, in hypothesis testing, the AdaBoost model proved to be a statistically significant result in having the lowest error rate, compared to the Random forests model. The analysis of ROC curves indicates that the AdaBoost model also had the highest AUC value. Overall, the AdaBoost model performs notably well in comparison to the other classifiers, and should be highly considered by stakeholders to model the success of Kickstarter campaigns.

9. References

[1] J. Han, M. Kamber and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. Morgan Kaufmann; 3 edition (July 6, 2011), 2011.

[2] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, and etal., "Top 10 algorithms in data mining," Knowledge and Information Systems, vol. 14, no. 1, pp. 1–37, 2007.

[3] "Logistic Regression", Saedsayad.com, 2019. [Online]. Available: https://www.saedsayad.com/logistic_regression.htm. [Accessed: 27- Mar- 2019].

[4] "Understanding stratified cross-validation", Cross Validated, 2019. [Online]. Available: https://stats.stackexchange.com/questions/49540/understanding-stratified-cross-validation. [Accessed: 27- Mar- 2019].

[5] R. Kohavi, A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. 1995.

[6] "F-Score Definition," DeepAI. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/f-score. [Accessed: 27-Mar-2019].

[7] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class AdaBoost," Statistics and Its Interface, vol. 2, no. 3, pp. 349–360, 2009.

[8] "Multicollinearity," Statistics Solutions. [Online]. Available: https://www.statisticssolutions.com/multicollinearity/. [Accessed: 31-Mar-2019].