

# It's All Funds & Games

## Predicting Kickstarter Success

Mark Giannini

University of Notre Dame  
mgianni1@nd.edu

Patrick Tinsley

University of Notre Dame  
ptinsley@nd.edu

Brian Tunnell

University of Notre Dame  
btunnell@nd.edu

### 1. Introduction

#### 1.1 Project Plan

For our semester project, we decided to test our ability to predict whether or not a given Kickstarter campaign will be successful. In order to be deemed a success, the proposed campaign needs to meet or exceed the funding goal proposed by the initial author by a predefined deadline; anyone can contribute as long as the campaign is still active. In the context of our project, each instance in our data set has a unique project ID and fourteen features; these include project name, project description, keywords, financial goal in US dollars, the project deadline and the number of backers contributing to and supporting the project. Using sentiment analysis and other logistic regression techniques we have learned in previous classes, we plan to predict the binary `final_status` field, which indicates a successful project (1) or a failed attempt (0).

#### 1.2 Data Sources

Initially, we planned to crawl the data from the Kickstarter website ourselves. However, upon browsing a plethora of Kaggle competitions, we found a pre-built data set that contains all our fields of interest. The supplied data has 108,129 rows, each corresponding to a project proposal submitted between May 2009 and May 2015. Each instance has the following features: Project ID, Name, Description, Funding Goal, Project Keywords, Disable Communication, Country, Currency, Deadline Date, Date Created, Date Launched, State Changed At, Launched At, Number of Backers and finally, the targeted response variable, Final Funding Status.

#### 1.3 Proposed Evaluation

To evaluate our models predictive power, we plan on splitting our data into two sets. The first partition will be the training set, and it will be used to build and train our model. The second partition will be the testing set, and it will be used to validate the model. If we split the data 66.6%-33.3% respectively, the training set will have 72,446 rows, and the testing set will have 35,683 rows. By withholding a subset of the full data set, we have the power to test our final model on unseen data, which can be used to evaluate estimator performance; this technique also helps to avoid over-

fitting, producing a more generalized model capable of predicting the success of future Kickstarter projects.

### 2. Related Work

Given the nature of Kaggle, there are several other kernels or projects that deal with predicting the success of Kickstarter campaigns. However, since the service is competition-based, other submissions are blocked. Nevertheless, many members of the data science community have tried their hand at this task. One such attempt included mostly spatial and geographical features, such as, country, state, and county; that work can be seen [here](#). Another attempt focused on the categories of the proposed project, such as music, theatre, and art; that work can be seen [here](#). In the latter attempt, the author found accuracy rates of 63%, 68%, and 65% for kNN, random forest, and logistic regression, respectively. For our project, we hope to surpass 80% accuracy.

### 3. Problem Definition

The problem of determining successful Kickstarter campaigns is inherently a classification problem; either the campaign succeeds or fails. In the context of the data, this is described in the `final_status` field, which contains only zeros and ones for failures and successes, respectively. Additionally, since the target variable is binary, logistic regression techniques will be used throughout the model evaluation process.

### 4. Proposed Methodology

To tackle this problem, we decided to apply several sklearn classifiers at the training data. The architecture for sklearn is very similar across models, which makes fitting the models very easy. Once the data has been split up into training and testing, constructing the model is as simple as calling a fit function on `X_train`. Listed below are the classifiers we built on the data.

- Decision Tree
- Logistic Regression
- Random Forest (Gini & Entropy)
- Naive Bayes (Bernoulli & Gaussian)

In order to evaluate the models, we looked at the following metrics as provided by the sklearn.metrics package: Confusion Matrix, Accuracy, Precision, Recall, F-1 Score, Classification Report, AUC, and Zero-One Loss. To keep our code clean, we wrote a function called `evaluate_model` that calculated each of these metrics given a vector of predicted labels on the `X_test` data and the actual labels (`y_test`).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

Copyright © 2018 ACM [to be supplied]. . . \$15.00.  
<http://dx.doi.org/10.1145/>

## 5. Data & Experiments

### 5.1 Data Set

As stated before, the fourteen features of this data set differ in type. The name, description, and keywords fields are string objects, disable\_communication is Boolean, goal is a float, country and currency are nominal, and the rest are integers (int64). For the first part of this project, we did not worry ourselves with the string objects, though they will be used later to cluster projects together. For the sake of dimensionality reduction, we removed the created\_at and currency fields. The created\_at column was removed since the overlap between created\_at and launched\_at was very strong; additionally, since the time of creation is hidden from the public, we do not wish to use it in predicting in real-world contexts. The currency column was essentially a proxy column for country, meaning it added little new information, so it was removed as well.

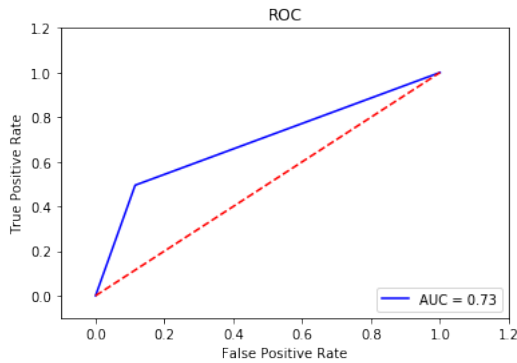
After stripping away the five columns mentioned previously, we entered the feature engineering phase. The date-related columns, originally integers representing time in the UTC format, were converted to Datetime objects using the pandas to\_datetime function. From these objects, we extracted the hour, day, month, and year for the launch and deadline of the proposed campaign. After label-encoding these fields with the pandas get\_dummies function, we encoded the country field as well. The new data set had 140 features, compared to the initial 14.

### 5.2 Experiments

#### 5.2.1 Decision Tree

For the first model, we decided to fit a decision tree to the training data. Using a grid-search package provided by sklearn, we found that the optimal decision tree consisted of a Gini index criterion and max depth of 9. A visualization rendered by the graphviz package can be seen by running the code; the tree is rather large, and is not legible at this size. The root node of the tree separates the data based on the goal field. According to this model, projects that have lower goals are more likely to succeed, which matches up with our intuition. Below are the evaluation metrics and ROC curve for the decision tree model.

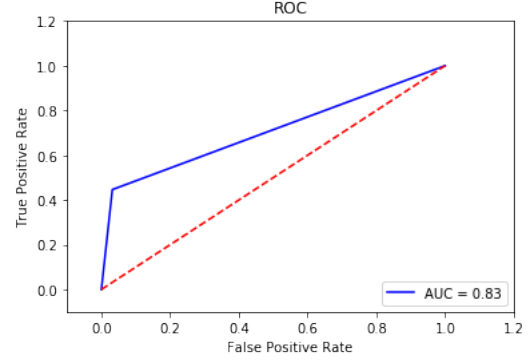
Accuracy	Precision	Recall	F-1	AUC
0.75985	0.49497	0.67057	0.56954	0.72907



#### 5.2.2 Logistic Regression

For the second model, we decided to fit a logistic regression model. As we can see, except for the precision metric, the regression outperformed the decision tree in the previous section. Below are the evaluation metrics and ROC curve for the logistic regression model.

Accuracy	Precision	Recall	F-1	AUC
0.80063	0.44643	0.86852	0.58973	0.82787



#### 5.2.3 Random Forest

We fit two Random Forest models using the python package sklearn and the function RandomForestClassifier. The Random Forest model is an ensemble method for classification, and can provide better predictive power than a simple Decision Tree model. Instead of relying on one decision tree for prediction, a Random Forest model constructs multiple decision trees and uses the mode of the classes for classification. The first Random Forest model was fit using

Gini Impurity as the split criteria, and the second used Entropy as the split criteria. Gini Impurity measures the probability that a randomly chosen element would be incorrectly labeled if it were labeled based on the distribution of labels within the data set. The following equation is used to calculate Gini Impurity for a set of items with  $J$  classes where  $p_i$  is the proportion of items with class  $i$  in the data set.

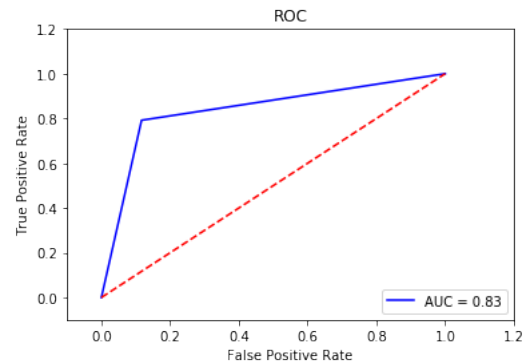
$$I_G(P) = 1 - \sum_{i=1}^J p_i^2 \quad (1)$$

Entropy, alternatively is used in the calculation of Information Gain, which chooses the split that results in the purest daughter nodes. Entropy is defined as the following.

$$H_T = - \sum_{i=1}^J p_i \log_2(p_i) \quad (2)$$

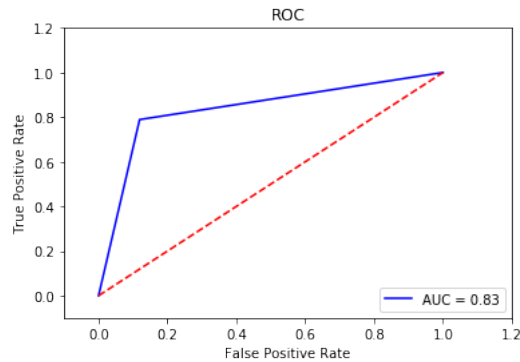
Information Gain is calculated by subtracting the weighted sum of Entropy(children) from Entropy(parent). The two Random Forest models had very similar results upon fitting them to the training data. When using Gini impurity as the split criteria, we obtained the results below.

Accuracy	Precision	Recall	F-1	AUC
0.85387	0.79193	0.76212	0.77674	0.83095



When using entropy as split criteria, we obtained similar evaluation metrics. However, this was a slightly less accurate method for creating our Random Forest, as seen in the results below.

Accuracy	Precision	Recall	F-1	AUC
0.85135	0.78905	0.75781	0.77312	0.828063

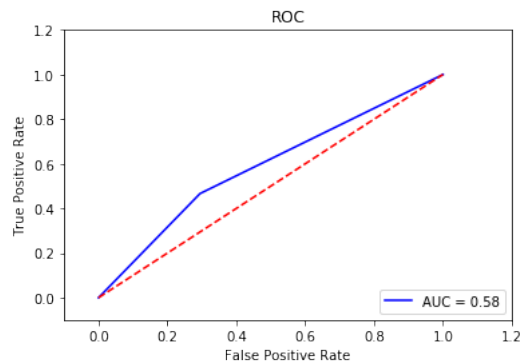


#### 5.2.4 Naive Bayes

We also chose to fit a Naive Bayes model with a Gaussian distribution as well as a Bernoulli distribution for the numeric variables. The Naive Bayes model is a learning algorithm based on applying Bayes Theorem along with the naive assumption that all data features are independent. The GaussianNB model within the python package sklearn seemed like a good initial model to examine because of its core assumption that the features are normally distributed. Further, the BernoulliNB model was a good juxtaposition because of its ability to work well with binary or Bernoulli distributed variables. Upon testing, the Gaussian model did not perform as well as the Bernoulli model. This makes sense with the large number of binary variables within the numeric feature data set. The metrics for each of the classifiers can be seen below.

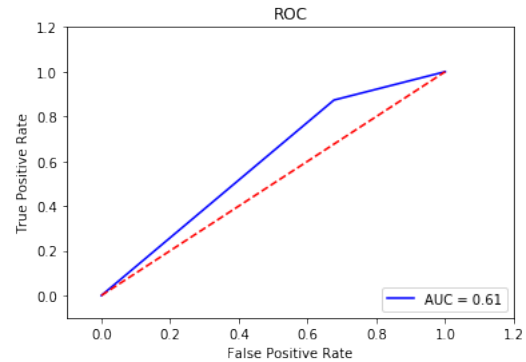
Naive Bayes - Bernoulli

Accuracy	Precision	Recall	F-1	AUC
0.62870	0.46651	0.428056	0.44645	0.58235



Naive Bayes - Gaussian

Accuracy	Precision	Recall	F-1	AUC
0.49995	0.87322	0.37894	0.52852	0.61133



For our final project, we intend to explore fitting a MultinomialNB model performs for our features requiring text classification as well as an out-of-core Naive Bayes model. This model would require pulling from each of the previously discussed models and combining each of their partial model fits into a full model capable of efficiently handling the different feature types within this set.

## 6. Conclusions

### A. Appendix Title

Appendix, if needed.

### Acknowledgments

Acknowledgments, if needed.