Launch Hard or Go Home!

Predicting the Success of Kickstarter Campaigns

Vincent Etter Matthias Grossglauser Patrick Thiran
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
firstname.lastname@epfl.ch

ABSTRACT

Crowdfunding websites such as Kickstarter are becoming increasingly popular, allowing project creators to raise hundreds of millions of dollars every year. However, only one out of two Kickstarter campaigns reaches its funding goal and is successful. It is therefore of prime importance, both for project creators and backers, to be able to know which campaigns are likely to succeed.

We propose a method for predicting the success of Kickstarter campaigns by using both direct information and social features. We introduce a first set of predictors that uses the time series of money pledges to classify campaigns as probable success or failure and a second set that uses information gathered from tweets and Kickstarter's projects/backers graph.

We show that even though the predictors that are based solely on the amount of money pledged reach a high accuracy, combining them with predictors using social features enables us to improve the performance significantly. In particular, only 4 hours after the launch of a campaign, the combined predictor reaches an accuracy of more than 76% (a relative improvement of 4%).

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

Crowdfunding; Kickstarter; time-series classification; success prediction; social features; Twitter

1. INTRODUCTION

Kickstarter[1] is a crowdfunding website: people with a creative idea can open a campaign on the website to gather money to make it happen. When launching a campaign, the creator sets a funding goal and a deadline. Then, people can pledge money towards the project, and receive various rewards in return. Rewards range from the acknowledgement of a backer's participation to deep involvement in a product's design.

The fundraising model is *all or nothing*: once its deadline is reached, a campaign is considered successful if and only if it has reached its goal. In this case, backers actually pay the money they pledged and the project idea is realized. In the case where the goal is not reached, the campaign has failed and no money is exchanged.

As only 44% of campaigns reach their goal overall, it is of high interest for creators to know early on the probability of success of their campaign, to be able to react accordingly. Users whose campaigns are failing to take off might want to increase their visibility and start a social media campaign, while those whose campaigns are highly likely to succeed could already start working on them to deliver faster, or look into possible extensions of their goal.

Similarly, backers could also benefit from such a prediction. They could engage their friends and social network in backing a campaign, if its probability of success is low shortly after its launch. When the success probability is high, backers could also adjust their pledge, maybe reducing it a little in order to support another campaign, while being confident that the campaign will still succeed. Some online tools, such as *Kicktraq*[2] and *CanHeKickIt*[3], provide tracking tools and basic trend estimators, but none has yet implemented proper success predictors.

There have been several studies published on crowdfunding platforms: Mollick [4] provides insights about of the dynamics of the success and failure of Kickstater campaigns. He presents various statistics about the determinant features for success and analyzes the correlation of many campaign characteristics with its outcome. Wash [7] focuses on a different platform, called Donors Choose, where people can donate money to buy supplies for school projects. He describes how backers tend to give larger donations when it allows a campaign to reach its goal, and also studies the predictability of campaigns over time.

Greenberg et al. [3] propose a success predictor for Kickstarter campaigns based solely on their *static* attributes, i.e., attributes available at the launch of a campaign. They obtain a prediction accuracy of 68%, which we will use as baseline

[1] `http://www.kickstarter.com`

COSN'13, October 7–8, 2013, Boston, Massachusetts, USA.

ACM 978-1-4503-2084-9/13/10 ...$15.00.
http://dx.doi.org/10.1145/2512938.2512957.

[2] `http://www.kicktraq.com`
[3] `http://canhekick.it`

when presenting the results in Section 3. However, to the best of our knowledge, no one has studied success prediction based on *dynamic* attributes of a campaign.

Of course, predicting a time series with a finite horizon has several other applications. An obvious extension of this framework could easily be applied to online auctions, where the final amount to be reached can be predicted. Financial products, such as options, could also benefit from such predictors. We focus on building models for predicting the success of crowdfunding campaigns, using Kickstarter as an example. The techniques and results presented below however are not restricted to this platform and should apply to any similar setting.

In Section 2, we describe our dataset, its main characteristics and the preprocessing we apply. We then present our different predictors in Section 3, explaining the models and showing their individual performance. We next propose a method to combine them that significantly improves the accuracy over individual predictors. Finally, we conclude in Section 4.

2. DATASET DESCRIPTION

Our dataset consists of data scraped from the Kickstarter website between September 2012 and May 2013. It consists of 16 042 campaigns that were backed by 1 309 295 users.

2.1 Collecting the Data

New campaigns are discovered on the *Recently Launched*[4] page of Kickstarter. Once a new campaign is detected, its main characteristics, such as its category, funding goal and deadline, are collected and stored in a database. Then, a crawler regularly checks each campaign's page to record the current amount of pledged money, as well as the number of backers, until the project's funding campaign reaches its end.

In parallel, we monitor[5] Twitter for any public tweet containing the keyword *kickstarter*. For each tweet matching our search, we record all its data in the database. To determine if the tweet is related to a particular campaign, we search for a campaign URL in its text. If any is found, the tweet is identified in the database as a reference to the corresponding campaign. We thus have, for each campaign, all public tweets related to it.

Along with Twitter, Kickstarter integrates Facebook on its website, as an other way of spreading the word about campaigns. However, contrary to Twitter, most Facebook posts are not public, being usually restricted to the user's friends. As a result, a search similar to the one described above performed on Facebook usually yields very few results. For this reason, we only use Twitter in our dataset.

Finally, we regularly crawl the *Backers* page of each campaign to get the list of users who pledged money, and store them in our database. This last step being time-consuming to perform, it is done every couple of days, resulting in only a few snapshots of the list of backers and therefore a coarse resolution of the time at which each backer joined a campaign.

[4]`http://www.kickstarter.com/discover/recently-launched`

[5]We use the Twitter Streaming API to search for the keyword *kickstarter*. Because few tweets match this search query compared to the global rate of tweets, we know that we get a large uniform fraction of the relevant tweets (usually close to 100%) [6].

	Successful	Failed	Total
Campaigns	7739	8303	16042
Proportion	48.24%	51.76%	100%
Users	1 207 777	171 450	1 309 295
Pledges	2 030 032	212 195	2 242 227
Pledged $	141 942 075	16 084 581	158 026 656
Tweets	564 329	173 069	737 398

Table 1: Global statistics of our dataset of Kickstarter campaigns. We show the values for successful and failed campaigns separately, as well as the combined total. Users are unique people who have backed at least one campaign.

	Successful	Failed	All
Goal ($)	9595	34 693	22 585
Duration (days)	30.89	33.50	32.24
Number of backers	262	25	139
Final amount	216.60%	11.40%	110.39%
Number of tweets	73	20	46

Table 2: Campaign statistics of our Kickstarter dataset. The average values for successful and failed campaigns are given, as well as the average over all campaigns. The final amount is relative to the campaign's goal.

2.2 Dataset Statistics

Table 1 describes the global statistics of our dataset, for successful and failed campaigns separately, as well as the combined total. Table 2 shows average statistics for individual campaigns. As expected, failed campaigns have a much higher goal on average (close to four times higher), but it is interesting to note that they also have a longer duration[6]. Moreover, we have a nearly even split between successful and failed campaigns, with more than 48% of campaigns that reach their funding goal. The reported[7] global success rate of Kickstarter is lower, with 44% of successful campaigns overall. This difference could be explained by the fact that our dataset only contains recent campaigns, that benefit from the growing popularity of crowdsourcing websites.

2.3 Dataset Preprocessing

As explained in Section 2.1, each campaign is regularly sampled by our crawler to get its current amount of pledged money and number of backers, until it ends. On average, a campaign's *state* is sampled every 15 minutes, resulting in hundreds of samples at irregular time intervals.

To be able to compare campaigns with each other, we resample each campaign's list of states to obtain a fixed number of $N_S = 1000$ states. The time of each state is normalized with respect to the campaign's launch date and duration. We divide the current amount of money pledged of each state by the goal amount to obtain a normalized amount.

A campaign c is thus characterized by its funding goal $G(c)$, launch date $L(c)$, duration $D(c)$, final state $F(c)$ (equal to 1 if the campaign succeeded, 0 otherwise) and a series of state samples $\{\mathbf{S}_i(c)\}_{i\in\{1,2,\ldots,N_S\}}$. Each state $\mathbf{S}_i(c)$ is itself

[6]Project creators can choose the duration of their campaign. The default value is 30 days, with a maximum of 60 days.

[7]`http://www.kickstarter.com/help/stats`

Variable	Description
$G(c)$	Funding goal
$L(c)$	Launch date
$D(c)$	Duration
$F(c)$	Final state (1 if successful, 0 otherwise)
$\{\mathbf{S}_i(c)\}$	Series of resampled states
$t_i(c)$	Sample time of the i^{th} state
$M_i(c)$	Pledged money at time t_i
$B_i(c)$	Number of backers at time t_i

Table 3: List and description of the variables describing a campaign c. The states $\{\mathbf{S}_i(c)\}$ are resampled to obtain $N_S = 1000$ states at regular time intervals, as explained in Section 2.3.

composed of the amount of money pledged $M_i(c)$ (normalized with respect to $G(c)$) and the number of backers $B_i(c)$.

Because each campaign is resampled to have N_S evenly-spaced states, the time $t_i(c)$ of the i^{th} state $\mathbf{S}_i(c)$ is simply defined as

$$t_i(c) = L(c) + \frac{i-1}{N_S - 1} D(c),\ i \in \{1, 2, \ldots, N_S\}.$$

Table 3 summarizes the variables describing a campaign c.

3. SUCCESS PREDICTORS

Given a campaign c and its associated variables described above, we now introduce the algorithms we chose to predict its success. Our predictors use partial information: to predict the success of c, they only consider a prefix $\{\mathbf{S}_i(c)\}_{i \in \mathcal{I}}$ of its series of states, where $\mathcal{I} = \{1, 2, \ldots, S\}$ and $1 \leq S < N_S$.

Below, we will present the results for various values of S, i.e., predictions made at different states of progress of the funding campaigns. Each result is obtained by a predictor that is trained independently. It would be possible to have predictors that are able to predict the success for several (or all) values of S, however, we chose to have separate predictors for each value of S. Global predictors would require a variable input size (as the length of the history depends on S), which is more complicated to handle.

3.1 Dataset Separation

In order to train our predictors, select their parameters and evaluate their performance, we separate the dataset into 3 parts: 70% of the campaigns are selected as the *training set*, 20% as the *validation set* and the remaining 10% as the *test set*. These sets are randomly chosen and all results presented below are averaged over 10 different assignments.

3.2 Money-Based Predictors

The first family of predictors that we define only uses the series of amounts of money pledged $\{M_i(c)\}_{i \in \mathcal{I}}$, which we call *trajectory*, to predict the outcome of a campaign c. The first predictor, described in Section 3.2.1, simply compares the trajectory of a campaign with other known campaigns and makes a decision based on the final state of the k closest ones. The second, described in Section 3.2.2, builds a probabilistic model of the evolution of trajectories and predicts the success probability of new campaigns using this model. The performances of these two predictors are shown in Section 3.2.3.

3.2.1 kNN Classifier

Our first model is a k-nearest neighbors (kNN) classifier [2]. Given a new campaign c, its partial trajectory $\{M_i(c)\}_{i \in \mathcal{I}}$ and a list of campaigns for which the ending state is known, kNN first computes the distance between c and each known campaign c'

$$d_{\mathcal{I}}(c, c') = \sqrt{\sum_{i \in \mathcal{I}} (M_i(c) - M_i(c'))^2}.$$

Then, it selects $\text{top}_{k,\mathcal{I}}(c)$, the k known campaigns that are the closest to c with respect to the distance defined above, and computes the probability of success $\phi_{\text{kNN}}(c, \mathcal{I})$ of c as the average final state of these k nearest neighbors:

$$\phi_{\text{kNN}}(c, \mathcal{I}) = \frac{1}{k} \sum_{c' \in \text{top}_{k,\mathcal{I}}(c)} F(c').$$

3.2.2 Markov Chain

Our second predictor also uses the campaign trajectories $\{M_i\}$, this time to build a time-inhomogeneous Markov Chain that characterizes their evolution over time. To do so, we first discretize the (time, money) space into a $N_S \times N_M$ grid. This means that we discretize each campaign trajectory $\{M_i(c)\}$ to map the pledged money to a set $\mathcal{M}$ of N_M equally-spaced values[8], ranging from 0 to 1. For example, if $N_M = 3$, $\mathcal{M} = \{0, 0.5, 1\}$.

We thus obtain for each campaign c a series of discretized amounts of money pledged $\{M'_i(c)\}_{1 \leq i \leq N_S}$. The Markov model defines, for each sample i, a transition probability

$$P_{m,m'}(i) = \mathbb{P}(M'_{i+1} = m' \mid M'_i = m),$$

defining a transition matrix $\mathbf{P}(i) \in [0,1]^{N_M \times N_M}$, $\forall i \in \{1, 2, \ldots, N_S - 1\}$. These transition matrices are not specific to a campaign but learned globally over all campaigns in the training set.

Success Prediction with the Markov Model.

Using the transition probabilities described above, predicting the success of a campaign c is straightforward given its discretized amount $M'_i(c)$ at time i. We compute its success probability $\phi_{\text{Markov}}(c, i)$ given its current discretized amount of pledged money $M'_i(c) = m$ as

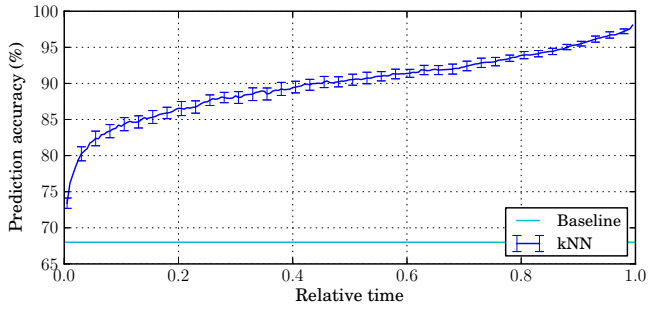
$$\begin{aligned}
\phi_{\text{Markov}}(c, i) &= \mathbb{P}(M'_{N_S}(c) = 1 \mid M'_i(c) = m) \\
&= \sum_{m'} \mathbb{P}(M'_{N_S}(c) = 1 \mid M'_{i+1}(c) = m') \cdot \\
&\qquad \mathbb{P}(M'_{i+1}(c) = m' \mid M'_i(c) = m) \\
&= \left[\prod_{i'=i}^{N_S - 1} \mathbf{P}(i') \right]_{m,1},
\end{aligned}$$

where the last step is obtained by repeatedly applying the law of total probability.

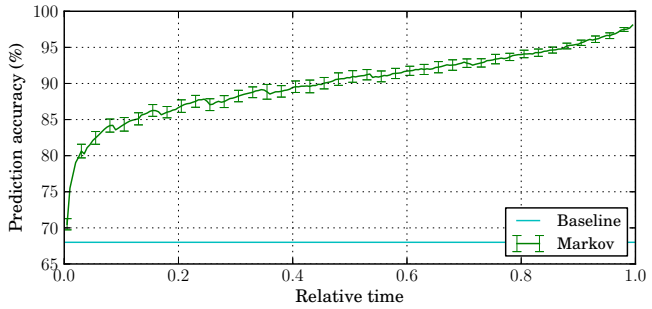
3.2.3 Results

We select the best parameters for each predictor by doing an exhaustive search on a wide range of values and evaluating the corresponding performances on the validation set. The optimal parameters found are $k = 25$ for *kNN* and $N_M = 30$ for *Markov*.

[8] All values higher than 1 are mapped to 1.



(a) *kNN* predictor



(b) *Markov* predictor

Figure 1: Prediction accuracy of the kNN and Markov predictors, along with the static baseline of Greenberg et al. [3]. For each relative time $t \in [0, 1]$, a predictor was trained using $\{M_i(c)\}_{i<\lfloor tN_S \rfloor}$ for all campaigns c in the training set. The value shown is the median accuracy over 10 runs and the error bars show the standard deviation over these runs.

Figure 1 shows the corresponding prediction accuracy over the test set for *kNN* (Figure 1a) and *Markov* (Figure 1b) predictors, along with the baseline of Greenberg et al. [3]. The baseline uses static campaign attributes, such as category, goal, and whether it has a video description or not, to predict the success of campaigns before their launch. The best accuracy obtained with this approach is 68%.

The two predictors perform similarly, and very well: after 15% of the duration of a campaign, its current amount of money pledged allows to predict its success with an accuracy higher than 85%. As time goes by, this accuracy steadily increases, to reach more than 97% in the very last moments.

However, *kNN* is very costly compared to *Markov*: it requires to keep all training samples in memory and to compute the distance to each of them when we want to classify a new sample. In contrast, *Markov* is compact, requiring to store only the matrices $\mathbf{P}(i)$, and computes the success probability of new samples very efficiently, requiring only matrix multiplications. It is thus noticeable that such a lightweight and elegant model performs as well as a more heavyweight method.

3.3 Social Predictors

Contrary to the predictors presented above, which use the amount of money pledged to predict the success of campaigns, the social predictors use side information, obtained from Twitter and Kickstarter's projects/backers graph. The first, described in Section 3.3.1, uses features extracted from the series of tweets related to a campaign, such as the number of retweets and the number of people who tweeted. The second, described in Section 3.3.2, considers a graph linking projects and backers to extract some project features such as its number of first-time backers and the number of other projects with common backers. Both predictors then use a support vector machine (SVM) [1] to predict the campaigns' success based on the extracted features. Their results are shown in Section 3.3.3.

3.3.1 Tweets

As mentioned in Section 2, we have, for each campaign c, the list of all public tweets $\{T_i(c)\}$ that mention it. As each tweet has a timestamp, we can select the subset of tweets $\mathcal{T}_t(c) = \{T_i(c) \mid \text{timestamp}(T_i(c)) < t\}$ that were published before a time t. Using $\mathcal{T}_t(c)$, we can extract the following features:

- number of tweets, replies and retweets,
- number of users who tweeted,
- estimated number of backers[9].

We then add the campaign's goal $G(c)$ and duration $D(c)$ to these features and feed them to an SVM, resulting in a predictor $\phi_{\text{tweets}}(c, t)$.

3.3.2 Projects/Backers Graph

To extract the second set of features, we first need to build the projects/backers graph G_1. This graph contains all projects and backers in our training set as vertices, and has an edge between a project p and a backer b if and only if b backed p. The resulting graph is an undirected and unweighted[10] bipartite graph.

From G_1, we can extract the co-backers graph G_2: it is the projection of G_1 onto the project vertices. G_2 is an undirected weighted graph, where vertices are projects and the weight of an edge between two projects p_1 and p_2 is the number of backers who have pledged money to both p_1 and p_2. Figure 2 shows an example of a projects/backers graph G_1 (Figure 2a) and the corresponding co-backers graph G_2 (Figure 2b).

Using G_1 and G_2 built from our training set, we can now consider a new campaign c whose probability of success we want to estimate at some time t. To do so, we add the project p corresponding to c to G_1 and G_2, using its list of backers at time t to add the necessary edges in both graph.

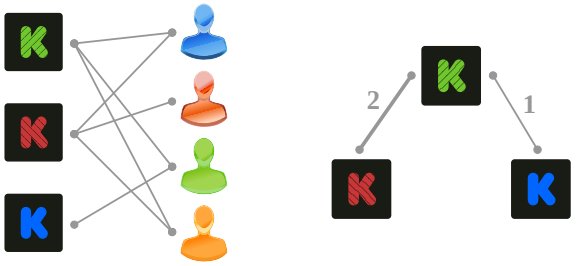
Then, we extract the following features of the project p:

- number of projects with co-backers (i.e., the degree of p in G_2),
- number and proportion of these projects whose campaigns are successful,
- number of backers,
- number and proportion of first-time backers[11].

[9]We estimate the number of backers by counting the number of tweets that contain texts such as "I just backed project X", which is the default message proposed by Kickstarter.

[10]It would be interesting to consider a weighted version of this graph, where the weight of each edge corresponds to the amount of money pledged. Unfortunately, we do not have access to this information, and thus can only consider the unweighted version.

[11]First-time backers are users that only pledged money to the current project, and no other.



(a) Project/backers graph G_1 (b) Co-backers graph G_2

Figure 2: Example of a project/backers graph G_1 and the corresponding co-backers graph G_2. G_1 contains both projects and backers as vertices, and has an edge between a project p and a backer b if and only if b pledged money to p. G_2 is the projection of G_1 onto the project vertices, where the weight of an edge between two projects represents their number of common backers.

As with tweets, we then add the campaign's goal $G(c)$ and duration $D(c)$ to these features and feed them to an SVM, resulting in a predictor $\phi_{\text{graph}}(c, t)$.

3.3.3 Results

We train[12] the two SVMs described above using a Gaussian radial basis function (RBF) as kernel, thus having two parameters to tune:

- C: the soft margin penalty parameter,
- γ: the kernel coefficient for the RBF.

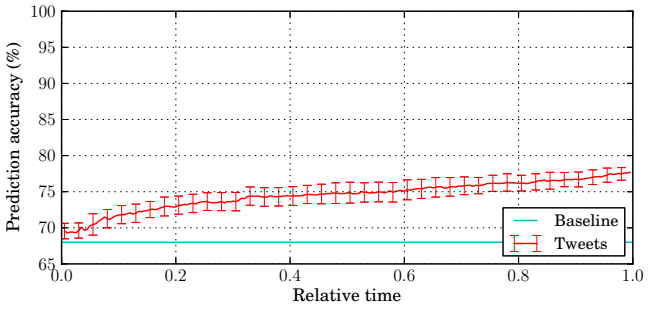
We perform an exhaustive search on a logarithmic scale for both parameters and evaluate the performance on the validation set to choose the best values. The best parameters for the *tweets* predictor are $C = 1000$ and $\gamma = 0.1$, whereas the best values for the *graph* predictor are $C = 100$ and $\gamma = 0.01$.

Figure 3 shows the corresponding prediction accuracy over the test set for the *tweets* predictor (Figure 3a) and the *graph* predictor (Figure 3b), along with the static baseline presented in Section 3.2.3. Although the performances are clearly inferior to those of the predictors that use the series of pledges, both social predictors quickly outperform the baseline performance of 68% obtained by Greenberg et al. [3]. The *graph* predictor has a fast increase in accuracy after a few time steps, then it decreases slightly towards the end. This effect could be countered by choosing the optimal values for SVM parameters independently at each time step, instead of once globally as we do now.

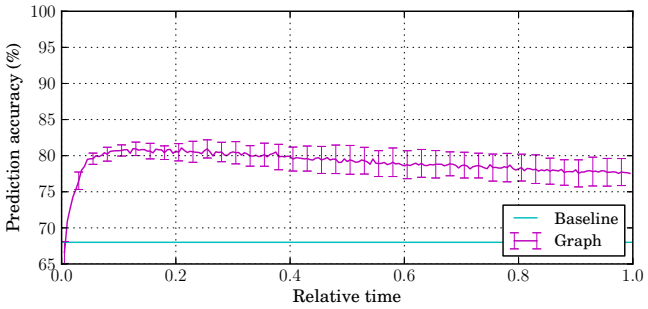
3.4 Combined Predictor

Predictors using the series of pledges show a good prediction accuracy, especially towards the end of the campaign. At the beginning, however, the accuracy could still be improved. Such improvement would be very useful to creators and backers, allowing them to react accordingly to correct the course of a campaign. A higher accuracy at later stages, however, would not be of high interest.

[12]We use a Python library [5] to train the SVMs.



(a) *tweets* predictor



(b) *graph* predictor

Figure 3: Prediction accuracy of the *tweets* and *graph* predictors, along with the static baseline of Greenberg et al. [3]. For each relative time $t \in [0, 1]$, a predictor was trained using features extracted from tweets or the projects/backers graph at time t, for all campaigns c in the training set. The value shown is the median accuracy over 10 runs and the error bars show the standard deviation over these runs.

To improve the accuracy of the predictors presented in Sections 3.2 and 3.3, we propose to train an SVM to take the individual predictions and combine them into a final prediction. The features used by the combiner are the campaign goal $G(c)$, its duration $D(c)$, along with the probabilities of success obtained using each of the four individual predictors.

3.4.1 Results

As with the social predictors described in Section 3.3.3, we use a RBF kernel for the SVM, thus having two parameters C and γ to tune. To do so, we run an exhaustive search on a logarithmic scale for both of them. The best parameter values we obtain are $C = 100$ and $\gamma = 0.1$. Figure 4a shows the corresponding prediction accuracy of the combiner, along with the static baseline presented in Section 3.2.3. Figure 4b highlights the early-stage performance of the combined predictor. Figure 4c shows the relative improvement of the combiner with respect to the best individual predictor, at each time step.

Overall, the improvement of the combiner is the strongest at the beginning of the campaign, increasing significantly the accuracy: the first combined prediction is 4% more accurate than any individual predictor. In other words, on average 4 hours after the launch of a campaign, the combined predictor can assess the campaign's probability of success with an accuracy higher than 76%.

4. CONCLUSION

In this paper, we introduce an exclusive dataset of Kickstarter campaigns. We study the prediction of their success, based on two kinds of features: the time-series of money pledged and social attributes. We show that predictors that use the series of money pledged a reach high prediction accuracy, with more than 85% of correct predictions after only 15% of the duration of a campaign. Although the social predictors reach a lower accuracy, we propose a way of combining them with time-series predictors. The combination results in a substantial increase in prediction accuracy in the very first moments of a campaign (4%), precisely when the ability to predict success has the most value. This can provide helpful directions to both project creators and backers.

There are many future research directions we would like to pursue. First, we should study the projects/backers graph, to explore its structure and main characteristics. We are especially interested in its dynamics: can we model the "diffusion" of success across this network? Another promising direction is the Twitter graph: how does the success of a campaign depend on the spread of messages on Twitter?

For now, our predictors only output a probability of success, but act as a black box: no reason for the probable success/failure is given. While this prediction itself can already be helpful to both campaign creators and backers, as discussed in the introduction, the next step would be to give them the specific characteristics of the campaign that could be improved.

5. REFERENCES

[1] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[2] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

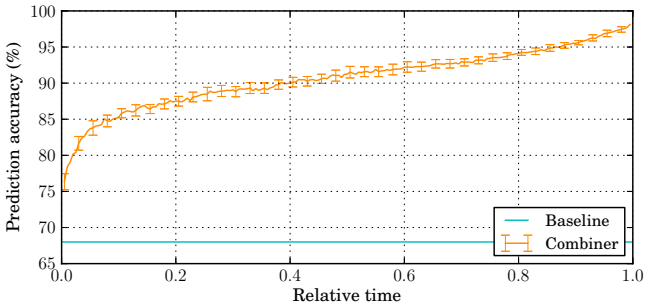
[3] Michael D Greenberg, Bryan Pardo, Karthic Hariharan, and Elizabeth Gerber. Crowdfunding support tools: predicting success & failure. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 1815–1820. ACM, 2013.

[4] Ethan Mollick. The dynamics of crowdfunding: An exploratory study. *Journal of Business Venturing*, 2013.

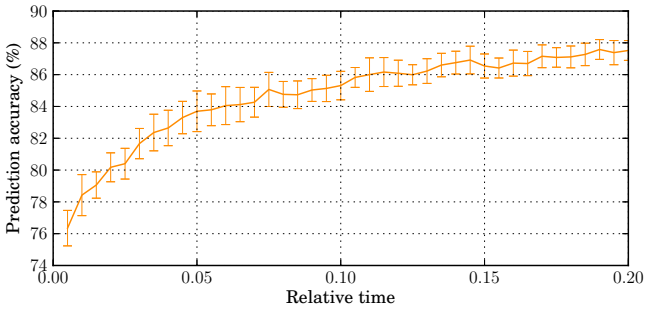
[5] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Twitter Developers. Frequently asked questions. Retreived August 24, 2013 from `https://dev.twitter.com/docs/faq#6861`.

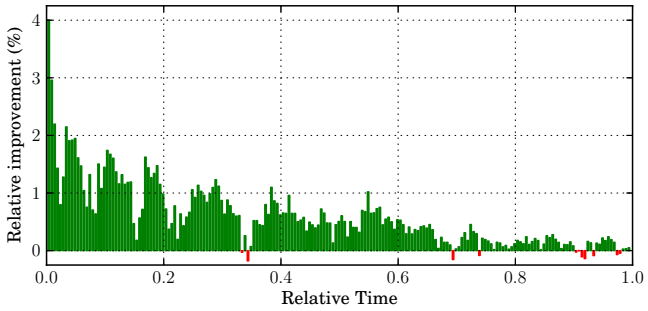
[7] Rick Wash. The value of completing crowdfunding projects. In *ICWSM'13: 7th International AAAI Conference on Weblogs and Social Media*, 2013.



(a) Combiner performance



(b) Detail of early-stage combiner performance



(c) Relative improvement over best predictor

Figure 4: Prediction accuracy of the combined predictor along with the static baseline of Greenberg et al. [3] (a), detail of the first 20% the campaign (b) and improvement of the combiner relative to the best individual predictor (c). For each relative time $t \in [0, 1]$, a combined predictor was trained using the the four predictions of individual predictors at time t, for all campaigns c in the training set. The value shown is the median accuracy over 10 runs and the error bars show the standard deviation over these runs.