Predicting Projects statuses proposed on Kickstarter platform

Vinícius Albuquerque

August 3, 2018

Abstract

To have your project turned into reality is what many people want. To do that, many people use crowdfunding method to raise the necessary money. This work proposes a model that could predict if the asked items at the Kickstarter system (a crowdfunding platform) would be successful or not. Since it is a classification problem, Machine Learning algorithms that solve this kind of problem were used, such as a Decision Tree. The results achieved in this paper show an accuracy of about 88%.

1 Introduction

Having projects (services, products) inserted on other people's lives is a great dream for many people. But, in order to have ideas turning into real projects, money is a major factor and not everybody has enough to make it turn to reality. There are a few ways of having the needed funding, some of them are:

- Equity Funding: When some company (or someone) gives part of the ownership of its company (or service/product) in exchange of the necessary capital;
- Debt Funding: It is essentially getting money from some place (or someone) that it will have to be paid back and, almost always, with interest;
- Crowdfunding: Source of financing that focuses on the "crowd" instead of specialized investor [BLS14]. Since it is the focus of this work, I will make a more detailed topic for that.

1.1 Crowdfunding

As Mollick says in [Mol14]:

"Crowdfunding allows founders of for-profit, artistic, and cultural ventures to fund their efforts by drawing on relatively small contributions from a relatively large number of individuals using the internet, without standard financial intermediaries.".

That means that people found another way of raising the necessary funds to execute their projects by having a bigger collective of individuals investing small amounts of capital to achieve a greater total. The internet has a great part on the success of this type of funding, some crowdfunding were created to facilitates the communication of the people who want to show their ideas with the people that can give some amount of money. These systems are usually donation-based, where people give money with the only purpose of support the idea to have that done, or rewards-based, where the owner of the project offers some type of advantage to the ones helping the project, for example, some offer a "first ride" others offer an premium account and that can vary in respect to what the product, or service, is.

1.2 Kickstarter

One of many existent Crowdfunding platforms, it has, according to [Cro], the second most amount of money raised, \$3B, number of supporters, 14M. Only losing to the GoFundMe platform which has a total of \$5B and 50M, respectively.

According to its website [Kica], by July 30th, it had more than 147 thousand funded projects and it had over 3500 still live projects. That means that more than 3500 projects are still waiting to achieve the total goal money to have their products viable.

Figure 1: Kickstarter Platform Logo.

1.3 Problem

According to [Kicc], in the Kickstarter platform, there is over $400M dollars that were donated but did not have the project reaching its goal. Also according to this source, the success rate of is only of 36,28%. In my opinion, that could be not only because of a "bad idea" but because of many other factors. Maybe the amount of money asked was too high, or maybe the period that the project was accepting donation was too short, or maybe another one.

By doing this work, I intend to help people to know how to ask other people for money so they have their projects done. By understanding what affects the successfulness of a pledge in the platform, we can try to adjust better the specific parameters, making more probable to a project to reach its goal of financing and, consequently, increasing the success rate of the platform.

1.4 Proposed Solution

This problem has different label classes that define if a pledge was successful or not, or if it was canceled. Thus, we can say that we have well defined label classes that we could use as our expected output. Therefore, we have a classification problem.

To solve this kind of problems, we can use some Machine Learning algorithms that work better with classification, such as a Decision Tree.

1.5 Metrics

To evaluate the model, I will use the F1-score to decide which of the trained models had the best results. With these results, a confusion matrix will be used so we can see have an even better notion if the model is working well or not. By doing the F1-score, we want to maximize the True Positives (TP) and the True Negatives (TN) and minimize the False Positives (FP) and False Negatives (FN).

The F1-score formula and its dependency calculations are described as follow:

$$F1score = 2 * ((Precision * Recall)/(Precision + Recall)) \tag{1}$$

$$Precision = \sum(TP)/(\sum(TP) + \sum(FP)) \tag{2}$$

$$Recall = \sum(TP)/(\sum(TP) + \sum(FN)) \tag{3}$$

2 Analysis

2.1 Data Exploration

To solve this problem, the chosen dataset will be one created by Mickaël Mouillé and can be found on [Kicb]. It is a dataset that includes two periods of time 2016 and 2018, both including information about projects that were using the Kickstarter platform as the crowdfunding system they needed to get the necessary funding.

Each of those datasets (2016 and 2018) have more than 300 thousand different projects with various features that we can use to extract information and, maybe, predict their final status. In this work, we will only work with the latest one, 2018, trying to make our model based on the most current projects.

2.1.1 Feature Description

In this section, the features from the 2018 Kickstarter projects are going to be described. The features are the following:

Feature	**Description**
ID	An unique integer which identifies the project
name	A string given by the owner of the project
category	A string indicating a more specific categorization for the project
main_category	A string indicating a more general categorization for the project
currency	A string indicating in which currency the project was proposed
deadline	A date indicating when is the final day to support the project
goal	A number indicating the amount needed to the project be executed
launched	A date indicating when the project was proposed
pledged	A number indicating the total collected amount in the project currency
state	A string indicating the final status of the project
backers	An integer indicating the total number of people that donated to the project
country	A string indicating the country which the project was launched
usd.pledged	A number indicating the total collected amount in dollars (converted by Kickstarter)
usd_pledged_real	A number indicating the total collected amount in dollars (converted by fixer.io api)
usd_goal_real	A number indicating the the goal amount in dollars (converted by fixer.io api)

2.1.2 Feature Selection

Now that we know all the available features, we can start to make some assumptions about what could be relevant or not. We can see that some of the Features give the same information only changing the universe where they are shown. For example, the features *'pledged', 'usd_pledged'* and *'usd_pledged_real'* give us the same information but in different unities. For that reason, we will only choose to work with only one of these 3 features. In that case, I am choosing *'usd_pledged_real'* because: first, the category *'pledged'* could represent values in different currencies and second, because according to the dataset creator, the Kickstarter conversion is less trustworthy than the one done by the *fixer.io* api. That login is also applied to *'goal'* and *'usd_goal_real'* where we will use the second one.

Another assumption made was that we did not need the *'ID'* when training the model since it is only an unique identifier that is given randomly. So, the *'ID'* feature was also excluded.

The feature *'name'* could be used if we had an text processing algorithm that could extract some additional information but since we are not doing that, this feature will also be excluded.

Another point that I would not want to influence the model is the feature *'backers'* and that's because I want to develop a model that would predict the final status of the project by the parameters defined before the project was launched. Thus, I would not want the project owner to calibrate its project predicting the number of *'backers'* their project has to have. Therefore, I removed this feature even though it is one that could really give us useful information.

If we also take a look on the features *'main_category'* and *'category'*, both give us information about a categorization, so I will choose to remove the more specific one so the grouping by it it's more difficult.

By removing those features, our model will only use the following features:

- main_category
- currency
- deadline
- launched

- state
- country
- usd_pledged_real
- usd_goal_real

2.1.3 Feature Exploration

Now that we defined the features that will be used in our model, it is time to define which one will be our label. The feature that gives us the information about the final status of the project is *'state'*. So we are going to use the values of this feature as our classification classes. For this dataset, we have the following values and total count of each for this column:

Value	Count
failed	197719
successful	133956
canceled	38779
undefined	3562
live	2799
suspended	1846

We are not interested in the projects that are still alive, so we are going to remove that. We are also going to exclude the projects with undefined state so we maintain our focus on the projects with status that are relevant to the problem we want to solve. At the end, we have the following statuses: *'failed'*, *'successful'*, *'canceled'* and *'suspended'*.

In Fig 2 we can see the correlations between the features. We can see that the only features with a real strong correlation is *'country'* and *'currency'*, that means that we could choose to remove one them since the behavior of one is explained by the other. Since we do not have a lot of features to work with anymore, I will not remove it.

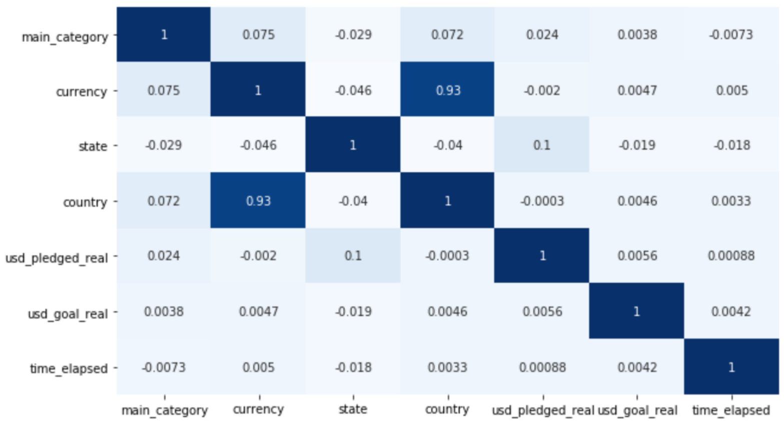

Figure 2: Feature Correlations

2.2 Algorithms and Techniques

Firstly, since we have features not defined as numbers and we cannot change directly the not number values to different numbers where each of one would represent a different value of the feature (that is because the absolute value of each defined integer could interfere in the model training since these values are mathematically greater or smaller than others), we will use a technique available on the pandas lib, the dummy values. What this does is: it gets every different value of a feature and creates a column for each one of them and assigns 1 to every datapoint that has this feature equals to this values. By doing that, we increase a lot the number of columns in our dataset.

To solve this problem - predicting if a project succeeds on collecting its goal or not - we are going to use some algorithms known as good solvers of classification. They are:

- Logistic Regression

  A mathematical modeling approach that can be used to describe the relationship of several independent variables (our features) to a dichotomous dependent variable (our class label) [KK10].

- Multilayer Perceptron (MLP)

  A system of interconnected neurons that represents a nonlinear mapping between an input vector and an output vector [GD98].

- KNN

  A simple nonparametric procedure for the assignment of a class label to the input pattern based on the class labels represented by the K-closest neighbors of the vector [KGG85].

- Decision Tree

  A decision tree is a directed tree with a root node where each path of the tree forms a decision rule that determines which class a new instance belongs to [DJ14].

Besides that, we will use a technique known as Grid Search which works in a way that we provide a lot of possible parameters for each different model and we compare the results for each combination of parameters used to train the model. The function that evaluates the quality of each result is the one shown in the Metrics section, the F1-score.

2.3 Benchmark

Before taking benchmarks into consideration, some personal goals were defined. When the initial analyze was made, I thought that a correct prediction of about 75% would be a good result because that would mean that the model would be correct guessing 3 out of 4 projects final state in the Kickstarter platform. We will see if that goal is achieved in next sections.

Now thinking about benchmarks, to compare the model proposed here, this work will compare with some models proposed in the Kaggle platform (source of where this dataset was acquired) in the Public Kernels section of the Kickstarter Projects dataset.

Comparing with the results in [ker], the model proposed here achieved better results with an accuracy of about 20% higher but the dataset *per se* is being more well visualized in this public Kernel and maybe more well represented which means that maybe the results are more reliable.

3 Methodology

In this section it will be explained the whole process of how the models proposed here were implemented and how we achieve the results that will be presented in the next section.

3.1 Data Preprocessing

To work with this dataset, I decided not to have the time of the year influencing our model but the period of time the project would be live to be supported on the crowdfunding platform. Thus, the features *'launched'* and *'deadline'* were reduced to one, *'time_elapsed'* that indicates what was previously explained.

To compare results, the final dataframe was also scaled and since it was using dummy values for some features and the number of features went from less than 10 to more than 50, PCA (Principal Component Analysis) was also used so we could try reducing the number of features to not only reduce the training time but also to remove maybe some of the information that did not accumulate a significant amount of information.

3.2 Implementation

To help implementing the code, this work made use of a popular lib used in the field of Machine Learn. This lib is called Sklearn. The Sklearn lib is done in a way where the classifiers available on there all have the *'fit'* and *'score'* function which trains and gives you the accuracy for the data you passed, respectively. So, in order to not repeat the code many times, the following code was implemented:

```
# Code implementing the function which will train and return
# us the accuracy of the passed classifier.

def clf_score(classifier, X_train, y_train, X_test, y_test):
    classifier.fit(X_train, y_train)
    return classifier.score(X_test, y_test)
```

After that, we only need to choose the classifiers and pass it to the function and the accuracy will be calculated. So, to test the models in the default configurations we only have to do the following:

```
from sklearn.model_selection import train_test_split

label_class = dataframe['state']
dataframe.drop(['state'], axis=1, inplace=True)

X_train, X_test, y_train, y_test =
        train_test_split(dataframe, label_class,
    test_size=0.2, random_state=89, shuffle=True)


from sklearn.linear_model import LogisticRegression
logistic_reg_score =
        clf_score(LogisticRegression(), X_train, y_train,
                X_test, y_test)

from sklearn.neural_network import MLPClassifier
mlp_score = clf_score(MLPClassifier(alpha=1),
                        X_train, y_train, X_test, y_test)

from sklearn.neighbors import KNeighborsClassifier
knn_score = (KNeighborsClassifier(n_neighbors=5),
                        X_train, y_train, X_test, y_test)


from sklearn.tree import DecisionTreeClassifier
dec_tree_score = clf_score(DecisionTreeClassifier(),
                        X_train, y_train, X_test, y_test)
```

3.3 Refinement

The initial results achieved were around 85% for the most of this algorithms, a good result but to try improving even more, the GridSearch technique was used. For each classifier, various values of some parameters were selected and were used by the algorithm to evaluate the model. The following code was used:

```
# The 'performance_metric' function will be passes as the
# 'scoring_func' parameter to the 'grid_search' function.

from sklearn.metrics import f1_score
def performance_metric(y_true, y_predict):
    return f1_score(y_true, y_predict, average='micro')
```

```python
# Grid Search implementation code

from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer

def grid_search(df, target, clf, parameters, scoring_func):
    sc_fnc = make_scorer(scoring_func)
    gs = GridSearchCV(clf, parameters, scoring=sc_fnc)
    gs.fit(df, target)
    return gs.best_estimator_
```

When we used that final refinement, that was when we got to the results that are described in the next section. To try to explain better what it was done by the GridSearch algorithm, I will give an example of one of the possible ways of working with that using the K-Nearest-Neighbor (KNN) as the classifier. The KNN works in a way that it checks the K points that are nearest to the point that we are predicting and the majority of these points would indicate which class this predicting point would be. So, to test our classifier with the GridSearch, we can, for example, define different values for K, such as 5, 7 or 9. Now the model will be trained and evaluated for each value of K and the GridSearch will return me which one has given me the highest value for the defined scoring function (F1-Score, in our case).

4 Results

For the algorithms chosen, the results can be seen in the Table 1. We can see that the final results were very similar and about 88% which is better than what we first expected. It is important to say that all of these results were achieved with a non-scaled and without the PCA applied on it. The results obtained without these factors had better accuracy.

Model	**Accuracy**
LogisticRegression	88.68%
KNN	88.17%
MLP	88.04%
DecisionTree	88.93%

Table 1: Results

Not by much, but the best algorithm was the DecisionTree. We can see the Confusion Matrix in Fig 3. By taking a look on this Confusion Matrix we can see that it works very well on predicting correctly what would happen with the project. And why I say that? If we take a look in the possible statuses of our dataset, we can see that have only have two final consequences: either achieve its goal or not. Thinking that way we can see that even though the model seems to get a considerable amount of wrong labels, we can see that the majority of the wrongly mislabeled was for a class with same consequence.

What I mean by that is: the model achieved 88% which is more than previously expected and the majority of its errors would not be so radical. When we come back to the use of what we want that model to have, if a person uses this model and the model predict it wrongly (fell into the 12%) it still have a great chance that the final consequence would be the same (failed or canceled), that is not putting your project with those requests and parameters.

4.1 Final Considerations and Future Works

In my opinion, for a not so complex dataset, the results were satisfactory. Thus, a system that could predict if a project would collect the final goal capital could be built using these kind of models. The data could be worked a little better and another concepts could be applied better. For example, applying the normalization correctly could have been good point of improvement in this work. Another point to take into consideration is the use of cross-validation so we have a better confidence that our model is not overfitting. Besides that, the removal of features with a

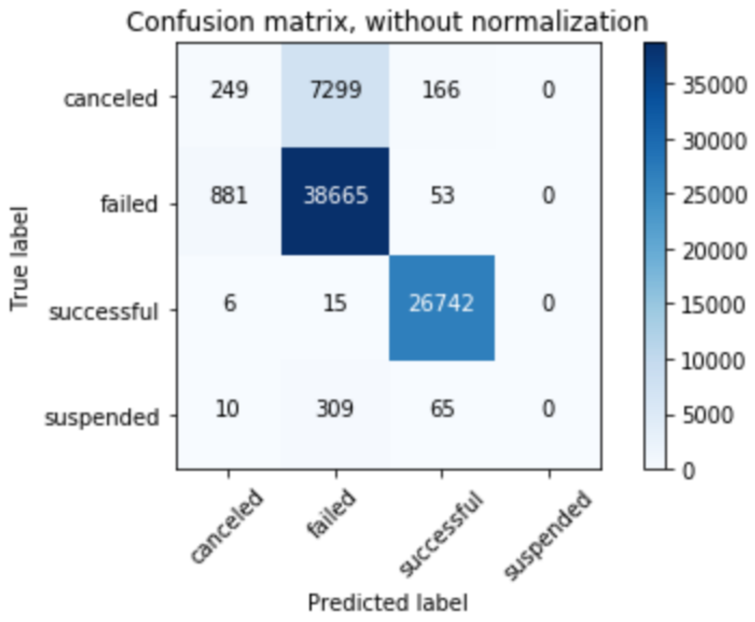

Figure 3: DecisionTree Confusion Matrix

strong correlation is another point of observation that could be better taken cared of in a way that we can improve not only the results of our models but also the training time of them.

Those were a few points of improvements that for sure deserved a better attention to it and I pretend to make a better job next time.

References

[BLS14] Paul Belleflamme, Thomas Lambert, and Armin Schwienbacher. Crowdfunding: Tapping the right crowd. *Journal of business venturing*, 29(5):585–609, 2014.

[Cro] Kickstarter kickstarter website. https://www.kickstarter.com/. Accessed: 2018-07-30.

[DJ14] Wei Dai and Wei Ji. A mapreduce implementation of c4. 5 decision tree algorithm. *International journal of database theory and application*, 7(1):49–60, 2014.

[GD98] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14):2627–2636, 1998.

[ker] KickstarterBM public kernel kickstarter dataset. https://www.kaggle.com/kosovanolexandr/kickstarter-lgbmclassifier-0-681. Accessed: 2018-08-02.

[KGG85] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4):580–585, 1985.

[Kica] Crowdfunding crowdfunding website. https://www.crowdfunding.com/. Accessed: 2018-07-30.

[Kicb] Kickstarter Dataset kickstarter dataset. https://www.kaggle.com/kemical/kickstarter-projects. Accessed: 2018-07-30.

[Kicc] Kickstarter Stats kickstarter stats website. https://www.kickstarter.com/help/stats. Accessed: 2018-07-30.

[KK10] David G Kleinbaum and Mitchel Klein. *Logistic regression: a self-learning text*. Springer Science & Business Media, 2010.

[Mol14] Ethan Mollick. The dynamics of crowdfunding: An exploratory study. *Journal of business venturing*, 29(1):1–16, 2014.