Stakeholder Report

Authors: Katharina Granberg, Sergiu Ropota and Johannes Hoseth

The present report is written to explain the project focus, methods and results of the M3 group assignment. All code that has laid the ground for this report can be found in the attached notebooks. Four notebooks are attached, with this recommended reading order: (1) the Data Cleaning and Exploration notebook, (2) the Baseline Categorical, (3) the Baseline NLP, and (4) the cumulative Network Analysis Notebook.

Problem statement

The present project group is a pretty interesting interdisciplinary mix of fields. With one from mathematics, one from design psychology and one from innovation management, we sought to find a problem that was of inherent interest to all of us. While a given data set and an underlying problem to solve are closely related, they are not mutually exclusive either. For a problem topic to be meaningful it needs solve an actual problem, within a specific context, derived by the data set at hand. Because of this, our approach for finding a meaningful problem was to discuss problems, not solutions or tasks, related to the data sets that we found during our exploration.

After searching for a while, one dataset seemed to equally match the interest for all group members. It was a Kickstarter data set, scraped by Webrobots - an EU backed project, providing publically available scraped data sets of dynamic javascript heavy websites.
The kickstarter data set peaked out interest for several reasons. The crowdsourcing platform, while at first glance appearing to be simple, consists of many stakeholders all invested in whether a project is backed or not. The future customers, or *backers*, are interested in the product made by the creators. For a person to potentially back a product, several things are clouded with uncertainty, i.e. '*Will this product be as good as they describe it to be?', 'What if I pay and the creators cannot keep up with the demand?',* or '*If the project is not successfully backed, am I just wasting my time emotionally investing myself in anticipating its completion?'*.
While backers are never at the risk of investing in a product that will not ship, many more subtle uncertainties exist and influence the user experience of Kickstarter. This is relevant, from a design psychology perspective, seeing as much of the experience is of anticipatory nature - something could very well be influenced by the display of information during times of uncertainties. From a mathematical standpoint, the dataset was nice since it allowed us to combine three different data types, posing an interesting coding challenge.
Besides the backers desiring a sense of security in their investment and creators needing to reach their backing goals in order to profit from the campaign, two other notable stakeholders exist. First, Kickstarter itself has an inherent incentive in as many campaigns becoming backed as possible, thereby earning a cut off the profits and increasing the popularity of the site for both consumers and creators. Second, quite paradoxically, Kickstarter has become crucial for investors. Investors and VCs actively evaluate the feasibility of potentially investing in a startup, by the success it is having on Kickstarter. Often, the Kickstarter funding is only a small part of a larger investment strategy planned by startups to draw in investors to fully cover their costs. In

terms of innovation management, the backing of campaigns thereby become an interesting and complex problem.
For these reasons, a prediction of whether a future campaign will successfully reaching its goal will actually have inherent value for both customers, Kickstarter, the creators and the investors - alongside academic value to us, the authors

Introduction

Using data found at: https://webrobots.io/kickstarter-datasets/ (the 2019-11-14 CSV datafile) we wish to predict if a kickstarter campaign will be successful or not. The data set has numeric and categorical data such as ‘category’, ‘goal’ and ‘staff_pick’, alongside textual data such as ‘name’ and ‘description’. Additionally the ‘photo’ column contains URLs for all the various photos used in each Kickstarter campaign.
The underlying desire is to create models for each of the different data types to evaluate their accuracies. Additionally, we are interested in attempting a *multi-modal* model, meaning the combination of the aforementioned models into one, utilizing all data types.

Initial Data Exploration and Data Cleaning

The first step was unpacking the data, and then merging the different datasets found in the zip file into one big dataset. For reference, this initial dataset consisted of 201500 Kickstarter projects, described by 38 columns.
From the very first inspection of the dataset, we knew that quite a few columns could be dropped. Various columns contained almost exclusively NaNs, which we deleted. Additionally, we wanted to make predictions on the success of Kickstarter projects, with no information that is unobtainable when launching a campaign. This implied that all columns pertaining information about the final amount of money that has been pledged were deleted.
Left were merely 15 columns aside of the target, used as features for the various models. The 15 features are mentioned in the notebook. We additionally saw that much several columns pertained currency/foreign exchange rates relevant to the numerical goal set by Kickstarter creators. We opted to simply convert all goals in the USD, using the foreign exchange rate. Additionally we One-Hot encoder categorical columns such as ‘country’ and ‘category’, after extracting the main categories using RegEx.

For further exploration, a prioritized first effort was to perform a simple value_counts() for the target column, ‘state’. This is important, seeing as working with binary predictions of an **imbalanced** dataset might cause the model to predict very accurately, because it guesses the most frequent state, regardless. In such a case other evaluation metrics would be more fruitful, such as 'precision'. Luckily, our dataset is fairly balanced and large enough that it should not cause an inconvenience. Aside from these two ‘successful’ and ‘failed’ states that we wanted to

predict, some campaigns were also ‘canceled’, ‘live’ and ‘suspended’. Since we had 180.000+ observations just in the two first states, we decided to drop canceled, live and suspended campaigns.

Second, we wanted to explore the distribution of ‘goal’ sizes according to the ‘failed’ and ‘successful’ campaigns, in order to see whether the might any patterns. We opted for the *relative* frequency distribution (as opposed to an *absolute*), in order to truly be able to compare the successes and failures that do not have equal value counts.

Figure 1 below, shows the distributions:

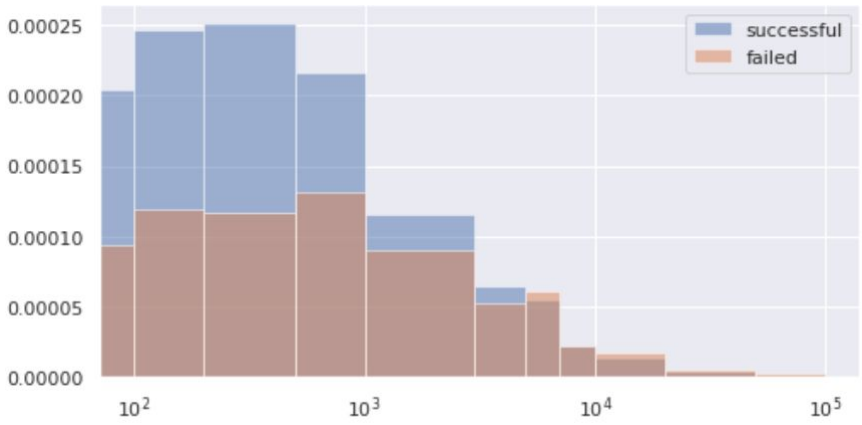

Figure 1 - A histogram plotting of the ‘goal’ sizes, split by ‘successful’ and ‘failed’.

As can be seen, a lot more successful campaigns in the first half of the graph, whereas for the bins indicating the largest goals, there are more failed campaigns.

Lastly, we wanted to inspect our dataset for any abnormal correlates to the target, using a Seaborn heatmap. This is seen below, in Figure 2.

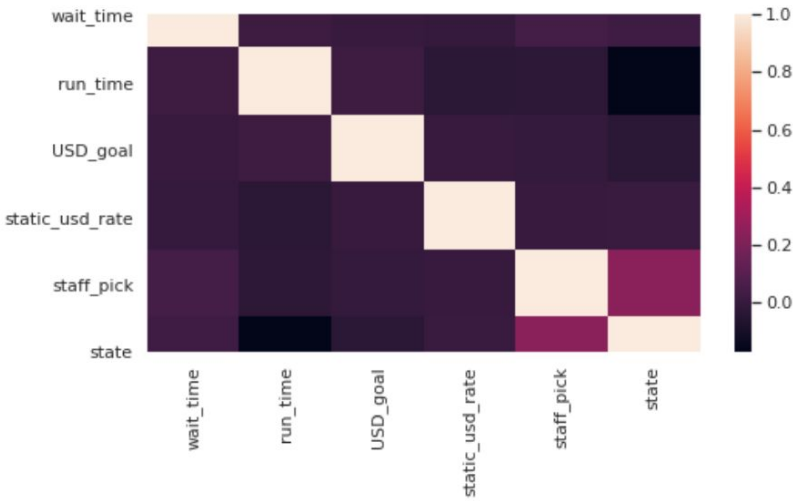

Figure 2 - A heatmap showing correlates between features and target.

From the heatmap no abnormal correlates appear, though the Kickstarter ‘staff_pick’ stamp of approval seems to strongly correlate with the success of a campaign. The heatmap had proven very valuable, as running an initial non-neural model yielded a strange 100% accuracy for both training and test, regardless of what (hyper)parameters being used. Upon revising the dataset for abnormalities via a heatmap we found out that the ‘spotlight’ boolean had a perfect correlate to the target. Apparently ‘spotlight’ is a stamp of approval given to campaigns, *after* they have reached their goals.

As a last thing, we checked for and removed any campaigns with missing values seeing as there were only a few, alongside identifying and duplicate campaigns. The cleaned data set was as such ready to build models from.

The Baseline Models

Categorical Data
We made two baseline models based on just the categorical data. The first one we made was just a logistic regression model with random parameters, and for the second one we used a randomized search to find good parameters for the logistic regression.
The model with the highest accuracy was the second model with an accuracy of 71.7%.

A possible way to get better results using a non-neural model on the categorical data could be to examine other models, for example a tree based model like a decision tree, or random forest, or a Gradient Boosting algorithm like XGBoost or CatBoost.

Doing a more extensive grid search to get the very best parameters could also be a way to squeeze the accuracy even higher.

Textual Data
We choose to also do a baseline model just using the descriptions. Here we used different techniques from M2.
We made three models:

1. TF-IDF and logistic regression
2. TF-IDF, TruncatedSVD and XGBoost
3. Tokens prepared with spacy, Word2Vec embedding and logistic regression

The model with the highest accuracy was the first model with an accuracy of 69.04%.

All these models also include a supervised machine learning algorithm to get the prediction, so all the extensive efforts mentioned above for the categorical data could also have been used here to get better results.
In the TfidfVectorizer there are some additional parameters like min_df and max_df, and in the Word2Vec there are parameters like min_count and window, these could be tuned separately to get the best input for the ML algorithms. We could also have used more of the NLP techniques from M2 like tried stemming instead of lemmatization, or made a list of custom stopwords to disregard.

The three distinct neural network models

In our Neural network model we have chosen to first use each type of data on its own as input for appropriate model architectures, before later combining them into one model.

As a first effort, we crop our dataset to contain only the first 150,000 campaigns, since those are the only pictures that we have scraped. Next, we then opted for splitting the data into train, test, and validate for both the categorical, textual and image-based models. The validation set is a split of the training data, providing the opportunity to train using ‘train’ and ‘val’ when fitting instead of ‘test, which is then kept for evaluating the model at the very end.

Categorical
The categorical model was designed as a simple Feed Forward network, consisting of Dense layers with Dropout layers and BatchNormalization. We opted to manually inspect its results by iteratively adding and subtracting layers relevant for this model architecture, achieving an accuracy of 66%. Subsequently we opted to perform a RandomizedSearch with cross-validation, in order to find the optimal parameters. Parameters included two different activation results (‘relu’ and ‘tanh’), various dropout rates and possible numbers of neurons for each hidden layer.

With the optimal selection of parameters, for our chosen model architecture, we achieved almost the same accuracy as without it.

Textual

Prior to building the model, we opted for the convenient keras Tokenizer, making our data workable. After preprocessing the NLP data, we downloaded the pre-trained word embeddings GloVe and used it to get the weights for our embedding layer. We then defined a recurrent neural network comprised of the embedding layer (which was trained on GloVe), a Bidirectional GRU layer, Batch Normalization, several Dense layers and two Dropout layers. GRU (Gated recurrent unit) is a recurrent layer which we use to be able to say something about the sequentiality of the words, much similar to LSTM. After trying both, the GRU yielded the best results. Then we fine-tuned the NLP model through RandomizedSearchCV to obtain the optimal amount of neurons, dropout rate and activations.

Image

For the image neural network we have defined two Conv2d layers with Maxpool layers, several Dense layers, and Dropout. Again, we wanted to do a RandomizedSearchCV to determine the optimal parameters. BUT we couldn't get the image data to work, and colab constantly crashed because of too little RAM, despite our efforts to reduce the computational requirements by substantially reducing our dataset size.
*Note: We hope to get this running and include it in our combined model to show at the exam.

Final model

So in the end we wanted to join all three models into one big model, but we had to make do with just the categorical and the textual, since the image didn't work. So what we did was we took the best version of each of the two models, removed the output layer, concatenated them and then added a dense layer with 2 neurons, one for each model, and then just one dense layer with a final Sigmoid activation, as the output, since we are trying to predict a classification problem.

Then had fit the final model on the training data, using the validation data for validation, and concluded the project by evaluating the model on the test data, getting a final accuracy of 67.8%. So it is slightly better than the two neural networks separately, but worse than our two baseline models.