IT University, Copenhagen

Frontpage Exam Hand In for Data Science in Games, Spring 2019

A. M. Belloso, 16479, albm@itu.dk

A. Kähler, 16757, akae@itu.dk

R. E. Odgaard, 16738, raod@itu.dk

Course number: KGDASCG1KU

GitHub repository: https://github.com/alexkahler/kickstarter-success

Analyzing Kickstarter Success with ANN (May 2019)

A. M. Belloso, A.Kähler, and R. E. Odgaard

***Abstract*—KickStarter is an ever-growing platform, with almost 500,000 projects launched as of April 2019 and 3.6 billion USD in funding. For investors, it can be difficult to correctly predict whether a project will be a success or failure, thus leading to our motivation for designing a model which can help in this aspect. To do this we utilized data scraped from KickStarter by WebRobots, where after preprocessing and exploratory analysis was conducted on the data. Several features were found to have the possibility to be used in a predictive model. Hereafter, two ANN was designed and tested on the data. The results were promising, with an accuracy of 72.06% (-0.5425 Loss) for the "alternative" ANN and 71.65% accuracy (-0.5520 Loss) for the "vanilla" ANN, however, the high Log Loss numbers reported showed that our model did not fit our data well.**

***Index Terms*—ANN, Artificial Neural Network, Data Science, Deep Learning, Kickstarter, Machine Learning, Supervised Learning**

I. Introduction

The basis of this project is to take a look at Kickstarter [1] projects and through the use of an artificial neural network (ANN) train multiple models for recognizing whether or not a Kickstarter project will succeed or fail.

As a baseline project, an article by L. Lewis [2] was used where the author by the use of the technologies: logistic regression, random forest, and XGBoost, analyzes the same dataset as is processed in this project. Through her efforts, L. Lewis reaches an accuracy of appx. 70%. It is the goal of this project to do a parallel analysis but with the use of deep learning, to reach similar or better results than in the aforementioned article. Furthermore, models will be trained on specific categories of Kickstarter campaigns e.g. games, technology, food, and journalism. This is done on the assumption that a general model for all kinds of Kickstarter projects might not be possible.

In regards to the algorithms, different architectures of deep neural networks will be tested and the results of each will be reported. Validation of the results will also be reported, along with an analysis of important features based on excluding and including them in the models.

II. Data

The dataset used [3] was chosen because of the large number of data points and features, as well as the possibility of comparing the results from this experiment with the one performed by L. Lewis [2]. It contains data extracted from Kickstarter projects since 2014 by the use of scraper robots and formatted as a CSV-file. However, some features like category were formatted as JSON objects inside the CSV.

In order to match the baseline [2], the data used for the algorithms were processed in mostly the same fashion as in said baseline. The data started out having 43 features including the state feature which were to be the correct label for the training of the binary classifier. However, as many of these categories including currency_symbol and source_url were of no use, they were dropped from the dataset. Furthermore, 5 features included incomplete or no data for all rows and were as such disposed of as well.

In the interest of practical uses to the experiment, data only available after the Kickstarter campaign had ended, such as number_of_backers were also removed. Having these included in the data would not give any information to potential Kickstarter users setting up a campaign.

The dataset had quite a few categorical features e.g. category, country, launch_day. These were all encoded to one hot matrix for better performance and more expressive results.

After all conversion and preprocessing all rows containing null values were dropped and the final dataset contained 15 features (83 columns because of one hot encoding using get_dummies() function from the pandas library) and 148,299 data points or rows.

All preprocessing was done in python by the use of mainly pandas [10].

III. Exploratory Analysis

For initial insights and to discover the main characteristics of our dataset, we conducted an exploratory analysis. This enabled us to uncover useful features which could be used in our predictive modeling. The data set used for exploratory analysis was the preprocessed data, where non-relevant projects were filtered out.

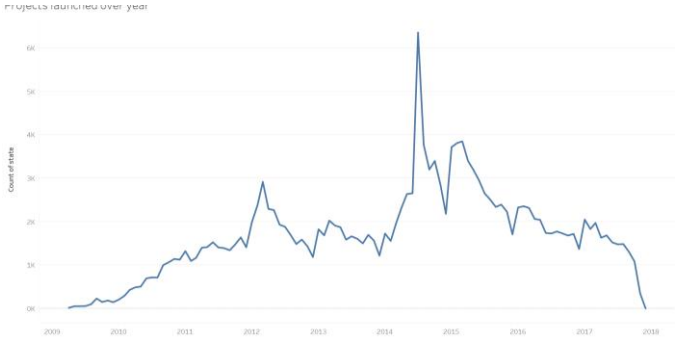

Figure 1. Amount of projects launched on Kickstarter.

Kickstarter projects have grown since its launch in 2009, as shown in figure 1. Particularly during its expansion during 2014. When examining the overall success and failures

of all projects, there is a surprisingly larger number of successful projects compared to failures, with a rough ratio of 1:1.75.

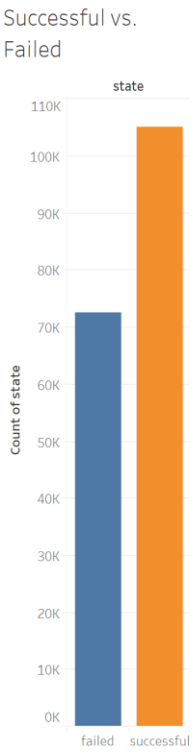

Figure 2. Successful vs. Failed projects

When examining this over the lifetime of Kickstarter, we noticed that the massive expansion was detrimental to the success ratio of Kickstarter projects – 2015 and 2016, after the expansion was the only years where there were more failed projects compared to successful.

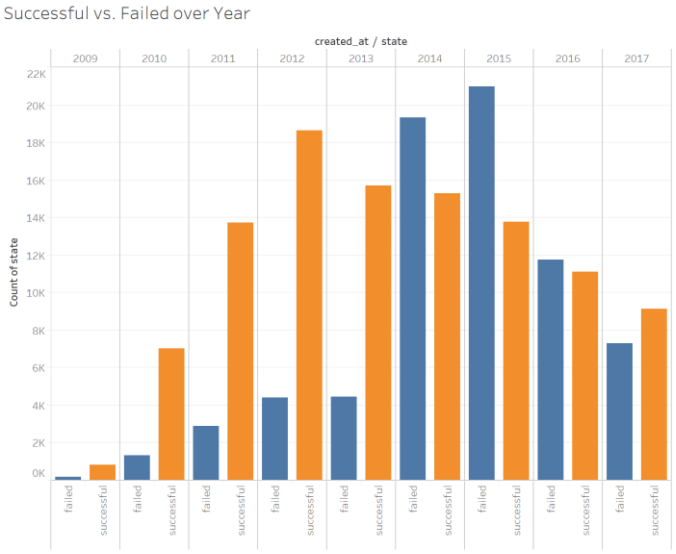

Figure 3. Successful vs. Failed projects over Year

For the variety of other features, we noticed in general that successful projects generally had more backers, as shown in figures 4, and unsurprisingly projects with a high goal had a higher chance of failure, while projects with a lower and realistic goal had a higher chance of success. The median amount sought by the successful projects were almost half that of the failed projects (figure 5).

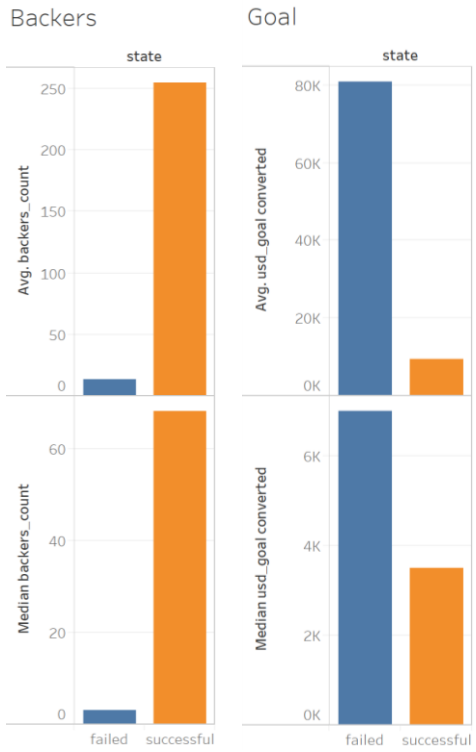

Figure 4 (left). Average and median amount of backers

Figure 5 (right). Average and median project goal

Moreover, successful projects typically also had a higher average and median pledged amount as shown in figure 6. The surprising thing here is that the median pledged amount for successful projects is higher than the median goal for successful projects, suggesting that successful projects had a tendency to be overfunded. As the discrepancy between successful and failed projects is also much larger in terms of pledged amount and the number of backers, compared to the goal amount, it also suggests that once a project indicates signs of success, then more backers will flock heightening its chances of success.

Finally, a few projects on Kickstarter are either spotlighted or hand-picked by staff, thus it is not unreasonable to suggest that projects which were either spotlighted or staff picked also had a higher chance of being successful (figure 7).

We also discovered that other features such as the length of the blurb, did not have much impact on the success of a project, and it would be much more beneficial to do sentiment analysis on the blurbs, for more insight. The duration of a project also had a small impact on success. Figures 8, 9, and 10, indicate that a project with a slightly shorter campaign length, but take slightly longer to launch from creation, have a slightly higher chance of success.

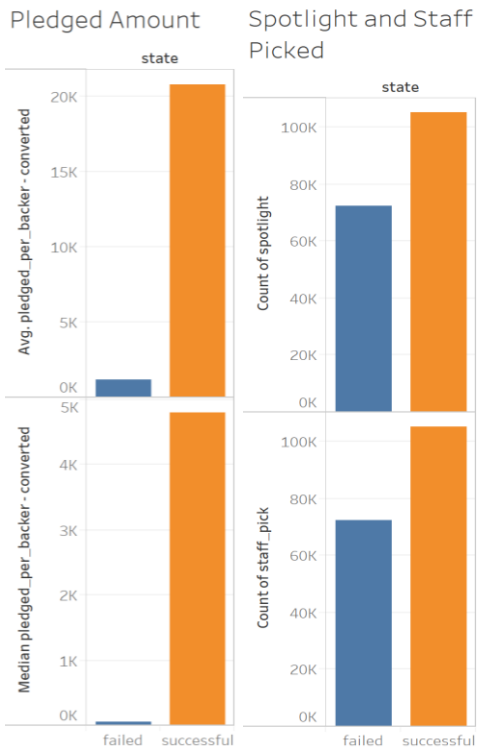

Figure 6 (left). Average and Median pledged amount
Figure 7 (right). Amount of Staff picked and Spotlighted projects

When creating a project on Kickstarter, it needs a category. The most popular kickstart category to launch in was "fiction", however, the highest amount of backers were "product design" and "wearables". This is also shown in the amount pledged, especially for "wearables", which exceeded 10 million, while "product design" was just under half that.
When examining success ratio for the categories, the best category to launch in was "comics" and "games", due to their high amount of backers and low goals.

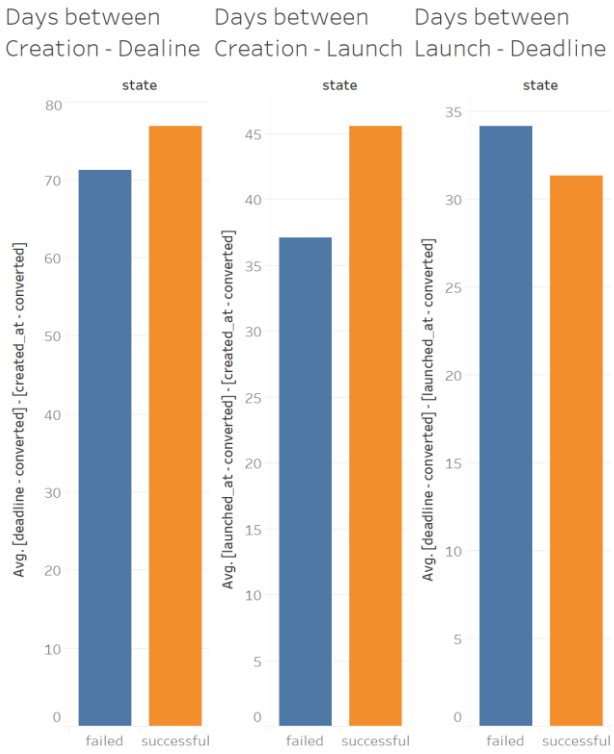

Figure 8 (right). Days between project creation and deadline.
Figure 9 (mid). Days between project creation and launch.
Figure 10 (right). Days between project launch and deadline.

IV. ALGORITHMS

The algorithm used in this project was an Artificial Neural Network (ANN), and this following section will briefly introduce the key concepts behind it.

An ANN, inspired by biological neural networks from nature, is a system which learns to perform tasks such as image recognition or labeling, without generally being programmed with any task-specific rule. In general, ANN solves tasks with classification, clustering, and predictive analysis (regression) and is used within fields of machine learning and data mining.[12]

An ANN typically constitutes of a series of nodes, also known as artificial neurons, that are interconnected. Each node connection functions like a synapse, which can transmit information from one neuron to another. Each neuron then processes the information and then forwards the processed information to the next neuron.

In typical ANN implementations, each neuron has a weight attached, which adjusts as the ANN is trained, where the function of the weight is to increase or decrease the strength of the signal between neurons. The sum of all neurons connections is then collected in the following neuron and then transformed with an activation function. This activation function has properties that benefit different kinds of output and common examples include the sigmoid function, step function and recently the Rectified Linear Unit (ReLU). Once the signal is transformed, it is sent to the next neurons, where the process is repeated, until the signal reaches its output layer.[12] The signal between the neurons is normally real numbers.

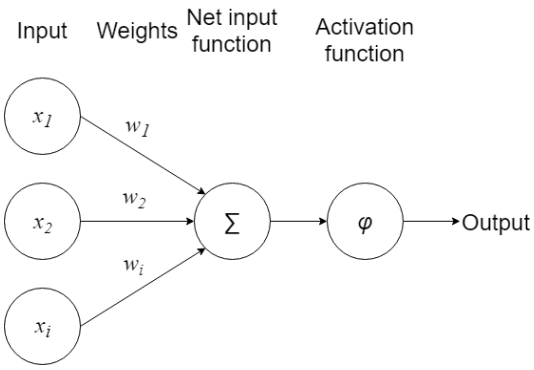

Figure 11. Example of a neuron connection.

A. Perceptron

Multilayer Perceptron (MLP) is a form of ANN which has an input layer, one or more hidden layers, and an output layer. MLP is one of the most commonly used ANN for classification, clustering and predictive analysis. MLP utilizes backpropagation as its supervised learning technique [12].

B. Backpropagation

For training the algorithm, backpropagation, a generalization of the least mean squares algorithm, is used to calculate the weights in the MLP following a gradient descent approach.
The advantage of backpropagation is that it is iterative and recursive and an efficient method for calculating the weights of each node in the MLP when the ANN is being trained. In this context, backpropagation is typically used with gradient descent optimization, in order to calculate the weights of each neuron in conjunction with the Log Loss Error.[13]

The gradient descent derivation for a single-layered network is calculated as follows.
If we assume that the squared error for one output neuron is;

$$E = L(t, v)$$

Where E is the loss for the output neuron, t is the target value and v is the actual output. The output o, for each neuron, is;

$$o_j = \varphi \sum_{k-1}^{n} w_{kj} o_k$$

Where, j is the j^{th} neuron in layer k, and w is the weight and φ is the activation function. The derivative function for the logistic function can then be calculated as;

$$\frac{d\varphi(z)}{dz} = \varphi(z)(1 - \varphi(z))$$

The gradient descent is used to update the weight w_{ij}, where E is decreased. This is reflected the Delta-rule;

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial o_j} = \eta o_i \delta_j$$

Where the learning rate is > 0.

The updated weight is then found as;

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$$

C. K-Folds Cross-Validation

In order to evaluate our algorithm, we have chosen to use a method called k-Fold Cross-Validation. With this method, we will divide the dataset into k-bins where each bin is used as a testing set at some point. Thus, k-Fold CV can be understood as an iterative process which repeats, until all bins have been used as a test set.

The problem with common hold-out methods, where the dataset is divided into a training set, validation set, and test set, is that due to sample variability between the training and test set, our model would give a better prediction on training data, but fail to generalize on test data. This, in turn, would result in a higher training accuracy but with low test accuracy.[15]

Moreover, as we split the dataset into training, validation and test set, we only use a subset of the data, thus resulting in a tendency to underestimate the test error rate for the model to fit the entire dataset.
These problems can be solved with k-Fold Cross Validation, as the repeatable nature of the method, reduces variability and bias.[15]

Figure 12 shows a graphical representation of the process of k-folds cross-validation with K = 5 number of folds, as is used in this project.[15]

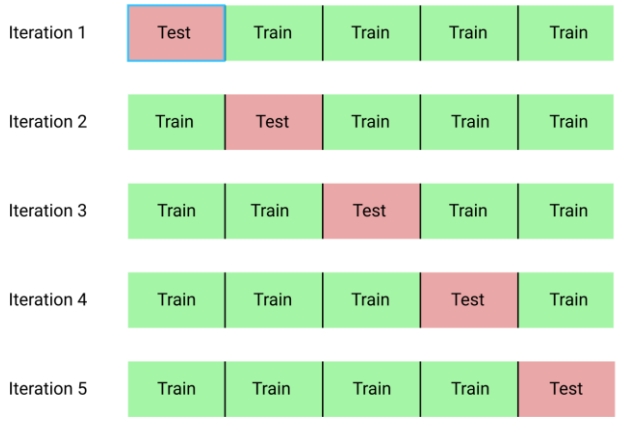

Figure 12. Graphical representation of the fold process [8]

V. Method

The method we used was supervised learning, given that we already know the features [7] of each project and whether it was successful or not, which is what we tried to predict in this project. For our supervised learning we used deep learning, different ANN and different subsections of the data.

We used Python and the framework, Keras [4], with Tensorflow [9] backend to build our different models, train and evaluate them. Furthermore, Scikit-learn [11] was used for facilitating training.

Different ANN models with a different number of layers were evaluated, the number of neurons of those layers modified, dropouts added and removed in order to avoid overfitting and we also tried different types of activations.

Lastly, we tried different optimizers. For the loss function, we used binary cross entropy or log loss/negative log loss as it is suited for binary classification and for the last layer we used a sigmoid activation which is also the standard for binary classification. The initial weights to start the training we used were normally distributed. We have also trained our models with different batch sizes and number of epochs. All these parameters were deduced by trial and error.

Finally, we ended up with three different models, one which we called simple or vanilla (figure 13), one we called alternative and the last called category. The first one, as its name indicates, is a simple model composed by 4 fully-connected layers with 32 neurons each except for the last one, which is of course only one neuron in charge of the binary classification. We used a 40% dropout layer after the two first fully-connected layers. Regarding the activation function, we used ReLU for the first 3 layers and sigmoid for the last one. Finally, the optimizer used was adam, which is a very effective optimizer that achieves good results faster than others.

```
Dense(32, kernel_initializer='normal', activation='relu')
Dense(32, kernel_initializer='normal', activation='relu')
Dropout(0.4)
Dense(32, kernel_initializer='normal', activation='relu')
Dense(1, kernel_initializer='normal', activation='sigmoid')
loss='binary_crossentropy', optimizer='adam', metrics=['accuracy']
```

Figure 14. Architecture of the simple model

The alternative model is composed of the same number of layers, but with 512, 256, 128 and 1 neurons respectively. Apart from that, the difference is that for this one we used leaky ReLU as the activation function instead of regular ReLU, which is not as sensitive to the "dead cell" activation problem when a unit becomes inactive [5]. Finally, as there are more neurons in each layer and it is more likely to have overfitting while training, we added more dropout layers, concretely 50% dropouts after each of the 3 first layers.

```
Dense(512, kernel_initializer='normal')
LeakyReLU(alpha=0.3)
Dropout(0.4)
Dense(256, kernel_initializer='normal')
LeakyReLU(alpha=0.3)
Dropout(0.4)
Dense(128, kernel_initializer='normal')
Dropout(0.3)
LeakyReLU(alpha=0.3)
Dense(1, kernel_initializer='normal', activation='sigmoid')
loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 15. The alternative model for the general data

The last model which was configured was specifically for training on the data sorted by category e.g. games, technology, food. These data subsets are smaller and as such a different smaller model was build and tweaked for this purpose as shown below in figure 16. Here the optimizer was rmsprop.

```
Dense(64, kernel_initializer='normal', activation='relu')
Dropout(0.5)
Dense(64, kernel_initializer='normal', activation='relu')
Dropout(0.5)
Dense(1, kernel_initializer='normal', activation='sigmoid')
loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy']
```

Figure 16. The model for the category data subsets

As mentioned, we have tried to train our artificial neural networks with different parameters and performed some tuning on the parameters in order to try to find a combination that would lead to better results.

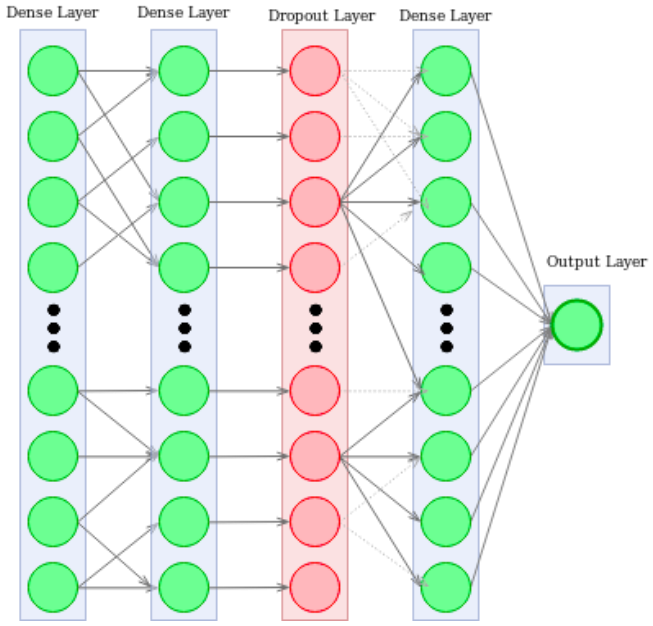

Figure 13: Vanilla ANN architecture

For performance reasons and with the intention of limiting the number of inputs to the algorithm a new dataset called "slim" was made, in which we removed all the time-related features like launch-day, deadline-month, etc. except for creation_to_launch_hours and the number of days the campaign ran and we also removed the countries features as there was plenty of categorical features. Apart from this having to move to another country to make a Kickstarter project seemed impractical and as such were somewhat counterintuitive to the motivation of the project.

For validation reason, it was decided to apply the k-folds cross-validation technique [6] for training the network. This was in order to estimate the accuracy across different subsets of the data and avoid groupings of similar data. This gives some performance overhead, but 5-folds cross-validation was considered applicable within the scope of the project.

VI. Results

Table 1 shows the accuracy and loss results that we obtained with our general models, both vanilla and alternative with the slim dataset and also the all included one. It also shows the different parameters that we have tried to modify in order to get better results which are the number of epochs the model was trained and the batch size. "K" represents the number of folds in our validation given that we used K-fold cross-validation to check how our models performed.

Dataset	Model	Epochs	K	batch_size	Acc	Loss
slim	vanilla	100	5	128	0.7228	-0.5430
slim	vanilla	100	5	256	0.7229	-0.5414
slim	alternative	50	5	256	0.7196	-0.5435
slim	alternative	250	5	256	0.7219	-0.5412
all incl	vanilla	100	5	256	0.7165	-0.5520
all incl	alternative	75	5	128	0.7206	-0.5425

Table 1: Results from general models

In regards to the models divided by category table 2 shows the results from the category model, as this provided the best results. All models were run with a smaller batch_size of 64, 100 epochs and with K = 5.

Category	Epochs	batch_size	Acc	Loss	Data points
games	100	64	0.6903	-0.5763	3457
technology	100	64	0.7140	-0.5461	8706
food	100	64	0.7197	-0.5339	9542

Table 2: Results from categories, with category model

The last set of results are a closer look at the technology category by using the alternate architecture. These models are based on the slim dataset and features are methodically excluded in order to see if any of them influences the accuracy and the log loss error, hence are important. All models were run with batch_size 64, K = 5 and 100 epochs.

Excluded feature	Acc	Loss
none	0.7140	-0.5461
usd_goal	0.6776	-0.5837
creation_to_launch_hours	0.6940	-0.5631
name_length	0.6891	-0.5727
campaign_days	0.6995	-0.5584

Table 3: Results from feature exclusion models with category technology

VII. Discussion

The results in Table 1 shows how the ANN handled the dataset with different architectures and versions of the dataset and with different hyperparameters. General for all versions of the ANN is actually both accuracy of around 71-72% and a negative log loss error of around -0.54. The accuracy in itself for binary classification problems can be misleading, without taking a look at the loss. For negative log loss error, a value of -0.693 is what corresponds to a 50% percent change of the answer being correct by the simple calculation ln(0.5).

This shows us that the models do indeed improve their performance from the training but the results are not a lot better than random. This could be mainly one of three things: either the architecture is simply not good enough, errors have been made in the calculations due to human error or the dataset simply does not have a structure that allows for predicting whether a Kickstarter is successful or not.

Regarding the robustness and design of the architecture, multiple versions and hyperparameters were tested for exactly this purpose. However, as the changes to the results were minimal, the optimal result for this entry level of machine learning must have been reached. Backing this up further was the use of k-folds cross-validation in order to assure that the random partings of the dataset do not influence the results. The number of epochs for the training were tweaked while training, as the training, was relatively fast and could be done multiple times with adjusted amounts of epochs.

In order to check for "dying cells" or permanently stuck units leaky ReLU activation was used, but it did not improve upon the results.

For the next set of results (table 2), the ANN was tested against a subsection of the dataset, that is category specific. By applying a lighter model (fewer layers) and the slim version of the dataset, as it provided better results, numbers close to those of the general model was achieved. However, by using the DummyClassifier (with stratified strategy, so ZeroR) of the sklearn library it was found that the category subsections of the dataset were very imbalanced. At times receiving dummy predictions of up to 70% dummy predictions. To avoid this the datasets were balanced, however, this reduced the already low number of data points, which most likely is responsible for the results being lower accuracy and higher loss than the general model. As such the results are not conclusive and could benefit from a larger sample of data.

Table 3 shows a closer inspection of the technology category. Here the importance of certain features was examined by excluding them and comparing the results. This did provide

some insight, particularly the usd_goal feature had an impact, by raising the loss and lowering the accuracy in its absence. Again however the results were not significant enough to conclude how much the different features impacted the model only that they did.

Based on the set of models, data subsets and hyperparameters used, it is likely that there is not a general rule to be concluded in regards to how to improve your Kickstarter campaign apart from what was shown in the exploratory analysis of the data. The accuracy from the baseline article was matched, however, the error is as significant as it makes us draw this conclusion.

VIII. Conclusion

For this project, we chose to discover, whether it would be possible to predict if a KickStarter project would be a failure or success. To do this, we first did an exploratory analysis on the data and found several features which could be used as predictive features for our model. We then designed two different ANN and carried out training on these models. With an accuracy of around 72%, our ANN is far from being perfect at predicting Kickstarter successes, especially when taking the Log Loss numbers into account, as this showed that our model did not fit our data well. However, compared to our inspiration source, who used logistic regression, reported an average accuracy of around 70%, it shows that with ANN we managed to have a slight improvement, or at least match the results.

For further improvements, we suggest that future work could be done, where Principal Component Analysis (PCA) is conducted upon the dataset features, in order to improve model fitting and accuracy. We also suggest to include some method of sentiment analysis which can analyze a project's description. Lastly, we suggest further study in the design of the ANN in order to optimize the model.

While our ANN was far from being perfect, it showed the possibility of a system, which is able to predict Kickstarter projects. If such a system is made, we can imagine it is highly relevant for investors and venture capitals, as it could help them in decision making. Other areas of application could also be for incubator programs and startup programs for new companies and projects.

References

[1] Kickstarter (2019, April) "Kickstarter" [Crowdfunding Website]. Available: https://www.kickstarter.com/

[2] L. Lewis (2019, April) "Using machine learning to predict Kickstarter success" [Online Article]. Available: https://towardsdatascience.com/using-machine-learning-to-predict-kickstarter-success-e371ab56a743

[3] WebRobots (2019, April) "Kickstarter dataset 2019-04-18" [Dataset]. Available:https://webrobots.io/kickstarter-datasets/

[4] Keras (2019, April) "Keras: The Python Deep Learning library" [Software Library] Available: https://keras.io/

[5] A. L. Maas, A. Y. Hannun, & A. Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." In *Proc. icml* (Vol. 30, No. 1, p. 3), June 2013.

[6] J. Han, M. Kamber and J. Pei, "Data Mining. Concepts and techniques" 3rd ed. Amsterdam: Elsevier/Morgan Kaufmann, 2012, pp. 327-393.

[7] J. Han, M. Kamber and J. Pei, "Data Mining. Concepts and techniques" 3rd ed. Amsterdam: Elsevier/Morgan Kaufmann, 2012, pp. 393-443.

[8] R. Shaikh (2018, November) "Cross-validation explained: evaluating estimator performance" [Figure]. Available: https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85

[9] Google Brain Team (2019, February) "Tensorflow" [Software Library]. Available: https://www.tensorflow.org/

[10] W. McKinney (2019, March) "pandas" [Software Library]. Available: https://pandas.pydata.org/

[11] F. Pedregosa, et al. "Scikit-learn: Machine learning in Python." in *Journal of machine learning research* 12.Oct (2011): 2825-2830.

[12] S. Sathyanarayana. (2014). "A Gentle Introduction to Backpropagation." Numeric Insight, Inc Whitepaper.

[13] R. Rojas. "Neural Networks - A Systematic Introduction" (ISBN 978-3540605058), Springer-Verlag, Berlin, 1996

[14] Krishni Hewa (2018, December). "K-Fold Cross Validation" [Online Article] Available: https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833