ECEN 649: Pattern Recognition - Project Report

KickStarterChance: Machine Learning Models to Predict KickStarter Projects' Success Outcome

Course Instructor

Prof. Tie Liu

Team Members and UIN

1) Khaled Nakhleh - **UIN:** 123001025
2) Drupad Khublani - **UIN:**

Due Date: December 12th, 2018.

Abstract

For the ECEN 649 final project, the goal was to build three different machine learning models. The models' function was to predict the success chances of KickStarter projects. Our methodology was to find the best combination of labels that would increase the models' accuracy rates for binary classification (succeeded, failed). Three models were used: fully connected neural network, logistic regression, and random forests. The random forests model had the highest accuracy rate out of the three, with an accuracy rate of %97.489. The following sections will discuss the results from each model, as well as offer graphs for data visualization.

The used dataset was found on Kaggle's database. The dataset we are using contained 378,661 projects that were hosted by Kickstarter, and last updated in 2018. The original dataset contained 15 labels (ID, name, main_category, etc). The dataset was missing a total of 3801 values across several labels, which resulted in a missing data percentage of %0.0669. Data cleaning techniques were utilized to convert strings into integers, fill the missing values with zeros, and remove redundant project outcomes (project gets cancelled, project gets suspended). The final clean dataset contained 10 labels, with partitioned sets for training (75% training size) and validation (25% validation size).

The label "name" was split into three labels: name length, uppercase count, and lowercase count. The three labels were introduced to identify characteristics in the project's title. Labels "project category" and "main category" were not used in the cleaned version. The other remaining labels where stored as found in the original file.

A brief results summary is provided below (for binary classification) :

1) **Fully connected neural network (Training epochs = 5, 5 hidden layers)**

Loss rate: 1.0962 Accuracy rate: %59.60

2) **Logistic Regression model**

Error rate: %37.97 Accuracy rate: %62.03

3) **Random Forests model**

Error rate: %2.51 Accuracy rate: %97.49

More details are provided on the next section. Areas of improvement are provided in the conclusion section. Full code implementation can be found on the GitHub repository [1].

Results

The results section has two subsections: the dataset cleaning process output, and the models' performance.

Data cleaning process

For the data cleaning part, a percentage of %0.0669 was found for the missing values in the dataset. Four missing values were found from the "name" label, and 3797 missing values were found for the "pledged usd" label. A total of 3801 missing values were replaced with zero. This process enabled the model to train on features that missed label values. It also prevented the model from basing false predictions based on undefined values.

Neural network Model performance

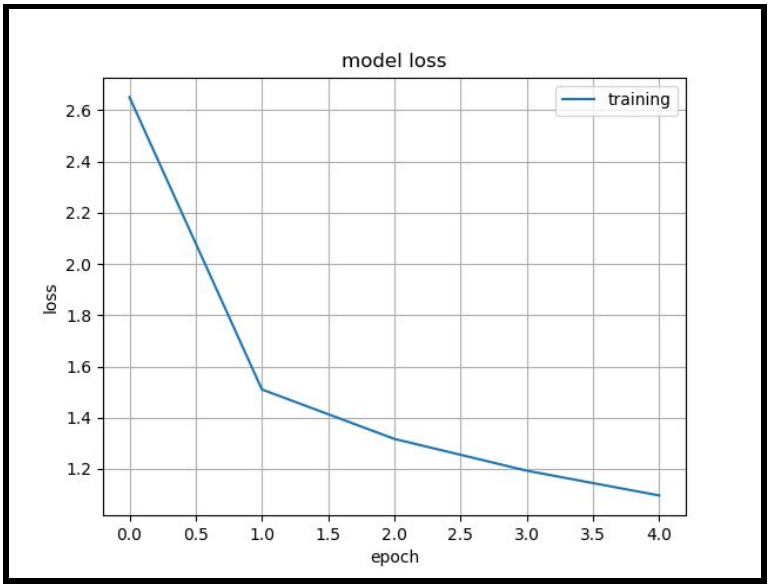

Figure 1: Neural network model loss rate

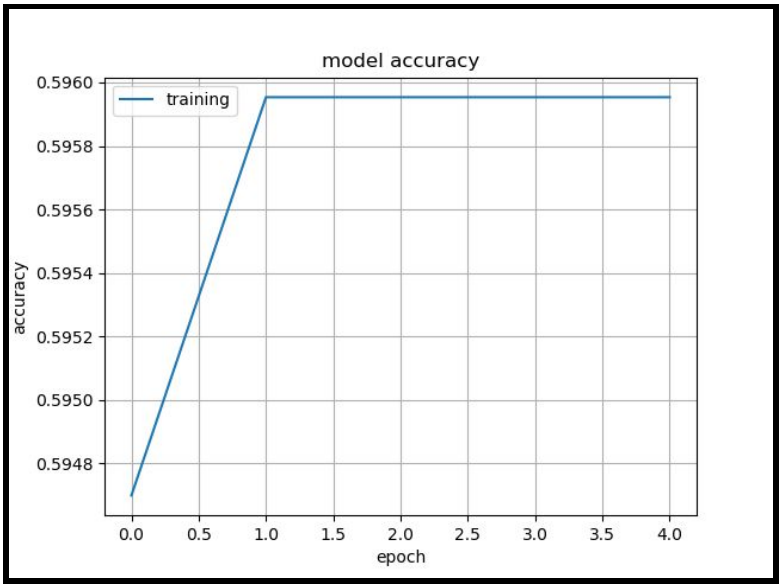

Figure 2: Neural network accuracy

For the neural network portion, the "Keras" package was used for building and training the model. The model consisted of five hidden layers, which were activated using "relu". The last output layer was activated using "softmax". The model was trained for 5 epochs, with a batch size of 500. The model accuracy was less than expected, at %59.60 accuracy rate. Figure 2 shows that the model plateaus after the first epoch. The model's accuracy rate showed an ambiguous relation for fully connected labels, which would require "windowing" the labels for better binary classification.

Logistic regression Model performance

The model was imported from Scikit-learn's Python module. This library class implements regularized logistic regression using the multiple solvers [2]. Since this dataset contains multiple classes, the solver "saga" was used. The number of iterations was arbitrarily chosen to have the model converge. The fit intercept was set to "True" to ensure convergence, otherwise the model would automatically set it to zero[2]. Figure 3 below gives the parameters chosen for this model:

Solver	Tolerance	Max iterations	Fit intercept
Saga	0.001	600	True

Figure 3: Logistic regression model's parameters

As mentioned in the abstract section, the logistic regression model had an accuracy rate of % 62.03 for binary classification (project succeeds, project fails). This is a better score than a fully connected neural network with five hidden layers.

Random forests Model performance

Random forests model was imported from Scikit-learn Python module [3]. Figure 4 gives the parameters chosen for this model:

No. estimators	Max depth	Random state
100	6	0

Figure 4: Random forests model's parameters

For this random forests model, an accuracy rate of %97.49 was achieved for binary classification. An error rate of %2.51 was recorded. This simple model was proven to be more robust than a fully connected neural network, and a logistic regression model.

The full code implementation, along with further comments, can be found on GitHub [1].

Conclusion and Future Improvements

Based on the previous results, it was found that a KickStarter project is difficult to estimate its success. The volatile nature of KickStarter projects, along with unpredicted business trends makes predictions cumbersome. This suggests that numerically predicting a project needs to be accompanied with a profound understanding of current business and societal trends. Preparing the finances alone will not benefit Kickstarter ventures.

Below, we identify future areas of improvements:

1) For the neural network mode, introduction 1-dimensional convolutional neural layers (CNN) promises improvements. A convolutional neural layer would "window" the labels, and propagate the activated neurons to the next layer. This has the potential to identify special data characteristics that a fully connected NN would not pick up. We also recommend readers to experiment with different activation functions, neuron numbers, and layers' arrangement.

2) Quantify\adding the remaining labels which were left out, such as the initial date and deadline. This can give a timeframe for which projects were getting funded on KickStarter. Quantifying the project's category would also prove useful in determining which product types have higher success probability.

3) Experiment with parameters for random forests and logistic regression models. Due to time constraints, we were not able to fully test out all possible combinations for logistic regression and random forests models. We recommend readers to start with the current parameters, and gradually add more from Scikit-learn's class allocated parameters.
4) Reducing the training size to prevent overfitting. The cleaned data version contained all 378,661 projects in the original Kaggle dataset. Removing older projects would give a better representation of current industry trends, as well as offering an overfitting buffer. Based on this approach, the generated models would be more aligned with the current business environment, since they only included newer Kickstarter projects.

For more information, and further code comments, please see the GitHub repository for further explanation [1].

References

[1] K. J. Nakhleh, “KickStarterChance” GitHub, 25-Nov-2018. [Online]. Available: https://github.com/khalednakhleh/KickStarterChance. [Accessed: 25-Nov-2018].

[2] “sklearn.linear model LogisticRegression” 1.4. Support Vector Machines - scikit-learn 0.19.2 documentation. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed: 01-Dec-2018].

[3] “sklearn ensemble RandomForestClassifier” 1.4. Support Vector Machines - scikit-learn 0.19.2 documentation. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 01-Dec-2018].