**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 Kumar Khublani - **UIN:** 526002470

**Due Date:** December 12th, 2018

**Abstract**

The goal 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 machine learning models were used: logistic regression, random forests and fully connected neural network. The random forests model had the highest accuracy rate out of the three, with an accuracy rate of 97.49%.

**Introduction to Dataset**

The KickStarter projects dataset was found on Kaggle. This dataset we are using contained 378,661 projects, and last updated in 2018. The original dataset contained 15 labels (ID, name, main\_category, etc). We divide the task of analysis into the following structure: -

1. Load Data and Modules
2. Initial Exploration and Visualization of data
3. Preparing/Cleaning data
4. Build models
5. **Load Data and Modules**

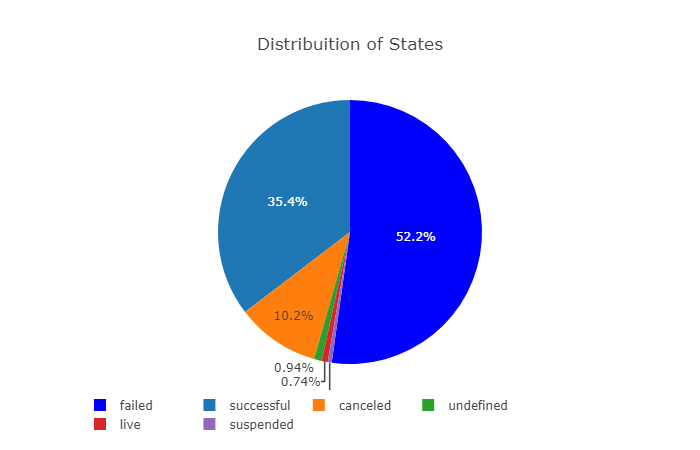
All the required modules like sklearn for logistic regression and random forest, pandas for database manipulation, matplotlib for visualization are loaded in python3. Data is imported from .csv file to data frame using pandas.

1. **Initial Exploration and Visualization of data**

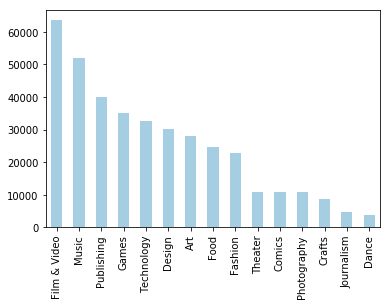
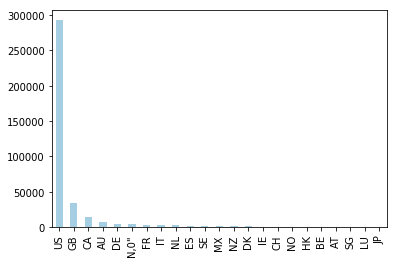
First thing we do it to see number of unique values corresponding to every column and that it gives us the following results: -

|  |  |
| --- | --- |
| ID | 378661 |
| name | 375764 |
| category | 159 |
| main\_category | 15 |
| currency | 14 |
| deadline | 3164 |
| goal | 8353 |
| launched | 378089 |
| pledged | 62130 |
| state | 6 |
| Backers | 3963 |
| country | 23 |
| usd pledged | 95455 |
| usd\_pledged\_real | 106065 |
| usd\_goal\_real | 50339 |

Our response is ‘state’ which has 6 classes, but we are aiming for binary classification. So, let’s visualize what are all the 6 classes and their distribution.



As we can see 52.2% - failed (We will use it for classification. It will be = 0) and 35.4% - successful (we will use it for classification. It will be =1). We won’t use others for our further process.

Next, we visualize the distribution of startups in different categories and from which country they are. We can see that most of the Kickstarter projects are U.S.

1. **Preparing/Cleaning data**

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.

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.

**d.Build Models**

As we are dealing with a classification problem we have to come with methods which can classify well and also capture the interactions between the predictors.

We decided to start with simple logistic model and then move to random forest. Finally, we also tried fully connected neural network on the dataset. Details of each model is mentioned below:-

1. **Logistic Regression model**

Logistic regression predicts probability of both the classes corresponding to every observation. Then it assigns the class with more probability. We can change threshold to set above which it should predict class 1 and below which it should predict class 0. 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]. The logistic regression model had an accuracy rate of % 62.03 for binary classification (project succeeds, project fails). 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**

1. **Random Forest Model**

Random Forest is an elegant classifier which as it has the property of decision tree reduces the variance and in addition the way it randomly selects predictors and its interactions to make the distribution an i.i.d also helps in reducing bias error. This is the reason we used random forest instead of normal decision trees.

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

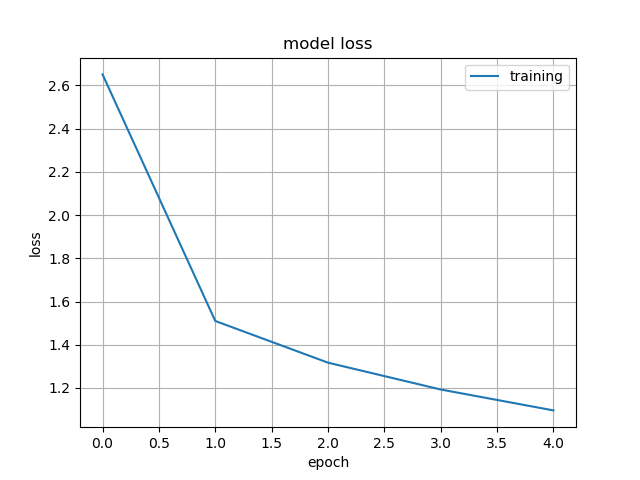
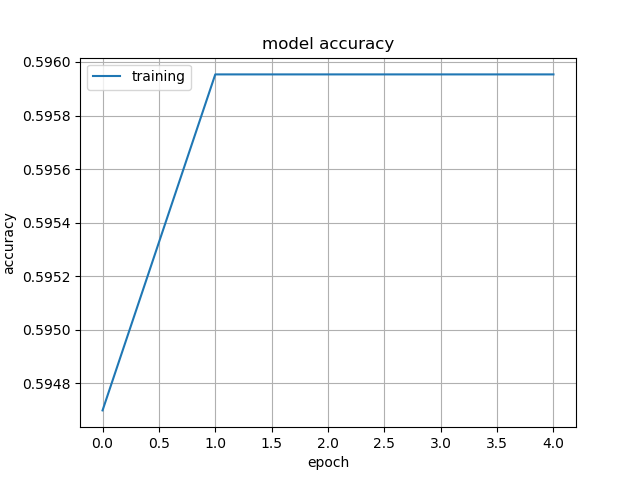
|  |  |  |
| --- | --- | --- |
| **No. estimators** | **Max depth** | **Random state** |
| 100 | 6 | 0 |

**Figure 4: Random forests model’s parameters**

1. **Fully connected neural network**

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

**Figure 1: Neural network model loss Figure 2: Neural network accuracy rate**



Areas of improvement are provided in the conclusion section. [Full code implementation can be found on the GitHub repository [1].](https://github.com/khalednakhleh/KickStarterChance)

**Results**

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

**Data cleaning process**

**Neural network Model performance**

**Random forests Model performance**

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].](https://github.com/khalednakhleh/KickStarterChance)

**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].](https://github.com/khalednakhleh/KickStarterChance)

**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].