Predicting the Success of Kickstarter Projects

INTRODUCTION

This paper is inspired by Szeto (2018) from the previous year, which predicts the success of Indiegogo projects. In that paper, we argue two problematic variables are included, namely Count (Number of backers investing into the project) and Balance (Total fund raised). Since they are supposed to be uncertain or on-going up to the time the crowdfunding reaches its deadline. Taking these variables into account provides no insights or predictive power for the next projects at its launch time. So, in the project, we aim to provide a new and precise method to predict the success of Kickstarter projects using features known when it launches.

PRELIMINARIES

Dataset Description

The Kickstarter dataset was extracted from Kaggle. It contains 378,661 observations in total spanning over from 2009 to 2018. The original dataset consists of 12 independent variables.

ID	name	category	main_category	currency	deadline	goal	launched	pledged	backers	country	usd pledged

The dependent variable “state” has six states: successful, failed, live, canceled, undefined and suspended.

PreProcessing

In terms of independent variables, ID, pledged, backers, currency and usd pledged are dropped due to irrelevancy. Text data such name is not intended to be covered. Category is also removed due to its high dimensionality. In addition, observations with missing data and state that is “live”, “canceled”, “undefined” and “suspended” are cleared out.

As a result, a total of five independent variables were extracted from the raw data and 331425 observations are kept (88% of the dataset). The dataset is reduced to 331425 rows with 6 columns including state. Independent variables include main category, deadline, goal, launched and country.

Features Engineering

State is extracted by using one-hot encoding and then concatenated to the dataset after dropping the state column. “Weekday” is generated from the launched date, 1 means “Sunday”. “Duration” is generated by calculating the number of days from “launched” and “deadline” and then concatenated to the dataset after dropping the launched and deadline column. Category is obtained by one-hot encoding due to its

sparsity in distribution (see appendix) and this generates 15 more binary features. Country is grouped into US and non-US since the US constitutes the large proportion of the dataset (see appendix). Subsequently, a dataset with size (331425, 20) is obtained.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 331425 entries, 0 to 378660
Data columns (total 20 columns):
Art             331425 non-null uint8
Comics          331425 non-null uint8
Crafts          331425 non-null uint8
Dance           331425 non-null uint8
Design          331425 non-null uint8
Fashion         331425 non-null uint8
Film & Video    331425 non-null uint8
Food            331425 non-null uint8
Games           331425 non-null uint8
Journalism      331425 non-null uint8
Music           331425 non-null uint8
Photography     331425 non-null uint8
Publishing      331425 non-null uint8
Technology      331425 non-null uint8
Theater         331425 non-null uint8
US              331425 non-null uint8
successful      331425 non-null uint8
goal            331425 non-null float64
duration        331425 non-null int64
weekday         331425 non-null int64
dtypes: float64(1), int64(2), uint8(17)
```

Next, we use the train_test_split algorithms from SciKit Learn to split the dataset into a training set and a test set. In our study, we adopted a ratio of 3:1 of the training set (75% of the dataset) versus test set (25% of the dataset). This gives us 248568 observations for training and 82857 observations for testing.

CLASSIFICATIONS

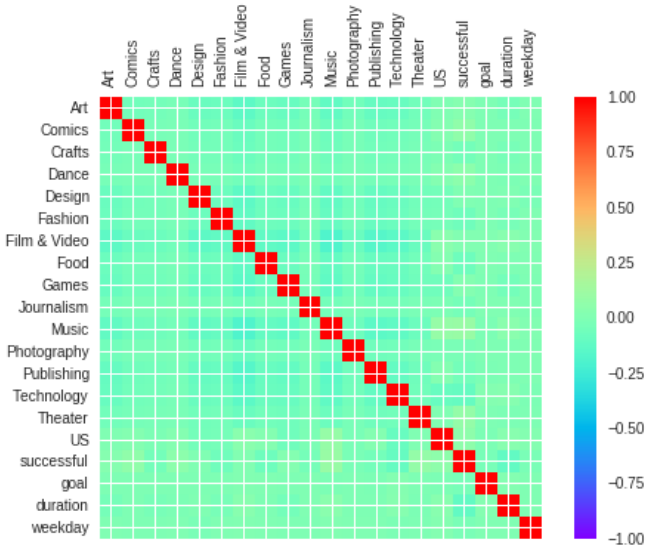
There are 6 classification algorithms that are commonly used:

1. LogisticRegression
2. Gaussian Naive Bayes
3. Support Vector Machine
4. DecisionTreeClassifier
5. KNeighborsClassifier
6. LinearDiscriminantAnalysis

We plan on applying LogisticRegression, Gaussian Naive Bayes and KNeighborsClassifier, and finally an ensemble learning in which voting is used after optimizing the algorithms. We abandon Support Vector Machine due to its time complexity which makes it difficult to run when the sample size is above 20000. Also, we avoid using DecisionTreeClassifier because of its overfitting problem.

LinearDiscriminantAnalysis assume that the features are correlated and that does not fit in our case.

Correlation Matrix:



RESULTS

1. LogisticRegression

In a binary classification problem, logistic regression would be the most convenient but powerful model for prediction. Regularization can be used to enhance the general performance of the model in the test set and regulate against overfitting. It is done by adjusting the hyperparameter, C, in the model. It acts as a penalty with a goal to minimizes the cost function.

$C = 1/\lambda$

C and λ represent two different objectives - a trade-off between the data loss and the regularization loss. A larger C which equivalent to a smaller λ represent a weaker penalty and regularization. Given a data with small disturbance, increase in the magnitude of the parameters would not improve much. In the training set, we further divide 240000 observations for training and 8568 observations for validation. We then ran the logistic regression with hyperparameters C among values of 0.001, 0.01, 0.1, 1, 5, 10, 20, 50, 100, 500, 1000. Nevertheless, there is no noteworthy change in the accuracy of the model.

```
C= 0.001 training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
C= 0.01  training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
```

```
C= 0.1   training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
C= 1     training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
C= 5     training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
C= 10    training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
C= 20    training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
C= 50    training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
C= 100   training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
C= 500   training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
C= 1000  training score: 0.5956 valid score: 0.6029 test score:
0.597088960498
Best alpha value: 0.001
```

A similar result shows on 5-fold Cross-Validation.

```
C: 0.001      test_score: 0.59619522
C: 0.01       test_score: 0.59619522
C: 0.1        test_score: 0.59619522
C: 1          test_score: 0.59619522
C: 5          test_score: 0.59619522
C: 10         test_score: 0.59619522
C: 20         test_score: 0.59619522
C: 50         test_score: 0.59619522
C: 100        test_score: 0.59619522
C: 500        test_score: 0.59619522
C: 1000       test_score: 0.59619522
Best alpha value: 0.001
```

2. Gaussian Naive Bayes

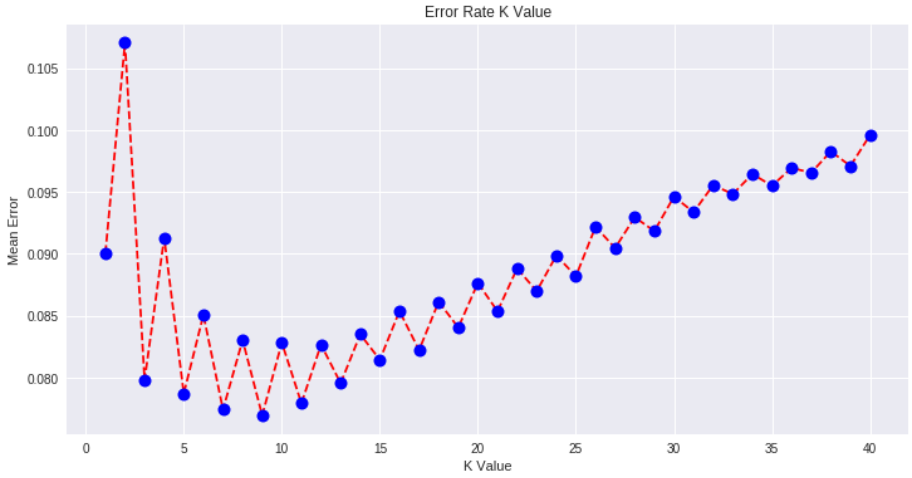
Native Bayes assumes that all the elements of regressors are independent and therefore a better fit in our case. Besides, there is no hyperparameters or tuning in Gaussian Naive Bayes. The accuracy is reported below and is fairly disappointing since it is below 0.5 (wild guess). It suggests that Gaussian Naive Bayes may not be a suitable candidate in predicting the success of crowdfunding projects, at least in our model.

```
Accuracy: 0.439021446589
```

5. KNeighborsClassifier

K-Nearest Neighbors is a non-parametric algorithm which makes no assumption about the probability distribution of the features. The number of neighbors can be determined based on a parameter - the weight function used in prediction. In our case, we use uniform weight and search for a K-value between 1 and 40 that minimize the mean error. The result has shown that nine is the best number of neighbors in our

case.



The accuracy score further confirms this finding.

```
n_neighbors: 1 test_score: 0.909784327214
n_neighbors: 2 test_score: 0.893732575401
n_neighbors: 3 test_score: 0.919994689646
n_neighbors: 4 test_score: 0.910737777134
n_neighbors: 5 test_score: 0.923965386147
n_neighbors: 6 test_score: 0.917363650627
n_neighbors: 7 test_score: 0.924665387354
n_neighbors: 8 test_score: 0.91889641189
n_neighbors: 9 test_score: 0.925063663903
n_neighbors: 10 test_score: 0.920525725044
```

ENSEMBLE LEARNING

Ensemble learning is used to enhance the performance of a single model by exploiting the old wisdom “Two heads are better than one”. Taking prediction from the heterogeneous models can yield a more accurate prediction since the probability of different model making the incorrect forecast altogether is low. Each model has equal weights though some models have high accuracy than the others. We pick the optimized model, namely C=0.01 in logistic regression and number of neighbors=9 in the K-Nearest Neighbors. And ensemble method here we used is majority voting which suggests that in our case it will only take the prediction which supported by more than half of the models. The results are reported below:

```
Accuracy_Ensemble (Logit, GaussianNB, KNN): 0.92326538494
Accuracy_Ensemble (Logit, KNN): 0.594373438575
```

The accuracy is actually lower than adopting KNeighborsClassifier alone. One of the possible reasons is the substandard performance in logistic regression and Gaussian Naive Bayes. However, if we dropped the Gaussian Naive Bayes which performs poorest, the accuracy decreases substantially. Considered the high accuracy (0.9233) using features known at its launch, this model is highly useful and one of the most

predictive models if you consider incorporating more than one classification algorithms.

ACCURACY SCORES of MODELS

Model	Accuracy score
Logistic Regression	0.597088960498
Gaussian Naïve Bayes	0.439021446589
K-Nearest Neighbors	0.925063663903
Ensemble	0.92326538494

AUC SCORE

AUC score allows us to evaluate the model independently of the threshold. It measures the tradeoff between how the positive rate (recall rate) and false positive rate. In fact, the AUC score usually takes precedence over accuracy in binary classification, especially on an unbalanced dataset. Summary of the AUC score is as follow:

```
*****Logit*****
Logit AUC score: 0.50
             precision    recall  f1-score   support

          0       0.59      1.00      0.75     49248
          1       0.00      0.00      0.00     33609

avg / total       0.35      0.59      0.44     82857


*****GaussianNB*****
GaussianNB AUC score: 0.53
             precision    recall  f1-score   support

          0       0.87      0.07      0.12     49248
          1       0.42      0.99      0.59     33609

avg / total       0.69      0.44      0.31     82857


*****KNN*****
KNN AUC score: 0.91
             precision    recall  f1-score   support

          0       0.91      0.97      0.94     49248
          1       0.95      0.85      0.90     33609

avg / total       0.92      0.92      0.92     82857
```

```
*****Ensemble*****
Ensemble AUC score: 0.92
             precision    recall  f1-score   support

          0       0.92      0.95      0.94     49248
          1       0.93      0.88      0.90     33609

avg / total       0.92      0.92      0.92     82857


*****Baseline*****
Baseline AUC score: 0.50
             precision    recall  f1-score   support

          0       0.60      0.59      0.60     49501
          1       0.40      0.41      0.41     33356

avg / total       0.52      0.52      0.52     82857
```

The AUC score for the Baseline is 0.5 as anticipated. Both Logistic Regression and Gaussian Naive Bayes show no significant difference than 0.5. However, K-nearest neighbours once again, outperform the two models with the AUC score of 0.91. One interesting fact is that ensemble learning performs better than K-nearest neighbours in terms of AUC score and may imply overfitting in using K-nearest neighbours. This suggests that ensemble learning may be the best model in predicting the success of Kickstarter projects due to the imbalance in the success or failure state.

CONCLUSION

In this paper, we criticized the previous work on the ground that some of the selected features are unrealistic. We then start off to build our model choosing variables that are acknowledged at its launch, using the Kickstarter dataset from Kaggle. Models adopted include Logistics Regression, Gaussian Naive Bayes, K-nearest neighbours and finally ensemble learning in which hard voting is used. Logistics Regression and Gaussian Naive Bayes perform fairly just as the baseline model. K-nearest neighbours and ensemble voting can successfully predict the outcome of a crowdfunding project over 92%. Ensemble voting although has a slightly lower accuracy score than K-nearest neighbours, we conclude ensemble voting is still the best model given the higher AUC score. Motivating as the result may be, we have to admit some important variables (e.g. text description and promotional video) and algorithms (e.g. Support Vector Machine) are left out due to the handle difficulties and limitation of technology and computation power.

Reference

Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures. (2016, November 11). Retrieved from http://blog.exsilio.com/all/accuracyprecision-recall-f1-score-interpretation-ofperformance-measures/

Mouillé, M. (2018, February 08). Kickstarter Projects. Retrieved May 13, 2019, from https://www.kaggle.com/kemical/kickstarter-projects

Robinson, S. (2018, February 15). K-Nearest Neighbors Algorithm in Python and Scikit-Learn. Retrieved May 13, 2019, from https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

Szeto, C., Shirley. (2018, April). Prediction of the Success of Crowdfunding Projects. Retrieved May 13, 2019.

Appendix

	main_category_%	main_category_count
Film & Video	17.047899	56501
Music	13.817908	45796
Publishing	10.684469	35411
Games	8.605265	28520
Technology	8.159010	27041
Art	7.735385	25637
Design	7.651505	25359
Food	6.654296	22054
Fashion	5.963642	19765
Theater	3.089688	10240
Comics	2.979860	9876
Photography	2.922531	9686
Crafts	2.358905	7818
Journalism	1.251565	4148
Dance	1.078072	3573

	country_%	country_count
US	78.851626	261334
GB	8.886777	29453
CA	3.731463	12367
AU	1.995323	6613
DE	1.036132	3434
FR	0.760353	2520
NL	0.727465	2411
IT	0.714490	2368
ES	0.564532	1871
SE	0.455307	1509
MX	0.425436	1410
NZ	0.384401	1274
DK	0.279400	926
IE	0.206080	683
CH	0.196425	651
NO	0.175605	582
BE	0.157803	523
AT	0.146338	485
HK	0.143924	477
SG	0.136984	454
LU	0.017198	57
JP	0.006940	23