Iteration3 OSAS: Success Rate Prediction Of Kickstarter Project

Jieyu KONG (UPI: jkon404)
jkon404@aucklanduni.ac.nz

Abstract

Kickstarter is a global crowdfunding platform focused on different domain, like Design, technology, music and so on. For the purpose of predicting the final state of Kickstarter project (success or fail), in this OSAS project, Python open sources would be implemented to complete the data KDD process. The Kickstarter project dataset is collected from Kaggle [1] and I would implement 8 phases to complete the task, including data preprocessing and modeling and evaluation. Besides using the open sources of python, tableau would also be applied to visualize the relation model.

Keywords Success rate, Kickstarter, OSAS, prediction, Python, classification, Supervise learning

1 Business and/or Situation understanding.

1.1 Identify the objectives of the business and/or situation

Analyze the most important element on the success of Kickstarter project campaigns and improve the chance of success.

1.2 Assess the situation;

Crowdfunding is a promising field for data mining, as it is an innovative business model nowadays and Kickstarter is the most prominent websites in the world. People are not familiar with the procedure as well as how to achieve success in Kickstart. According to the statistic, the success rate for Kickstarter campaign is approximately 35% and the success rate decreases over time. It's worth to figure out what kind of elements will have more or less effect on the success of the Kickstarter campaign. One of the first tasks is to assess the reliable historical data of the Kickstart project.

Data: Since there are plenty of data platforms nowadays, like Data world, KDnuggets, Kaggle, Kickstart project Kickstart dataset could be available on those platforms.

Risks: It is hard to access an accurate the dataset. In addition, Using MacBook Pro might not be able to process large amount of data, professional devices should be implemented.

Constrains: As I can not access to the backend data of Kickstart platform, Limited fields of data can be accessed. I can only use all the fields (category, goal, Month of the campaign ...) I can access to build the model to make prediction of success rate.

1.3 Determine data mining objectives

- build relation models of [category, date, goal and other fields]-[success] using data mining on big data;
- build relation models of [year-success] using data mining on big data;
- build relation models of [main_category-success] using data mining on big data;

1.4 Project plan for OSAS

The whole project analysis including 8 phases, which are listed in Table.1 and visualize in Fig.1 For the project of OSAS, I am going to focus on the last 6 phases, as the first two steps have been analyzed in July. Python open sources would be the main tool for this OSAS project.

Phase	Days	Start Date
Business understanding	1 week	10th July
Data Understanding	1 week	17th July
Data Preparation	3 days	25th Aug
Data transformation	2 days	28th Aug
Data-mining methods selection	3 days	31st Aug
Data-mining algorithms selection	3 days	2nd Sep
Data Mining	1 week	5th Sep
Interpretation	3 days	10th Sep

Table 1. Classification of state of the project.

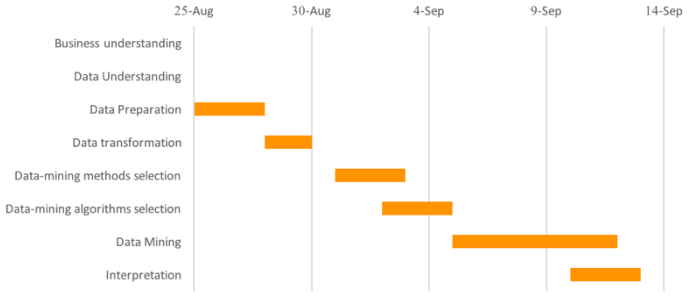

Figure 1. Project plan (from 25th Aug – 14th Sep).

2 Data understanding

2.1 Collect initial data

I collected my data from Kaggle [1], a platform for predictive modelling and analytics competitions. raw data is available for people to download and process. In the process of collecting the data, I searched Kickstarter dataset in the official Kickstarter Website first, but project data is not able to access there, and

then I tried different platform, such as kaggle, data world to collect the relevant dataset. I initially got a dataset of Kickstart project before 2016, but there are lots of blank value of that dataset. Finally I got a satisfied dataset with more than 300000 projects are recorded from Kaggle.

2.2 Describe the data

Kickstarter dataset is an csv file which including 378661 rows of project information. The dataset include the 15 fields: ID, name, category, main_category, currency, deadline, goal, launch_date, pledge, state, backers, country, usd_pleadged, usd_pleadged_real and usd_goal_real. The column of State denotes the finally state of the project, including failed, successful, canceled, live, suspended and undefined, which are target column I am going to predict in my data mining project. In order to predict the successful rate the project, I would reclassify the the value of state in the 3 and 4 phases;
For the original dataset, the explanations of each column are listed as following Table 2.

Column	Column Explanation
ID	The ID number of the project, which are now shown in order in the original table.(type: numeric)
name	name of project(type: String)
main_category	category of campaign, including 15 kinds of main_catetory, such as Art, Craft, Technology and so on. (type: String)
category	For each main_category, there are various subdivided categories. For example, for the main category of music, there are categories of pop, rock, Jazz. (Type: String)
currency	The Currency used for pledge. (Type: String)
deadline	The deadline for the campaign. (Type: String)
goal	The aimed amount of money to collect. (Type: numeric)
launched	The launched date of the project
pledged	The amount of money they collected. (Type: String)
state	The state of the project, fail, success, cancelled, live, suspended, undefined. This is the target column to predict in my data mining project. (Type: String)
backers	The number of people who support the project. (Type: numeric)
country	The country where the project from. (Type: String)
usd pledged	conversion in US dollars of the pledged column. (Type: numeric)
usd_pledged_real	conversion in US dollars of the pledged column (real situation). (Type: numeric)
usd_goal_real	conversion in US dollars of the goal column. (Type: numeric)

Table 2. Explanation of each column.

2.3 Explore the data

From the visualized chart Fig.2, there are six kind of states of the Kickstarts project in the dataset, including failed, successful, canceled, live, suspended and undefined. Failed project accounts for largest number of projects and the number of successful project are ranked the second. In order to predict the state of project in the future, the state of original dataset (which is also the tranning set) should ne processed. With the preliminary understanding of the dataset, I predicted that the main-category, category Fig.3, the goal, or the duration of the campaign of the would have a main influence on the successful rate of the project. I would use python to prove calculate the importance rate of each field in the following phase. Other fields might also have more or less influence on the success rate of the project. For the field of launched and deadline, field of the duration of the campaign can be generated, which can affect the final state of the project. From the dataset, the year or the month can also be extracted from the field of launched date. With the data of launched year, we might be able to see whether the successful rate is different in each year, as well as the tendency of the successful rate.

```
df['state'].value_counts().plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f35847cc8d0>
```

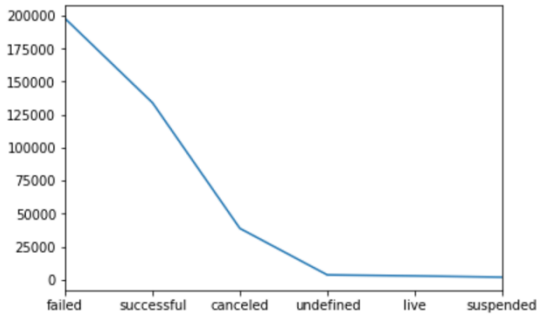

Figure 2. The number of project in different states.

```
df['main_category'].value_counts().plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f35853bdb38>
```

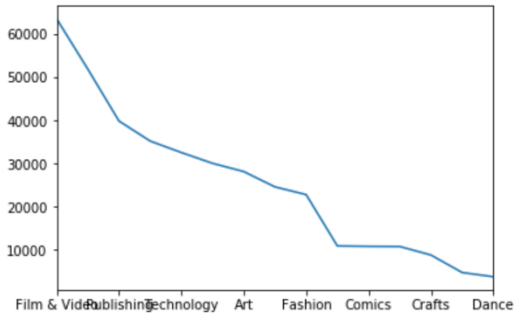

Figure 3. The number of project in different categories.

2.4 Verify the data quality

Missing Data: there are missing values in two column, the column of name and the column of usd pledged, 4 and 3797 respectively Fig.4.

```
df.isnull().sum()
```

```
KsID                0
name                4
category            0
main_category       0
currency            0
deadline            0
goal                0
launched            0
pledged             0
state               0
backers             0
country             0
usd pledged      3797
usd_pledged_real    0
usd_goal_real       0
dtype: int64
```

Figure 4. Missing value evaluation.

Deviation in Data: In the column of goal, there are some extreme values and outliers according to the histogram chart shown below Fig.5. Log function is applied to normalize the distribution of the value of goal.

```
df['goal_log'] = np.log(df['goal'])
```

```
df['goal_log'].hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f35846f4c
```

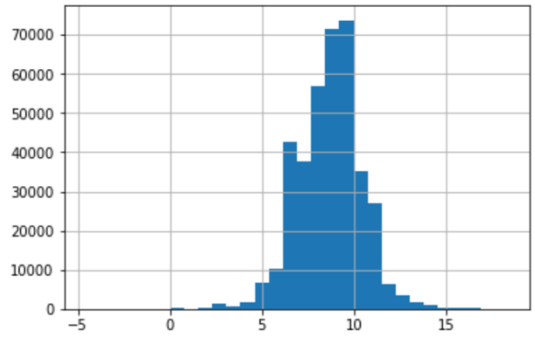

Figure 5. The distribution of the value of goal.

3 Data preparation

3.1 Select the data

In the column of state, there are six value, however, only the state of success and fail are useful in the process of building the predictive model. In order to make a binary prediction, I classify the state of canceled and the state of suspended as failed, as well as drop the rows which state are undefined and live Fig.6. After processing the column of state, I manage to deleted the repetitive columns in the dataset, such as the column of ['goal'] and ['usd_goal_real'], the column of ['pledged'], ['usd_pledged'] and ['usd_pledged_real']. In addition, as the column of ['ksID'] is not an automatically generated ID number for each rows, it will not affect the final state of the project, the ['KsID'] column is excluded in this stages. Without calculation of the important scale of other columns, it's hard to determine the column to keep or drop at this stage it's reasonable to keep them for the next phase.

```
newState01 = df['state'].replace(['canceled','suspended'],'failed',inplace=True)
```

```
df = df[df.state != 'undefined']
```

```
df = df[df.state != 'live']
```

```
df['state'].value_counts()
```

```
failed        238340
successful    133956
Name: state, dtype: int64
```

Figure 6. Result of processing the column of stage.

3.2 Clean the data

Missing Value: there are 4 and 3797 missing value in the column of ['name'] and ['usd pledged'] respectively Fig.4. 4 missing value in the column of ['name'] are excluded while the blank values in the column of ['usd pledged'] are filled with the values in the column of ['usd_pledged_real'].
Data Error: In the column of ['goal'], there are unreasonable values there, such as the goal is 0 dollars, which means this campaign is not aimed at collecting any funds, which is meaningless. And there is some extremely high number in the column of ['goal']. Using the histogram to visualize the distribution of goal value Fig.5, and clean the extreme values and outlier which might have a negative impact on the accuracy of the predictive model.
Ambiguous Data: In the column of ['state'], there are six values (failed, successful, canceled, live, suspended and undefined), however, only the state of success and failed are useful in the process of building the predictive model. It is hard to classify the state of undefined and live into fail or success, so the rows with state of undefined or live would be excluded.
After Cleaning the Data, the size of dataset changed: After clean the rows with missing data, error data, and the ambiguous data, the size of the dataset shrink to 369936 rows Fig.7, (originally 378661 rows) which is still massive enough for machine learning.

3.3 Construct the data

There are four important columns are generated Table.3 and implemented in the following phases. The LaunchedYear, LaunchedMonth, LaunchedWeekday, campaignDuration. The first three fields are derived form the fields ['launched']. In order to study whether the year, month or even weekday would affect on the successful state, as the market sentiment is changing through the year, I extract those fields from the column of ['launched']. For the field of campaignDuration, which is the measure the duration of the campaign, using the deadline subtract the launched date.

```
In [27]: # according to the histogram chart above
         df['goal_logmin']= df['goal_log']<=2.5

In [28]: # the sum of the extreme low value
         df['goal_logmin'].sum()

Out[28]: 1299

In [29]: # according to the histogram chart above
         df['goal_logmax']=df['goal_log']>=14

In [30]: # the sum of the extreme high value
         df['goal_logmax'].sum()

Out[30]: 1061

In [31]: # delete the extreme value
         df = df[df.goal_log > 2.5]
         df = df[df.goal_log < 14]

In [33]: df.shape

Out[33]: (369936, 18)
```

Figure 7. The size of dataset after cleaning the data.

New Column	Derived from
LaunchedYear	launched
LaunchedMonth	launched
LaunchedWeekday	launched
campaignDuration	deadline - launched'

Table 4. Constructing the data.

3.4 Format the data

In order to implement supervised learning to generate the classification model. Data in several fields are being formatted Table.4.

column	Original format	New format
Main_category	The name of the main category. (type:object)	Number(type:int)
category	The name of the category (type:object)	Number(type:int)
launched Fig.8	DD/MM/YYYY HH:MM (type:object)	YYYY-MM-DD (type:datetime)
deadline	DD/MM/YYYY (type:object)	YYYY-MM-DD (type:datetime)
state	Successful or failed (type:object)	0, or 1 (type:int)
country	The abbreviation of the country name (type: object)	Number (type:int)

Table 5. Formatting the data.

```
In [31]: df['launched'].head()

Out[31]: 0    11/08/2015 12:12
         1     02/09/2017 4:43
         2     12/01/2013 0:20
         3     17/03/2012 3:24
         4     04/07/2015 8:35
         Name: launched, dtype: object

In [64]: # formatting the field of lauched. from DD/MM/Y
         df['launched'] = df.launched.str.slice(0, 11)

In [70]: df['launched']=pd.to_datetime(df.launched, dayf

In [69]: df['launched'].head()

Out[69]: 0   2015-08-11
         1   2017-09-02
         2   2013-01-12
         3   2012-03-17
         4   2015-07-04
         Name: launched, dtype: datetime64[ns]
```

Figure 8. Data format conversion of column ['launched'].

4 Data transformation

4.1 Reduce the data

There are horizontal reduction and vertical reduction in my dataset. In terms of horizontal reduction, repetitive columns as well as the column of ID are excluded at the beginning. In the phase of modeling, I also apply a sorting function to filter the most important fields related to the successful state Fig.9. In the aspect of vertical reduction, rows with missing values, error values, extreme values and ambiguous values are dropped, the size of the dataset become 369936 rows and 17 columns Fig.10.

```
In [48]: df.drop(columns=['goal_logmin', 'goal_logmax','

In [49]: df.dtypes

Out[49]: KsID                int64
         name               object
         category            int64
         main_category       int64
         currency           object
         deadline    datetime64[ns]
         launched    datetime64[ns]
         state               int64
         backers             int64
         country             int64
         usd_pledged_real  float64
         usd_goal_real     float64
         goal_normalized   float64
         launchedYear        int64
         launchedMonth       int64
         launchedWeekday     int64
         campaignDur         int64
         dtype: object

In [50]: df.shape

Out[50]: (369936, 17)
```

Figure 10. Reducing the data.

4.2 Project the data

For the data in the column of ['goal'], data ranged from 0 to millions of dollars, log function is applied to project the data Fig.11.

```
df['goal_log'] = np.log(df['goal'])
```

```
df['goal_log'].hist(bins=30)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a142bb
```

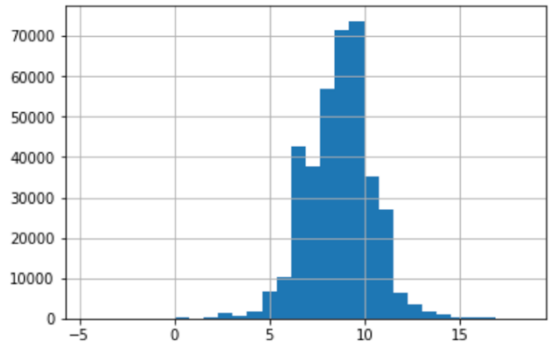

Figure 11. Project the data.

5 Data-mining method(s) selection

5.1 Match and discuss the objectives of data mining to data mining methods

The objective of this data mining project is to accurately predict the final state (successful or failed) of the Kickstarter project. That means the target is ['state'], and the target attribute has only two possible values, 0 or 1(which means failed or successful). It is a binary classification problem. As the dataset, in which the final state of the projects as well as different labels (different fields) are known, supervised learning can be implemented.

5.2 Select the appropriate data-mining method(s)

There are many data mining methods for different cases, such as Association learning, Clustering, Regression and Classification. For each of them, association learning is used to analyzed the things that occur together, clustering is applied to recognize distinct group, regression is used to make prediction based on the relationship of the dataset (always use to predict a numerical target). Classification is also used to make prediction, but the target can be discrete. **The data type available for mining:** the variables of [launchedYear], [launchedMonth], [launchedWeekday], [category], [main_category], [campaignDur], [country] are categorical type. It's essential for running a binary classification data-mining algorithm.
Data mining goal: my project is a binary predictive cases. classification data-mining methods is fit for my case.
In conclusion: classification data-mining methods is appropriate to applied in my case, classification data-mining algorithm will be described in the next step.

6 Data-mining algorithm(s) selection

6.1 Exploratory analysis and discuss

Kickstarter data mining project is a classification problem, predicting one variable - the state of the project, based on the other attributes in the kickstarter dataset. With the analysis of the dataset, I can preliminarily filter the data-mining algorithm by type. There are various types of algorithms, including classification algorithm, regression algorithm, segmentation algorithm, association algorithm as well as sequence analysis algorithm. For the type of classification algorithm, there are different algorithms for predicting a discrete attribute in classification can be used, such as decision tree, random forest, neural network, naïve bayes and so on. In my project, different algorithms would be applied to test and determine the most effective algorithm for this case.

6.2 Select data-mining algorithm based on discussion

Decision tree, which is a classical binary classification algorithm would be applied in this predictive project, although it might not have the best performance. In contrast to the performance of decision tree, random forest with different parameters will be applied to predict the goal variable. Random forest is an ensemble of decision trees, which builds multiple decision trees and merge them together to get a more accurate and stable prediction. As the size of the dataset is reasonable massive, it can be split into training set and testing set. Cross-validation would also be implemented to assure the robust of the model.

6.3 Build/Select appropriate models and choose relevant parameter

We can from the figure Fig.12 below, a function called classification_model is defined, which requires four parameters, model, data, predictors. Different algorithm can be applied in this function. Inside the function, cross-validation is defined, which help me to split the dataset into a training set and testing set. After defining the model of classification, I determined the outcome, which is the 'state' and the variables used for data mining, which shown in line [59].

```
In [57]: def classification_model (model, data, predictors, outcome):
             model.fit(data[predictors], data[outcome])
             predictions = model.predict(data[predictors])
             accuracy = metrics.accuracy_score(predictions, data[outcome])
             print('Accuracy:{0}%'.format(accuracy))
             kf = KFold(n_splits=5)
             kf.get_n_splits(data)
             error = []
             for train, test in kf.split(data):
                 train_predictors = (data[predictors].iloc[train, :])
                 train_target = data[outcome].iloc[train]
                 model.fit(train_predictors, train_target)
                 error.append(model.score(data[predictors].iloc[test, :], dat
                 print('plot:',model.score(data[predictors].iloc[test, :], da
             print("Cross-Validation_Score {0}".format(np.mean(error)))
             model.fit(data[predictors], data[outcome])

In [58]: outcome_var ='state'

In [59]: predictor_var = ['usd_goal_real','launchedYear', 'launchedMonth','la
```

Figure 12. model of Decision tree

Firstly, I train the decision tree model Fig.13. The accuracy is quite high, but the cross-validation is only 60.8%. This implies the model is over-fitting the data.

```
In [60]: model = DecisionTreeClassifier()

In [61]: classification_model(model, df, predictor_var, outcome_var)

         Accuracy:0.9620583019765581%
         plot: 0.6075850137860194
         plot: 0.6063767959236082
         plot: 0.6098503791206563
         plot: 0.6089988781813022
         plot: 0.6105532052928218
         Cross-Validation_Score 0.6086728544608816
```

Figure 13. Accuracy of decision tree.

Secondly, I try the random forest model with default parameters Fig.14. The cross-validation performs better than the decision tree one, but the gap between the accuracy and the cross-validation is still quite obvious.

```
In [62]: model = RandomForestClassifier(n_estimators=100)

In [63]: classification_model(model, newdf, predictor_var, outcome_var)

         Accuracy:0.96204208295488955%
         plot: 0.658944693734119
         plot: 0.6573181775176721
         plot: 0.659264465379053
         plot: 0.6601159663184073
         plot: 0.6612918485679917
         Cross-Validation_Score 0.6593870303034486
```

Figure 14. Accuracy of decision tree.

In order to improve the performance of the random forest model, two methods are applied. One is filtering the variables used in data mining and selecting the most important variables Fig.15. Another method is to tune the parameter of the random forest model Fig.16, the parameters of n_estimators, min_samples_split, min_samples_leaf, max_features, random_state, max_depth and so on are reset. There is a big improvement after implementing those two methods. Although the accuracy reduced, the cross-validation score is improving, which is nearly 70%, showing that the model is generalizing well.

```
In [69]: #Create a series with feature importances:
         featimp = pd.Series(model.feature_importances_,

In [70]: featimp

Out[70]: usd_goal_real       0.365627
         campaignDur         0.214238
         main_category       0.170511
         launchedYear        0.115119
         launchedMonth       0.059211
         launchedWeekday     0.041828
         country             0.033465
         dtype: float64
```

Figure 15. Create a series of important feature.

```
In [71]: predictor_var = ['usd_goal_real','launchedYear', 'laun

In [72]: r(n_estimators=80,min_samples_split=100,min_samples_le
         f, predictor_var, outcome_var)

         Accuracy:0.7104526188313655%
         plot: 0.6874898632210629
         plot: 0.6874045440415208
         plot: 0.6893778636787544
         plot: 0.6878235365672348
         plot: 0.6908105477989377
         Cross-Validation_Score 0.6885812710615021
```

Figure 16. Better performance of random forest.

During the process of setting the parameters, there are some parameters that make a different on the performance and the computation speed of the model. Such as n_estimators, max_depth. The number of n_estimators would affect the computation speed of the model, the larger the number is, the slower the model run. It's common to set it around 100. For the parameter of max_depth, it control the depth of the tree, it's necessary to set this parameter, if the dataset is massive to avoid overfitting. I set it as 60.

7 Data Mining

7.1 Create and justify test design

In the process of building predictive model, cross-validation is being implemented. Dataset is divided into multiple cross-section and then iteratively test the structure and the data mining models. Based on the analysis, it outputs a set of accuracy measures for each model, and an average value of cross-validation.

At the beginning, I set the fold as 5. Why I choose 5? In terms of the volume of dataset, my dataset is not too massive, setting the fold as 5 can produce a reasonable robust accuracy. In terms of time consuming, setting the fold as 10 would take a long time to run and it doesn't make a big different on the result, which shown in Fig.17, the value of cross-validation is nearly the same as the previous one Fig.16.

```
In [89]: model = RandomForestClassifier(n_estimators=100,min_samples_sp
         classification_model(model, df, predictor_var, outcome_var)

         Accuracy:0.7256471389645777%
         plot: 0.6867600151375899
         plot: 0.6863815753906038
         plot: 0.6856517273071309
         plot: 0.6898686273449749
         plot: 0.6883008055360329
         plot: 0.6895983132399849
         plot: 0.6865623226015732
         plot: 0.6891303760170843
         plot: 0.6916173330089477
         plot: 0.6878868975211526
         Cross-Validation_Score 0.6881757993105075
```

Figure 17. Setting the fold as 10

7.2 Conduct data mining – classify

In my project, I conduct the data mining using classification method. Two classification algorithms are implemented, the decision tree and random forest. The models run perfectly, which is shown below Fig.18.

```
In [57]: def classification_model (model, data, predictors, outcome):
             model.fit(data[predictors], data[outcome])
             predictions = model.predict(data[predictors])
             accuracy = metrics.accuracy_score(predictions, data[outcome])
             print('Accuracy:{0}%'.format(accuracy))
             kf = KFold(n_splits=5)
             kf.get_n_splits(data)
             error = []
             for train, test in kf.split(data):
                 train_predictors = (data[predictors].iloc[train, :])
                 train_target = data[outcome].iloc[train]
                 model.fit(train_predictors, train_target)
                 error.append(model.score(data[predictors].iloc[test, :], c
                 print('plot:',model.score(data[predictors].iloc[test, :],
             print("Cross-Validation_Score {0}".format(np.mean(error)))
             model.fit(data[predictors], data[outcome])

In [58]: outcome_var ='state'

In [59]: dictor_var = ['usd_goal_real','launchedYear', 'launchedMonth','laur

In [60]: model = DecisionTreeClassifier()

In [61]: classification_model(model, df, predictor_var, outcome_var)

         Accuracy:0.9620583019765581%
         plot: 0.6075850137860194
         plot: 0.6063767959236082
         plot: 0.6098503791206563
         plot: 0.6089988781813022
         plot: 0.6105532052928218
         Cross-Validation_Score 0.6086728544608816

In [62]: model = RandomForestClassifier(n_estimators=100)

In [63]: classification_model(model, newdf, predictor_var, outcome_var)

         Accuracy:0.9620420829548895%
         plot: 0.658944693734119
         plot: 0.6573181775176721
         plot: 0.659264465379053
         plot: 0.6601159663184073
         plot: 0.6612918485679917
         Cross-Validation_Score 0.6593870303034486
```

Figure 18. The screen shoot of running the model.

8 Interpretation

8.1 Study and discuss the mined pattern

Compare with the model of decision tree, the model of random forest performs better, even though it takes longer time to run. Although the accuracy for training is not high, but the cross-validation score (accuracy for predicting) is almost the same as the value for training, nearly 70%, showing that the model is generalizing well Fig.19.

In terms of the metrics, the precision rate is around 70%, which is not bad, but the recall rate of 1 is a little bit weak. More detail would be elaborated in 8.3.

```
In [122]: print("randomforest's Accuracy for trainning: ", metrics.accu
          print("randomforest's Accuracy for predicting: ", metrics.acc

          randomforest's Accuracy for trainning:  0.7262445395960382
          randomforest's Accuracy for predicting:  0.6880974006314606

In [123]: print(confusion_matrix(y_testset, predTreeTest))
          print(classification_report(y_testset, predTreeTest))

          [[51193  8056]
           [20790 12445]]
                        precision    recall  f1-score   support

                     0       0.71      0.86      0.78     59249
                     1       0.61      0.37      0.46     33235

              avg / total       0.67      0.69      0.67     92484
```

Figure 19. Model evaluation.

8.2 Visualize the data, result, models, and patterns

To visualize the data, I use another open source software, called tableau, which is proficient in visualizing data. In this part, I am going to visualize the objectives for this project.

In term of building relation models of [main_category-success] Fig.20. The orange parts mean successful while blue parts mean failed. According to the figure below, we can see that there are some categories have a higher success rate, such as 3, which is Dance. Even though there are very limited projects belong to 3, but the success rate is quite high, which is apparently over 50%. However, the category of 12, which is publishing, seems to have a low success rate.

In terms of build relation models of [year-success], we can see from the histogram below Fig.21, the amount of successful project is stable, while the total amount of kickstart project increased since 2011. It's undoubted that making success in kickstart campaign would be harder and harder.

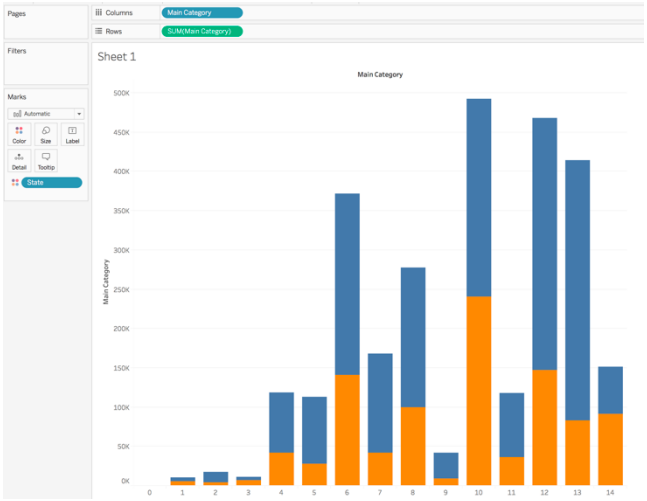

Figure 20. relation models of [main_category-success]

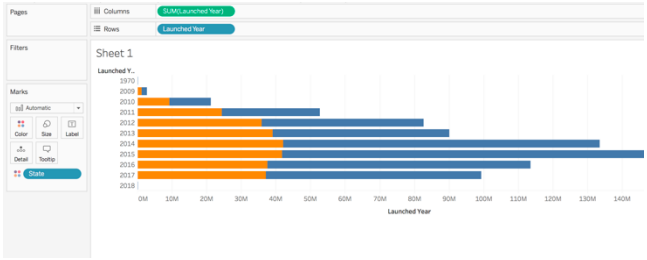

Figure 22. relation models of [year-success]

8.3 Interpret the results, models, and patterns

After filtering the features and tuning the parameters, the performance of random forest model becomes better. The accuracy of training model is 72.6% while the accuracy of predicting model is 68.8% Fig.19. With the implement of cross-validation, the deviation between those two values becomes small.

In terms of the metrics Fig.19, TP (true positive) is 51193, TN (true negative) is 20790, FP (false positive) is 8056, FN (false

negative) is 12445. In details, true positive means, the true label is 1, and the predictive label is 1 too. True negative means, the true label is 1, but the predictive label is 0.

In terms of the metrics of precision and recall Fig.23, Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. I get 71% for 0, which is relatively good. Recall means the ratio of correctly predicted positive observations to the all observations in actual class. I got 86% for 0, which is good for this model as it above 0.5, but the recall value for 1 is relatively weak. For these two values, the higher the better.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Figure 23. The formula of Precision and Recall.

For the score of F1, it is the weighted average of Precision and Recall Fig.24. Therefore, this score takes both false positives and false negatives into account. In my case, the F1 score for 1 is 78%, which performs excellent, while the F1 score for 0 is 46%. F1 is usually more useful than accuracy, especially if you have an uneven class distribution.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Figure 24. The formula of f1-score.

8.4 Assess and evaluate results, models, and patterns

the overall result of Kickstart project with data mining is fairly easy to communicate from the business perspective. The study helps us understand how to increase the possibility of making a success in kickstart campaign. The accuracy of the random forest model is relatively good.

9 Action

9.1 How I would apply the knowledge and deploy the implementation

As the top elements affect the success rate are pledged (the amount of money you need to pledge), goal, and backers. When a new project is launched, launcher need to consider seriously the goal of money they want to pledge totally. And it is also important to attract more people to browse your project and get support from them.

9.2 How to enhance the solution in the future?

In this project, I used decision tree and random forest to train the predictive model. In the future, I can try more algorithm to enhance the performance of the model. What's more, method of stacking and bagging can also be used to improve the performance;

References

[1] https://www.kaggle.com/kemical/kickstarter-projects/home