INSY662 Individual Project Report

I. Introduction

Kickstarter is a crowdfunding website that has helped bring many creative projects to life. It does this by allowing the project author to create their project first while filling in basic project information in the process, then submit the project for review, after the review is approved the project author can then create a pre-launch page to advertise, and finally, when the project author is ready, he can officially launch the project and start crowdfunding.

My project mainly focuses on two points: building a classification model to help Kickstart and its users in predicting whether their projects will be successful when the project is launched and building a clustering model to help Kickstarter segment the projects on their website.

II. Classification Model

2.1 Data Cleaning & Preprocessing

To clean the data, firstly, I dropped all features that can only be realized after the project is launched since the prediction happens at the time each project is launched. Secondly, I dropped rows that contain missing values and columns that main compose of missing values. Thirdly, I dropped the features which are highly correlated with the target variable by drawing a heatmap. Fourthly, I dropped rows whose state columns value are not ‘fail’ or ‘success’. Lastly, I dropped the columns which should not be related to the target variable intuitively, and columns that contain mainly the DateTime variable. To preprocess the data for the model, firstly, I dummified all non-numerical variables. Secondly, I spitted out the target and predictor variables and standardized the predictor variables.

2.2 Feature Engineering & Model Constructing

I constructed five models and compared their performance, and selected out the best models finally. The first model is using the LASSO method to perform feature selection tasks and the KNN algorithm to construct the classification model. The LASSO method regularizes model parameters by shrinking the regression coefficients, reducing some of them to zero. Meanwhile, those predictors who got assigned zero coefficients are relatively meaningless. Hence, I removed them and only used those being selected by LASSO to train the KNN model. To train the model, I used the cross-validation method, spitted the train, and test data by 70% and 30%, to compute the accuracy of the model. Lastly, I generated the confusion matrix and classification report of the KNN model. The accuracy of the first model is around 62.32%.

The second model is still using the LASSO method to perform feature selection tasks but changed the classification model from the KNN model to the Gradient-Tree-Boosting (GBT) model. I used the same train and test data as the first model. The accuracy of the second model is around 67.81%. As one can see, the model performance improves significantly with the Tree-based models. Hence, in my following tests, I mainly focused on advanced tree-based models such as random forest and GBT.

The third model is using the random forest to perform feature selection tasks and the GBT model to perform classification tasks. When using the random forest model to select features, I set the threshold to 0.04 and discarded all the other predictors whose feature importance is lower than 0.04. Then I standardized the selected predictor variables and use them to train the GBT model. Note that the cross-validation method is also used here. The accuracy of this model is around 69.31%.

The fourth model is using the GBT model to perform feature selection tasks and the GBT model to perform classification tasks. It follows the same procedure as the third model except that here when selecting features, the threshold is to be set as 0.05, and since those selected features contain some of the ‘category’ variables such as ‘category_Web’, hence, we need to include all the other ‘category’ variables because the feature importance of one category is estimated based on other categories. Finally, the accuracy of this model is around 72.99%

The fifth model is using the PCA technique to perform dimension reduction and feature selection tasks and still uses the GBT model to perform classification tasks. When building the PCA model, the number of components is determined by the number of features selected by the random forest model before. Then, we use the new predictors/components generated by PCA to train the GBT model. During this process, a cross-validation technique is applied to calculate the accuracy of the model. The accuracy of this model is around 69.78%.

Overall, the fourth model is the optimal one with the highest accuracy and best performance. Therefore, I selected the fourth model as my final classification model and performed hyperparameter tuning to make its performance even higher.

2.3 Hyperparameter tunning

Hyperparameter tuning techniques are performed on the final selected model to further increase the performance. The two main parameters of the GBT model: min_samples_split and n_estimators are being tunned within the grid that min_samples split ranges from 2 to 4 and min_samples_split ranges within 120,130,140,150. Then, the accuracy corresponding to each combination of these two parameters is stored as a list of tuples. A helper function is then implemented to return the left component of a tuple if its right component is the maximum value within the list of tuples. Finally, the optimal min_samples and n_estimators returned by the helper function are used to train the GBT model, this time, we can see that the accuracy of our final model increases from around 72.99% to around 73.34%.

III. Clustering Model

3.1 Data Cleaning & Preprocessing

To clean the data, firstly, I dropped the rows whose state is not ‘success ‘fail’. Secondly, I dropped the features which are meaningless for clustering, such as the project_id, name, and DateTime variables. Note that the state column is also dropped in this step to prevent the case that interpretation of the model is limited to happen. Lastly, I dropped the columns which mainly compose of missing variables as well as the rows which contain missing values. Note that here I did not drop those columns that are considered to be invalid for the classification model since this is not required here. To preprocess the data, I dummified all non-numerical variables such as country, currency.

3.2 Feature Selection & Model Constructing

I constructed two clustering models for this section, one of them has a high performance in the sense of silhouette score and another is more practical as it makes use of most of the data.

The first model is implemented using the predictors which perform well independently. To get these high-performance predictors, three helper function is used. The first helper function can convert a series object to a NumPy array with the shape that can be used for training clustering models. The second helper function can return the performance of a model trained by a given array. The third helper function will loop through all the features in the data and use each of them to train the KMeans model and store results in a list, finally, the result list will be sorted based on their silhouette scores. From the result list l_k2_sorted, we can see that the silhouette score starts decreasing significantly after the static_usd_rate feature. Hence, I selected all the features starting from the first one to static_usd_rate as our features. Finally, I standardized these chosen features, used them to fit the KMeans clustering model, and calculated the silhouette score, and centers of each cluster. As we can see, the silhouette score of the first model is around 0.9232, which is a relatively high number.

The second model uses all the predictors as it aims to make use of all the useful data on our hands to segment the projects. Hence, it standardized the preprocessed data and directly used it to fit the model. When fitting the model, I used the Elbow method to find the optimal k for the model, which is k=2. Finally, I used the standardized data and the optimal k to construct my KMeans model and calculated its silhouette score and centers of each cluster.

3.3 Interpretation of Clustering Result

From the result of the first model, we can see that projects that belong to the second center have a higher goal, longer create_to_launch_days, lower static_usd_rate, and way more backers, extremely higher money pledged and usd_pledged than the projects in the first cluster. This tells that the projects in the second cluster are mainly composed of those projects who set a higher goal at the beginning and spent more time doing advertising during the pre-launch process, and may then attract more followers, which later after the projects are launched, converted to its backers and eventually more money was raised for these projects. Besides these, we also can see those projects in the second cluster may come from those countries with a lower currency than the projects in the first cluster. However, note that since we did not include state when fitting the clustering model, so we cannot get the conclusion that the projects in the second cluster tend to be more successful than the projects in the first cluster.

From the result of the second model, since we want to include as much data as we can, there are 117 features for each cluster. Therefore, I will only explain some features that have a relatively more significant difference for the sake of space. As we can see from the centers of the second model, projects in the first cluster have relatively higher goals, more money pledged, longer name_len, lower blurb_len, earlier deadline_yr, earlier created_at_yr, and so on than the projects in the second cluster. This tells that the projects in the first cluster are ‘older’ than the projects in the second cluster in terms of their ‘birthday’, and projects in the first cluster have more money pledged in the end and higher goals in the beginning, their names are also longer but their blurbs are shorter than the projects in the second cluster. However, similarly, as we have excluded the variable ‘state’ when fitting the KMeans model, we cannot argue whether projects in the first cluster tend to be more successful than projects in the second cluster.

IV. Summary & Business Applications

The first model is powerful in generating business profits for Kickstarter website since Kickstarter website can promote a premium package for its users, those who purchased the premium package can use the function built in the first model to predict whether their project will succeed when they are about to launch their projects. This is a promising function since it allows project owners to know their chance to succeed before they launch so that if the chance is too low, they could stop launching, stay in the pre-launch stage and keep preparing, advertising and improving their projects until they are ready to launch. Moreover, the Kickstarter website can also use this model internally for selecting the ‘staff-pick’ projects. Although this does not bring direct revenue, it can improve the company's reputation as their decisions are more data-driven now.

The second model is meaningful for doing the market segmentation as it can segment the projects on Kickstarter website into several groups. Using the second model, Kickstarter can target a specific segment that is likely to be interested in their content or products. For instance, Kickstarter can target those projects with longer blurbs and offer them the paid blurb advising service just as the resume check service that LinkedIn uses. Kickstarter can also target the project owners who have ‘older’ projects and send them healthcare products advertisements if applicable, as this group of people might also be relatively older. By doing so, Kickstarter can reach out to its target audience more effectively and save huge costs. Without the help of our model, Kickstarter will advertise to the entire market, which will eventually cost a massive amount of money on ads, but the conversion rate will be relatively low.