Predicting the Effectiveness of Customers in Repurchase

Sheetal Mangesh Pandrekar
Stony Brook University
sheetalmangesh.pandrekar@stonybrook.edu

Shashank Jain
Stony Brook University
shashank.jain@stonybrook.edu

ABSTRACT

In this project we have tried to address the famous customer churn prediction problem. We took a Kaggle problem to help us with it [1]. We have provided details about the dataset, how we went about with feature selection, followed up by the problems we faced in the models tried by us and our possible approaches to solve them. We concluded by comparing our models with the best result from Kaggle, and a very simple baseline.

1 Problem definition

In this project, we attempt to solve the Kaggle problem of predicting customer repurchase behavior given an offer [1]. The data captures the process of offering incentives to a large number of customers. We need to predict which of these customers will purchase the same item again. Thus, our goal is to see how effective the incentives provided to the customers prove to be helpful in avoiding customer churn. This will help to decide if a certain offer should be continued or not.

2 Prior Work

As it is a project from Kaggle [1], different approaches include Logistic Regression, Naive Bayes, Random Forest and Neural Networks. The problem of customer churn prediction has been dealt with in different researches. [2] proposes an Improved Random Forest Algorithm by combining weighted and random balanced forests to predict customer churn. A Hybrid KNN-LR Classifier is proposed in [3] to address the problem of customer churn prediction. It compares the performance with Logistic Regression, C4.5 and RBF.

3 Dataset

Kaggle dataset had the following data files with it:-

1) Train and test files consisting of customer id, offers and other relevant parameters etc offered to them. Approximately 160000 rows each.

2) Offer file consisting of details about 38 offers.

3) Customer transactions of 22GB size over the past few years.

Initially, we started only with the transactions which are related to the offers i.e. the transactions which have the company, category and brand as any offer. This set of transactions is 4GB. We later included the additional feature by considering the entire 22GB of transactions.

4 Features

4.1 Choosing Features

The goal was to design features that capture the behavioral response of the customer when he is given an offer. We have a total of 68 features for each user. Each offer has a brand, category and company.

We can categorize the features as follows:

1) Quality of the offer, considering the quantity and amount of the offer.

2) Popularity of the brand, company and category associated with the offer.

3) How much the user prefers the brand, company and category associated with the offer. For this we consider the past transactions to get the number of transactions, amount spent, number of returns.

4) General purchase behavior of the user considering the transactions and spending on all products in the past 30, 60, 90, 120, 180 days.

Some of the features have been inspired from [1]. The new features that we have considered are: Number of returns made for each brand, company and category associated with the offer.

4.2 Feature Scaling

We plotted the histograms of the data in each feature and observed that the values have a high range (0 - 200000). Using the histogram, we divided the real data to categorical set.

5 Experiments and Results

We experimented with different models on the data. With every model, we analyze why the performance is not up to mark and choose the next model that can help to deal with the situation. Following are some of the key points common to all of our experiments:

1)We used StratifiedShuffleSplit of scikit-learn, and divided our training set into two parts namely train(80%)

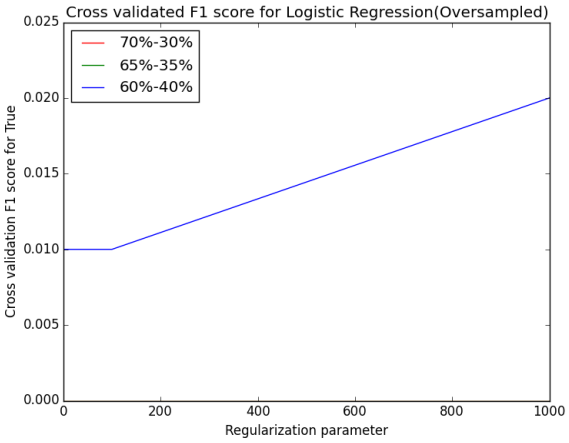
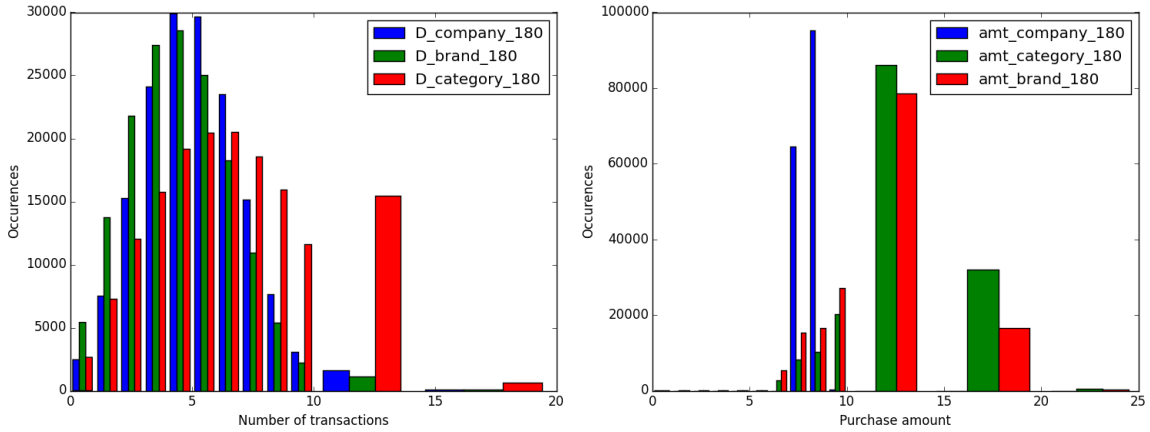


Figure 1: Cross validated(10 fold) F1 score for Logistic Regression(Oversampled)



(a) Distinct transactions of company, brand and category

(b) Purchase amount of company, brand and category

Figure 2: Features as Gaussian distribution

and eval set(20%).

2)It is observed that all models are performing well at predicting the False Class because of class imbalance problem. Therefore, we choose the F-1 score of the True class as our performance criteria.F1 score results have been calculated on the eval set.This helps to find out the best hyper-parameters for our model to try on test set.

3) We also keep a track of the AUC score for the best hyper parameters on Eval and Test Data as kaggle reports only AUC score on its test dataset.

4) We have shown our results using both initial features, and the one additional feature which we took from the entirety of dataset.

5) Our dataset had a class imbalance problem. Hence we oversampled by following three distributions in favor of majority class: (60%-40%),(65%-35%), and the original dataset of (70%-30%). We did not take extra samples by changing feature values, instead we took the above distribution randomly from our original dataset of majority and minority class.

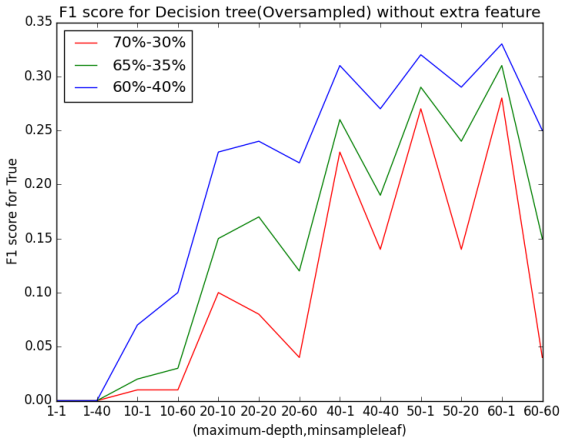
6 Baseline

The total training data is 160057 rows which consists of True Class (27%) and False Class (73%). Our baseline is always predicting the majority class. This gives an AUC score of 0.49905 on the test data on Kaggle. Our AUC score on evaluation data is 0.5

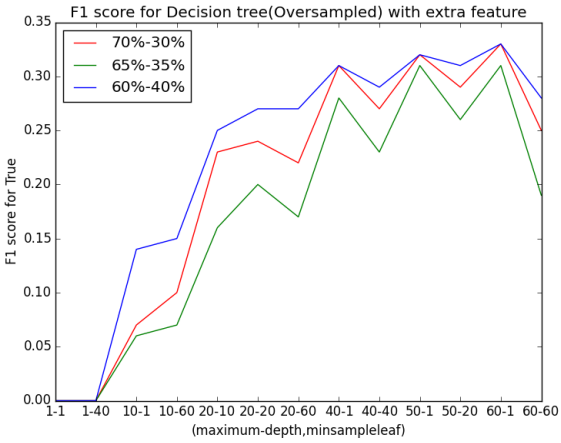
7 Models

7.1 Model1:Logistic Regression(LR)

Our motive for trying LR was to check if the data is linearly separable or not. The Figure1 shows the results for LR with L1 penalty for different values of regularization parameter (smaller value specifies stronger regularization). It is observed that for our initial data distribution of 73%:27%, the model gives a 0 F-1 score at predicting the True label. Thus, we tried different distributions of the data ie. 65%:35% and 60%:40% out of which the 65%:35% distribution also gives 0 F-1 score while 60%:40% gives non-zero F-1 score.

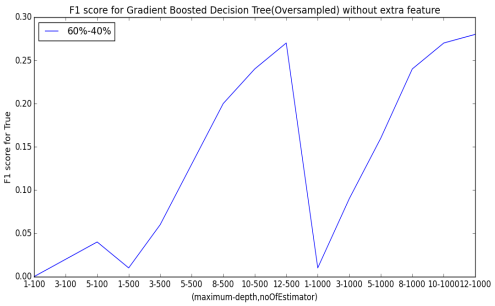


(a) Without extra feature

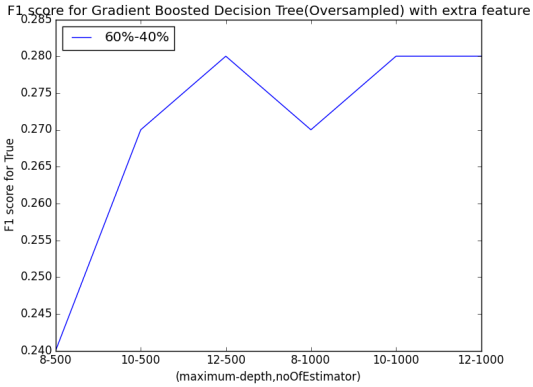


(b) With extra feature

Figure 3: F1 score(True) Decision Tree



(a) Without extra feature



(b) With extra feature

Figure 4: F1 score(True) Gbdt

The F-1 score obtained is poor for smaller values of regularization. Although the results improve slightly with larger value of regularization, it leads to learning a complex model which is not desirable. Thus, we can conclude that the data is not linearly separable. Also, we observed that the performance for oversampled data will be helpful in the further models that we try.

7.2 Model2:Gaussian Naive Bayes(GNB)

Figure2. shows the histogram plots of the data for some features. It is seen that most of the features follow a nearly Gaussian distribution. The results of GNB are presented in Table1.

The F-1 score does not improve even for different oversampled data. Hence, it was desirable to try a model that does not make strict assumptions of the features distribution.

7.3 Model3:Decision Tree

Decision Tree classifier was a natural choice since our features are categorical. A decision tree would be able to segregate the categories and learn to distinguish the two classes. Fig3a) and Fig3b) shows the results obtained with the original features and with the new feature respectively.

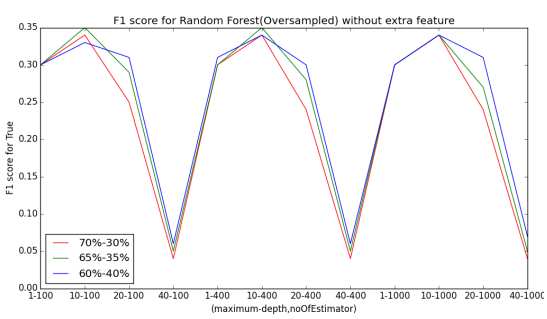
Distribution	F1-Score
70%-30%	0.20
65%-35%	0.22
60%-40%	0.23

Table 1: F1 score for True in GNB

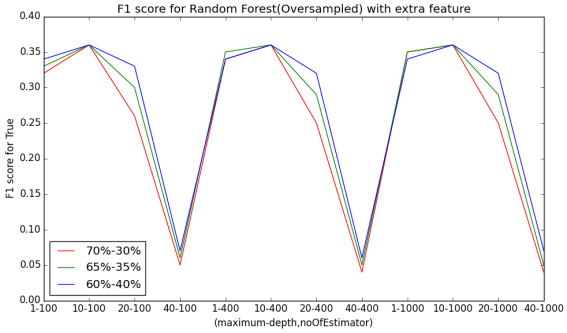
It is seen that the results improve with the new feature (the graph has fewer spikes) and also oversampling the data gives the best results. However, the best F-1 score of 0.33 is not acceptable and thus there is a need to try different ensemble classifiers.

7.4 Model4

One decision tree for the whole dataset may not learn well on all parts of the dataset. Therefore we move to trying two ensemble-based methods and the results are presented in this section.

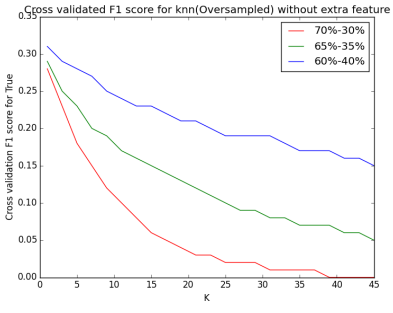


(a) Without extra feature

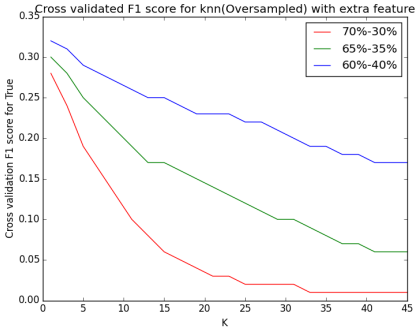


(b) With extra feature

Figure 5: F1 score(True) RandomForest



(a) Without extra feature



(b) With extra feature

Figure 6: Cross validated(10 fold) F1 score(True) KNN

7.4.1 Gradient Boosted Decision Tree (GBDT)

Initially the GBDT classifier was applied using some values of maximum depth and number of estimators and the results are shown in Figure4a. It is seen that good results are obtained for maximum depth greater than 8 and also for number of estimators greater than 500. Thus, a fine-grained search is performed with more values of maximum depth for larger number of estimators. The results of the fine-grained search are shown in Figure4b.

The best F-1 score is obtained for Maximum Depth of 12 and 1000 number of estimators. The best F-1 score is 0.285. However, this result is less than that obtained with the decision tree. Since, GBDT classifier has high variance, it is possible that it has overfitted the data.

7.4.2 Random Forest

After poor results are obtained with GBDT, we move to Random Forest classifier which has a low variance and thus a less chance of overfitting. The results of applying the Random forest classifier on the original dataset and the dataset with a new feature, for different oversampled data are shown in Fig5a. and Fig5b. respectively.

The Random Forest classifier gave an F-1 score of 0.35 which is better than that of the GBDT. Among the classifiers that we tried so far, the Random Forest gives us the best result. But this result is still not good enough. Thus, there is a need to try a different approach other than decision trees.

7.5 KNN

The figure9. shows the hex bin plot for two features in the True class and the False class respectively. We can see that the True/False are packed very closely together but the density of false is more. Hence for smaller values of k KNN classifier could perform well.

Fig6a. and Fig6b. shows the cross validation F-1 score by increasing the value of K on the three oversampled datasets.

It is observed that as K increases, the performance of the classifier drops. As there is not much difference between the feature values of the True and False class, and False class being more in density, the increase in value of K would always return False. Hence, the reason for performance drop. From this we can infer that in order to distinguish the True class from the False class, it is important to look beyond just the nearest neighbors while performing the classification.

7.6 Clustering

The inference from the KNN classifier, motivated us to consider the fact that there will exist patterns in the transaction behaviors of the customers. To observe the pattern, we formed clusters of the

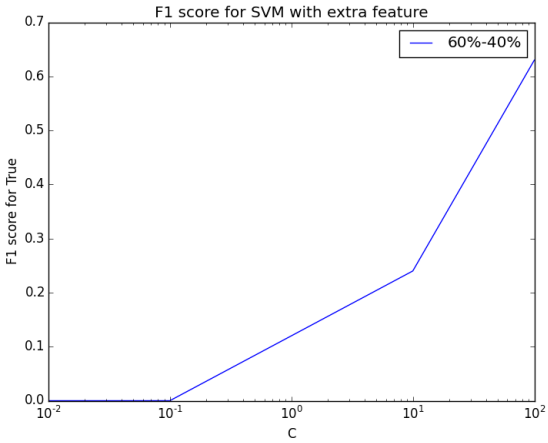


Figure 7: F1 score for svm(Oversampled)

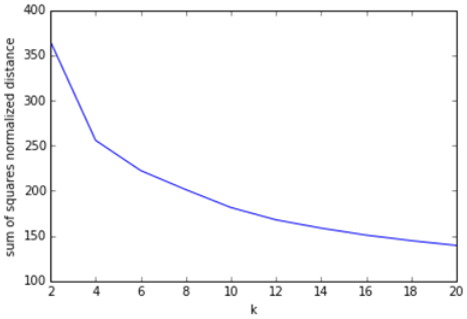


Figure 8: Knee in K-means

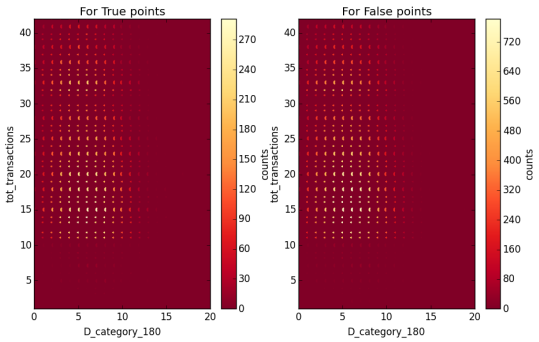


Figure 9: Hexbin of some features for True/False

customers depending on their behavioral patterns. Figure8. shows the plot of K versus the Sum of distances of samples to their closest cluster center(inertia). We choose the knee as k =12.Thus, our dataset contains 12 different behavioral patterns of customers.

We drew a histogram plot of various features of clusters which are shown in Figure10,11,12.In each of the figures: x axis represents the number of transactions done by the customer for respective company, category and brand respectively. And y axis is simply the count for x axis. Our analysis of the first two clusters is that customers in Cluster0 tend to buy more from same brand and company a lot more than Cluster1. However, they don't buy same kind of product every time which shows the low count of categories in the histogram. Hence, customers in Cluster 0 are more loyal to company, brand. But they would like to try different categories of product rather than buying the same thing every time.

After the clusters have been formed, our task is to perform classification on each cluster. For a point to be classified, we find its nearest cluster by checking its distance from the means of the 12 clusters. Two different classification methods are tried, and the results are outlined below.

7.6.1 Clustering followed by Decision Tree

As the feature values for the True and False class are not very distinguishable in each class, the decision tree is not able to perform well and gave similar performance when applied on the whole data set. Decision tree with clustering result is shown in Table2.

7.6.2 Clustering followed by Support Vector Machines

As decision tree failed so we thought of trying an approach to separate the feature values for the True and False class at higher dimensions. Hence, SVM with rbf kernel was trained for each cluster. The best F-1 score of 0.63 was obtained by using SVM within clusters for C=100. We could not try higher values of C because it slowed down our system. For completion We also tried to run svm on the whole dataset and have reported the results in Table2.

8 Summary of Results

The table2 presents the summary of the results of all the approaches that were tried. For each method we provide the AUC score on evaluation data and test data.

The best result on eval set was obtained with SVM combined with clustering. This result was

Model	Eval	Test
DecisionTree	0.5153	0.50164
GradientBoosting	0.5116	0.50202
Random Forest	0.50703	0.50062
KNN	0.50703	0.49908
Clustering+DecisonTree	0.5126	0.50034
Clustering+SVM	0.5315	0.50171
SVM	0.5081	0.5031
Baseline	0.5	0.49905
Kaggles best		0.627

Table 2: AUC score of our model on eval and test set

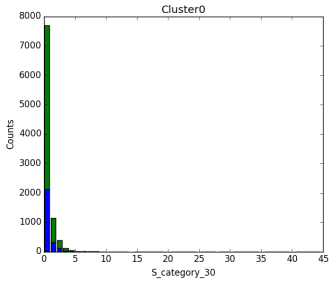
even better than a single SVM trained over the entire dataset. However, both of these reported similar scores when computing auc scores on the test set.

9 Conclusion

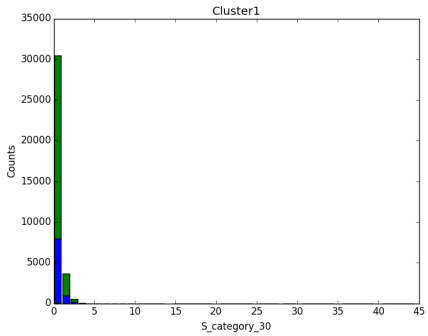
The Kaggle problem of predicting if a customer would repurchase the same item given an offer was tackled by using different approaches. With every approach, an analysis is provided about why the approach would have failed and how a new approach is motivated. Logistic Regression, Decision Tree, Random Forest and Gradient boosted Decision Tree were already tried on Kaggle. Gaussian Naive Bayes to check whether features follow some distribution, KNN classifier and Clustering followed by Classification(SVM and decision trees) were new approaches that we introduced. Also, we observed the results for different distributions of the data, which was not done on Kaggle. Oversampling ratio of 60:40 gave us the best results.

10 References

1) https://www.kaggle.com/c/acquire-valued-shoppers-challenge
2) Customer churn prediction using improved balanced random forests: Xie,Li,Ngai, Ying
3) A Hybrid KNN-LR Classifier and its Application in Customer Churn Prediction, Yangming Zhang, Jiayin Qi, Huaying Shu, Jiantong Cao

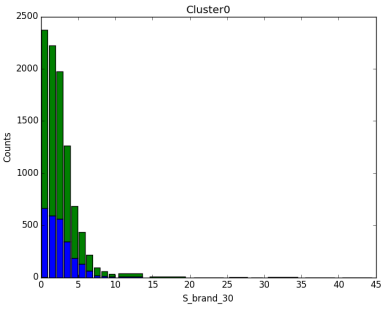


(a) Cluster0

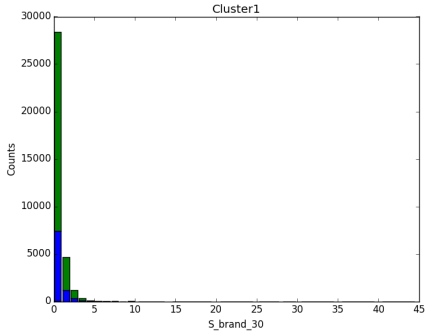


(b) Cluster1

Figure 10: Distribution for same category in past 30 days(True=Blue,False=Green)

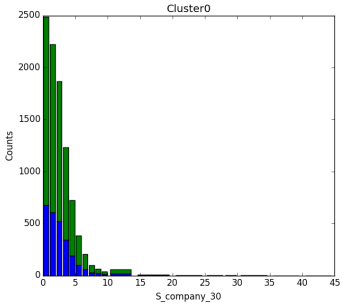


(a) Cluster0

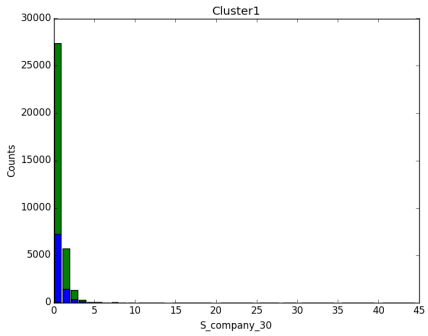


(b) Cluster1

Figure 11: Distribution for same brand in past 30 days(True=Blue,False=Green)



(a) Cluster0



(b) Cluster1

Figure 12: Distribution for same company in past 30 days(True=Blue,False=Green)