CMPE 239 Project II: *Prediction of Consumer Disputes about Financial Complaints Response*

Team 2: Fengmei Liu, Xiaoyan Chong, Weiqian Hou

Ch.1 Introduction

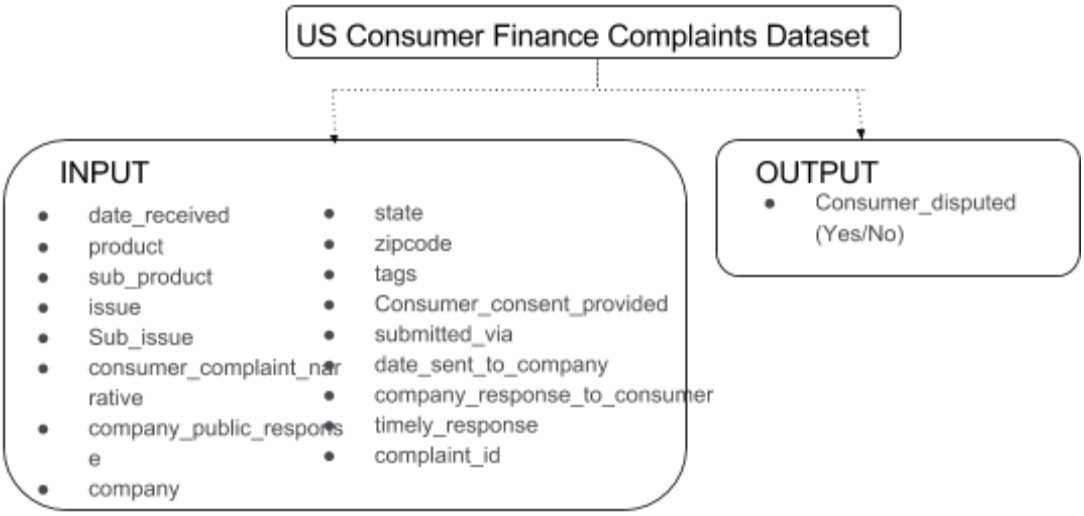
I. Motivation

We are trying to solve the problem from Kaggle dataset "US Consumer Finance Complaints". Background knowledge are from Kaggle's description, "Each week the Consumer Financial Protection Bureau (CFPB) sends thousands of consumers' complaints about financial products and services to companies for response. Those complaints are published here after the company responds or after 15 days. By adding their voice, consumer help improve the financial marketplace." As we choose Classification as our project scope, this can be a suitable dataset to explore and practice.

II. Objective

Data is confined to be in year 2016 with 50853 items, every row represents a customer complaint. Each column describe one feature of a specific complaint item. And we treat every variable to be a categorical variable. See below for the detail input variables and our selected output.

We choose the **Consumer Disputed** as the classification objective. The classification problem at hand is to figure out whether the consumer with dispute with the complaint response or not from a knowledge of complaint patterns. The result could be used as a reference for companies to understand their customer service, meanwhile, guide their customers to follow the correct way of feedback in order to get complains successively solved.



Ch.2 System Design & Implementation details

I. Methods

We use programming language R to clean the data and to perform the three classification algorithms.

II. Algorithms

For the reason of categorical variables, we exclude such classifiers like KNN, linear regression etc.. Logistic Regression, Naive Bayes and decision tree classifiers are selected and implemented in this case.

A. Logistic regression

Logistic regression is a model when the dependent variable is categorical. When the dependent variable has two levels, we call it a binary logistic model, and it is used to estimate the probability of a binary response based on the independent variables. This estimation of probability is fulfilled by using a logistic function. In the introduction part, we know the response in our project is "consumer disputed" and it is a binary response with two categories "Yes" and "No". Besides, all the predictors are categorical variables with different levels. Many algorithms are applicable to this case, and logistic regression is one of the most popular algorithms for dealing with this kind of problem.

B. Naive Bayes/Tree Augmented Naive Bayes

Naive Bayes is always a first try when we try to address the classification problem with categorical variables. It is easy to implement and fast, and has the assumption that classes are conditional independent. Most applications can get good prediction result although sometimes there's violation of the assumption. A Tree Augmented Naive Bayes model (TAN) imposes a tree structure on the Naive Bayes model, by allowing correlation between variables and restricting the interaction among the variables to a single level. It can significantly improve the accuracy as proved. We tried packages in R to implement Naive Bayes/TAN, including library e1071 and klaR, caret, bnlearn.

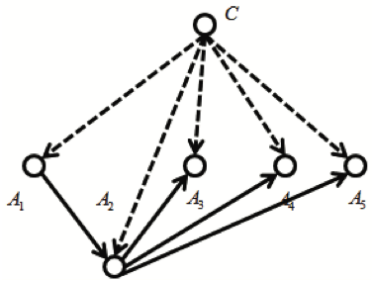


Fig Tree Augmented Naive Bayes

C. Decision tree

Since all the variables in this dataset are categorical data, the classification tree algorithm might fit for this case very well. We tried bagging, random forest, and boosting methods to reduce the instability. However, the bagging method is way too slow, and random forest has the limitation of 53 categories in each variable. The training data whose dimension is 20063 x 15 is a relative large one, and some features has more than 60 categories. Finally, we have to give up the bagging and random forest methods and turn to the boosting. Boosting is an ensemble algorithm to reduce the bias. In each iteration, it fits a tree model based on the last tree, that is, giving more weight to the misclassified points in the last run. In this project, we would use gradient boosting (gbm package of R). To avoid overfitting, we will adjust two parameters (shrinkage and n.trees) to get the best accuracy rate and show the process of finding the best parameters in experiments.

III. Process Flow

Below diagram shows the workflow of the whole system.

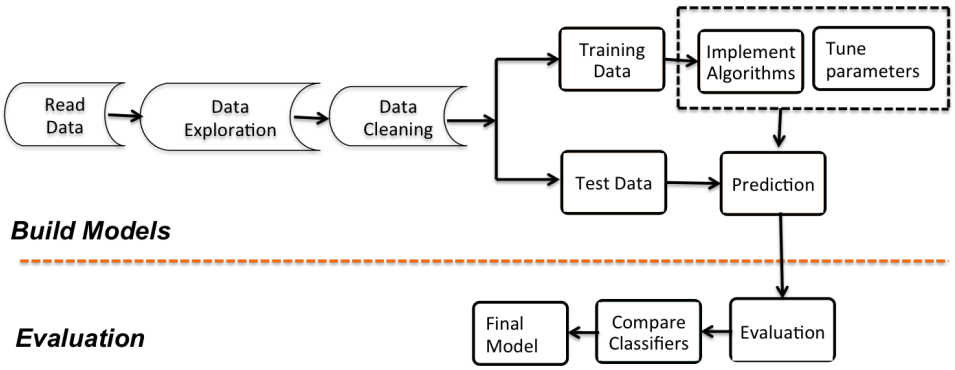


Fig Process Flow

Ch.3 Experiments / Proof of concept evaluation

I. Dataset description and preprocessing

This dataset is called "US Consumer Finance Complaints" which is downloaded from Kaggle, and we pick "Consumer Disputed" as the response. It has two categories "Yes" and "No". The raw dataset has 50,853 observations and 18 variables. During the analysis, we decide to preprocess it in order to let the model make more sense, and get a size of 30,095 by 15 including 1 response an 14 predictors in the end. When applying the three algorithms, we divide this dataset into two parts: 2/3 as training set (20,063 by 15) for building models and 1/3 as testing set (10,032 by 15) for evaluation. The details of preprocessing are as follows.

Step 1: Delete columns including information of ID, zipcode, date received complaints.

Step 2: Make all the missing values as a single category for each variable since we believes the

missing parts are also very informative in our case.

Step 3: Delete observations which includes non-existing 'State'.

Step 4: Make the column 'date_sent_to_company' four levels according to month.

Step 5: Based on observations for each company (over 1800 companies in total), we select observations including top-20 companies only, since a large amount of the rest companies only have less than 10 observations.

Step 6: Delete observations based on three predictors: issue, sub-issue, and sub_product. They have too many levels (almost 100), but many of the levels only include single number of observations which make building models and evaluation more difficult. We delete these rows which is about 300 in total. Till this step, our data size is 30,095 by 15.

Step 7: Divide the dataset into training set and testing set randomly. In the end, we have 20,063 by 15 for training models and 10,032 by 15 for testing.

II. Logistic Regression

A. Experiment of logistic regression

In this part, we are trying to use training dataset to build a model based on logistic regression algorithm. When performing logistic regression, we assume the error distribution is binomial. The dependent variable "Consumer Disputed" and all the independent variables are treated as factors with different levels.

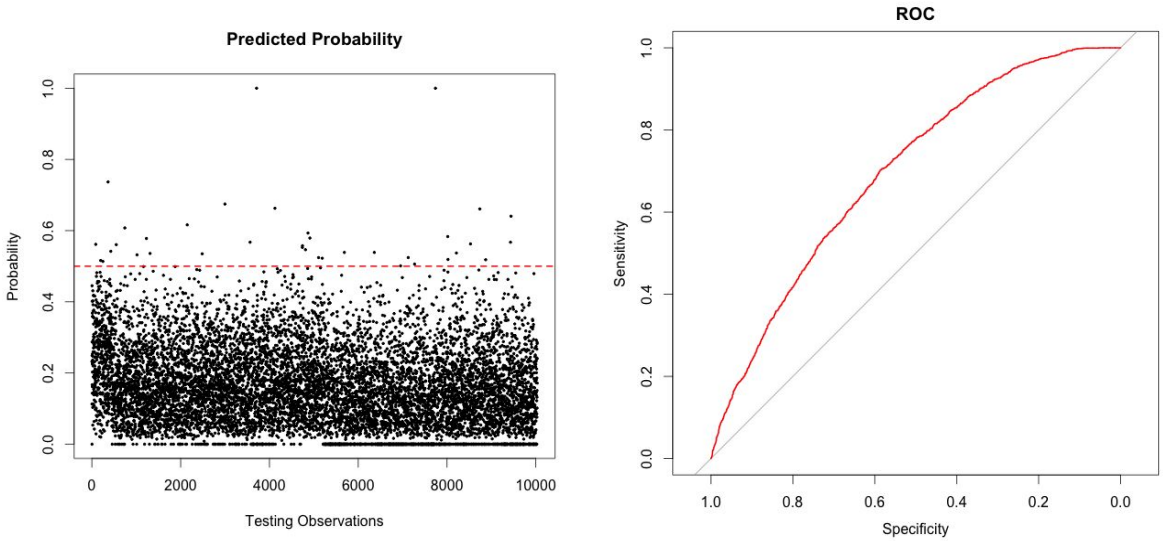
The model building process is easy since there is no parameters need to tune. We only need to specify the response, predictors, and also the distribution of error we plan to use. By simply used R's base package, we can have a result logistic regression based on training dataset.

Since each variable has many levels, and we have many independent variables, the output model is huge and we decide not to report the final model with details. Instead, we would like to evaluate the model on the testing dataset.

B. Evaluation

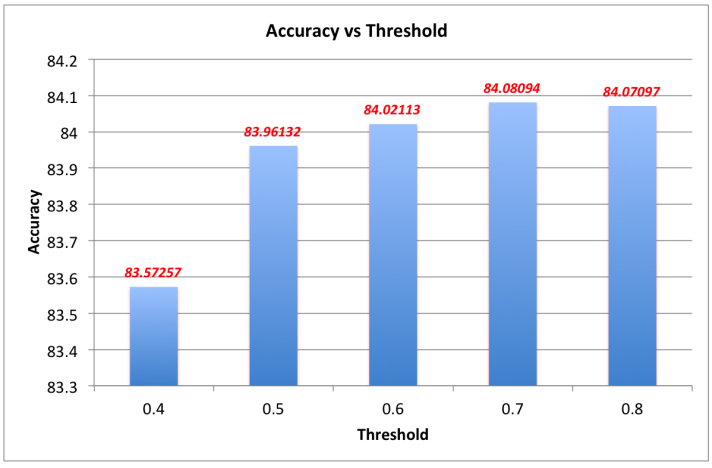
In the section, we conduct the evaluation process on testing dataset. Since we know that logistic model can predict the probability of a binary response based on the independent variables. The below left figure shows the predicted probabilities of each observation of testing dataset. The below right figure is the ROC curve based on predicting result. By R, we calculated the area under the curve is 0.6963. The result seems not very promising. There are many reasons lead to this, one of the biggest issue is that in the original dataset, the ratio of the two categories of our response is about 5:1 ("No" to "Yes"). This will make the

prediction for “No” with high accuracy and “Yes” with very low accuracy. As a result, the overall prediction is not very promising and the area under the ROC curve is also very low.



In order to make the prediction result more straightforward, we decide to calculate the accuracies. As we know, the prediction outputs are probabilities, so we need to pick a threshold for the classification purpose. In practice, “0.5” is usually selected for convenience. We can also make selection on the threshold based on our own case. The figure below show that the relation between the thresholds we choose and the accuracies. Based on the figure, we find that in our experiment of choosing different threshold, the number of 0.7 give the highest accuracy on our testing data.

Finally, we conclude based on logistic regression method, the best accuracy we get is 84.08%. The corresponding confusion matrix is shown in the table below.



III. Naive Bayes

A. Simple Naive Bayes

Use the same training dataset and test dataset, the simple naive bayes implemented in R using library e1071 and klaR. Choose the 10-fold cross validation method to get the best model.
Result: Prediction Accuracy: **0.763** (by package e1071,klaR), **0.794**(by package bnlearn)

B. Naive Bayes With Laplace smoothing

10-fold cross validation are used to choose best model. The Laplace smoothing parameters tested are 0.001, 0.1, 0.5, 1, and 2.
Result: Prediction Accuracy: **0.770 ~ 0.787**

C. Tree Augmented Naive Bayes

The model is implemented using the bnlearn package, using default parameter setting.
Result: Prediction Accuracy: **0.865**

TAN_PRED	NO	YES
NO	8178	1094
YES	256	504

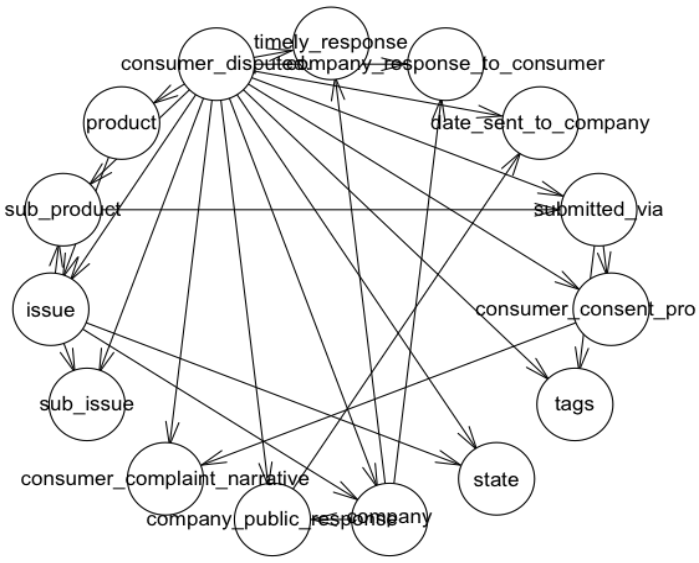


Fig Plot of TAN Network

Analysis of the Naive Bayes result:
Between the various Naive Bayes models, we can see that the TAN gives the best prediction result, and the smallest confidence interval. Another advantage of TAN is that we can see the main correlation between input variables. Although most of the time Laplace smoothing is useful, but here because our test data are almost have no new categories, so it's no major improvement by adding the smoothing.

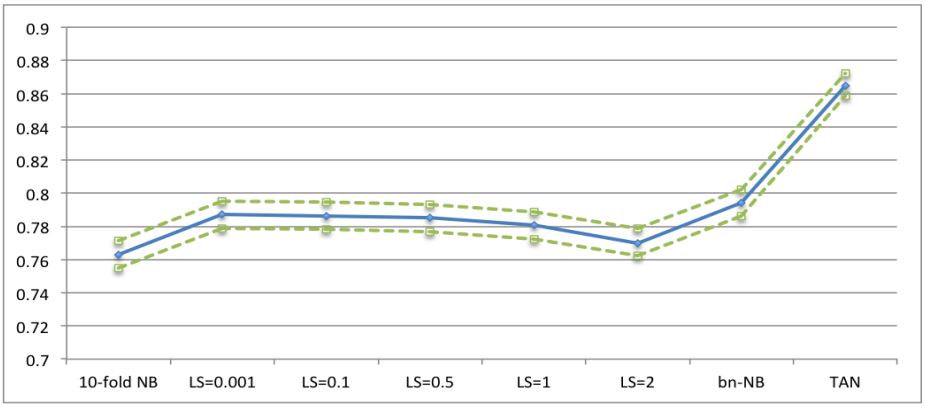
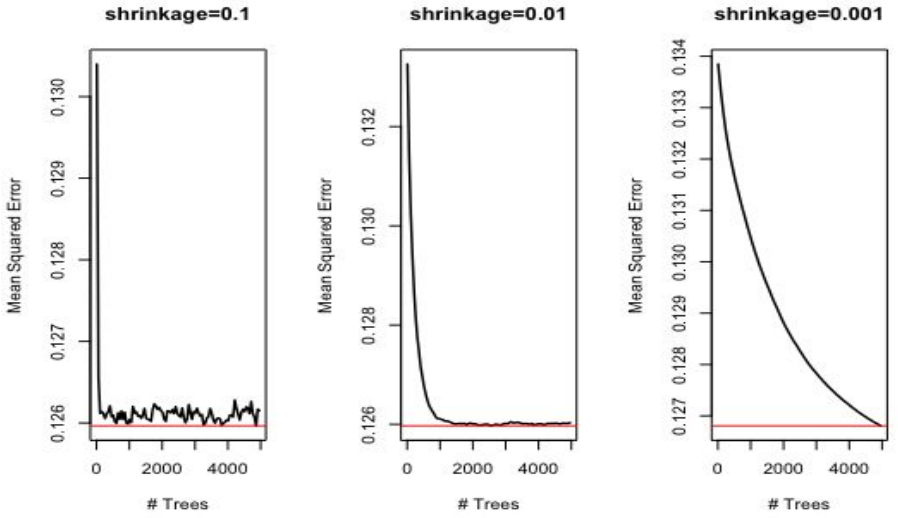


Fig Comparison of different Naive Bayes Accuracy

IV. Decision Tree (Gradient Boosting)

A. Experiments for finding the optimal shrinkage

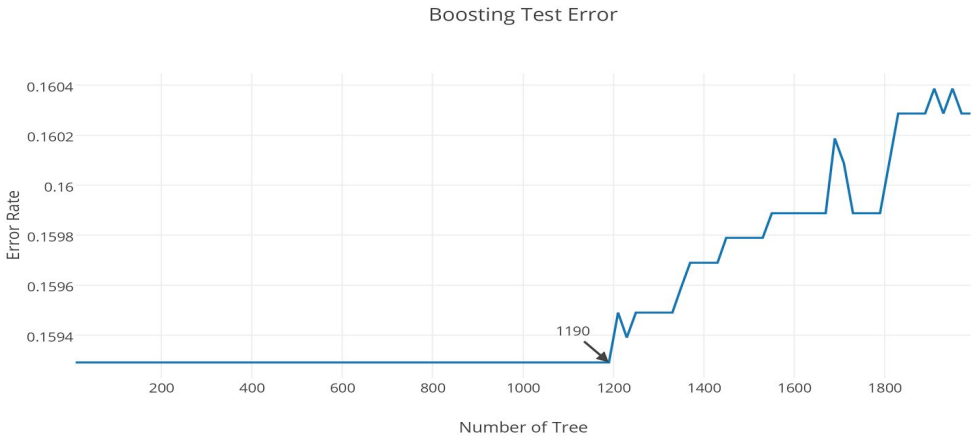
Gradient boosting can be seen as a gradient descent algorithm or an optimization algorithm on a cost function. Its goal is to minimize the mean square error. There is a gbm package in R can be used. The first thing we need to do is choosing a suitable shrinkage value. It is like the length of a step. If a step is too large, it will be hard to converge; however, if it is too small, it will take very long time to converge. We tried different value of shrinkage: 0.1, 0.01 and 0.001 when fitting 5000 tree models and test them using the test dataset.



We find the mean square error (MSE) going down super fast when shrinkage = 0.1, the large fluctuation on the plot shows it is hard to converge to the minimum. As for shrinkage = 0.001, it takes 5000 iterations to reach the minimum. We will use shrinkage 0.01 which make MSE converge around 1200-1300 iterations. So we would like to set n.trees = 2000 in next experiment to find the optimal number of trees.

B. Experiments for finding the optimal number of trees

The response in this case is binary (0 for "No", and 1 for "Yes"). Thus, it is meaningless to use MSE to evaluate the performance of classification. Calculating error rate or accuracy makes more sense.



The plot shows that the error rate goes up after 1190 iterations because of overfitting. It is surprising that the error rate is same from 1 to 1190 iterations, while the MSE is changing within that period. The reason is the predicted value is a probability which between 0 and 1. Most of them are below 0.5, which are converted to 0 and compared with true label when we calculate the error rate. If the number of probability less than 0.5 is same for different iterations, the error rate will be the same. However, MSE measures the distance between the predicted value and response value (0, 1) which will cause a difference.

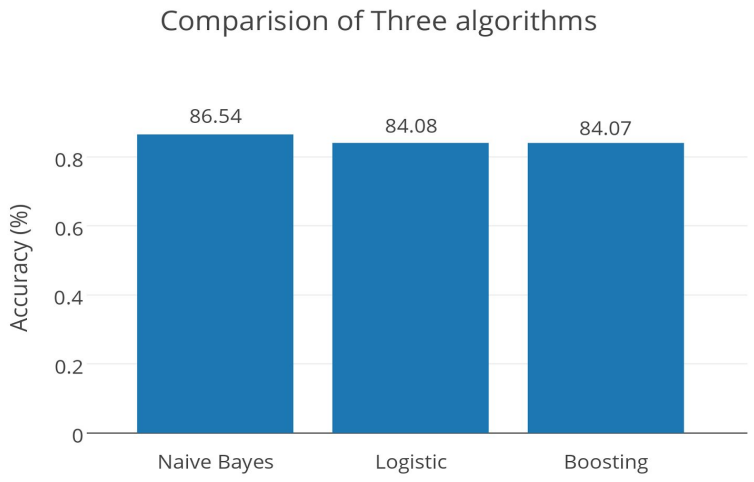
C. Evaluation

Since the error rate is same within 1190 iterations, we can select any n.trees less than 1190 to calculate the accuracy rate which is 0.8407.

Ch.4 Discussion & Conclusions

I. Comparison of three algorithms

In this part, we want to compare the performance of three algorithms: Logistic regression, Tree Augmented Naive Bayes, and Gradient Boosting Tree. The bar chart below shows that Tree Augmented Naive Bayes gives us the most accurate predicted label for test data. The results of logistic regression and gradient boosting method are also good and similar. Their difference of accuracy is only 0.01%.



II. Discussion

A. Logistic Regression

Pros: This method is really fast and is intrinsically simple, it has low variance and so it is less prone to overfitting.

Cons: Doesn't handle large number of categorical features/variables well. But in our case, it performs well. The only thing is since it is simple and straightforward, there is no parameters needed to tune when fitting the model, so it does not give us much space to improve the final predicting accuracy.

B. Naive Bayes

Difficulties: There are many Naive Bayes libraries to try and challenge is to learn and tune the best parameters for each one.

Things that worked: The Naive Bayes and TAN models can work well and efficient. K-fold cross validation can be applied.

Things didn't work: Laplace Smoothing didn't work well here for this dataset split. Didn't figure out how to tune TAN parameters to make it work better.

C. Gradient Boosting

Pros: 1. Gradient boosting tree is a very good model which yields an accuracy classifier.

2. It is faster than other boosting methods such as adaptive boosting.

Cons: 1. To get good results, we need to tune the model with these parameters. Usually, it is hard to get the optimal values for all the parameters.

2. Like some other boosting methods, it is easy to get overfitting.

3. It is not very speedy.

III. Conclusion

After comparing the three classification algorithms, we would like to choose the Tree Augmented Naive Bayes as the classifier for this case because of its nice accuracy rate.

Ch.5 Project Plan / Task Distribution

I. Project Plan

A. Discuss the topic and choose a suitable dataset together.

B. Everyone try several algorithms and figure out a better one.

C. Work together on final report and slides, and each person is in charge of one algorithm.

II. Task Distribution

Fengmei: Introduction and Naive Bayes/TAN Bayes algorithm part.

Xiaoyan: Data processing and Logistic Regression algorithm part.

Weiqian: Conclusion and Gradient Boosting Tree algorithm part.