STAT 641 Final Project
— Credit Card Customers Prediction (Classification)
— Car Insurance Claim Amount Prediction (Prediction)

Bohan Shen(30157005) Yonghan Qiu(30158194)
Mengzhao Xu(30189444) Jinghui Lu(30157954)

2022-12-14

Contents

Part 1: Credit Card Customers Prediction (Classification Method)

Introduction

Credit card has served merchants and individuals since its inception. It is not only a payment but also an auxiliary bookkeeping tool. Compared with traditional cash, it is safer and more convenient. In addition, it can build up personal credit history and stimulate transaction through cashback. But at the same time, the credit card with prepaid function has increased people's overconsumption behavior. If the person owes money to the bank for a long time, he/she will not only lose his/her personal credit score, but also affect the financial security of the bank.

As the pandemic spreads and the economy recovers, people have more ways to use their credit cards in daily business. Meantime, banks are trying to incentivize people to use credit cards with a series of initiatives. On the contrary, many people lost their jobs during the epidemic. They were unable to repay the debt they owed before, so they finally chose to give up using credit cards.

Statement of the problem

The purpose of this report is to help bank managers predict what types of customers will choose to continue using credit cards or customers with what characteristics will cancel the credit card service, so as to provide better service and quality for customers who use credit cards and to ensure the safety of funds.

Motivation

1 Fit interpretable models (Logistic regression, LDA, QDA, KNN, SVM, Classification tree, Bagging, Random forest, Boosting, and BART) and use validation set to evaluate the prediction.

2 Examine if the models are sufficient and select the best model by using the confusion

matrix (Accuracy and Sensitivity).

3 All the predictors have been used except the first (Clientnum) and last two (Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_

Difficulties in Dealing with Dataset

- The dataset contains missing values that may affect the final predictions.
- The variable Credit_limit has high correlation with the variable Ave_Open_To_Buy which may cause multicollinearity.
- The dataset does not indicate a release date, so it may not be possible to predict the real situation.

Statistical analysis

Preprocessing

The dataset contains 10127 data with 23 columns. We treated Attrition_flag as a response variable and deleted first and last two columns since the first column is customer ID and the last two columns are the predictions made by author which are not our consideration. The response has two categories (one is existing customer, one is attrited customer), we thus used classification methods to predict the results. Note that some of the variables in the dataset are characters, we first converted them to factor variables. To make the work reproducible, we set the random seed in the R code using set.seed(2022) and used the sample() function to split the dataset into a training set with 5064 data and 20 columns and a test set with 5063 data and 20 columns.

The dataset contains missing values, we firstly treat missing values as one category in each column and use all of them to fit the models. Then we removed these missing values and fit models again to find out if the accuracy will improve.

In real situation, some questions may ask private information and people may not want to provide. For example, when people buy insurance or open bank account, the website

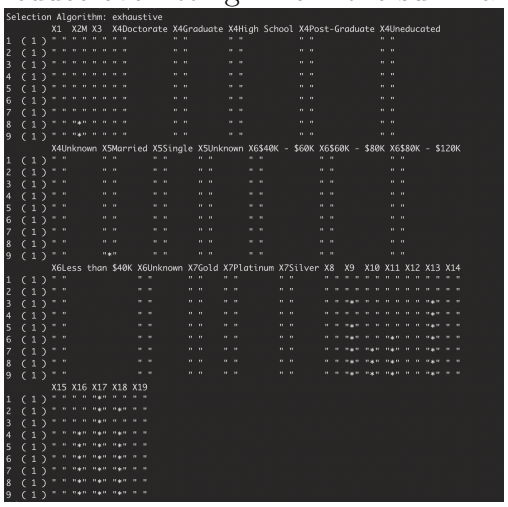
always has the option “Help Me Choose”, whenever we click it, we will be asked some questions. Some questions may be sensitive for some group of people, for example the income or the age. If people prefer not to say, we still hope the model can roughly help customers to predict the type of products that they need, so keeping missing values is necessary in the dataset.

logistic regression

Logistic regression is a statistical analysis method to predict the probability of an event occurring and the response variable is bounded between 0 and 1.

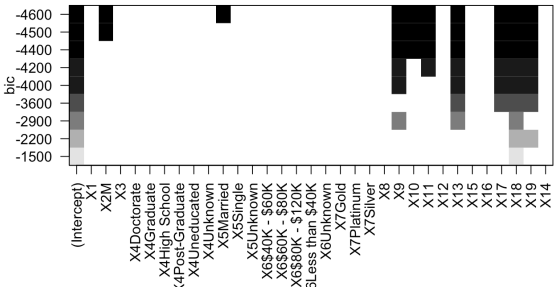
We used glm() function to fit a logistic regression model on training set with the argument family = binomial. We then used predict() function to predict the probability that if the customer will continue using credit card or not on test set. The type = ‘response’ will yield the probabilities of the form P(Y=1|X), and we set the decision rule to 0.5 by using ifelse() function, that is if the predicted probability is less than 0.5, it is classified as the attrited customer, if the predicted probability is larger or equal to 0.5, it is classified as the existing customer. Finally we used the table() function to generate the confusion matrix to determine 89.61% of observations were correctly classified, and the test error is 1 - 0.8961 = 0.1039.

In order to find the best model, we performed the best subset selection with regsubsets() function along with AIC and BIC, it is able to simplify the model, shorten the training time, and reduce overfitting. From the summary table as shown below:

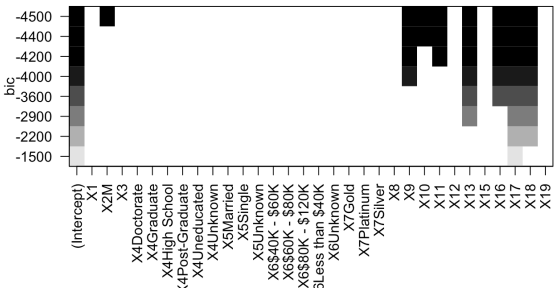


x_1	x_2	x_3	x_4
Attrition_Flag	Customer_Age	Gender	Dependent_count
x_5	x_6	x_7	x_8
Education_Level	Marital_Status	Income_Category	Card_Category
x_9	x_10	x_11	x_12
Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_m
x_13	x_14	x_15	x_16
Credit_Limit	Total_Revolving_Bal	Avg_Open_To_Buy	Total_Amt_Chng_Q4
x_17	x_18	x_19	x_20
Total_Trans_Amt	Total_Trans_Ct	Total_Ct_Chng_Q4_Q1	Avg_Utilization_Rat

We found that regsubsets() function selects X2, X5, X9, X10, X11, X13, X16, X17, X18 as important variables. While from the plots, both forward stepwise selection and backward stepwise selection obtain the same results as those in best subset selection, they select X2, X5, X9, X10, X11, X13, X17, X18, X19 as important variables, this may not be reasonable. Take BIC plot as an example shown below:



From the plot, we saw all of the variables list orderly except X14 which comes after X19 meaning that X14 has a high correlation with other variables. In order to solve this problem we use cor() function to test each predictor and found the correlation between X12 and X14 is 0.996, and removing either one would solve the problem. We then dropped X14 which corresponds to 15th variable in the dataset, since we set X1 as response variable, which caused each variable to be shifted backwards by one unit. Repeating the steps above, the plots marks X2, X5, X9, X10, X11, X13, X16, X17, X18 as important variables which is consistent with the summary table. Take BIC plot as an example shown below:



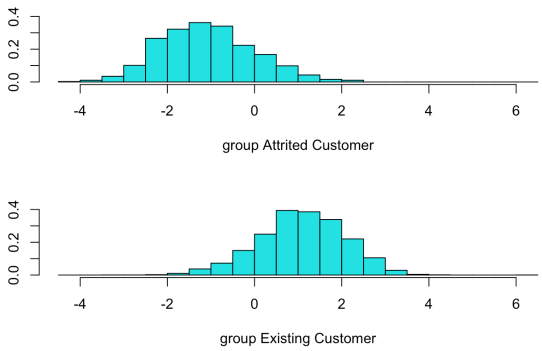
By using the variables selected from the best model selection to generate a logistic model,

the same procedure as we did in glm, the prediction accuracy is 0.8969 which improves slightly compare to the model without doing subset selection.

LDA and QDA

LDA is based on Bayes's theorem, assuming the distribution of predictors X follows normal distribution with the different means and the same variance. We used lda() function to fit the model, but r outputs an error message indicating rank deficiency. This is due to the high correlation between X12 and X14 in the original data set, we, therefore, removed the 15th variable and compute the confusion matrix, the prediction accuracy is 0.8959 which means 89.59% of observations were correctly classified, it's very close to the result from logistic regression.

The decision boundary is defined as $w^T x + b = 0$, where w is the coefficient which computed by coef(lda()) function and x are the predictors, if the output of equation $w^T x + b$ is large than 0, it will classify as an existing customer and vice versa. It is consistent with the plot shown below with plot() function:



0 is the decision boundary in the plot, when the number falls on the left, it will be classified into the attrited group, if it falls on the right it will be classified into the existing group, and the overlapping area in the middle is the number of misclassifications.

The idea of QDA is similar to LDA, but it assumes normal distribution with the different means and variances. The prediction accuracy is 0.8987 which means 89.87% of observations were correctly classified, it's very close to the result from LDA. The test error therefore is 1 - 0.8987 = 0.1013.

KNN

KNN is nonlinear model with featurization and supervised learning, it will predict accourding to a "majority vote" among training set. Before performing KNN, we need to change the class of response variable from factor to binary. The model is then generated using the knn() function, by setting k = 1, 5, and 100 where K controls the number of nearest neighbors to be used by the classifier, the corresponding prediction accuracy is 0.8681, 0.8874, and 0.8637. We noticed that the result improves slightly from k = 1 to k = 5, but increasing k further turns out to provide no further improvements.

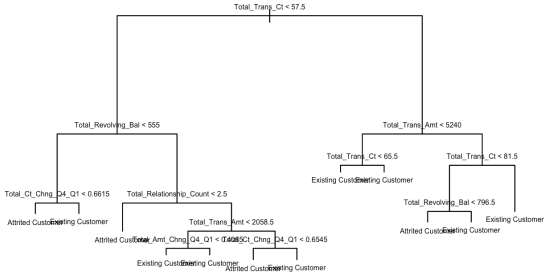
SVM

We used tune() function to perform ten-fold cross-validation on training set by default and set kernel = 'linear' means the decision boundary is linear since all the variables are degree 1. By setting a range of values of the cost parameter, the attribute of best.model will yield the best model along with the best hyperparameter C where C controls how heavily the slackness is penalized compared to maximizing the margin. Large value of parameter C leads to small margin, and vice versa. The summary table tells when C = 10, it gets the best result, the prediction accuracy is 0.8985 which means 89.85% of observations were correctly classified, the test error therefore is 1 - 0.8981 = 0.1015.

Classification tree

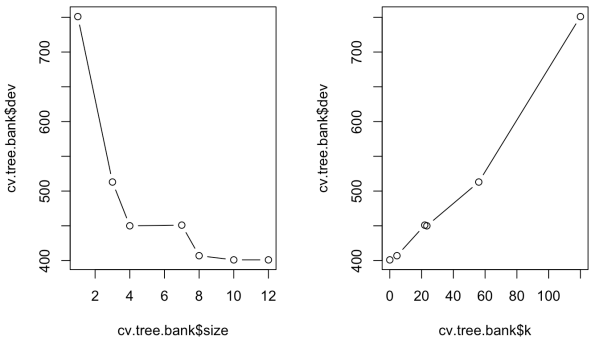
A decision tree is a supervised learning algorithm that is used for classification and regression model.

we used default settings to fit the tree model, by viewing the summary table, r chooses the best nodes as 12 that is consistent with the plot shown below:



Then we used predict() function on test set and confusion matrix to compare the predicted and actual results, the prediction accuracy is 0.9111 which means 91.11% of observations were correctly classified, the test error therefore is 1 - 0.9111 = 0.0889.

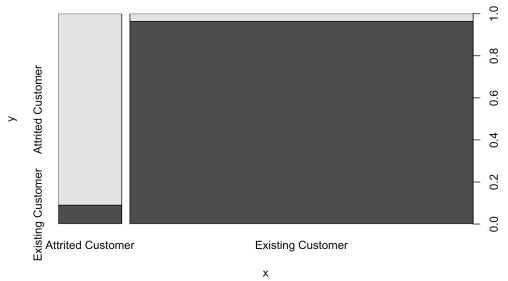
Next, we consider whether pruning the tree might lead to improved results. We used cv.tree() function to perform the cross-validation to determine the optimal level of tree complexity, and the argument FUN = prune.misclass to display the classification error rate to guide the CV and pruning process. The function will show the number of terminal nodes of each tree, the corresponding error rate, and the value of k. The plot shown below:



From the result, when tree has 10 or 12 terminal nodes, both results yield a minimum of 394 cross-validation errors. By using prune.misclass() function and setting best = 10, the model shows the exactly same prediction accuracy.

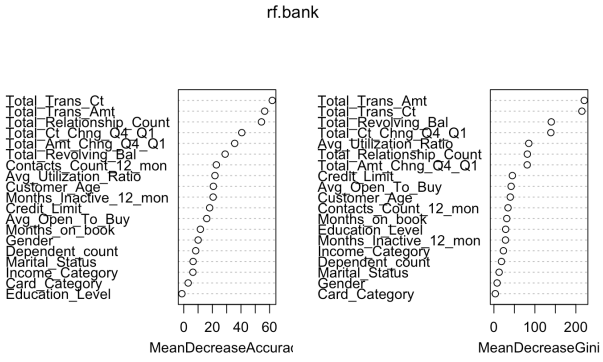
Bagging and Random forest

Bagging takes samples from observed dataset with replacement and it is a special case of a random forest with m = p where p is the number of features, so by setting mtry = 18 indicates 18 predictor should be considered for each split of the tree. The prediction accuracy is 0.9565 which improves a lot compared to the previous algorithms.



From the plot above, one can see the majority of the data is classified correctly.

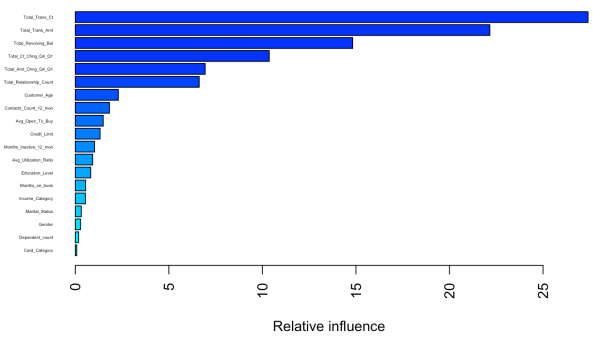
Random forest requires the smaller value of features. In the code, we skipped this parameter, which means by default, r automatically uses the sqrt(18) variables when building a random forest. The prediction accuracy is 0.9550 that is quite similar to what we got with bagging. By measuring the importance of variables with the varImpPlot() function, the plots shown below:



we can clearly see the results indicate all of the trees considered in the random forest, the Total_Trans_Ct and the Total_Trans_Amt are by far the two most important variables.

Boosting

Boosting algorithm trains the model with a focus on what the previous model have gotten wrong. We run gbm() function with the argument distribution = 'bernoulli' since it is a binary classification problem. n.trees = 5000 means we want 5000 trees, as N (n.trees) increases, the error on the training set will decrease but it will face overfitting problem. Interaction.depth = 4 limits the depth of the tree and controls the interaction order of the boosted model, d = 1 often works well. We used default shrinkage number, that is the learning rate $\lambda = 0.001$, since very small λ requires a large value of B, thus it is reasonable for us to choose the number of trees to be 5000. The summary() function generates a relative influence plot and the relative influence statistics, the plot shown below:



Both plot and statistics suggest the "Total_Trans_Ct" and the "Total_Trans_Amt" are by far the two most important variables, which is consistent with what we saw in the random forest. The confusion matrix shows the prediction accuracy is 0.9589 which means 95.89% of observations were correctly classified, the test error therefore is 1 - 0.9689 = 0.0411.

We still tried the different values of hyperparameters by setting n.trees = 1000, interaction.depth = 1, and shrinkage = 0.1, the prediction accuracy is 0.9528 which is slightly lower than the previous one.

Bart

Under BART package, we chose lbart() function since the response is binary variable. The prediction accuracy is 0.9492, which means that 94.92% of observations were correctly classified, and the test error is 1 - 0.9492 = 0.0508.

Treating Unknown as Missing Data

Theoretically, missing data would lead to model bias, which may affect other variables. In our data, the missing variables are categorical (marital status,educational level and income), which are difficult to fill in. Since the dimension of data is large enough, we could delete all those missing variables and analyze the rest of them to check whether the missing data affect our model selection or not. The remaining dataset has 7081 data with 19 columns.

In this section, we would focus on the hyperparameter tuning, since the syntax for the algorithms such as logistic regression, random forest, KNN, LDA, QDA and bart are the same as we did in the last section except keeping missing values.

SVM

We set different kernels which are 'linear' and 'radial' and used tune() function to select the best choice of γ and cost value C. We first used linear kernel and tested C with range from 0.01 to 10. The summary table tells when cost=0.1, it gets the best result, the prediction accuracy is 0.8949, which means 89.49% of observations were correctly classified. The model's confusion matrix is:

Table 1: Confusion matrix.

		Predicted	Predicted
		0	1
Actual	0	280	80
Actual	1	292	2888

0 means Attrited Customer, and 1 means Existing Customer.

We can find that the Accuracy is (TP+TN)/N= 0.8949.

The precision is TP/(TP+FP)=0.9730. This means 97.30% of customers who we predicted are existing are actually existing.

The recall is TP/(TP+FN)=0.9082. This means 90.82% of customers who are actually existing are predicted to be existed. The sensitivity (True Positive Rate) is TP/(TP+FN)=0.9082, which is the same with recall.

The specificity (True Negative Rate) is TN/(TN+FP)=0.7778. This means, 77.78% of customers who are actually attrited are predicted to be attrited.

For helping bank managers, we don't want customers to leave. That is, if the customer actually exists but we predict that he/she will leave, it's worse than the opposite. Therefore, we pay more attention on the false negatives (sensitivity).

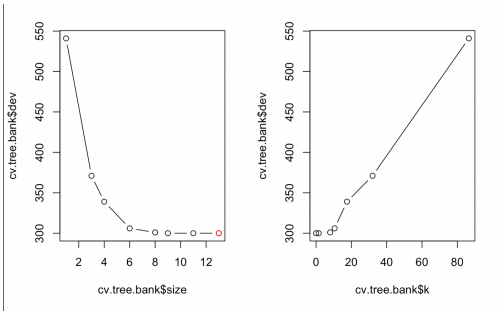
A radial kernel is defined as $k(x, x') = exp(-\gamma||x - x'||^2)$ where k is a valid kernel from $R^p \to R$. γ is a free parameter. Radial kernel expands into an infinite number of dimensions. This is due to the expansion of the exponential term. We tested C with range from 0.01 to 10, and γ with range from 0.001 to 10. The summary table tells when cost=10, $\gamma = 0.02$, the best prediction accuracy is 0.9339, which means 93.39% of observations were correctly classified and the sensitivity is 0.9440.

As 93.39% $>$ 89.49% in accuracy and 94.40% $>$ 90.82% in sensitivity, we concluded the

radial kernel is better than the linear kernel.

Classification Tree

We used cv.tree() function to test if pruning is necessary in the model,



From the plot above, we noticed that when tree has 9, 11, and 13 terminal nodes, a minimum of 300 cross-validation errors were generated, which has the same prediction accuracy (0.9133) with the same sensitivity (0.9567) as classification tree yields by default setting.

Boosting

Unlike bagging and random forests, boosting can overfit the training set if the number of trees is too large, although this overfitting tends to occur slowly. Therefore, we used cross-validation to select the number of trees. Since the shrinkage parameter λ controls the rate at which boosting learn and applying a very small λ requires using a very large number of trees for the model to achieve a good performance. As the interaction depth controls the complexity and interaction order of the boosted ensembles, we need to find the best one to fit the model. Based on the summary table, we found the best combination is when number of trees is 1000, $\lambda = 0.1$ and interaction depth is 4, the best accuracy is 0.9653 with sensitivity of 0.9718.

Other models

The following table shows the results we run on the same model as we did in the last section with no missing values:

Table 2: Other models Accuracy summary.

Response type	Accuracy	Sensitivity
Logistic Regression	0.8990	0.9172
Best model after Regfit	0.8977	0.9152
LDA	0.8969	0.9192
QDA	0.8989	0.9333
Bagging	0.9557	0.9649
Bart	0.9497	0.9550
KNN(K=1)	0.8638	0.9092
KNN(K=5)	0.8822	0.9084
KNN(K=100)	0.8667	0.8743
RF(mtry=$\sqrt{18}$)	0.9520	0.9536
RF(mtry=8)	0.9593	0.9647
RF(mtry=10)	0.9585	0.9636

Conclusion

We fit lots of models and use hyperparameter tuning to find which models perform high accuracy results. In real situation (before deleting missing values), we noticed that results from nonlinear methods are better than linear methods. Especially, boosting algorithm gave us the highest accuracy among all the models we fit. In addition, using variables for best subset selection in logistic regression did improve the results. Highly correlated predictors lead to LDA/QDA rank deficiencies, while trees are not sensitive to the effects of multicollinearity. Increasing the value C in SVM lead to small margin and overfitting problems. The "Total_Trans_Ct" and the "Total_Trans_Amt" are by far the two most important variables in the data set. And in most cases, R will automatically run the models with the optimal hyperparameters. In addition, we tried various models to classify the dataset after deleting missing values, and all these models classify the dataset well. The best model is the boosting model when taking the best combination of hyperparameters: the number of trees is 1000, $\lambda = 0.1$ and interaction depth is 4. The highest accuracy is 0.9653 with the highest sensitivity of 0.9718. We noticed that the accuracies are quite similar before and after deleting missing values, because the predictors of marital status, educational level and income play less significant roles on the response and there are other predictors such as total transaction amount and credit limit are highly correlated with the deleted variables.

Part 2: Car Insurance Claim Amount Prediction (Prediction Method)

Introduction

As most of premium revenue is spent on claim, using empirical data to identify patterns and trends to predict future claim is essential for an insurance company.

1 It can help insurance company to arrange reserves reasonably, in order to satisfy future payments.

2 According to the overall claims, the insurance company can adjust the premium of the next period in time to achieve balance of incomes and payments.

3 It is useful to identify high-quality customers with low claim rate.

Purpose/Motivation

As the importance of prediction for claim, our purpose of this project is to find a high accuracy algorithm, not only accurately predicts the amount of claim, but also evaluates the degree of influence of different predictors.

For original dataset, whose sample size is 9134. We select 19 out of 24 variables used as predictors and the variable of "Total Claim Amount" used as response variable. Then we fit some prediction models, such as linear regression, lasso regression, ridge regression, regression tree, Bagging, Random forest, Boosting, BART and so on, and test whether the models are sufficient and select the best model by using the test MSE.

Data Collection

We get the dataset from Kaggle.com. The problem of the dataset is which variables should be used in models for prediction. Through the analysis of the dataset, we decide to delete 4 variables.

1 Customer: ID of customers, which is useless obviously.

2 State: In order to reduce the running time, we focus on predicting claim amount of California, and delete the observations of other states, so that the sample size drops to 3150.

3 Effective to data: It is a variable about time. We delete it for two reasons, one reason is that keeping it in the model will overfit, and the other reason is that the span of time is just about one year and its influence is slight.

4 Policy Type: We both have 'Policy type' and 'Policy', and 'Policy' is a finer division of 'Policy type', which means that the variable "Policy" can determine "Policy type", so we delete the latter one.

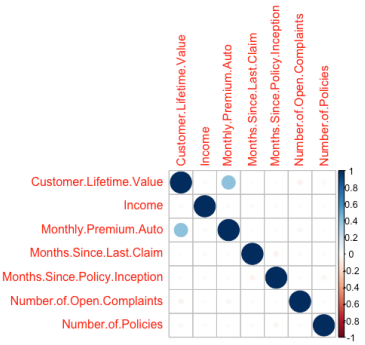
Statistical analysis

Preprocessing

After deleting four variables, we have 1 response variable and 19 predictors which includes 13 categorical variables and 6 continuous variables. Nextly, we make some preparations for modelling.

1 We check the completeness of the dataset, and find that the dataset has no missing values.

2 For continuous variables, we test the correlation pairwise. From the result below we can see they don't have high correlation, so we hold all continuous variables.



3 Some of the variables in the dataset are characters, we convert them to factor variables.

4 We use half of the sample as the training set (the size is 1575), and the other half as the test set. In addition, in order to make the work reproducible, we set the random seed in the R code using set.seed(2022).

Lingear Regression

Firstly, we use "lm" function with all the variables to build our full model. Then we use "step" function with several ways to do the variables selection including 'Forward', 'Backward', 'AIC', 'BIC', 'Adjusted R Square'.

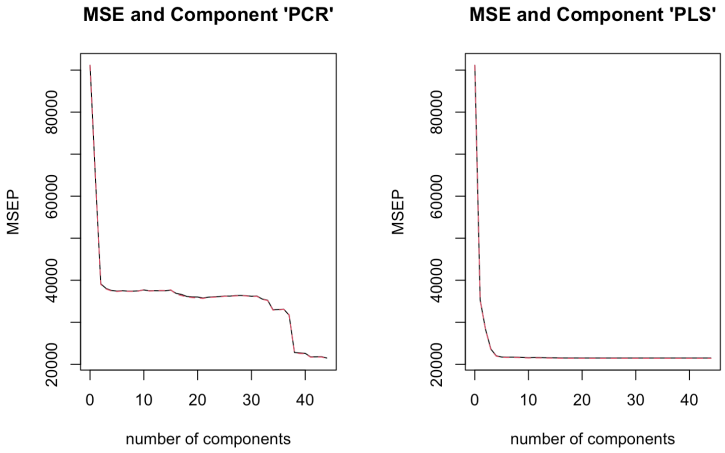
From these methods we get two different models, 'Forward' method implies a full model, other ways choose the same '10' variables. We compare the Test Error(MSE) and R Square for the two cases and find the full model holds a better result.(MSE_{full} = 20603.07<MSE_{others} = 20719.93,R^2_{full} = 0.7485989>R^2_{others} = 0.747173.

Method	Selected Variables	MSE	R-square
Forward	Full	20603.07	0.7485989
others	Coverage, Gender…(10 in total)	20719.93	0.747173

Next, we use 'pcr' and 'plsr' function to do variable selection with 'Polymerase Chain Reaction' and 'Primary lateral sclerosis', respectively.

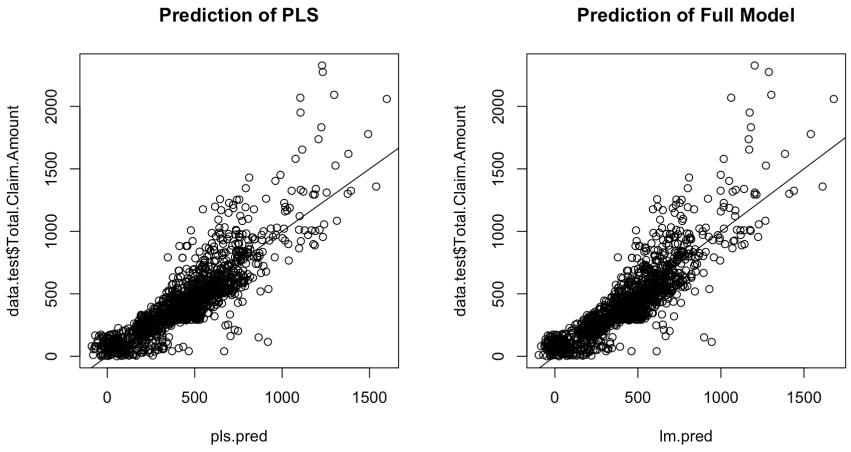
For 'PCR', we learn the correlation between 'MSE' and 'number of components'. The following picture implies when all the components are selected we get the best model, which means it has the same result with the full model.

For 'PLS', from the following picture we find when the components equal to 4, the model almostly get the smallest MSE about 21279.94. After 4, little MSE decreases with per component increasing, we think the model with less components is more effective when they have same MSE, so we choose the 4 components to fit the model.



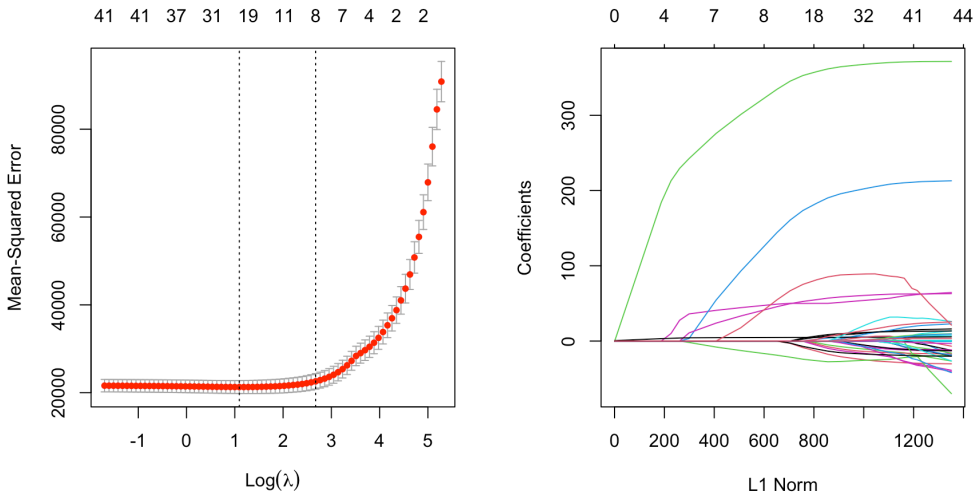
Above all, using the linear models with different methods to build models, we have 2 different results, they are 'full model' and 'PLS model'. By comparing the MSE we find the 'full model' holds a better result than 'PLS model', the following pictures and table also show this.

Method	MSE	R-square
PCR	20603.07	0.7485989
PLS	21279.94	0.7403397



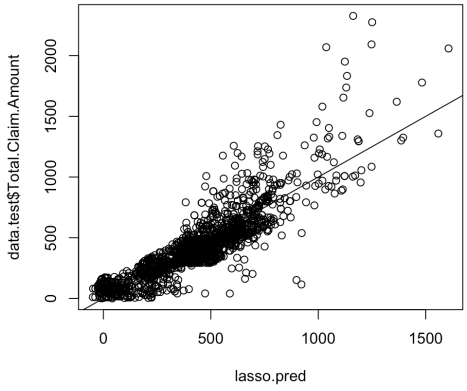
LASSO

For the LASSO regression, the regularization is added, we need to choose our best hyperparameter λ firstly, we use the function 'glment' in the 'glment' package, from the following picture we find with the increases of λ, the MSE also increases. By cross-validation, we get the best λ_{best}, which is 3.27. When we use 3.27 as our best lambda, the coefficiencts change like the right picture.



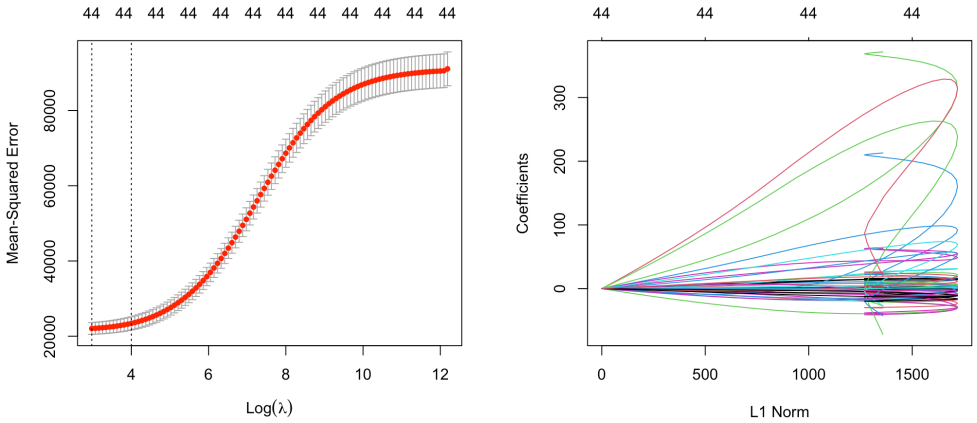
The MSE and R-square are shown in the table, we find the MSE and R square values are better than them in PLS model but still not as good as the full model.

Method	MSE	R-square
LASSO	20660.44	0.7485989



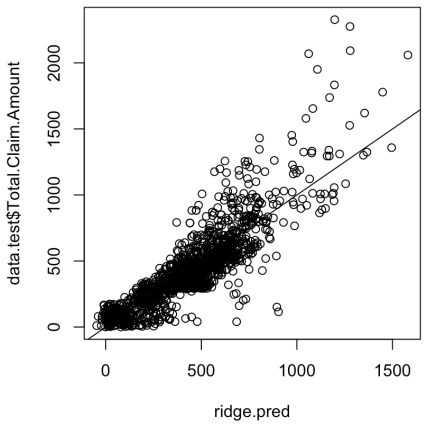
Ridge

For the Ridge regression, another regularization is added. Similarly to Lasso, from the following picture on the left we find that with the increases of λ, the MSE also increases. By cross-validation, we get the best λ_{best}, which is 19.62127, and the coefficiencts change like the right picture. Comparing with the coefficients in LASSO, more variables are used to do the prediction.



However, Ridge model doesn't give us a better result than LASSO, the MSE value is much bigger. Ridge model holds the worst result so far.

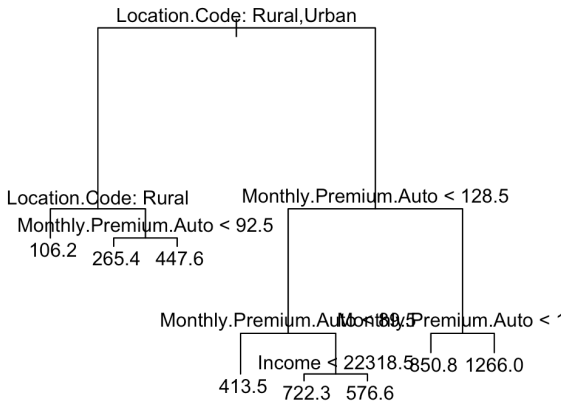
Method	MSE	R-square
Ridge	21245.18	0.7407638

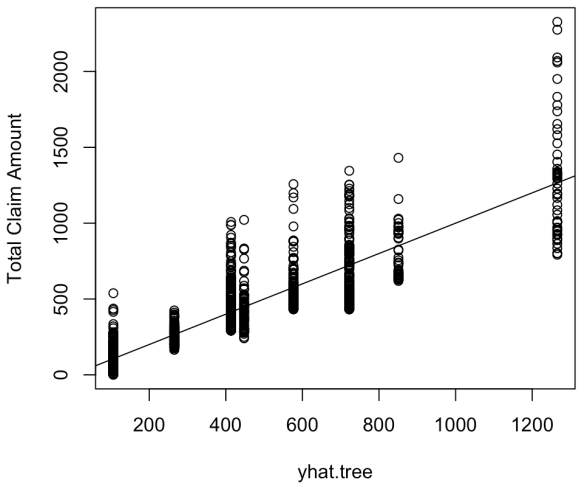


Regression Tree

We fit our data with Regression Tree model and find only three variables "Location Code", "Monthly Premium Auto" and "Income" are used. Our regression tree indicates that living in Rural, lower Monthly Premium Auto and lower Income will result in a lower Total Claim Amount.

The MSE is 22095.71, so the square root of it is 168.64, it means the true total claim amount for the census tract is within approximately 164.64.





Random Forest and Bagging

We test different values of 'ntree' ('100','200','500','1000') and 'mtry' (from 1 to 19) in Random Forest model, and compute the MSE of each combination, and the MSE is shown in the following table. Note that when 'mtry' = 19 (all parameters are included), it is the Bagging model.

We find when mtry = 8 and ntree =500, the smallest MSE occurs so that we build our best RF model with this combination, and the prediction result is shown as below.

Method	MSE	R-square
RandomForest	14629.22	0.8214926

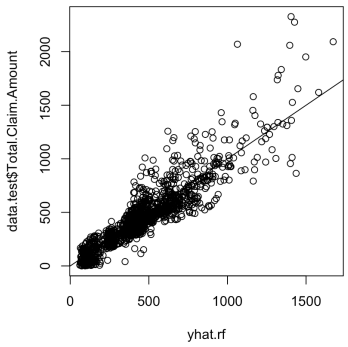


Table 8: Part of Hyper Parameter Tuning's MSE of RandomForest.

mtry and ntree	100	200	500	1000
1	35431.07	35367.74	34577.40	34638.26
2	21337.59	21334.53	21151.87	20786.36
3	17491.51	17248.31	17206.21	17287.99
4	16010.75	16016.28	16079.37	16008.74
5	15626.12	15498.10	15447.53	15375.52
6	15269.56	15113.14	14883.15	14866.29
7	15288.18	15015.44	14902.27	14815.27
8	15029.04	14844.16	14629.22	14661.75
9	14739.64	14846.10	14788.64	14750.66
10	14894.38	14738.11	14660.10	14737.93
19	15596.80	15405.57	15418.59	15350.88

Boosting

We choose various values of 'ntree' and 'depth' in Boosting model, and compute the MSE of each combination.

Table 9: Hyperparameter Tuning's MSE of Boosting(ntree and depth).

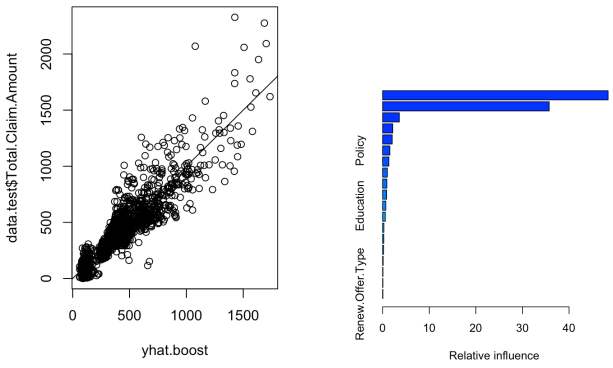
ntree and depth	1	2	3	4
100	21725.74	16456.29	15823.88	15563.45
200	21616.23	15833.62	15967.71	16000.08
500	21492.17	15779.22	16137.02	15904.90
1000	21717.68	16085.90	16674.19	16646.10

Table 10: Shrinkage Tuning's MSE.

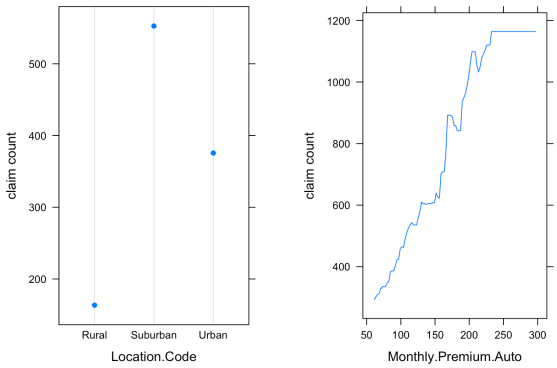
shrinkage	0.01	0.1	0.05
MSE	72220.56	15823.88	15382.96

We find when depth = 4, ntree =100 and shrinkage=0.15, the smallest MSE (15382.96) occurs so that we build our Boosting model with this combination. However, its MSE still bigger than Random Forest model.

Method	MSE	R-square
Boosting	15382.96	0.8100929



The picture above on the right side indicates that across all of the trees and depth considered in the boosting, the Location Code and the Monthly Premium Auto are by far the two most important variables.



Form the left chart we can see the customers live in Suburban have the highest claim, compared with the customers live in Rural get the lowest claim amount.

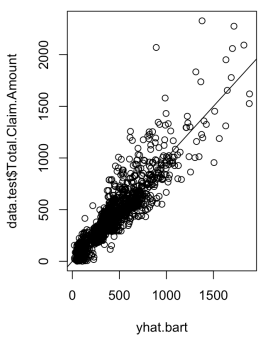
From the right chart we can find the Monthly Premium Auto has the positive correlation with claim amount.

Bart

We build our BART model with gbrat function in BART package, and the MSE of it is 15267.31, which is smaller than it in Boosting model but still not as effective as Random Forest model.

Variable	Monthly.Premium.Auto	Location.Code1	EmploymentStatus5
Times	18.750	8.535	8.494

The table above shows how many times a variable appears in the collections. Here we list three highest variables. We can see Monthly Premium Auto and Location Code are still in the first two places.



Conclusion

Table 13: Model Comparation.

Model	MSE
Random Forest(ntree=500,mtry=8)	14629.22
BART	15267.31
Boosting(ntree=100,depth=4,shrinkage=0.15)	15382.96
Full-model/Forward	20603.07
PCR(ncomp=44)	20603.07
LASSO(lambda = 3.27)	20660.44
AIC/BIC/Adjustable R Square/Backward	20719.93
Ridge(lambda=19.62)	21245.18
PLC(ncomp=4)	21279.94
Rgression Tree	22095.71

We try various models to predict the amount of claim, and the prediction results of these models are quite different. By comparing all the models' MSEs, we find the results from nonlinear methods are better than linear methods on the whole, and the best model is the Random Forest with ntree=500 and mtry=8, which has the highest R-square (about 82%). In addition, we also can see the methods of variable selection don't improve the prediction accuracy.

On the other hand, from the models we learn, Monthly Premium Auto and Location code have the greatest importance of Total Claim Amount, which means if customers who live closer to urban or pay more car insurance premium per month were more likely to have a higher claim amount.

References

Lecture notes and labs

https://www.kaggle.com/datasets/whenamancodes/credit-card-customers-prediction

https://www.kaggle.com/datasets/pankajjsh06/ibm-watson-marketing-customer-value-data

Appendix

Part 1

```
## import data and data preprocessing
library(leaps)
library(MASS)
library(e1071)
library(tree)
library(randomForest)
library(gbm)
library(BART)
bank <- read.csv("BankChurners.csv")
head(bank)
dim(bank)
str(bank)
bank <- bank[, 2:21]
for (i in 1:ncol(bank)) {
  if (class(bank[, i]) == "character") {
    bank[, i] = factor(bank[,i ])
  }
}
str(bank)


set.seed(2022)
train <- sample(1:nrow(bank), (nrow(bank)+1)/2)
dim(bank[train,])
bank.test <- bank[-train, ]
dim(bank.test)
```

```r
## logistic regression
glm.bank <- glm(Attrition_Flag ~., data = bank, subset = train, family = binomial)
glm.pred <- predict(glm.bank, bank.test, type = "response")
glm.pred <- ifelse(glm.pred < 0.5, "Attrited Customer", "Existing Customer")
table(glm.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using logistic regression is: ", (482+4057)/nrow(bank.test)


set.seed(2022)
bank.test <- bank[-train, ]
x_1 <- bank$Attrition_Flag
x_2 <- bank$Customer_Age
x_3 <- bank$Gender
x_4 <- bank$Dependent_count
x_5 <- bank$Education_Level
x_6 <- bank$Marital_Status
x_7 <- bank$Income_Category
x_8 <- bank$Card_Category
x_9 <- bank$Months_on_book
x_10 <- bank$Total_Relationship_Count
x_11 <- bank$Months_Inactive_12_mon
x_12 <- bank$Contacts_Count_12_mon
x_13 <- bank$Credit_Limit
x_14 <- bank$Total_Revolving_Bal
x_15 <- bank$Avg_Open_To_Buy
x_16 <- bank$Total_Amt_Chng_Q4_Q1
x_17 <- bank$Total_Trans_Amt
x_18 <- bank$Total_Trans_Ct
x_19 <- bank$Total_Ct_Chng_Q4_Q1
x_20 <- bank$Avg_Utilization_Ratio
```

```
fulldata1 <- data.frame('Y' = x_1, 'X1' = x_2, 'X2' = x_3,'X3'=x_4,'X4'=x_5,'X5'=x_6,'X6
bank.test1 <- fulldata1[-train, ]
glm.fit <- glm(Y ~.,data = fulldata1, subset = train, family = binomial)
glm.pred1 <- predict(glm.fit, bank.test1, type = "response")
glm.pred1 <- ifelse(glm.pred1 < 0.5, "Attrited Customer", "Existing Customer")
table(glm.pred1, bank.test1$Y) # confusion matrix
cat("The prediction accuracy using logistic regression is: ", (482+4057)/nrow(bank.test1
regfit.full<-regsubsets(Y~.,data=fulldata1[train,])
summary(regfit.full)
par(mfrow=c(1,3))
plot(regfit.full,scale="Cp")
plot(regfit.full,scale="bic")
plot(regfit.full,scale="adjr2")
#AIC
regfit.fwd<-regsubsets(Y~.,data=fulldata1,method="forward")
reg.fwd.summary<-summary(regfit.fwd)
plot(regfit.fwd,scale="Cp")
plot(regfit.fwd,scale="bic")
plot(regfit.fwd,scale="adjr2")
#BIC
regfit.bwd<-regsubsets(Y~.,data=fulldata1,method="backward")
plot(regfit.bwd,scale="Cp")
plot(regfit.bwd,scale="bic")
plot(regfit.bwd,scale="adjr2")

#High correlation problem
cor(fulldata1[,c(13,15)])
fulldata1.drop <- fulldata1[,-15]
regfit.drop<-regsubsets(Y~.,data=fulldata1.drop[train,])
```

```
summary(regfit.drop)


plot(regfit.drop,scale="Cp")
plot(regfit.drop,scale="bic")
plot(regfit.drop,scale="adjr2")
#AIC
regfit.fwd<-regsubsets(Y~.,data=fulldata1.drop,method="forward")
reg.fwd.summary<-summary(regfit.fwd)
plot(regfit.fwd,scale="Cp")
plot(regfit.fwd,scale="bic")
plot(regfit.fwd,scale="adjr2")
#BIC
regfit.bwd<-regsubsets(Y~.,data=fulldata1.drop,method="backward")
plot(regfit.bwd,scale="Cp")
plot(regfit.bwd,scale="bic")
plot(regfit.bwd,scale="adjr2")
par(mfrow=c(1,1))



##best model
fulldata2 <- data.frame('Y' = x_1,'X2' = x_3,'X5'=x_6,'X9'=x_10,'X10'=x_11,'X11'=x_12,'X
bank.test2 <- fulldata2[-train, ]
glm.fit2 <- glm(Y ~.,data = fulldata2, subset = train, family = binomial)
glm.pred2 <- predict(glm.fit2, bank.test2, type = "response")
glm.pred2 <- ifelse(glm.pred2 < 0.5, "Attrited Customer", "Existing Customer")
table(glm.pred2, bank.test2$Y) # confusion matrix
cat("The prediction accuracy using logistic regression is: ", (485+4056)/nrow(bank.test)


## LDA
```

```r
lda.bank <- lda(Attrition_Flag ~ ., data = bank[,-15], subset = train)
coef(lda.bank)
lda.pred <- predict(lda.bank, bank.test)
table(lda.pred$class, bank.test$Attrition_Flag)
cat("The prediction accuracy using LDA is: ", (511+4025)/nrow(bank.test)) #0.8959
plot(lda.bank)

## QDA
qda.bank <- qda(Attrition_Flag ~ ., data = bank[,-15], subset = train)
qda.pred <- predict(qda.bank, bank.test[,-15])
table(qda.pred$class, bank.test$Attrition_Flag)
cat("The prediction accuracy using LDA is: ", (540+4010)/nrow(bank.test))  #0.8987

## KNN
library(class)
bank2 = bank[,-15]
bank2 <- matrix(0, nrow(bank),ncol(bank))
for (i in 1:ncol(bank)){
  bank2[,i] <- as.numeric(bank[,i])
}
train.X <- bank2[train,-1]
test.X <- bank2[-train, -1]
train.Y <- bank2[train,1]
knn.pred <- knn(train.X,test.X,train.Y, k = 1)
table(knn.pred,bank.test$Attrition_Flag)
cat("The prediction accuracy using KNN is: ", (487+3908)/nrow(bank.test)) #0.8681
knn.pred2 <- knn(train.X,test.X,train.Y, k = 5)
table(knn.pred2, bank.test$Attrition_Flag)
cat("The prediction accuracy using KNN is: ", (468+4025)/nrow(bank.test))  #0.8874
```

```r
knn.pred3 <- knn(train.X,test.X,train.Y, k = 100)
table(knn.pred3, bank.test$Attrition_Flag)
cat("The prediction accuracy using KNN is: ", (267+4106)/nrow(bank.test))  #0.8637


## SVM
bank <- bank[,-15]
tune.out <- tune(svm, Attrition_Flag ~ ., data = bank[train, ], kernel= "linear", ranges
bestmod <- tune.out$best.model
summary(bestmod)
svm.pred <- predict(bestmod, bank.test)
table(svm.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using SVM is: ", (480+4069)/nrow(bank.test)) #0.8985


## classification tree
tree.bank <- tree(Attrition_Flag ~ ., data = bank, subset = train)
summary(tree.bank)  #default nodes = 12
plot(tree.bank)
text(tree.bank, pretty = 0,cex=0.5)
tree.pred <- predict(tree.bank, bank.test, type = "class")
table(tree.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using classfication tree is: ", (620+3993)/nrow(bank.test))

##pruning
set.seed(2022)
cv.tree.bank <- cv.tree(tree.bank, FUN = prune.misclass)
cv.tree.bank
str(cv.tree.bank)
```

```
par(mfrow = c(1, 2))
plot(cv.tree.bank$size, cv.tree.bank$dev, type = "b")
plot(cv.tree.bank$k, cv.tree.bank$dev, type = "b")
par(mfrow = c(1, 1))

prune.bank <- prune.misclass(tree.bank, best = 10)
plot(prune.bank)
text(prune.bank, pretty = 0,cex = 0.5)
prune.tree.pred <- predict(prune.bank, bank.test, type = "class")
table(prune.tree.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using pruned classfication tree is: ", (620+3993)/nrow(bank

## bagging
set.seed(2022)
dim(bank)
bag.bank <- randomForest(Attrition_Flag ~., data = bank, subset = train, mtry = 18, impo
bag.bank  #training data set confusion matrix
bag.pred <- predict(bag.bank, newdata = bank.test)
table(bag.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using bagging is: ", (724+4119)/nrow(bank.test))  #0.9565
plot(bag.pred, bank.test$Attrition_Flag)

## randomForest
set.seed(2022)
rf.bank <- randomForest(Attrition_Flag ~ ., data = bank, subset = train, importance = TR
rf.pred <- predict(rf.bank, newdata = bank.test)
table(rf.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using randomForest is: ", (693+4142)/nrow(bank.test)) #0.95
varImpPlot(rf.bank) #CT and AMT are important
```

```
plot(rf.pred, bank.test$Attrition_Flag)

## Boosting
bank.cat = bank
bank.cat$Attrition_Flag <- ifelse(bank.cat$Attrition_Flag == "Attrited Customer", 0, 1)
bank.test.cat <- bank.cat[-train, ]



boost.bank <- gbm(Attrition_Flag ~ ., data = bank.cat[train, ], distribution = "bernoull
summary(boost.bank,las = 2,cex.names = 0.3)
boost.pred <- predict(boost.bank, newdata = bank.test.cat, n.trees = 5000, type = "respo
boost.pred <- ifelse(boost.pred > 0.5, 1, 0)
table(boost.pred, bank.test.cat$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (732+4123)/nrow(bank.test))  #0.9589

boost.bank2 <- gbm(Attrition_Flag ~ ., data = bank.cat[train, ], distribution = "bernoul
boost.pred2 <- predict(boost.bank2, newdata = bank.test.cat, n.trees = 1000, type = "res
boost.pred2 <- ifelse(boost.pred2 > 0.5, 1, 0)
table(boost.pred2, bank.test.cat$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (697+4127)/nrow(bank.test)) #0.9528



##Bart
library(BART)
#y <- rep(0,nrow(bank))
#y[bank$Attrition_Flag=='Existing Customer'] <- 1
#bart.bank <- lbart(bank[train,-1],y[train],bank[-train,-1])
bart.bank <- lbart(bank[train,-1], bank.cat$Attrition_Flag[train], bank[-train,-1])
pred.bart <- bart.bank$prob.test.mean
```

```r
pred.bart <- ifelse(pred.bart<0.5,0,1)
table(pred.bart, bank.cat$Attrition_Flag[-train])
cat("The prediction accuracy using bart is: ", (682+4124)/nrow(bank.test))   #0.9492

##treat Unknown catrgory as NA

bank[bank == "Unknown"] <- NA
bank.na <- na.omit(bank)
dim(bank.na)
for (i in 1:ncol(bank.na)) {
  if (class(bank.na[, i]) == "character") {
    bank.na[, i] = factor(bank.na[,i ])
  }
}
set.seed(2022)
train.na <- sample(1:nrow(bank.na), (nrow(bank.na)+1)/2)
bank.test.na <- bank.na[-train.na, ]

## logistic regression
set.seed(2022)
glm.bank.na <- glm(Attrition_Flag ~., data = bank.na, subset = train.na, family = binomi
glm.pred.na <- predict(glm.bank.na, bank.test.na, type = "response")
glm.pred.na <- ifelse(glm.pred.na < 0.5, "Attrited Customer", "Existing Customer")
table(glm.pred.na, bank.test.na$Attrition_Flag) # confusion matrix
cat("The prediction accuracy using logistic regression is: ", (313+2870)/nrow(bank.test.


## LDA
bank.test.na2 <- matrix(0,nrow(bank.test.na),ncol(bank.test.na))
```

```r
for (i in 1:ncol(bank.test.na)){
  bank.test.na2[,i]<- as.numeric(bank.test.na[,i])
}
lda.bank.na <- lda(bank.na[train.na,-1], grouping = bank.na[train.na,1])
lda.pred.na <- predict(lda.bank.na, bank.test.na2[,-1])
table(lda.pred.na$class, bank.test.na$Attrition_Flag)
cat("The prediction accuracy using LDA is: ", (321+2854)/nrow(bank.test.na)) #0.8969


## QDA
qda.bank.na <- qda(bank.na[train.na, -c(1,15)], group = bank.na[train.na,1])
qda.pred.na <- predict(qda.bank.na, bank.test.na2[,-c(1,15)])
table(qda.pred.na$class,bank.test.na$Attrition_Flag)
cat("The prediction accuracy using QDA is: ", (371+2811)/nrow(bank.test.na)) #0.8989

## SVM
tune.out.na <- tune(svm, Attrition_Flag ~ ., data = bank.na[train.na, ], kernel= "linear
bestmod.na <- tune.out.na$best.model
summary(bestmod.na)
svm.pred.na <- predict(bestmod.na, bank.test.na)
table(svm.pred.na, bank.test.na$Attrition_Flag)
cat("The prediction accuracy using SVM is: ", (280+2888)/nrow(bank.test.na))  #0.8949

## Boosting
bank.cat.na = bank.na
bank.cat.na$Attrition_Flag <- ifelse(bank.cat.na$Attrition_Flag == "Attrited Customer",
bank.test.cat.na <- bank.cat.na[-train.na, ]
boost.bank.na <- gbm(Attrition_Flag ~ ., data = bank.cat.na[train.na, ], distribution =
summary(boost.bank.na,las = 2,cex.names = 0.3)
```

```
boost.pred.na <- predict(boost.bank.na, newdata = bank.test.cat.na, n.trees = 5000, type
boost.pred.na <- ifelse(boost.pred.na > 0.5, 1, 0)
table(boost.pred.na, bank.test.cat.na$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (488+2926)/nrow(bank.test.cat.na))  #

boost.bank.na2 <- gbm(Attrition_Flag ~ ., data = bank.cat.na[train.na, ], distribution =
boost.pred.na2 <- predict(boost.bank.na2, newdata = bank.test.cat.na, n.trees = 1000, ty
boost.pred.na2 <- ifelse(boost.pred.na2 > 0.5, 1, 0)
table(boost.pred.na2, bank.test.cat.na$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (463+2906)/nrow(bank.test.na)) #0.951


## Delete the unknown elements

## import data and data preprocessing

library(leaps)
library(MASS)
library(e1071)
library(tree)
library(randomForest)
library(gbm)
library(BART)
bank <- read.csv("BankChurners.csv")
head(bank)
dim(bank)
str(bank)
bank <- bank[, 2:21]
```

```r
str(bank)
bank <- subset(bank, Education_Level != 'Unknown')
bank <- subset(bank, Marital_Status !='Unknown')
bank <- subset(bank, Income_Category!='Unknown')
for (i in 1:ncol(bank)) {
  if (class(bank[, i]) == "character") {
    bank[, i] = factor(bank[,i ])
  }
}



set.seed(2022)
train <- sample(1:nrow(bank), (nrow(bank)+1)/2)
bank.train<- bank[train,]
bank.test <- bank[-train, ]


## logistic regression
set.seed(2022)
glm.bank <- glm(Attrition_Flag ~., data = bank, subset = train, family = binomial)
glm.pred <- predict(glm.bank, bank.test, type = "response")
glm.pred <- ifelse(glm.pred < 0.5, "Attrited Customer", "Existing Customer")
table(glm.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using logistic regression is: ", mean(glm.pred==bank.test$A

set.seed(2022)
bank.test <- bank[-train, ]
x_1 <- bank$Attrition_Flag
```

```
x_2 <- bank$Customer_Age
x_3 <- bank$Gender
x_4 <- bank$Dependent_count
x_5 <- bank$Education_Level
x_6 <- bank$Marital_Status
x_7 <- bank$Income_Category
x_8 <- bank$Card_Category
x_9 <- bank$Months_on_book
x_10 <- bank$Total_Relationship_Count
x_11 <- bank$Months_Inactive_12_mon
x_12 <- bank$Contacts_Count_12_mon
x_13 <- bank$Credit_Limit
x_14 <- bank$Total_Revolving_Bal
x_15 <- bank$Avg_Open_To_Buy
x_16 <- bank$Total_Amt_Chng_Q4_Q1
x_17 <- bank$Total_Trans_Amt
x_18 <- bank$Total_Trans_Ct
x_19 <- bank$Total_Ct_Chng_Q4_Q1
x_20 <- bank$Avg_Utilization_Ratio


fulldata1 <- data.frame('Y' = x_1, 'X1' = x_2, 'X2' = x_3,'X3'=x_4,'X4'=x_5,'X5'=x_6,'X6
bank.test1 <- fulldata1[-train, ]
fulldata0 <- data.frame('Y' = x_1, 'X1' = x_2, 'X2' =as.numeric(x_3),'X3'=x_4,'X4'=as.nu
glm.fit <- glm(Y ~.,data = fulldata1, subset = train, family = binomial)
glm.pred1 <- predict(glm.fit, bank.test1, type = "response")
glm.pred1 <- ifelse(glm.pred1 < 0.5, "Attrited Customer", "Existing Customer")
table(glm.pred1, bank.test1$Y) # confusion matrix
cat("The prediction accuracy using logistic regression is: ", mean(glm.pred1==bank.test1
regfit.full<-regsubsets(Y~.,data=fulldata1)
```

```
summary(regfit.full)
par(mfrow=c(1,3))
plot(regfit.full,scale="Cp")
plot(regfit.full,scale="bic")
plot(regfit.full,scale="adjr2")
#AIC
regfit.fwd<-regsubsets(Y~.,data=fulldata1,method="forward")
reg.fwd.summary<-summary(regfit.fwd)
plot(regfit.fwd,scale="Cp")
plot(regfit.fwd,scale="bic")
plot(regfit.fwd,scale="adjr2")
#BIC
regfit.bwd<-regsubsets(Y~.,data=fulldata1,method="backward")
plot(regfit.bwd,scale="Cp")
plot(regfit.bwd,scale="bic")
plot(regfit.bwd,scale="adjr2")
par(mfrow=c(1,1))
#High correlation problem
cor(fulldata1[,c(13,15)])
cor(fulldata0[,c(2,9)])
fulldata1.drop <- fulldata1[,-15]
regfit.drop<-regsubsets(Y~.,data=fulldata1.drop)
summary(regfit.drop)
par(mfrow=c(1,3))
plot(regfit.drop,scale="Cp")
plot(regfit.drop,scale="bic")
plot(regfit.drop,scale="adjr2")
#AIC
regfit.fwd<-regsubsets(Y~.,data=fulldata1.drop,method="forward")
```

```
reg.fwd.summary<-summary(regfit.fwd)
plot(regfit.fwd,scale="Cp")
plot(regfit.fwd,scale="bic")
plot(regfit.fwd,scale="adjr2")
#BIC
regfit.bwd<-regsubsets(Y~.,data=fulldata1.drop,method="backward")
plot(regfit.bwd,scale="Cp")
plot(regfit.bwd,scale="bic")
plot(regfit.bwd,scale="adjr2")
par(mfrow=c(1,1))



##best model

set.seed(2022)
fulldata2 <- data.frame('Y' = x_1,'X2' = x_3,'X5'=x_6,'X9'=x_10,'X10'=x_11,'X11'=x_12,'X
bank.test2 <- fulldata2[-train, ]
glm.fit2 <- glm(Y ~.,data = fulldata2, subset = train, family = binomial)
glm.pred2 <- predict(glm.fit2, bank.test2, type = "response")
glm.pred2 <- ifelse(glm.pred2 < 0.5, "Attrited Customer", "Existing Customer")
table(glm.pred2, bank.test2$Y) # confusion matrix
cat("The prediction accuracy using logistic regression is: ", mean(glm.pred2==bank.test2

set.seed(2022)
bank<-bank[,-15]
train <- sample(1:nrow(bank), (nrow(bank)+1)/2)
bank.train<- bank[train,]
bank.test <- bank[-train, ]
```

```r
## LDA

set.seed(2022)
bank.test2 <- matrix(0,nrow(bank.test),ncol(bank.test))
for (i in 1:ncol(bank.test)){
  bank.test2[,i]<- as.numeric(bank.test[,i])
}
lda.bank <- lda(bank[train,-1], grouping = bank[train,1])
lda.pred <- predict(lda.bank, bank.test2[,-1])
table(lda.pred$class, bank.test$Attrition_Flag)
cat("The prediction accuracy using LDA is: ", (321+2854)/nrow(bank.test)) #0.8969


## QDA

set.seed(2022)
qda.bank <- qda(bank[train, -1], group = bank[train,1])
qda.pred <- predict(qda.bank, bank.test2[,-1])
table(qda.pred$class,bank.test$Attrition_Flag)
cat("The prediction accuracy using QDA is: ", (371+2811)/nrow(bank.test)) #0.8989


## KNN


library(class)
set.seed(2022)
bank2 <- matrix(0, nrow(bank),ncol(bank))
```

```r
for (i in 1:ncol(bank)){
  bank2[,i] <- as.numeric(bank[,i])
}
train.X <- bank2[train,-1]
test.X <- bank2[-train, -1]
train.Y <- bank2[train,1]
knn.pred <- knn(train.X,test.X,train.Y, k = 1)
table(knn.pred,bank.test$Attrition_Flag)
cat("The prediction accuracy using KNN is: ", (296+2762)/nrow(bank.test)) #0.86384
knn.pred2 <- knn(train.X,test.X,train.Y, k = 5)
table(knn.pred2, bank.test$Attrition_Flag)
cat("The prediction accuracy using KNN is: ", (286+2837)/nrow(bank.test))  #0.88220
knn.pred3 <- knn(train.X,test.X,train.Y, k = 100)
table(knn.pred3, bank.test$Attrition_Flag)
cat("The prediction accuracy using KNN is: ", (153+2915)/nrow(bank.test))  #0.86667


## SVM

set.seed(2022)
tune.out <- tune(svm, Attrition_Flag ~ ., data = bank[train, ], kernel= "linear", ranges
summary(tune.out) # error rate=0.09461367
bestmod <- tune.out$best.model
summary(bestmod)
svm.pred <- predict(bestmod, bank.test)
table(svm.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using SVM is: ", (280+2888)/nrow(bank.test)) #0.8949153 co
```

```r
set.seed(2022)
tune0.out <- tune(svm, Attrition_Flag ~ ., data = bank[train, ], ranges = list(cost = c(
summary(tune0.out) #error rate 0.06749741
bestmod0 <- tune0.out$best.model
summary(bestmod0)
svm.pred0 <- predict(bestmod0, bank.test)
table(svm.pred0, bank.test$Attrition_Flag)
cat("The prediction accuracy using SVM is: ", (401+2905)/nrow(bank.test)) #0.93390,cost=


## classification tree

set.seed(2022)
tree.bank <- tree(Attrition_Flag ~ ., data = bank, subset = train)
summary(tree.bank)  #default nodes = 13
plot(tree.bank)
text(tree.bank, pretty = 0,cex=0.5)
tree.pred <- predict(tree.bank, bank.test, type = "class")
table(tree.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using classfication tree is: ", (446+2787)/nrow(bank.test))


##pruning

set.seed(2022)
cv.tree.bank <- cv.tree(tree.bank, FUN = prune.misclass)
cv.tree.bank
str(cv.tree.bank)
par(mfrow = c(1, 2))
```

```
plot(cv.tree.bank$size, cv.tree.bank$dev, type = "b")
points(cv.tree.bank$size[which.min(cv.tree.bank$dev)], min(cv.tree.bank$dev), col = "red
plot(cv.tree.bank$k, cv.tree.bank$dev, type = "b")
par(mfrow = c(1, 1))

prune.bank <- prune.misclass(tree.bank, best = 13)
plot(prune.bank)
text(prune.bank, pretty = 0,cex = 0.5)
prune.tree.pred <- predict(prune.bank, bank.test, type = "class")
table(prune.tree.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using pruned classfication tree is: ", (446+2787)/nrow(bank


## bagging

set.seed(2022)
dim(bank)
bag.bank <- randomForest(Attrition_Flag ~., data = bank, subset = train, mtry = 18, impo
bag.bank  #training data set confusion matrix
bag.pred <- predict(bag.bank, newdata = bank.test)
table(bag.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using bagging is: ", (465+2917)/nrow(bank.test))  #0.95565
plot(bag.pred, bank.test$Attrition_Flag)

## randomForest

set.seed(2022)
rf.bank <- randomForest(Attrition_Flag ~ ., data = bank, subset = train, importance = TR
rf.pred <- predict(rf.bank, newdata = bank.test)
```

```
table(rf.pred, bank.test$Attrition_Flag)
cat("The prediction accuracy using randomForest is: ", (429+2941)/nrow(bank.test)) #0.95
varImpPlot(rf.bank) #CT and AMT are important
plot(rf.pred, bank.test$Attrition_Flag)

rf.bank2 <- randomForest(Attrition_Flag ~ ., data = bank, subset = train,mtry=10 , impor
rf.pred2 <- predict(rf.bank2, newdata = bank.test)
table(rf.pred2, bank.test$Attrition_Flag)
cat("The prediction accuracy using randomForest is: ", (465+2928)/nrow(bank.test)) #0.95

rf.bank3 <- randomForest(Attrition_Flag ~ ., data = bank, subset = train,mtry=8 , import
rf.pred3 <- predict(rf.bank3, newdata = bank.test)
table(rf.pred3, bank.test$Attrition_Flag)
cat("The prediction accuracy using randomForest is: ", (453+2933)/nrow(bank.test)) #0.95

## Boosting


set.seed(2022)
bank.cat = bank
bank.cat$Attrition_Flag <- ifelse(bank.cat$Attrition_Flag == "Attrited Customer", 0, 1)
bank.test.cat <- bank.cat[-train, ]


set.seed(2022)
boost.bank <- gbm(Attrition_Flag ~ ., data = bank.cat[train, ], distribution = "bernoull
summary(boost.bank,las = 2,cex.names = 0.3)

plot(boost.bank,i="Total_Trans_Ct")
```

```r
plot(boost.bank,i="Total_Trans_Amt")

boost.pred <- predict(boost.bank, newdata = bank.test.cat, n.trees = 5000, type = "respo
boost.pred <- ifelse(boost.pred > 0.5, 1, 0)
table(boost.pred, bank.test.cat$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (489+2925)/nrow(bank.test))  #0.9644

boost.bank2 <- gbm(Attrition_Flag ~ ., data = bank.cat[train, ], distribution = "bernoul
boost.pred2 <- predict(boost.bank2, newdata = bank.test.cat, n.trees = 1000, type = "res
boost.pred2 <- ifelse(boost.pred2 > 0.5, 1, 0)
table(boost.pred2, bank.test.cat$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (492+2925)/nrow(bank.test)) #0.96525

boost.bank3 <- gbm(Attrition_Flag ~ ., data = bank.cat[train, ], distribution = "bernoul
boost.pred3 <- predict(boost.bank3, newdata = bank.test.cat, n.trees = 10000, type = "re
boost.pred3 <- ifelse(boost.pred3 > 0.5, 1, 0)
table(boost.pred3, bank.test.cat$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (487+2927)/nrow(bank.test)) #0.96441



boost.bank5 <- gbm(Attrition_Flag ~ ., data = bank.cat[train, ], distribution = "bernoul
boost.pred5 <- predict(boost.bank5, newdata = bank.test.cat, n.trees = 1000, type = "res
boost.pred5 <- ifelse(boost.pred5 > 0.5, 1, 0)
table(boost.pred5, bank.test.cat$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (481+2925)/nrow(bank.test)) #0.96215

boost.bank6 <- gbm(Attrition_Flag ~ ., data = bank.cat[train, ], distribution = "bernoul
boost.pred6 <- predict(boost.bank6, newdata = bank.test.cat, n.trees = 1000, type = "res
boost.pred6 <- ifelse(boost.pred6 > 0.5, 1, 0)
```

```
table(boost.pred6, bank.test.cat$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (488+2930)/nrow(bank.test)) #0.96554

boost.bank7 <- gbm(Attrition_Flag ~ ., data = bank.cat[train, ], distribution = "bernoul
boost.pred7 <- predict(boost.bank7, newdata = bank.test.cat, n.trees = 1000, type = "res
boost.pred7 <- ifelse(boost.pred7 > 0.5, 1, 0)
table(boost.pred7, bank.test.cat$Attrition_Flag)
cat("The prediction accuracy using boosting is: ", (458+2937)/nrow(bank.test)) #0.95904



##Bart
library(BART)
set.seed(2022)
#y <- rep(0,nrow(bank))
#y[bank$Attrition_Flag=='Existing Customer'] <- 1
#bart.bank <- lbart(bank[train,-1],y[train],bank[-train,-1])
bart.bank <- lbart(bank[train,-1], bank.cat$Attrition_Flag[train], bank[-train,-1])
pred.bart <- bart.bank$prob.test.mean
pred.bart <- ifelse(pred.bart<0.5,0,1)
table(pred.bart, bank.cat$Attrition_Flag[-train])
cat("The prediction accuracy using bart is: ", (434+2928)/nrow(bank.test))   #0.94972
```

Part 2

```
dff=read.csv('Car_Insurance_Claim_Amount_Prediction.csv')
df=dff[dff$State=="California",] #choose California to analyse
data=df[,c(-1,-2,-7,-18)] #delete ID, effective to date, policy type
head(data)
names(data)
```

```r
for(i in 1:ncol(data)){
  if (class(data[,i])=="character"){
    data[,i]=factor(data[,i])
  }
}

#cor(data[,c(-2,-3)])
set.seed(2022)
train.size = dim(data)[1] / 2
train = sample(1:dim(data)[1], train.size)
data.train=data[train,]
data.test=data[-train,]

#linear
set.seed(2022)
lm.fit = lm(Total.Claim.Amount~., data=data.train)
summary(lm.fit)

#coef(lm.fit)
lm.pred = predict(lm.fit, data.test)
test.error.lm=mean((data.test$Total.Claim.Amount - lm.pred)^2)
test.error.lm

test.avg=mean((mean(data.test$Total.Claim.Amount)-data.test$Total.Claim.Amount)^2)
test.error.lm.r2=1-test.error.lm/test.avg
test.error.lm.r2
#fit
qqplot(c(1:train.size),data.test$Total.Claim.Amount,main="linear fit result",xlab="Sampl
lines(c(1:train.size),sort(lm.pred),col="red")
```

```
plot(lm.pred, data.test$Total.Claim.Amount);
abline(0,1)



#AIC BIC Adjust Rsq
aic=step(lm.fit,type="AIC",trace=FALSE)
summary(aic)
aic.pred = predict(aic, data.test)
test.error.aic=mean((data.test$Total.Claim.Amount - aic.pred)^2)
test.error.aic
test.error.aic.r2=1-test.error.aic/test.avg
test.error.aic.r2

bic=step(lm.fit,type="BIC",trace=FALSE)
summary(bic)
bic.pred = predict(bic, data.test)
test.error.bic=mean((data.test$Total.Claim.Amount - bic.pred)^2)
test.error.bic



adjr2=step(lm.fit,type="adjr2",trace=FALSE)
summary(adjr2)
adjr2.pred = predict(adjr2, data.test)
test.error.adjr2=mean((data.test$Total.Claim.Amount - adjr2.pred)^2)
test.error.adjr2

bkd=step(lm.fit,direction = "backward",trace=FALSE)
```

```
summary(bkd)
bkd.pred = predict(bkd, data.test)
test.error.bkd=mean((data.test$Total.Claim.Amount - bkd.pred)^2)
test.error.bkd

fwd=step(lm.fit,direction = "forward",trace=FALSE)
summary(fwd)
fwd.pred = predict(fwd, data.test)
test.error.fwd=mean((data.test$Total.Claim.Amount - fwd.pred)^2)
test.error.fwd

#PCR
library(pls)
set.seed(2022)
pcr.fit <- pcr(Total.Claim.Amount~., data = data.train, scale = TRUE, validation = "CV")
summary(pcr.fit)

validationplot(pcr.fit, val.type = "MSEP")

pcr.pred <- predict(pcr.fit, data.test, ncomp = 32)
error.pcr=mean((pcr.pred-data.test$Total.Claim.Amount)^2)
error.pcr
plot(pcr.pred, data.test$Total.Claim.Amount)
abline(0,1)

#PLS
set.seed(2022)
pls.fit <- plsr(Total.Claim.Amount ~ ., data =data, subset = train, scale = TRUE, valida
summary(pls.fit)
```

```
validationplot(pls.fit, val.type = "MSEP")

pls.pred <- predict(pls.fit, data.test, ncomp = 4)
error.pls=mean((pls.pred-data.test$Total.Claim.Amount)^2)
error.pls

test.error.pls.r2=1-error.pls/test.avg
test.error.pls.r2

plot(pls.pred, data.test$Total.Claim.Amount)
abline(0,1)

#lasso
library(glmnet)
set.seed(2022)
train.mat = model.matrix(Total.Claim.Amount~., data=data.train)
train.mat
test.mat = model.matrix(Total.Claim.Amount~., data=data.test)
cv.out = cv.glmnet(train.mat, data.train$Total.Claim.Amount,alpha=1)
plot(cv.out)
bestlam = cv.out$lambda.min
bestlam
lasso.mod=glmnet(train.mat, data.train$Total.Claim.Amount,alpha=1, lambda=bestlam)
summary(lasso.mod)
#lasso.mod=glmnet(train.mat, data.train$Total.Claim.Amount,alpha=1)
#plot(lasso.mod)
lasso.pred = predict(lasso.mod, s=bestlam,newx=test.mat)
test.error.lasso=mean((data.test$Total.Claim.Amount - lasso.pred)^2)
test.error.lasso
```

```
lasso.coef=predict(lasso.mod,type="coefficients",s=bestlam)
lasso.coef
test.error.lasso.r2=1-test.error.lasso/test.avg
test.error.lasso.r2

plot(lasso.pred, data.test$Total.Claim.Amount)
abline(0,1)

grid = 10^seq(10, -2, length = 100)
lasso.mod1 = glmnet(train.mat, data.train$Total.Claim.Amount, alpha = 1,lambda = grid, t
plot(lasso.mod1)




qqplot(c(1:train.size),data.test$Total.Claim.Amount,main="LASSO fit result",xlab="Sample
lines(c(1:train.size),sort(lasso.pred),col="red")
#rigde
set.seed(2022)
train.mat = model.matrix(Total.Claim.Amount~., data=data.train)
test.mat = model.matrix(Total.Claim.Amount~., data=data.test)
cv.out = cv.glmnet(train.mat, data.train$Total.Claim.Amount,alpha=0)
plot(cv.out)
bestlam = cv.out$lambda.min
bestlam
ridge.mod=glmnet(train.mat, data.train$Total.Claim.Amount,alpha=0,lambda = bestlam)
summary(ridge.mod)
ridge.pred = predict(ridge.mod,s=bestlam, newx=test.mat)
test.error.ridge=mean((data.test$Total.Claim.Amount - ridge.pred)^2)
```

```
test.error.ridge



test.error.ridge.r2=1-test.error.ridge/test.avg
test.error.ridge.r2

ridge.coef=predict(ridge.mod,type="coefficients",s=bestlam)
ridge.coef

qqplot(c(1:train.size),data.test$Total.Claim.Amount,main="Ridge fit result",xlab="Sample
lines(c(1:train.size),sort(ridge.pred),col="red")

plot(ridge.pred, data.test$Total.Claim.Amount)
abline(0,1)

ridge.mod1 = glmnet(train.mat, data.train$Total.Claim.Amount, alpha = 0,lambda = grid, t
plot(ridge.mod1)

#regression tree
set.seed(2022)
library(tree)
tree.auto = tree(Total.Claim.Amount~., data = data, subset=train)
summary(tree.auto)
plot(tree.auto)
text(tree.auto,pretty = 0)
yhat.tree = predict(tree.auto, newdata  = data.test)
error.tree=mean((yhat.tree-data.test$Total.Claim.Amount)^2)
error.tree
plot(yhat.tree,data.test$Total.Claim.Amount, ylab="Total Claim Amount")
abline(0,1)
```

```
cv.auto=cv.tree(tree.auto)
summary(cv.auto)
prune.tree=prune.tree(tree.auto,best = 8)
plot(prune.tree)
plot(cv.auto$size,cv.auto$dev,type = "b")
pyhat.tree = predict(prune.tree, newdata  = data.test)
perror.tree=mean((pyhat.tree-data.test$Total.Claim.Amount)^2)
perror.tree

plot(yhat.tree, data.test$Total.Claim.Amount)
abline(0,1)

#rf
set.seed(2022)
library(randomForest)
error.rf=rep(NA,86)
nt=c(100,200,500,1000)
n0=0
for(i in 1:19){
  for(j in 1:4){
  set.seed(2022)
    n0=n0+1
rf.auto <- randomForest(Total.Claim.Amount ~., data = data, subset=train,ntree=nt[j],mtr
yhat.rf <- predict(rf.auto, newdata = data.test)
error.rf[n0]=(mean((yhat.rf -data.test$Total.Claim.Amount)^2))
}}
error.rf

which.min=which.min(error.rf)
```

```
which.min




set.seed(2022)
rf.auto <- randomForest(Total.Claim.Amount ~., data = data, subset=train,mtry = 8, impor
yhat.rf <- predict(rf.auto, newdata = data.test)
error.rf=(mean((yhat.rf -data.test$Total.Claim.Amount)^2))
error.rf
test.error.rf.r2=1-error.rf/test.avg
test.error.rf.r2

importance(rf.auto)
varImpPlot(rf.auto,main="Importance of Parameters")



qqplot(c(1:train.size),data.test$Total.Claim.Amount,main="RF fit result",xlab="Sample nu
lines(c(1:train.size),sort(yhat.rf),col="red")

plot(yhat.rf, data.test$Total.Claim.Amount)
abline(0,1)

#boosting
set.seed(2022)
library(gbm)
error.boost=rep(0,48)
t=c(100,200,500,1000)
d=c(1,2,3,4)
s=c(0.001,0.1,0.15)
n=0
```

```
for(i in 1:4){
  for (j in 1:4) {
    for(k in 1:3){
  set.seed(2022)
  n=n+1
boost.auto = gbm(Total.Claim.Amount~., data = data.train, distribution = "gaussian", n.t
yhat.boost = predict(boost.auto, newdata  = data.test, n.trees = t[i])
error.boost[n]=mean((yhat.boost-data.test$Total.Claim.Amount)^2)
}
  }
    }
error.boost
which.min=which.min(error.boost)
which.min

set.seed(2022)
boost.auto = gbm(Total.Claim.Amount~., data = data.train, distribution = "gaussian", n.t
summary(boost.auto)
yhat.boost = predict(boost.auto, newdata  = data.test, n.trees = 100)
error.boost=mean((yhat.boost-data.test$Total.Claim.Amount)^2)
error.boost
test.error.boost.r2=1-error.boost/test.avg
test.error.boost.r2

plot(boost.auto, i = "Location.Code",ylab = "claim count")
plot(boost.auto, i = "Monthly.Premium.Auto",ylab = "claim count")

qqplot(c(1:train.size),data.test$Total.Claim.Amount,main="Boosting fit result",xlab="Sam
lines(c(1:train.size),sort(yhat.boost),col="red")
```

```r
plot(yhat.boost, data.test$Total.Claim.Amount)
abline(0,1)
#BART
set.seed(2022)
library(BART)
x <- data[, c(1:17,19:20)]
y <- data[, "Total.Claim.Amount"]
xtrain <- x[train, ]
ytrain <- y[train]
xtest <- x[-train, ]
ytest <- y[-train]
set.seed(2022)
bartfit <- gbart(xtrain, ytrain, x.test = xtest)
yhat.bart <- bartfit$yhat.test.mean
error.bart=mean((ytest - yhat.bart)^2)
error.bart
#we check how many times each variable appeared in the collection of trees
ord <- order(bartfit$varcount.mean, decreasing = T)
bartfit$varcount.mean[ord]

plot(yhat.bart, data.test$Total.Claim.Amount)
abline(0,1)
```