Case study for Bayesian Statistics

吴文韬

10165000225

Contents

1 Data description **3**

2 Exploratory data analysis **4**

2.1 Summary of numerical variables 4

2.2 Summary of qualitative variables 5

2.3 Data preprocessing 6

3 Method used in data analysis **9**

3.1 Naive Bayes classifier 9

3.2 Xgboost algorithm 9

3.3 Random Forest algorithm 10

3.4 Bayesian inference based on MCMC 10

4 Main results **11**

4.1 Naive Bayes 11

4.2 Using Xgboost and RandomForest to select variables 13

4.3 MCMC 16

5 Conclusions and remarks **20**

6 References **22**

7 Appendix: R code **22**

Abstract:

World Health Organization has estimated 12 million deaths occur worldwide, every year due to Heart diseases. Half the deaths in the United States and other developed countries are due to cardio vascular diseases. The early prognosis of cardiovascular diseases can aid in making decisions on lifestyle changes in high risk patients and in turn reduce the complications. This research intends to pinpoint the most relevant/risk factors of heart disease as well as predict the overall risk using logistic regression Data Preparation. So we want to use OpenBugs to establish the Logistics regression model to predict it.

For this artical, in first part we discribe the data set, which has 4238 obvservations and 16 variables, the variable TenYearCHD is our target. In the second part, we give data a exploratory analysis, containing summary table, correlation plot and barplot for differnet type of variables. And then we do the Data preprocessing, using RandomForest to impute the missing values, remove a variable and subsample the data. We let 2/3 of the original data be the training set, the other be the test set.

The third part is about the method we about to use, and in the fourth part we create the model based on Naive Bayes and MCMC. Before using OpenBugs, we using Xgboost and RandomForest to select important variables glucose, age, sysBP, BMP and diaBP, and further binning into 32 observations as new data which is used in OpenBugs. We assume the distribution of parameters are normal, and give the estimate of them to build MCMC model.

The final part is to compare these classifiers. We calculate Accuracy and Kappa value for the four model: NaiveBayes, MCMC, Xgboost and RandomForest. The result is RandomForest model is the best, while MCMC model is not as good as expected. The reasons might be wrong variable selecting, wrong further binning and improper prior distribution.

1 Data description

Source The dataset is publically available on the Kaggle website, and it is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The classification goal is to predict whether the patient has 10-year risk of future coronary heart disease (CHD).The dataset provides the patients' information. It includes over 4,000 records and 15 attributes. Each attribute is a potential risk factor. There are both demographic, behavioral and medical risk factors.

表 1: Variables and explanation

Variable name	Type	Explanation
Sex	factor	Male for '1';Female for '0'
Age	numeric	Age of the patient(Continuous)
Education	factor	The education level of the patient: 1 = Some High School;2 = High School or GED; 3 = Some College or Vocational School;4 = college
Current Smoker	factor	Whether or not the patient is a current smoker: 0 = nonsmoker; 1 = smoker
Cigs Per Day	numeric	The number of cigarettes that the person smoked on average in one day
BP Meds	factor	Whether or not the patient was on blood pressure medication: 0 = no;1 = yes
Prevalent Stroke	factor	Whether or not the patient had previously had a stroke: 0 = no;1 = yes
Prevalent Hyp	factor	Whether or not the patient was hypertensive: 0 = no;1 = yes
Diabetes	factor	Whether or not the patient had diabetes:0 = no;1 = yes
Tot Chol	numeric	Total cholesterol level(mg/dL)
Sys BP	numeric	Systolic blood pressure(mmHg)
Dia BP	numeric	Diastolic blood pressure(mmHg)
BMI	numeric	Body Mass Index[Weight(kg)/Height(meter-squared)]
Heart Rate	numeric	Heart rate[Beats/Min(Ventricular)]
Glucose	numeric	Glucose level(mg/dL)
TenYearCHD*	factor	10 year risk of coronary heart disease CHD:0 = no;1 = yes

***** is the variable we interest, in other words, **dependent variable**

And we have some notes for some variables:

i. For variable Age: Although the recorded ages have been truncated to whole numbers, the concept of age is continuous.

ii. For variable Cigs per day: It can be considered continuous as one can have any number of cigarettes, even half a cigarette.

iii. For variable Heart Rate: In medical research, variables such as heart rate though in fact discrete, yet are considered continuous because of large number of possible values.

2 Exploratory data analysis

2.1 Summary of numerical variables

We give the summary table of numerical variables:

表 2: Summary of numerical variables

Variable name	Mean	Sd	Median	Min	Max	1st Qu	3rd Qu	NA's
Age	49.58	8.57	49	32	70	42	56	0
Cigs Per Day	9	11.92	0	0	70	0	20	29
Tot Chol	236.72	44.59	234	107	696	206	263	50
Sys BP	132.35	22.04	128	83.5	295	117	144	0
Dia BP	82.89	11.91	82	48	142.5	75	89.88	0
BMI	25.8	4.08	25.4	15.54	56.8	23.07	28.04	19
Heart Rate	75.88	12.03	75	44	143	68	83	1
Glucose	81.97	23.96	78	40	394	71	87	388

Sd means Standard deviation; **NA's** means the number of missing values for the variable.

From the table above, we can find that the patient's age range is 32 to 70, and the average of age is almost 50, saying it is more evenly distributed. For cigs per day, this variable actually is related to the variable Current Smoker, the people no smoking would not smoke any cigarette per day. And for variable Glucose, this variable has many missing numbers, we have two choices to fix that: remove this variable or remove all of items that miss the data of this variable. We will talk that in data preprocessing.

We also plot the correlation plot between every two of numerical variables:

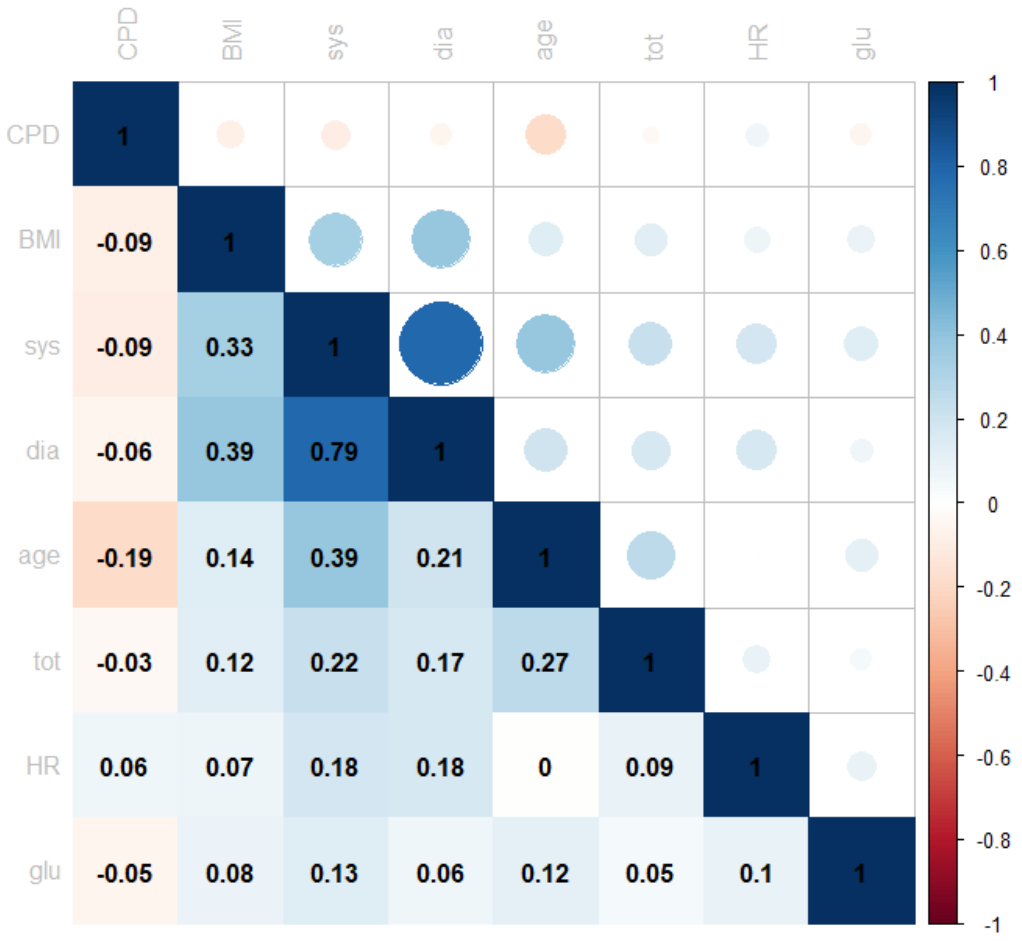


图 1: The visualization of the correlation among variables

From the correlation plot we can find that the variable sysBP and diaBP have a strong correlation, which also is within reason. Other variables every two of which do not have a strong correlation.

2.2 Summary of qualitative variables

We have 8 qualitative variables in this dataset: Sex, Education, Current Smoker, BP Meds, Prevalent Stroke, Prevalent Hyp, Diabetes and TenYearCHD. Except variable Education has 4 levels, the rest of them are binary variables, so we plot the barplot of them:

图 2: Barplots of qualitative variables

Considering these plots, we find that the numbers of 0 and 1 are not balanced, with the number of 0 is quite more than the other, for variable BPMeds, Prevalent Stroke and diabetes. The variable we are intersted in also has unbalanced number of 0 and 1, we will discuss this in data preprocessing.

2.3 Data preprocessing

First we check the missing value of the data, from the summary table above we get variables heartRate, BMI, cigsPerDay, totChol, BPMeds, education and glucose contain missing values, in total, up to 645, and the observations not completed is 582. We use function to see the missing data pattern.

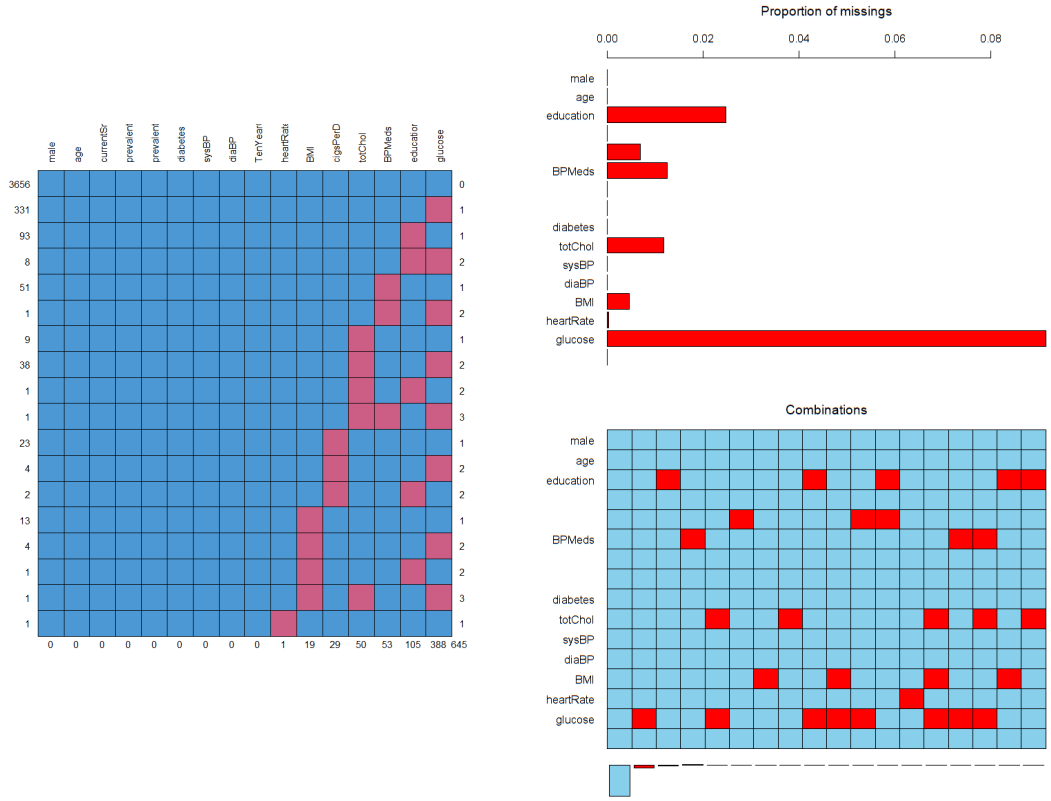


图 3: Aggregations for missing values

The missing data pattern shows that the missing type is sort of randomly scattered, so we judge the missing of data does not depend on any incomplete or complete variables, accordingly, use imputation.

Since since having 4238 records, we treat items with missing values depending on the type of variable. For numerical variables, we use knn imputation, setting 5 to the number of nearest neighbors from set to use for imputation. For qualitative variables, we take Nonparametric Missing Value Imputation using Random Forest, setting 10 to maximum number of iterations to be performed given the stopping criterion is not met beforehand, and 100 to number of trees to grow in each forest.

The first six rows of processing results shows in the following table:

```
##    cigsPerDay totChol BMI heartRate glucose
## 15          9      226  22        85      76
## 22          0      185  30        70      82
## 27          0      260  27        65      82
## 34          5      175  19        72      75
## 37          0      257  28        72      75
## 43          0      251  25        70      79

##     education BPMeds
```

```
## 34          1       0
## 37          1       0
## 50          3       0
## 73          2       0
## 86          2       0
## 185         1       0
```

Then we get data set without missing values. Next move is to remove the variables with near zero variance, but no variable has near zero variance in this data set.

And as mentioned earlier, the variable cigs per day depend to some extent on variable Current Smoker. This article offers two ideas: one is to simply remove the variable cigs per day, the other is to further binning varibale Current Smoker according to variable cigs per day. Consider reduce the work, we choose the first treatment.

Then we split the data. Since the propotion of 1 of target variable TenYearCHD is so small that we choose to create a Generation of synthetic data by Randomly Over Sampling Examples. This process is to create a sample of synthetic data by enlarging the features space of minority and majority class examples. Operationally, the new examples are drawn from a conditional kernel density estimate of the two classes, as described in Menardi and Torelli (2013).

```
library(ROSE)
set.seed(1)
data3 <- ROSE(y~., data=data2)$data
```

```
table(data3$y)
```

```
##
##    0    1
## 2196 2042
```

The final step is to use the subsample to create test/training partitions. Setting seed 1, let 2/3 of the original data be the training set, the other be the test set.

The results are shown below.

表 3: Data splitting

Data set	0	1
train	1464	1362
test	732	680

3 Method used in data analysis

3.1 Naive Bayes classifier

Naive bayes algorithm is a classification algorithm based on probability statistics, which is widely used in solving classification problems in machine learning. For classification tasks, bayesian decision theory considers how to select the optimal category markers based on these probabilities and misjudgment losses in the ideal case where all relevant probabilities are known. Bayesian criterion: to minimize the overall risk, that is, choosing the category tag on each sample that can minimize the conditional risk $R(c|\mathbf{x})$,

$$h^*(\mathbf{x}) = arg\ \underset{c\in\varphi}{min} R(c|\mathbf{x})$$

where φ is possible category tags, $\varphi = \{c_1, c_2, ..., c_N\}$, then we call h^* the best bayesian optimal classifier. In the classification problem, our goal is to minimize the classification error rate, that is, conditional risk $R(c|\mathbf{x}) = 1 - P(c|\mathbf{x})$. So we need to get a posterior probability $P(c|\mathbf{x})$. The posterior probability can be obtained by bayes formula. Here's a major difficulty: the conditional probability $P(\mathbf{x}|c)$ is a joint probability of all the attributes, so naive bayes classifier assume "attribute conditional independence hypothesis": For a given category, all attributes are assumed to be independent to each other.

Based on the assumption of attribute conditional independence, bayesian formula can be written as

$$P(c|\mathbf{x}) = \frac{P(c)\,P(\mathbf{x}|c)}{P(\mathbf{x})} = \frac{P(c)}{P(\mathbf{x})}\prod_{i=1}^{d} P(x_i|c)$$

where d is property entries, and x_i is the ith value of $\mathbf{x}$.

Since $P(\mathbf{x})$ is the same for all categories, the bayesian criterion based on the former formula is

$$h_{nb}(\mathbf{x}) = arg\ \underset{c\in\varphi}{max}\ P(c)\prod_{i=1}^{d} P(x_i|c)$$

Obviously, the training process of naive bayes classifier is to estimate the class prior probability $P(c)$ based on the training set D, and estimate the conditional probability $P(x_I|c)$ for each attribute.

3.2 Xgboost algorithm

Xgboost, which is short for "eXtreme Gradient Boosting", namely, eXtreme Gradient Boosting tree, is an improved algorithm based on GBDT, which was proposed by Chen equals in 2015.Gradient Boosting, which belongs to a category in the Boosting method of the ensemble algorithm, and the Boosting classifier, which belongs to the ensemble learning model, whose basic idea is to combine hundreds of tree models with low classification accuracy into a model with

high accuracy. Gradient Boosting by adding new weak learning to improve the existing study of residual and multiple learning add up to get the final prediction results, the concrete is when every tree generated by adopting the idea of Gradient descent, the basis of the above step to generate all the trees, to minimize the objective function given direction. XGBoost algorithm is an improved version of GBDT algorithm. Compared with GBDT, which only USES the first derivative information in optimization, XGBoost carries out the second order Taylor expansion of the loss function, making full use of the first and second derivatives, and obtains the optimal solution for the regular term besides the loss function. And Xgboost automatically leverages the CPU's multithreading for parallelism and improves the algorithm to improve accuracy.

3.3 Random Forest algorithm

Random Forest is an algorithm that integrates multiple trees through the idea of ensemble learning. Its basic unit is decision tree.On the basis of Bagging integration based on decision tree, RF further introduces random attribute selection in the training process of decision tree.In particular, the traditional decision tree when choosing partition attribute is in the current node in the collection of attributes to select a optimal properties, and in the RF, on the base of decision tree each node, start with the attribute of the node set contain a random selection of k attribute subset, and then select an optimal attribute from the subset is used to divide.Due to bagging, which is the idea of integration, the method of random forest is actually equivalent to sampling both samples and features (if the training data is regarded as a matrix, as is common in practice, it is a process in which both rows and columns are sampled), so overfitting can be avoided.

Based on bagging, random forest goes further:

1) Select N samples randomly from the sample set by Bootstrap(random sampling with relocations).
2) Select K randomly from all the attributes, and select the best segmentation attribute as the node to establish the CART decision tree (which can also be other types of classifier).
3) Repeat the above two steps m times, namely, establish m CART decision trees.
4) These m carts form a random forest and decide which kind of data belongs to through voting results.

3.4 Bayesian inference based on MCMC

Markov chain Monte Carlo (MCMC) methods. This essentially is a continuousvalued generalization of the discrete Markov chain setup described in the previous section. The MCMC sampling strategy sets up an irreducible, aperiodic Markov chain for which the stationary distribution equals the posterior distribution of interest.

One of the attractive methods for setting up an MCMC algorithm is Gibbs sampling. Suppose that the parameter vector of interest is $\theta = (\theta_1, \cdots, \theta_p)$. The joint posterior distribution of θ, which we denote by $[\theta \mid data]$, may be of high dimension and difficult to summarize. Suppose we define the set of conditional distributions.

$$[\theta_1|\theta_2, ..., \theta_p]$$

$$[\theta_2|\theta_1, \theta_3, ..., \theta_p]$$

$$...$$

$$[\theta_p|\theta_1, \theta_2, ..., \theta_{p-1}]$$

where $[X \mid Y, Z\ ]$ represents the distribution of X conditional on values of the random variables Y and Z. The idea behind Gibbs sampling is that we can set up a Markov chain simulation algorithm from the joint posterior distribution by successfully simulating individual parameters from the set of p conditional distributions. Simulating one value of each individual parameter from these distributions in turn is called one cycle of Gibbs sampling. Under general conditions, draws from this simulation algorithm will converge to the target distribution (the joint posterior of $\mid \theta\ )$ of interest.

In this articel, we use OpenBugs to realize MCMC based on Gibbs sampling.

4 Main results

4.1 Naive Bayes

In simple naive bayesian modeling, the optimal parameter needed to be found is usekernel=FALSE or TRUE, indicating whether nuclear density estimation is used, and if TRUE, nuclear density estimation is used.If false, normal density is used to estimate; And parameter fL=0 or 1, indicating whether to use the Laplace correction, not if it is 0, and not if it is 1. This article uses the method of Leave-One-Out-Cross-Validation to find the excellent solution of these two parameters, and the error rate when taking different parameters is arranged into the table as follows:

表 4: Error rate table

usekernel \ fL	0	1
TRUE	0.329	0.317
FALSE	0.329	0.3174

It can be seen from the above table that the optimal parameter is usekernel = TRUE and fL = 0 with the minimum error rate. Use them to build Naive bayes model, and use it to predict the value of variable we care in the test set. Consturct the confusionmatrix by comparing the predicted results with true values in the test set, as shown in the following table:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 518 241
##          1 214 439
##
##                Accuracy : 0.6778
##                  95% CI : (0.6527, 0.7021)
##     No Information Rate : 0.5184
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.3537
##
##  Mcnemar's Test P-Value : 0.2229
##
##             Sensitivity : 0.6456
##             Specificity : 0.7077
##          Pos Pred Value : 0.6723
##          Neg Pred Value : 0.6825
##              Prevalence : 0.4816
##          Detection Rate : 0.3109
##    Detection Prevalence : 0.4625
##       Balanced Accuracy : 0.6766
##
##        'Positive' Class : 1
##
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6455882            0.7076503            0.6722818
##       Neg Pred Value            Precision               Recall
##            0.6824769            0.6722818            0.6455882
##                   F1           Prevalence       Detection Rate
```

```
##            0.6586647            0.4815864            0.3109065
## Detection Prevalence    Balanced Accuracy
##            0.4624646            0.6766193
```

4.2 Using Xgboost and RandomForest to select variables

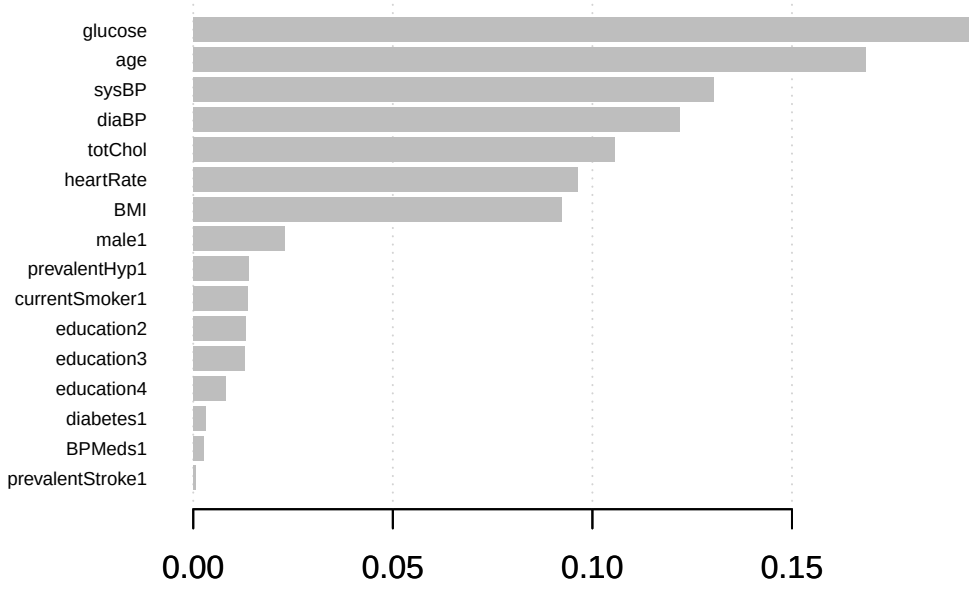
Since to establish a logistics model, we need further process the data. In fact, each variable is of different importance to the variable we care about. Therefore, we first process the data through Xgboost algorithm and RandomForest algorithm to find some important variables.

4.2.1 Xgboost to select variables

For Xgboost modeling, there are a lot of parameters to set, since author does not have a deep understanding of Xgboost, it is difficult to do parameter optimization accurately. The parameters taken here are obtained by the experience, and trying different parameters to compare so that find the relatively superior parameters. Since the problem involved in this paper is a binary classification problem, and there are factor-type variables in the independent variables, the training set and the test set should be transformed into Design Matrices. The following table shows some of the parameters to be optimized in Xgboost:

* booster: which booster to use, can be gbtree or gblinear. In this case, we choose gbtree.
* nrounds: max number of boosting iterations. In this case we take 200.
* min_child_weight: minimum sum of instance weight (hessian) needed in a child. The larger, the more conservative the algorithm will be. In this case, we take 0.1.
* max_depth: maximum depth of a tree. In this case, we take 5.
* subsample: subsample ratio of the training instance. It makes computation shorter. In this case, we take 1.
* colsample_bytree: subsample ratio of columns when constructing each tree. In this case, we take 1.
* objective: specify the learning task and the corresponding learning objective, users can pass a self-defined function to it. In this case, we take "binary:logistic".

Using the parameters above, the obtained variable importance is shown in the figure below:



4.2.2 RandomForest to select variables

For random forest modeling, there are also many parameters need to set. In this paper, only a few of them are selected for optimization.

* ntree: Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
* mtry: Number of variables randomly sampled as candidates at each split.
* importance: Should importance of predictors be assessed?
* nodesize: Minimum size of terminal nodes. In this case, we take 1.
* maxnodes: Maximum number of terminal nodes trees in the forest can have. In this case, we take 2ˆ10

The selection of ntree and mtry can be referred to in the figure below:

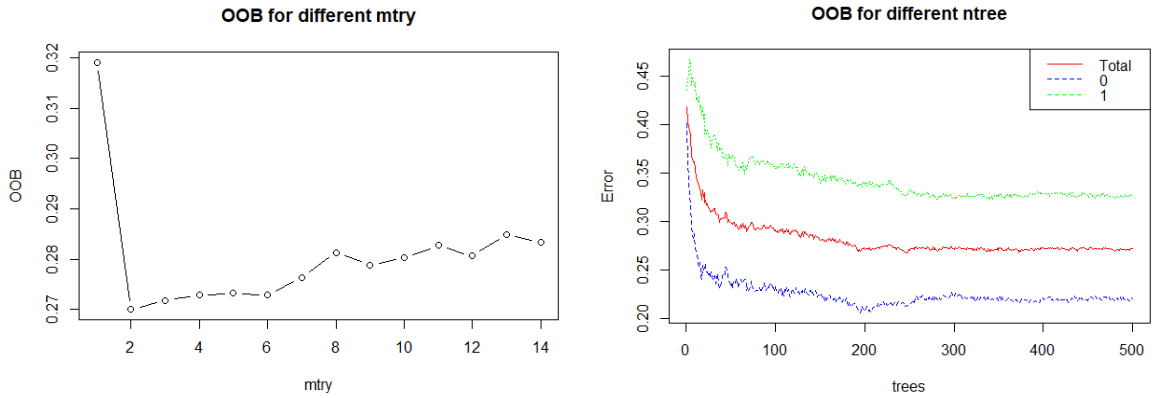
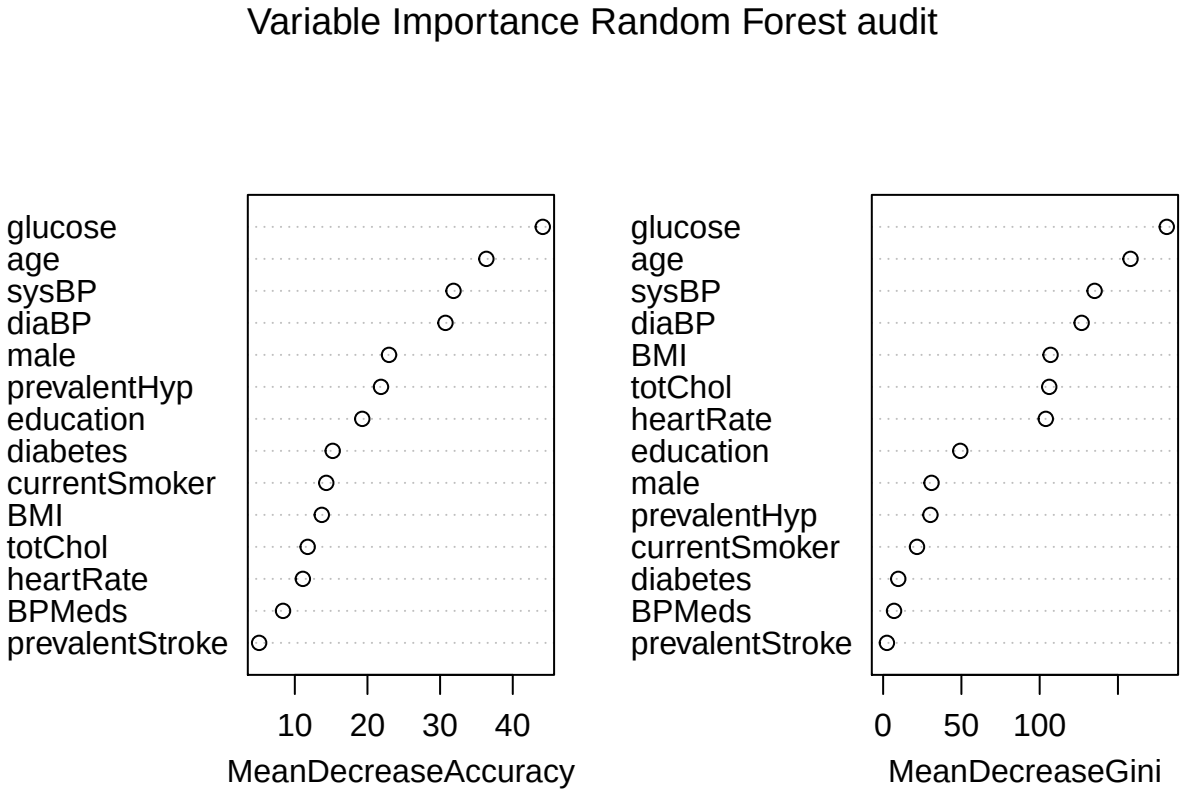


图 4: Aggregations for missing values

By observing the above figure, we can see that OOB is minimum when mtry = 2. Looking at the relationship between the ntrees and the OOB error rate, red line represents the overall OOB error rate, blue line and green line represent the OOB error rate of "no" and "yes" respectively. Code result display the value of ntree making OOB minimum under mtry = 2 is about 253.

Using these parameters to build RandomForest model, and get the plot of variable importance which as follows:



4.3 MCMC

From the result above, we finally choose five variables as independent variables: glucose, age, sysBP, BMP and diaBP. Using the training set, we binning every independent variables into two categories:

表 5: Binning reference

Variable name	0	1
age	age $\leqslant 50$	> 50
sysBP	$90 <$ sysBP < 140	$\leqslant 90$ *or* $\geqslant 140$
diaBP	$60 <$ diaBP < 90	$\leqslant 60$ *or* $\geqslant 90$
BMI	$18.5 <$ BMI < 23.9	$\leqslant 18.5$ *or* $\geqslant 23.9$
glucose	$70 <$ glucose < 120	$\leqslant 70$ *or* $\geqslant 120$

For variable age, the critical value of partition is 50, according to the statistical regularity that people over 50 years old have a significantly increased risk of heart disease. For other variables, the critical value of partition is taken from a normal range of adults. There are five independent variables each of which has two categories, therefore, we can organize the training set with 2826 observations into a data set with only $2^5 = 32$ observations:

表 6: The new data set

Observation	age	sysBP	diaBP	BMI	glucose	n_i	N_i
1	0	0	0	0	0	47	232
2	0	0	0	0	1	31	84
...	...	...	...	...	...	...	...
31	1	1	1	1	0	151	233
32	1	1	1	1	1	106	145

Then we can use this data set to create a logistics regression model:

$$n_i \sim Binomial\,(p_i, N_i) \tag{1}$$

$$logit\,(p_i) = \alpha_0 + \sum_{i=1}^{5} \alpha_i x_{ij}, \quad j = 1, ..., 32. \tag{2}$$

Where $x_1 \sim x_5$ represent age, sysBP, diaBP, BMP and glucose respectively. And ausume $\alpha_0 - \alpha_5 \sim N(\mu, \sigma^2)$.

Use OpenBugs to build Bayesian models. First we plot the model:

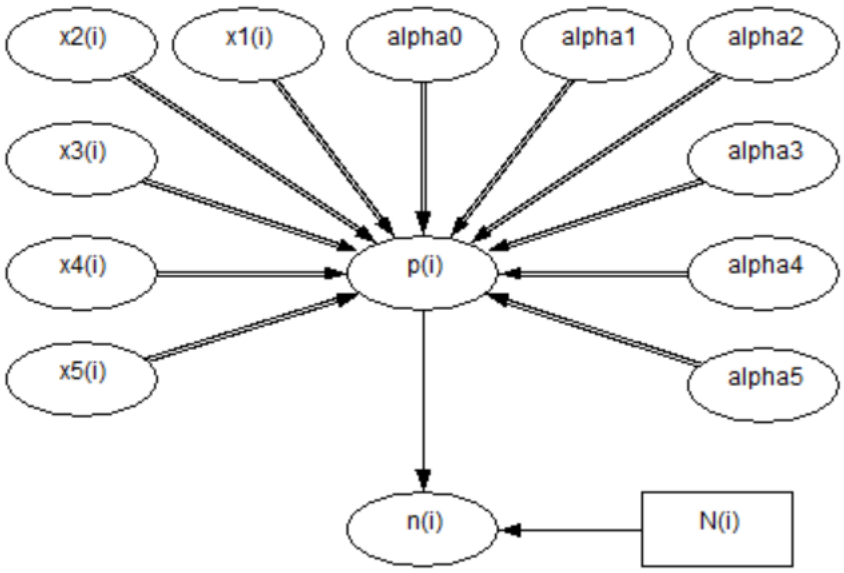


图 5: Logistics regression model

Then load data and initial values = list(alpha0 = 0, alpha1 = 0, alpha2 = 0, alpha3 = 0, alpha4 = 0, alpha5 = 0). Consider the stability of the model, we carry out 10000 MCMC updates, and store 50000 updates into MCMC simulation, and check the result:

表 7: Node statistics

Parameter	mean	sd	MC_error	val2.5pc	median	val97.5pc	start	sample
alpha0	-1.125	0.09232	4.344E-4	-1.308	-1.124	-0.9458	10001	50000
alpha1	0.8821	0.0828	3.838E-4	0.7191	0.8824	1.044	10001	50000
alpha2	0.5285	0.09128	4.219E-4	0.3504	0.5282	0.7085	10001	50000
alpha3	0.4181	0.09353	4.559E-4	0.2349	0.4179	0.6008	10001	50000
alpha4	0.009286	0.0894	4.323E-4	-0.1664	0.008747	0.1843	10001	50000
alpha5	0.6427	0.0851	4.289E-4	0.4743	0.6433	0.8086	10001	50000

The following plots are examples of: (i) chains for all parameters (left-hand-side); and (ii) Auto-correlation plot (right-hand-side).

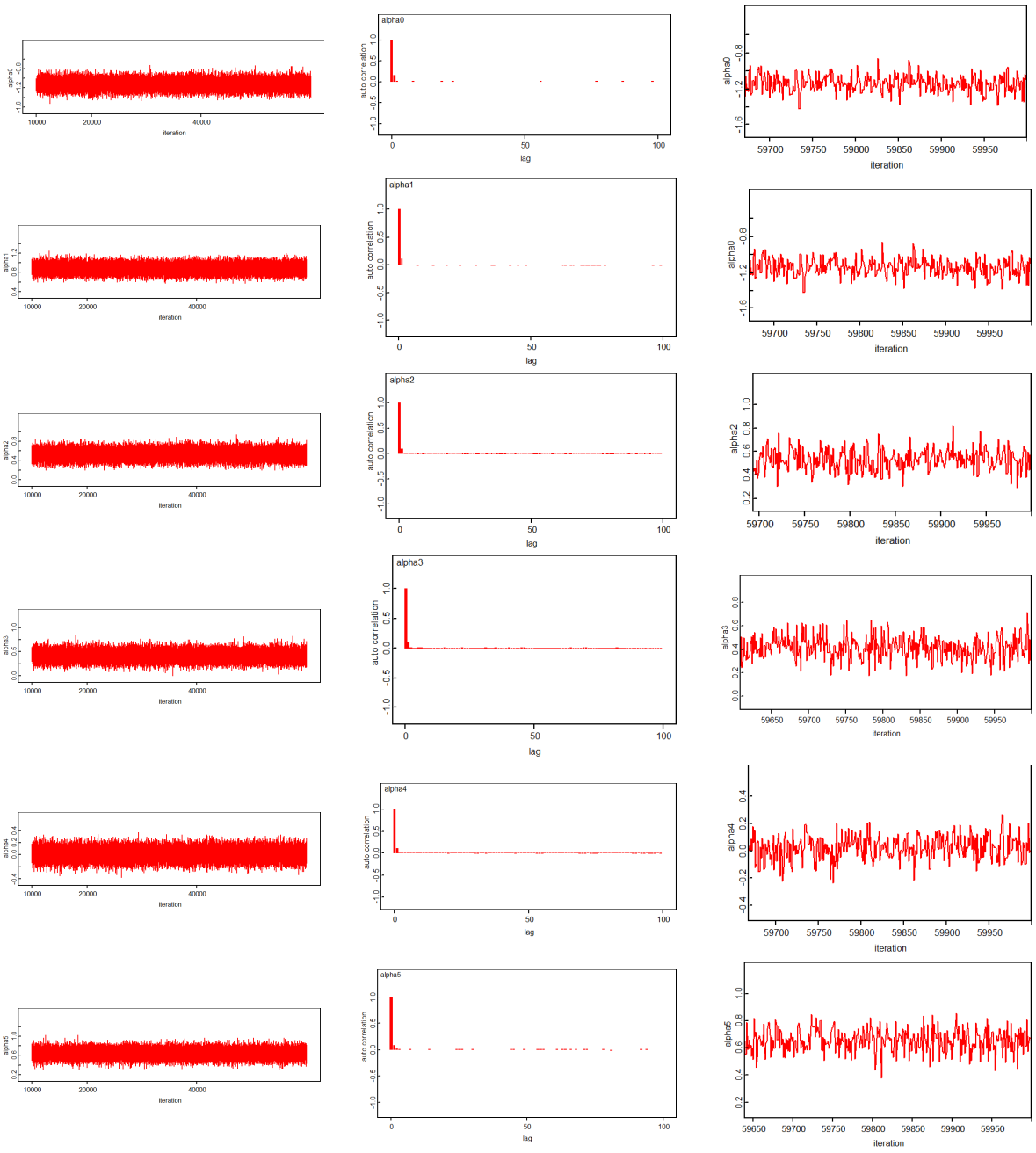


图 6: Chains and auto-correlation

The stat table produces summary statistics for the variable, pooling over the chains selected. The required percentiles can be selected using the percentile selection box. The quantity reported in the MC error column gives an estimate of $\sigma/N^{1/2}$, the Monte Carlo standard error of the mean. The batch means method outlined by Roberts (1996; p.50) is used to estimate σ.

The history chain, auto-correlation and trace plot can help us check convergence. When trace and history chain tend to be stable, it can be considered that the model converges. From the graphics avove for each parameter, we can think of it as stable. And another way to judge the convergence is that if the MC error of parameter estimation is less than 3% of its standard

deviation, it means that it has converged. Obviously our results are consistent with this criterion according to the stat table.

So we get the logistics model:

$$logit\,(p_i) = -1.125 + 0.8821x_1 + 0.5285x_2 + 0.4181x_3 + 0.009286x_4 + 0.6427x_5$$

$$n_i \sim Binomial\,(p_i, N_i)$$

Then we use this model to calculate the probability for each obvservation in test set, and classify them into 0 and 1 so that we can create the confusion matrix.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 503 264
##          1 229 416
##
##                Accuracy : 0.6508
##                  95% CI : (0.6253, 0.6757)
##     No Information Rate : 0.5184
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.2995
##
##  Mcnemar's Test P-Value : 0.1257
##
##             Sensitivity : 0.6118
##             Specificity : 0.6872
##          Pos Pred Value : 0.6450
##          Neg Pred Value : 0.6558
##              Prevalence : 0.4816
##          Detection Rate : 0.2946
##    Detection Prevalence : 0.4568
##       Balanced Accuracy : 0.6495
##
##        'Positive' Class : 1
##
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6117647            0.6871585            0.6449612
##       Neg Pred Value            Precision               Recall
##            0.6558018            0.6449612            0.6117647
##                   F1           Prevalence       Detection Rate
##            0.6279245            0.4815864            0.2946176
## Detection Prevalence    Balanced Accuracy
##            0.4567989            0.6494616
```

5 Conclusions and remarks

We also use Xgboost and RandomForest to predict the test set, and compare the confusion-matrix:

表 8: Comparison of four models

index \ model	NaiveBayes	MCMC	RandomForest	Xgboost
Accuracy	0.677	0.651	0.721	0.6891
Precision	0.672	0.645	0.725	0.679
Sensitivity	0.645	0.612	0.676	0.6721
Specificity	0.707	0.687	0.762	0.7049
F1 Score	0.658	0.628	0.700	0.675
Kappa	0.353	0.300	0.440	0.3771
AUC	0.761	0.701	0.805	0.756

We also draw the index images of these four models according to the forecast situation:

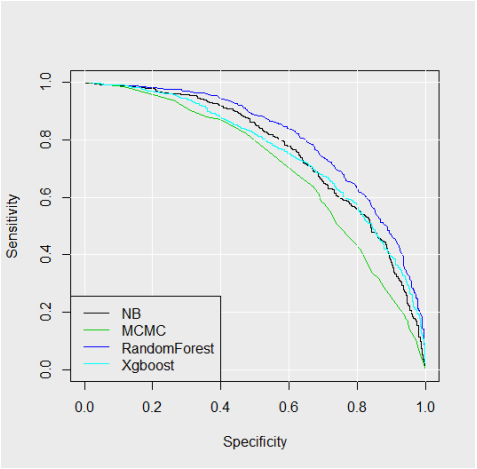


图 7: Sensitivity-Specifity Curves

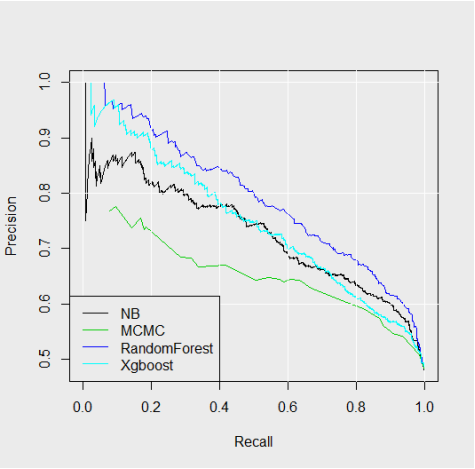


图 8: Recall-Precision Curves

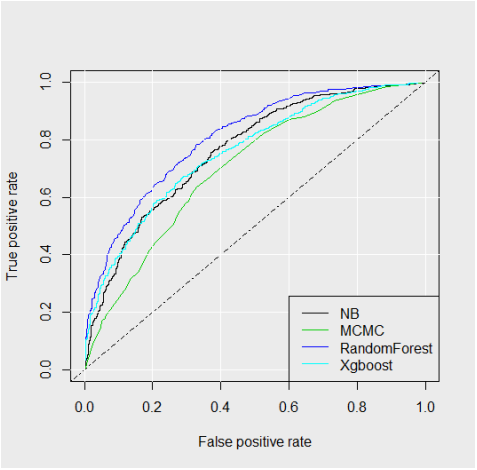


图 9: ROC

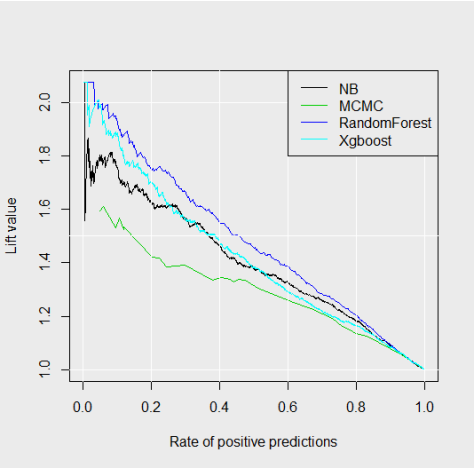


图 10: Lift Charts

图 11: Index Images

By observing the index table, from the point of view of Kappa value, RandomForest model has the highest Kappa value, then Xgboost, two of which have good consistency, and the Kappa value of MCMC is the lowest. In terms of accuracy, RandomForest and Xgboost models are the most accurate. The probability of prediction error is relatively small. MCMC model has the lowest prediction accuracy, but it is also within the acceptable range.

From index lines, we can see that for ROC curve, the curves of four models are above the diagonal line, so that the effect is superior to random classifier, and obviously ROC curve of RF is over the rest of the three curves, which means RF model is superior to other models. And MCMC model perfrom not good.

Analyze why MCMC does not have a good classification effect, the author thinks there are several reasons:

* We didn't select enough important variables, which lead us miss some important information.
* The subsampling(ROSE) didn't offer a positive help.

* The futher binning method is improper.
* The distribution of parameters are not normal.

And our next goal is to try more variables and new binning method, and we will figure out the distribution of different parameters for prior, maybe non-informative priors or exponential.

6 References

[1] 张继巍, 高文龙, 秦天燕, 刘建正, 李学朝, 拉扎提木拉提, 李娟生.OpenBUGS 软件介绍及应用 [J]. 中国卫生统计,2017,34(01):170-172+176.

[2] 方兴华, 宋明顺, 张月义, 顾龙芳. 非共轭先验分布下的贝叶斯统计质量控制研究 [J]. 工业工程与管理,2012,17(05):72-75.

[3] 胡仕强. 基于贝叶斯 MCMC 方法的我国人口死亡率预测 [J]. 保险研究,2015(10):70-83.

[4] 姚丹丹, 雷相东, 张则路. 基于贝叶斯法的长白落叶松林分优势高生长模型研究 [J]. 北京林业大学学报,2015,37(03):94-100.

[5] WinBUGS User Manual

[6] OpenBUGS User Manual,Version 3.2.1.

[7] https://www.kaggle.com/naveengowda16/logistic-regression-heart-disease-prediction

[8] https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[9] https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset/data

7 Appendix: R code

```
library(mice);library(VIM);library(klaR);library(ROCR);library(pROC)
library(randomForest);library(rpart);library(rpart.plot)
setwd('C:\\Users\\lenovo\\Desktop')
data <- read.csv('framingham_heart_disease.csv',header = T)
colnames(data)[16] <- 'y'
str(data)
table(data$y)
#  调整变量类型
data$male <- factor(data$male)
data$education <- factor(data$education)
data$currentSmoker <- factor(data$currentSmoker)
data$prevalentStroke <- factor(data$prevalentStroke)
data$BPMeds <- factor(data$BPMeds)
data$prevalentHyp <- factor(data$prevalentHyp)
data$diabetes <- factor(data$diabetes)
```

```r
#data$TenYearCHD <- factor(data$TenYearCHD)
data$y <- factor(data$y)
str(data)
summary(data)
# 数值型变量 描述性统计
library(psych)
describe(data,quant = c(.25,.75))
# 相关性可视化
library(corrplot)
# 要去掉所有缺失值
data1.1 <- data[,c(2,5,10,11,12,13,14,15)]
str(data1.1)
data1.1 <- data1.1[complete.cases(data),]
colnames(data1.1) <- c('age','CPD','tot','sys','dia','BMI','HR','glu')
# 数值型变量为 2,5,10,11,12,13,14,15
data.corr <- cor(data1.1);data.corr
corrplot(data.corr,type = 'upper',method = 'circle',tl.col = 'grey',order = 'hclust',
         tl.pos = 'lt',diag = T)
corrplot(data.corr,type = 'lower',method="color",addCoef.col="black",add = T,
         order = 'hclust',tl.pos = "n",tl.col = 'grey',cl.pos = "n",diag = T)
#   描述性统计
library(ggplot2)
ggplot(data=data, mapping=aes(x=male,fill=male))+
  geom_bar(stat="count",width=0.6)+
  scale_color_manual(values=c("#999999", "#E69F00"))+
  geom_text(stat='count',aes(label=..count..), vjust=1.6, color="white", size=3.5)+
  theme_minimal()+
  labs(title = 'The barplot of variable male')+
  theme(plot.title = element_text(size = 14, hjust = .5, color = "gray30"))
ggplot(data=data, mapping=aes(x=education,fill=education))+
  geom_bar(stat="count",width=0.6)+
  scale_color_manual(values=c("#999999", "#E69F00"))+
  geom_text(stat='count',aes(label=..count..), vjust=1.6, color="white", size=4)+
  theme_minimal()+
  labs(title = 'The barplot of variable education')+
  theme(plot.title = element_text(size = 18, hjust = .5, color = "gray30"))
```

```
ggplot(data=data, mapping=aes(x=currentSmoker,fill=currentSmoker))+
  geom_bar(stat="count",width=0.6)+
  scale_color_manual(values=c("#999999", "#E69F00"))+
  geom_text(stat='count',aes(label=..count..), vjust=1.6, color="white", size=4)+
  theme_minimal()+
  labs(title = 'The barplot of variable currentSmoker')+
  theme(plot.title = element_text(size = 18, hjust = .5, color = "gray30"))
ggplot(data=data, mapping=aes(x=BPMeds,fill=BPMeds))+
  geom_bar(stat="count",width=0.6)+
  scale_color_manual(values=c("#999999", "#E69F00"))+
  geom_text(stat='count',aes(label=..count..), vjust=1.6, color="white", size=4)+
  theme_minimal()+
  labs(title = 'The barplot of variable BPMeds')+
  theme(plot.title = element_text(size = 18, hjust = .5, color = "gray30"))
ggplot(data=data, mapping=aes(x=prevalentStroke,fill=prevalentStroke))+
  geom_bar(stat="count",width=0.6)+
  scale_color_manual(values=c("#999999", "#E69F00"))+
  geom_text(stat='count',aes(label=..count..), vjust=1.6, color="white", size=4)+
  theme_minimal()+
  labs(title = 'The barplot of variable prevalentStroke')+
  theme(plot.title = element_text(size = 18, hjust = .5, color = "gray30"))
ggplot(data=data, mapping=aes(x=prevalentHyp,fill=prevalentHyp))+
  geom_bar(stat="count",width=0.6)+
  scale_color_manual(values=c("#999999", "#E69F00"))+
  geom_text(stat='count',aes(label=..count..), vjust=1.6, color="white", size=4)+
  theme_minimal()+
  labs(title = 'The barplot of variable prevalentHyp')+
  theme(plot.title = element_text(size = 18, hjust = .5, color = "gray30"))
ggplot(data=data, mapping=aes(x=diabetes,fill=diabetes))+
  geom_bar(stat="count",width=0.6)+
  scale_color_manual(values=c("#999999", "#E69F00"))+
  geom_text(stat='count',aes(label=..count..), vjust=1.6, color="white", size=4)+
  theme_minimal()+
  labs(title = 'The barplot of variable diabetes')+
  theme(plot.title = element_text(size = 18, hjust = .5, color = "gray30"))
ggplot(data=data, mapping=aes(x=TenYearCHD,fill=TenYearCHD))+
```

```
  geom_bar(stat="count",width=0.6)+
  scale_color_manual(values=c("#999999", "#E69F00"))+
  geom_text(stat='count',aes(label=..count..), vjust=1.6, color="white", size=4)+
  theme_minimal()+
  labs(title = 'The barplot of variable TenYearCHD')+
  theme(plot.title = element_text(size = 18, hjust = .5, color = "gray30"))
#   数据预处理
sum(is.na(data))
sum(!complete.cases(data)) # 查看空缺行
md.pattern(data,rotate.names = T)  # 查看缺失模式
aggr(data,prop=T,numbders=T)
# 直接用随机森林插补全部缺失
mf1 <- missForest(data, maxiter = 5)  #用随机森林迭代弥补缺失值
data2 <- mf1$ximp # 得到插补后的数据
data2 <- data2[,-which(colnames(data2)%in%c('cigsPerDay'))]
#  子抽样
library(ROSE)
set.seed(1)
data3 <- ROSE(y~., data=data2)$data
table(data3$y)
# 划分
coly <- which(colnames(data3)%in%c('y'))
set.seed(1)
trainIndex <- createDataPartition(data3$y , p=2/3 ,list=F,times=1)
train <- data3[trainIndex,]
test <- data3[-trainIndex,]
#  Naive Bayes
#留一交叉验证寻找最优参数usekernel和fL
pred.NB3 <- factor(rep("2",nrow(train)*4),levels=c("0","1"))
ntrain <- nrow(train)
istart <- -1
for (ifL in 0:1) {
  for (iusekernel in 0:1) {
    istart <- istart+1
    for (i in 1:nrow(train)){
      NB.model <- NaiveBayes(y ~ ., train[-i,],usekernel=iusekernel,fL=ifL)
```

```
      pred.NB <- predict(NB.model,train[i,-coly])
      pred.NB3[istart*ntrain+i] <- pred.NB$class
    }
  }
}
NB.accuracy <- c(rep(0,4))
for (i in 1:4) {
  pred.NB4 <- pred.NB3[((i-1)*nrow(train)+1):(i*nrow(train))]
  NB.measure <- confusionMatrix(pred.NB4,train[,coly],positive="1")
  NB.accuracy[i] <- NB.measure$overall[1]
}
NB.accuracy
#把上述结果整理为一个表格
para.table <- matrix(rep(NA,4),nrow=2,ncol=2)
colnames(para.table)=c("usekernel=F","usekernel=T")
rownames(para.table)=c("fL=0","fL=1")
para.table[1,1]=1-NB.accuracy[1]
para.table[1,2]=1-NB.accuracy[2]
para.table[2,1]=1-NB.accuracy[3]
para.table[2,2]=1-NB.accuracy[4]
para.table #结果显示usekernel=T, fL=0或1为最优
# 使用最优参数建模
NB.model <- NaiveBayes(y ~ ., train, usekernel=T,fL=0)
pred.NB.test <- predict(NB.model,test[,-coly])
con.NB <- confusionMatrix(pred.NB.test$class,test[,coly],positive="1")
con.NB
con.NB$byClass
#  Random forest
# 确定随机森林的最优参数
p <- ncol(train)-1
err <- rep(0,p)
B <- rep(0,p)
for (i in 1:p){
  set.seed(1)
  rfmodel <- randomForest(y ~ .,data=train,ntree=500,mtry=i,nodesize=1,maxnodes=2^10)
  err[i] <- min(rfmodel$err.rate[,1])
```

```
  B[i] <- which.min(rfmodel$err.rate[,1])
}
err
B
mtry.optimal <- which.min(err)
ntree.optimal <- B[which.min(err)]
c(mtry.optimal,ntree.optimal)
#用最优参数建模
set.seed(1)
model.rf <- randomForest(y ~ .,data=train,ntree=500,mtry=mtry.optimal,nodesize=1,maxnodes=
c(min(model.rf$err.rate[,1]),which.min(model.rf$err.rate[,1]))
plot(1:14,err,xlab = 'mtry',type = 'b',ylab = 'OOB',main = 'OOB for different mtry')
plot(model.rf,col=c("red","blue","green"),main = 'OOB for different ntree')
legend("topright",c("Total","0","1"),lty=c(1,2,2),col=c("red","blue","green"))
model.rf$importance#各变量的重要程度
varImpPlot(model.rf, main="Variable Importance Random Forest audit")#作图
# 预测
pred.rf <- predict(model.rf,test)
con.RF <- confusionMatrix(pred.rf,test[,coly],positive="1")
con.RF
con.RF$byClass
#                                XGboost
library(xgboost)
# xgboost仅接受数值型变量，故对因子型变量要先做one hot编码
trainx <- model.matrix(~.,data=train[,-coly])[,-1]
trainy <- as.numeric(as.factor(train[,coly]))-1 #将因变量转化为numeric,且取值为0和1
testx<- model.matrix(~.,data=test[,-coly])[,-1]
param <- list(seed=1,objective="binary:logistic",subsample = 1,max_depth=5,
              colsample_bytree = 1,min_child_weight=.1,eval_metric ="auc")#,gamma = 5)
#建模
model.xgb <- xgboost(data=trainx,label=trainy,params=param,nrounds=200)
names <- dimnames(data.matrix(trainx))[[2]]#获得特征的真实名称
importance_matrix <- xgb.importance(names, model = model.xgb)#计算特征重要性矩阵
xgb.plot.importance(importance_matrix)#变量重要性作图
#预测
pred.xgb <- predict(model.xgb,testx)
```

```
pred.xgboost <- ifelse(pred.xgb<.5,0,1)
pred.xgboost <- factor(pred.xgboost);levels(pred.xgboost) <- c('0','1')
con.XB <- confusionMatrix(pred.xgboost,test$y,positive = '1')
con.XB
con.XB$byClass
#                      MCMC
data <- train
#   根据上面的结果 我们选择的变量为glucose, age, sysBP, BMI, diaBP
data.fin <- data[,which(colnames(data)%in%c('glucose','age','sysBP','BMI','diaBP','y'))]
summary(data.fin)
# 尝试分箱
data.b <- data.fin
data.b$age <- factor(ifelse(data.b$age<=50,'0','1'))
data.b$sysBP <- factor(ifelse(data.b$sysBP<140&data.b$sysBP>90,'0','1'))
data.b$diaBP <- factor(ifelse(data.b$diaBP<90&data.b$diaBP>60,'0','1'))
data.b$BMI <- factor(ifelse(data.b$BMI<23.9&data.b$BMI>18.5,'0','1'))
data.b$glucose <- factor(ifelse(data.b$glucose<120&data.b$glucose>70,'0','1'))
str(data.b)
N <- c(); fin <- matrix(rep(0,7*32),ncol = 7)
obj <- c();n <- 1;pro <- c()
lel <- c('0','1')
for(i in 1:2){for(j in 1:2){for(k in 1:2){for(l in 1:2){for(m in 1:2){
  N[n] <- sum(data.b$age==lel[i]&data.b$sysBP==lel[j]&data.b$diaBP==lel[k]&
                        data.b$BMI==lel[l]&data.b$glucose==lel[m])
  pro[n] <- sum(data.b$age==lel[i]&data.b$sysBP==lel[j]&data.b$diaBP==lel[k]&
                          data.b$BMI==lel[l]&data.b$glucose==lel[m]&data.b$y=='1')
  names(pro)[n] <- paste('(',i-1,',',j-1,',',k-1,',',l-1,',',m-1,')',sep = '')
  names(N)[n] <- paste('(',i-1,',',j-1,',',k-1,',',l-1,',',m-1,')',sep = '')
  fin[n,1]<-i-1;fin[n,2]<-j-1;fin[n,3]<-k-1;fin[n,4]<-l-1;fin[n,5]<-m-1;fin[n,6]<-pro[n];f
  n <- n+1}}}}}
fin <- as.data.frame(fin)
# 通过OpenBugs我们得到参数估计
# 先将测试集提取处理
data <- test
#   根据上面的结果 我们选择的变量为glucose, age, sysBP, BMI, diaBP
data.fin1 <- data[,which(colnames(data)%in%c('glucose','age','sysBP','BMI','diaBP','y'))]
```

```
# 尝试分箱
data.b1 <- data.fin1
data.b1$age <- as.integer(ifelse(data.b1$age<=50,0,1))
data.b1$sysBP <- as.integer(ifelse(data.b1$sysBP<140&data.b1$sysBP>90,0,1))
data.b1$diaBP <- as.integer(ifelse(data.b1$diaBP<90&data.b1$diaBP>60,0,1))
data.b1$BMI <- as.integer(ifelse(data.b1$BMI<23.9&data.b1$BMI>18.5,0,1))
data.b1$glucose <- as.integer(ifelse(data.b1$glucose<120&data.b1$glucose>70,0,1))
# 计算概率
pro <- c()
#data.b1 <- as.matrix(data.b1)
for(i in 1:nrow(data.b1)){
  logit <- -1.125+0.8821*data.b1[i,1]+0.5285*data.b1[i,2]+
    0.4181*data.b1[i,3]+0.009286*data.b1[i,4]+0.6427*data.b1[i,5]
  pro[i] <- exp(logit)/(1+exp(logit))
}
cla <- factor(ifelse(pro<0.5,0,1))
con.MC <- confusionMatrix(cla,data.b1$y,positive = '1')
con.MC
con.MC$byClass
#########       模型效果比较
pr.NB <- prediction(pred.NB.test$posterior[,2],test$y)
pr.CART <- prediction(pro, test[,coly])
pr.RF <- prediction(pred.rf1[,2],test$y)
pr.XG <- prediction(pred.xgb,test$y)
pred11 <- performance(pr.NB,'auc');pred12 <- performance(pr.CART,'auc')
pred13 <- performance(pr.RF,'auc');pred14 <- performance(pr.XG,'auc')
unlist(pred11@y.values);unlist(pred12@y.values)
unlist(pred13@y.values);unlist(pred14@y.values)#AUC值
#敏感度-特异度曲线
perf.NB1 <- performance(pr.NB,"sens","spec")
perf.CART1 <- performance(pr.CART,"sens","spec")
perf.RF1 <- performance(pr.RF,"sens","spec")
perf.XG1 <- performance(pr.XG,"sens","spec")
par(bg='grey92')
plot(perf.NB1,col=1)
plot(perf.CART1,col=3,add=T)
```

```
plot(perf.RF1,col=4,add=T)
plot(perf.XG1,col=5,add=T)
abline(h = seq(0,1,.2),v = seq(0,1,.2),col = 'white')
legend("bottomleft",c("NB","MCMC","RandomForest","Xgboost"),col=c(1,3,4,5),lty=1)
#查全率-查准率曲线
perf.NB12 <- performance(pr.NB,"prec","rec")
perf.CART12 <- performance(pr.CART,"prec","rec")
perf.RF12 <- performance(pr.RF,"prec","rec")
perf.XG12 <- performance(pr.XG,"prec","rec")
par(bg='grey92')
plot(perf.NB12,col=1)
plot(perf.CART12,col=3,add=T)
plot(perf.RF12,col=4,add=T)
plot(perf.XG12,col=5,add=T)
abline(h = seq(0,1,.2),v = seq(0,1,.2),col = 'white')
legend("bottomleft",c("NB","MCMC","RandomForest","Xgboost"),col=c(1,3,4,5),lty=1)
#ROC
perf.NB121 <- performance(pr.NB,"tpr","fpr")
perf.CART121 <- performance(pr.CART,"tpr","fpr")
perf.RF121 <- performance(pr.RF,"tpr","fpr")
perf.XG121 <- performance(pr.XG,"tpr","fpr")
par(bg='grey92')
plot(perf.NB121,col=1)
plot(perf.CART121,col=3,add=T)
plot(perf.RF121,col=4,add=T)
plot(perf.XG121,col=5,add=T)
abline(0,1,lty = 4)
abline(h = seq(0,1,.2),v = seq(0,1,.2),col = 'white')
legend("bottomright",c("NB","MCMC","RandomForest","Xgboost"),col=c(1,3,4,5),lty=1)
#提升图
perf.NB1212 <- performance(pr.NB,"lift","rpp")
perf.CART1212 <- performance(pr.CART,"lift","rpp")
perf.RF1212 <- performance(pr.RF,"lift","rpp")
perf.XG1212 <- performance(pr.XG,"lift","rpp")
par(bg='grey92')
plot(perf.NB1212,col=1)
```

```
plot(perf.CART1212,col=3,add=T)
plot(perf.RF1212,col=4,add=T)
plot(perf.XG1212,col=5,add=T)
abline(h = seq(0,4,.5),v = seq(0,1,.2),col = 'white')
legend("topright",c("NB","MCMC","RandomForest","Xgboost"),col=c(1,3,4,5),lty=1)
```

```
##############################
# CODE FOR OPENBUGS
##############################
# MODEL
model{
 for( i in 1 : N ) {
  r[i] ~ dbin(p[i],n[i])
  logit(p[i]) <- alpha0 + alpha1 * x1[i] + alpha2 * x2[i] +
   alpha3 * x3[i] + alpha4 * x4[i] + alpha5 * x5[i] }
 alpha0 ~ dnorm(0.0,1.0E-6)
 alpha1 ~ dnorm(0.0,1.0E-6)
 alpha2 ~ dnorm(0.0,1.0E-6)
 alpha3 ~ dnorm(0.0,1.0E-6)
 alpha4 ~ dnorm(0.0,1.0E-6)
 alpha5 ~ dnorm(0.0,1.0E-6)
}
# DATA
list(r = c(47,31,72,73,15,6,36,13,6,13,22,23,14,8,42,32,55,25,132,81,7,10,33,25,38,28,96,5
   n = c(232,84,323,181,27,16,81,33,26,17,60,40,25,11,99,51,124,51,289,128,14,11,59,37,64,
   x1 = c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),
   x2 = c(0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1),
   x3 = c(0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1),
   x4 = c(0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1),
   x5 = c(0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1),
   N = 32)
# INIT
list(alpha0 = 0, alpha1 = 0, alpha2 = 0, alpha3 = 0, alpha4 = 0, alpha5 = 0)
```