EMIS 7331 Data Mining

Project 3

Zheqi Wang 47711564
Beichen Hu 47809766

Table of Contents

1. Data Preprocess

In this part, we do the initial preprocess of data. We summarized the variables and types as following.

FEATURE NAME	EXPLAINATION	TYPE
Agency	Name of insurance agency	factor
Agency Type	Type of insurance agency: Airline and Travel Agency	factor
Distribution Channel	Insurance distribution channel: Online and Offline	factor
Product Name	Name of insurance product	factor
Claim	Whether the insurance is claim or not	factor
Duration	Insurance duration. Value Range: -2 to 4881	integer
Destination	Travel insurance destination	factor
Net Sales	Net sales income. Value Range: -389 to 810	numeric
Commision in value	Value range: 0 to 283	numeric
Gender	No value, F and M	factor
Age	Value range: 0 to 118	integer

1.1. Missing Value

We count the missing value (N/A and NULL) of each feature. We could find that there is no NA and NULL value. However, using summary() function to see the dataset, we find that 'Gender' has 45107 "" value, which is not useful for our analysis.

Because the "" value is too much for our dataset (total has 63326 records), which is about 71%. So many meaningless values will affect our subsequent models building, and for the gender attribute we can't effectively fill them. So we decide to delete this feature.

```
> summary(df)
     Agency            Agency.Type    Distribution.Channel                          Product.Name
 EPX    :35119   Airlines     :17457   Offline: 1107        Cancellation Plan              :18630
 CWT    : 8580   Travel Agency:45869   Online :62219        2 way Comprehensive Plan       :13158
 C2B    : 8267                                             Rental Vehicle Excess Insurance: 8580
 JZI    : 6329                                             Basic Plan                     : 5469
 SSI    : 1056                                             Bronze Plan                    : 4049
 JWT    :  749                                             1 way Comprehensive Plan       : 3331
 (Other): 3226                                             (Other)                        :10109
 Claim          Duration          Destination      Net.Sales       Commision..in.value. Gender
 No :62399   Min.   :  -2.00   SINGAPORE:13255   Min.   :-389.00   Min.   :  0.00        :45107
 Yes:  927   1st Qu.:   9.00   MALAYSIA : 5930   1st Qu.:  18.00   1st Qu.:  0.00       F: 8872
             Median :  22.00   THAILAND : 5894   Median :  26.53   Median :  0.00       M: 9347
             Mean   :  49.32   CHINA    : 4796   Mean   :  40.70   Mean   :  9.81
             3rd Qu.:  53.00   AUSTRALIA: 3694   3rd Qu.:  48.00   3rd Qu.: 11.55
             Max.   :4881.00   INDONESIA: 3452   Max.   : 810.00   Max.   :283.50
                               (Other)  :26305
      Age
 Min.   :  0.00
 1st Qu.: 35.00
 Median : 36.00
 Mean   : 39.97
 3rd Qu.: 43.00
 Max.   :118.00
```

1.2. Outliers/Invalid Value

We find that 'Destination' has too much (149) levels, which is not good for the following model build. We decide to delete this column.

```
df <- df[ ,-which(names(df)%in%c("Destination"))]
```

For 'Age' attribute, current oldest living person is 114 years old. We delete rows with 'Age' is greater than 114. After deleting, our 'Age' value range is 0-88. [1]

```
      Age
 Min.   : 0.00
 1st Qu.:33.00
 Median :36.00
 Mean   :39.72
 3rd Qu.:47.00
 Max.   :88.00
```

'Duration' has negative value, we think 'Duration' should more than zero. We delete record with 'Duration' greater than zero.

```
    Duration
 Min.   :   0.00
 1st Qu.:  12.00
 Median :  29.00
 Mean   :  60.02
 3rd Qu.:  67.00
 Max.   :4881.00
```

1.3. Duplicated value

We check duplicated data record and delete them. There are 42879 unique value record in dataset.

```
> dim(unique(df))
[1] 42879     9
```

1.4. Response feature

We choose feature 'Claim' from the processed data that will serve as the response-feature. Because our analysis of this data set is from the perspective of insurance companies, and insurance companies are most concerned about the probability of selling products will be claimed. The situation of claims can determine the income of insurance companies. Another reason we choose 'Claim' as our response is that 'Claim' is a factor with two values: Yes and No.

2. Classification Methods

We split the processed data into training and testing sets. Number of train data: Number of test data = 0.67. And for now, we use unbalanced data which that 'Claim' has 41973 Yes and 906 No. This dataset is unbalanced dataset, so the assessment method should be changed, and this will be explained in detail later.

```
> summary(df$Claim)
   No   Yes
41973   906
```

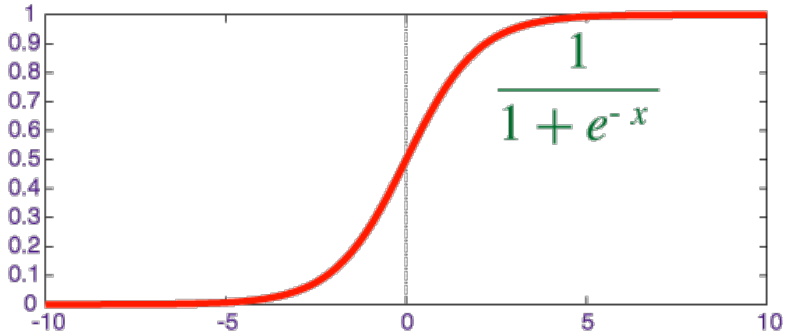
We will apply three classification methods to train this dataset. They are Logistic Regression, Naïve Bayes and Random Forest.

2.1. Logistic Regression

Logistic Regression is a machine learning method used to solve a two-class (0 or 1) problem, which is used to estimate the likelihood of something. Logistic regression assumes that the dependent variable y obeys the Bernoulli distribution. [2]
The Logistic Regression algorithm maps the results of linear functions to sigmoid functions. The sigmoid function shows as following.

$$y = \frac{1}{1 + e^{-x}}$$

Thus, the Logistic Regression function is: $h_\theta(x) = \frac{1}{1+ e^{-\theta^T x}}$, θ is weight of features.
$h_\theta(x) < 0.5$ indicates that the current data belongs to class A.
$h_\theta(x) > 0.5$ indicates that the current data belongs to class B.
The diagram of sigmoid function shows as following.



The probabilities for inputting x-category results to category 1 and category 0 are:

$$P(y = 1|x, \theta) = h_\theta(x)$$

$$P(y = 0|x, \theta) = 1 - h_\theta(x)$$

According to the above formula, we can use the method of maximum likelihood estimation in probability theory to solve the loss function. First, we obtain the probability function as:

$$P(y|x, \theta) = \left(h_\theta(x)\right)^y \times (1 - h_\theta(x))^{1-y}$$

Since the sample data (sample size is n) are independent, their joint distribution can be expressed as the product of the marginal distributions, and the likelihood function is:

$$L(\theta) = \prod_{i=1}^{n} \left(h_\theta(x^i)\right)^{y^i} \times (1 - h_\theta(x^i))^{1-y^i}$$

Log likelihood function

$$l(\theta) = log(L(\theta)) = \sum_{i=1}^{n} y^i log\left(h_\theta(x^i)\right) + (1 - y^i)\log\left(1 - h_\theta(x^i)\right)$$

The maximum likelihood estimation is the θ required to make l(θ) take the maximum value. We could use Gradient descent to calculate the maximum value. However, for this project, we use R function 'glm' to do the above work.

To use logistic regression, we need to transfer all factor features to numeric. The features are 'Product.Name', 'Agency', 'Agency.Type' and 'Distribution.Channel'. Response value 'Claim' is also factor. Due to logistic regression response value need to be {0, 1}. We need transfer 'Yes' to 0 and 'No' to 1.

Then we build the model using 'glm' function. Then we make a prediction using test data to predict 'Claim'. We decide to choose mean value of probability as our threshold, which means if the probability greater than mean value, it is classified to class A; if the probability less than mean value, it is classified to class B. The mean value of this model is 0.02.

```
> summary(prob)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.002407 0.009274 0.010807 0.020495 0.015219 0.452639
```

The confusion matrix of logistic regression is shown as following.

```
> TP_log_table
   pred
      0     1
0 11139  2696
1   123   192
```

After calculating, the misclassification rate based on the testing set is **19.92226%.**

2.2. Naïve Bayes

Naive Bayes classification is a very simple classification algorithm because the idea of this method is really simple and make a strong assumption that assuming each condition independent. The idea of Naive Bayes: for the given classification item, solves the probability of occurrence of each category under the condition of this occurrence, and which is the largest, which category is considered to belong to this category.

Given a random variable X$\in R^d$ and a dependent variable $Y \in R$, the naïve Bayes model defines joint distribution. The assumption is that each condition independent.
The formal definition of Naïve Bayes classifier is:

1. Set $x = \{a_1, a_2, \dots, a_n\}$ as category to be classified, and each 'a' is a feature attribute of x.
2. Category set $C = \{y, y_2, \dots, y_m\}$
3. Calculate $P(y_1|x)$, $P(y_1|x)$, …, $P(y_1|x)$
   According to the assumption: each feature is independent, it is derived from Bayes' theorem as follows.
   $$P(y_i|x) = \frac{P(x|y_i)P(y_i)}{P(x)}$$
   To calculate this, we need obtain conditional probability estimates for each feature under each category. In other words, calculate $P(a_1|y_1)$, $P(a_2|y_1)$, …, $P(a_n|y_1)$; $P(a_1|y_2)$, $P(a_2|y_2)$, …, $P(a_n|y_2)$; …; $P(a_1|y_m)$, $P(a_2|y_m)$, …, $P(a_n|y_m)$;
   Because the denominator is constant for all categories, we only need to maximize the molecule. Each feature is conditionally independent, there are:
   $$P(x|y_i)P(y_i) = P(a_1|y_i)P(a_2|y_i) \dots P(a_n|y_i) = P(y_i)\prod_{j=1}^{n} P(a_j|y_i)$$
4. If $P(y_k|x) = \max\{P(y_1|x), P(y_2|x), \dots, P(y_n|x)\}$, then $x \in y_k$ [3]

We use package 'e1071' and function 'naiveBayes' in R to train our model. Then we use 'Claim' in test dataset to predict.
The confusion matrix of prediction is shown as following.

```
nb_pred    No   Yes
    No  13293   238
    Yes   542    77
```

And misclassification rate based on the testing set is **5.512367%.**

2.3. Random Forest

Random forests, as the name implies, establish a forest in a random way. The tree in the forest is called decision tree. What is decision tree?

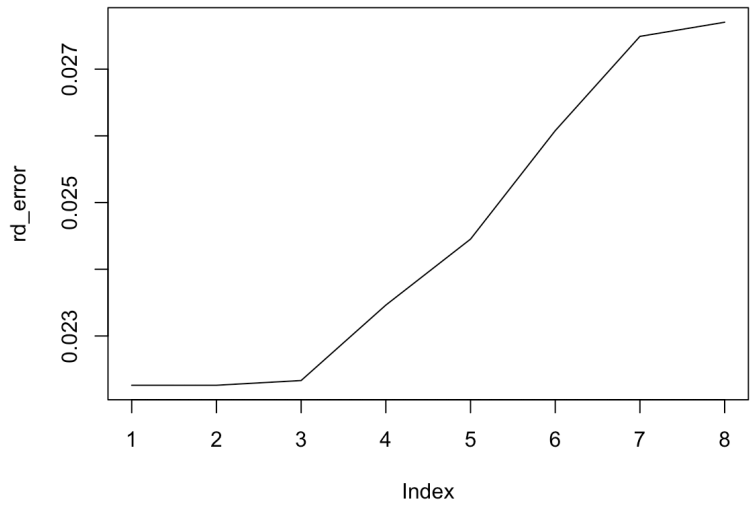
The decision tree is a tree structure (which can be a binary tree or a non-binary tree). Each of its non-leaf nodes represents a test on a feature, each branch representing the output of the feature over a range of values, and each leaf node storing a category. The decision process using the decision tree is to start from the root node, test the corresponding feature attributes in the item to be classified, and select the output branch according to its value until the leaf node is reached, and the category stored by the leaf node is used as the decision result.

Random forest consists of many decision trees. There is no correlation between each decision tree in the random forest. After getting the forest, when a new input sample enters, let each decision tree in the forest make a separate judgment to see which class the sample should belong to (for the classification algorithm), and then see which one has the maximum information gain, predicting which sample is the same category.[4]

The following is the Random forest generation method [5]:

1. Generate n samples from the sample set by resampling
2. Assuming that the number of sample features is a, select k features in a for n samples, and obtain the best segmentation point by establishing a decision tree.
3. Repeat m times to generate m decision trees
4. When make predictions, choose the one with maximum information gain

To generate the model, we use package ‘randomForest’. This package need to give a parameter to decide how many features to consider when splitting. Our dataset has 8 features totally. We decide to train the model using number of features from 1 to 8 (hyper-parameter), in other words, make 8 model. Then find a best model.
From the plot shows below, we could find that elbow point is 5, which means that after 5, the model's error rises faster. So we use 5 features to consider when splitting.



Using the best model to predict response value. The confusion matrix shows following.

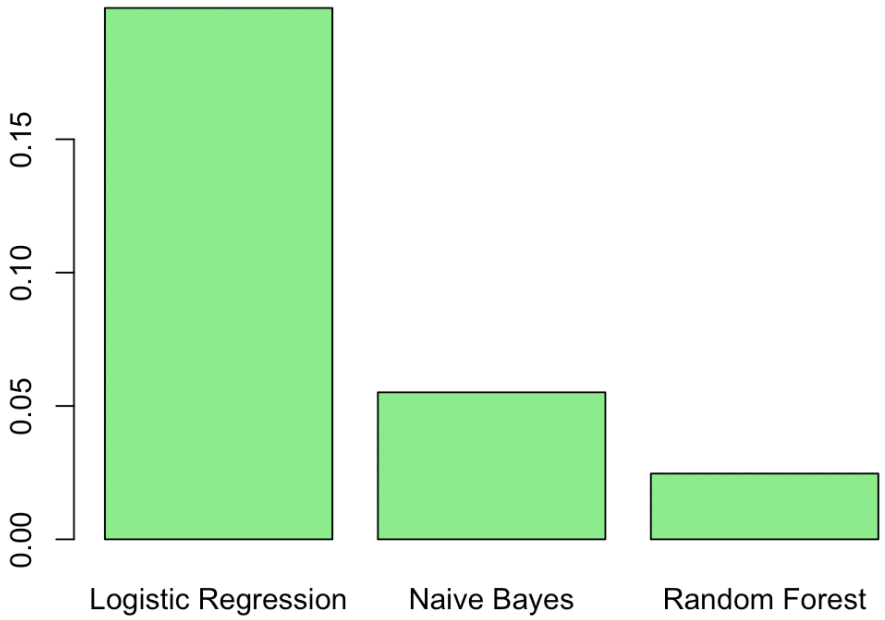
baggingPred	No	Yes
No	13797	311
Yes	38	4

The misclassification rate based on the testing set is **2.4664%**

```
> rd_error
[1] 0.02466431
```

2.4. Comparison

In this part, we will compare performance of the selected classification methods. We draw a histogram to show different classification method misclassification rate.



In the figure, the error rate of logistic regression is the highest, followed by naive Bayes, and the highest accuracy is random forest. The error rate of logistic regression is much larger than the other two methods.

3. Linear Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.[7]

The above definition is for general classification (for high dimension space). In this project, we only focus on Linear Support Vector Machine, which is two dimensional space. Then the question become: dividing points into two classes using a line, and make margin as big as possible.

Given training dataset $\{ (x^i, y_i) \}^N_{i=1}$. Assume that $x^i \in R^d$ and $y_i \in R$ for each sample i.

The line's function, also called SVM's objective function:

$$\omega^T x + b = 0$$

ω is normal vector (weight vector) to the line, b is the bias, x is the dataset point.

The corresponding decision function, which is the function used for prediction is:

$$\text{sign}(\omega^T x_i + b) = \begin{cases} -1, & \omega^T x_i + b < 0 \\ 0, & \omega^T x_i + b = 0 \\ 1, & \omega^T x_i + b > 0 \end{cases}$$

This decision function could be written in another format: $y * sign(\omega^T x_i + b) > 0$

With normalization, function of margin becomes: $\frac{1}{||\omega||} y(\omega^T x_i + b)$

We define that this line's interval of the dataset is the minimum value of all sample points' interval in the dataset. It is $\min_n [y_i(\omega^T x_i + b)]$. n is sample points number.

To dataset with n data record, which each data is (x_i, y_i), we could conduct the following function to represent this line with the maximum margin:

$$\arg\max_{\omega,b} \left\{ \frac{1}{||\omega||} \min_n [y_i(\omega^T x_i + b)] \right\}$$

$$y_i(\omega^T x_i + b) \geq 1$$

The above function is support vector machine's loss function.

In the case of correct classification, $y * sign(\omega^T x_i + b) > 0$. By scaling, this function could be written to $y * sign(\omega^T x_i + b) \geq 1$. $y * sign(\omega^T x_i + b)$'s value doesn't affect optimal question's result. Finally, what we need to solve is:

$$\arg\max_{\omega,b} \left\{ \frac{1}{||\omega||} \right\}$$

$$y_i(\omega^T x_i + b) \geq 1$$

In the optimization problem, it is more difficult to solve the maximum value, and it is better to convert it to the minimum value. Then the function becomes:

$$\min_{\omega,b} \left(\frac{1}{2} ||\omega||^2 \right)$$

$$y_i(\omega^T x_i + b) \geq 1$$

Using hinge loss function, we can rewrite the above function as:

$$\min_{\omega,b,z} \sum_{i=1}^{N} z_i + \frac{\lambda}{2} ||\omega||_2^2$$

Denote $z = (z_1, z_2, \ldots\ldots, z_N) \in R^n$, λ is a hyper parameter.

This is the primal problem of Support Vector Machine formulated by without transformation of the feature vectors. In project, we use a modified formulation of SVM is defined as following:

$$\min_{\omega,b,z} C\sum_{i=1}^{N} z_i + \frac{1}{2}||\omega||_2^2 + \frac{C_1}{2}b^2 + \frac{C_2}{2}||z||_2^2$$

$$s.t.\ 1 - y_i(\omega^T x_i + b) \le z_i$$

$$0 \le z_i$$

for all i = 1, 2, ……, N

where $w \in R^d$, $b \in R$, and $z=(z_1,...,z_N) \in R^N$ for some constant C, C_1, and C_2. Given C = 1, C_1 = C_2 = 0.00001

As shown in class, the above problem is a convex quadratic program (QP). What is quadratic problem? Quadratic programming (QP) is the process of solving a special type of mathematical optimization problem—specifically, a (linearly constrained) quadratic optimization problem, that is, the problem of optimizing (minimizing or maximizing) a quadratic function of several variables subject to linear constraints on these variables. Quadratic programming is a particular type of nonlinear programming. [6]

Based on an existing QP solver (function solve.QP in R-package quadprog), we are going to implement and solve Linear Support Vector Machine on R. The function called lsvm().
For a quadratic program, we need to rewrite the above function to the following format.

$$\min_V \frac{1}{2}V^T QV + q^T V$$

$$s.t.\ D^T V \ge d$$

$V = (\omega_1, \omega_2, b, z_1, \dots\dots, z_n)^T$ where n equals to number of samples.
Q is an identity matrix with all diagonals being 1 and all other elements being 0. $Q \in R^{(n+3)\times(n+3)}$
q = (0, 0, 0, 1, ……, 1)^T where $q \in R^{(n+3)}$
d = (0, 0, 0, 1, ……, 1)^T where $d \in R^{(n+3)}$

$$Q = \begin{bmatrix} A1 & A2 \\ A3 & A2 \end{bmatrix}$$

A1 is a matrix with all elements is zero. $A1 \in R^{n\times 3}$
A2 is an identity matrix with all diagonals being 1 and all other elements being 0. $A2 \in R^{n\times n}$
A3 is a matrix which each row contains response value and response value multiple feature value.

$$A3 = \begin{bmatrix} response^1 \times sample^1_{feature1} & response^1 \times sample^1_{feature2} & response^1 \\ \dots & \dots & \dots \\ response^n \times sample^n_{feature1} & response^n \times sample^n_{feature2} & response^n \end{bmatrix}$$

To use function solve.QP in R-package quadprog, we need to put Q, V, D (in code is called A) and d as input parameters. And the results are in 'solution'. It's a vector [b, w1, w2, z1, z2, ……, zn]. As requested, we return the w and b as output.
Source code show as following.

```
lsvm <- function(X, Y, C, C1, C2) {
  row_num <- nrow(X) # row number
  row_num
  col_num <- ncol(X) # feature number/column number
  col_num

  # min(1/2 * w^2 + C1/2 * b^2 + C2/2 * z^2 + C * sum(z))
  # s.t. 1 - yi * (w^T+ b) <= zi
  # min (1/2 * V^T * Q * V + q^T * V)
  # s.t. D^T * V >= d
  V <- c(0, rep(0, col_num), rep(-C, row_num)) # V - dvec in function solve.QP, V is a vector
  Q <- diag(c(C1, rep(1, col_num), rep(C2, row_num))) # Q - Dmat in function solve.QP
  dim(Q)
  d <- c(rep(0, row_num), rep(1, row_num)) # d - bvec in function solve.QP, d is a vector

  I_N <- diag(row_num) # row_num x row_num identity matrix
  A_1 <- cbind(matrix(0, ncol = col_num + 1, nrow = row_num), I_N) # upper part of D^T
  A_2 <- as.matrix(cbind(as.matrix(Y), X * as.matrix(Y)[, rep(1, col_num)], I_N)) # bottom part of D^T
  rownames(A_1) <- NULL; rownames(A_2) <- NULL
  colnames(A_1) <- NULL; colnames(A_2) <- NULL
  A <- t(rbind(A_1, A_2)) # D^T - Amat in function solve.QP
  dim(A)

  # solve QP
  library(quadprog)
  results <- solve.QP(Q, V, A, d)

  # Retrieve the results
  # results$solution => [b, w1, w2, z1, z2, z3,......]
  # b <- results$solution[1]
  # w <- results$solution[1 + (1:col_num)]
  # z <- results$solution[(col_num + 1) + (1:row_num)] # z1,z2,z3,......
  return(results$solution[0:col_num + 1]) # return b and w
}
```

In the function lsvm(), firstly, we build vector V, d, matrix Q and A. Then use function solve.QP to calculate w and b. Then return the result.

Then we use the dataset train as input of the lsvm( ) function. Use the columns 'Net.Sales' and 'Age' as features, which is X in the function. And the column 'Claim' as response, which is Y in previous source code. Given C = 1, $C_1 = C_2 = 0.00001$. The result of lsvm() is:

```
> lsvm_res
[1] -1.058601134  0.016068053 -0.002835539
```

b is -1.058601134, w1 is 0.016068053, w2 is -0.002835539. Using the result, the SVM objective function is:

$$(0.016068053)\, x_1 + (-0.002835539)\, x_2 - 1.058601134 = 0$$

For this dataset, x_1 is 'Net.Sales' and x_2 is 'Age' values of the dataset test.

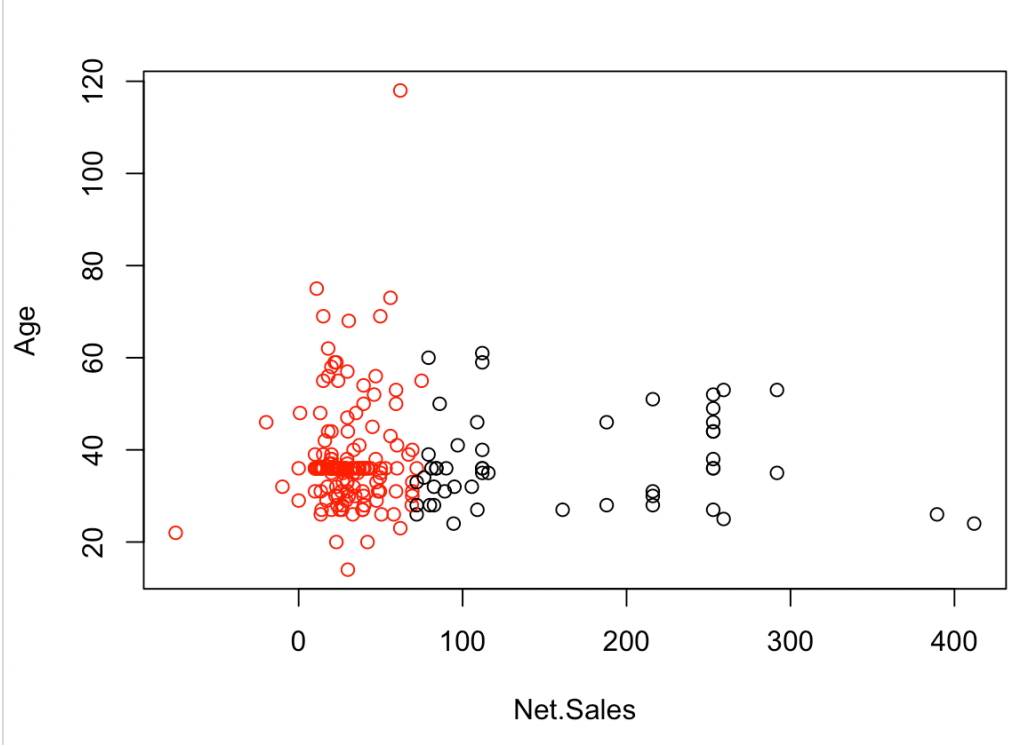
The using 'Net.Sales' and 'Age' in test dataset, we predict 'Claim' based on the solution obtained from the previous part and calculate the misclassification rate.

```
Y_pred <- sign(apply(X_test, 1, function(x) beta_0 + sum(w * as.vector(x))))
```

The table of prediction result is:

```
Y_pred -1  1
    -1 91 59
     1 15 35
```

The misclassification rate is **37%**. So the accuracy of prediction is 63%.
Then we draw a scatter plot which illustrates the above prediction-results. Use 'Net.Sales' as the horizontal axis and 'Age' as the vertical axis. Use black and red to represent different classes. The diagram shows as following.



4. Exceptional Work

We have used an unbalance dataset in classification method before. An unbalanced dataset is one in which the number of samples varies widely from one category to another. Taking the dichotomy problem as an example, it is assumed that the number of samples of the positive class is much larger than the number of samples of the negative class. In this case, the data is called unbalanced data.

If 90% of the samples of the training set belong to the same class, and our classifier classifies all the samples into this class. In this case, the classifier is invalid, although the final classification accuracy is 90%. Therefore, when the data is not balanced, accuracy is of little reference significance. In fact, if the imbalance ratio exceeds 4:1, the classifier will favor the larger category and almost prediction value will be larger category.

Take our dataset as an example, 'travel_insurance.csv', after we preprocess our dataset, then we can check the rate of target feature, Claim.

```
Claim
No :41973
Yes:  906
```

906 / 41973 = 2.1585% which means, the number of 'No' is much larger than the number of 'Yes'. That is an unbalance dataset. If we use logistic regression model to predict response value, the incoming data will be predicted as 'No' because of the unbalance dataset.

For logistic regression, confusion matrix is:

```
pred
      0     1
0 11139  2696
1   123   192
```

The error of prediction is:

```
> lg_error
[1] 0.1992226
```

From the confusion matrix, we can see that almost prediction values are belonged to 'No' (0), and a little 'Yes' (1). Also, the mean error is 0.1992 which means that the accuracy is 1 – 0.1992 = 0.8008. It is great.

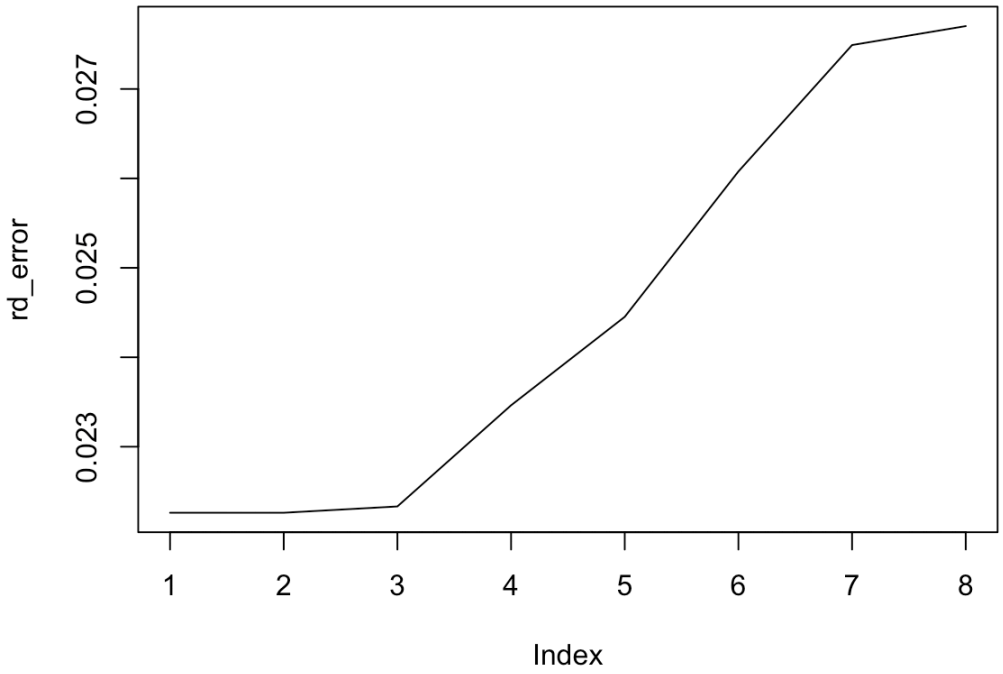
For Naïve Bayes, confusion matrix is:

```
nb_pred    No   Yes
    No  13293   238
    Yes   542    77
```

The error of prediction is:

```
> nb_error
[1] 0.05512367
```

The less 'No' prediction values and more 'Yes' values. However, the almost prediction values are 'No', and the accurarcy is 1 – 0.0551 = 0.9449 which is also great.
For Random Forest, we use plot to determine number of feature to split.



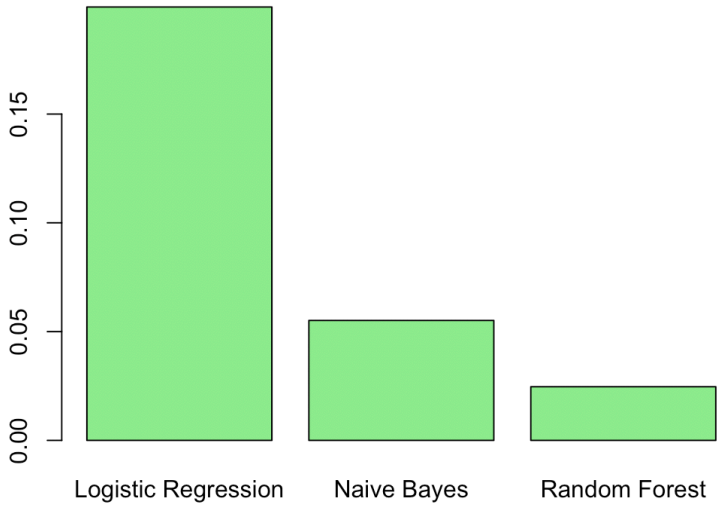
From the elbow plot, we catch up 5 as our features number in bagging to build up the model. Because the slope is much higher than before when features number is 5. Thus, after prediction, confusion matrix is:

baggingPred	No	Yes
No	13797	311
Yes	38	4

The error of prediction is:

```
> rd_error
[1] 0.02466431
```

When we pick up 3 features to build up the random forest model, our confusion matrix shows that almost value is No but not Yes. However, the accuracy is 1 – 0.0247 = 0.9753.
Compare with three model we build, 0.8008 (logistic regression), 0.9449 (naïve bayes), 0.9753 (random forest). All of them own high accuracy.



4.1. Assessment Indicator

However, we use unbalance data to do analysis and prediction. Is the accuracy or mean error reliable? Is there any other method to measure the prediction result? Furthermore, if we use a balance dataset to predict the value. What is the performance according to different measure approaches?

Choosing the appropriate evaluation measure is the key step of unbalanced data analysis. Most classification algorithms only measure the accuracy through the correct classification rate. But for our dataset it is not enough.

Imagine using a model that the classifier classifies all test samples as “0” or “1”, having good accuracy (more than 90%) if accuracy is used to measure the quality of the model, but obviously this model does not provide us with any valuable information.

In this case, other alternative assessment indicators can be applied, such as:

Precision: In positive samples, the number of samples is really positive (True).

Recall: The ratio of positive samples detected to the total number of positive samples.

Here is the formulation:

$$\text{Precision} = TP / (TP + FP)$$

$$\text{Recall} = TP / (TP + FN)$$

Confusion Matrix	Prediction value is (positive)	Prediction value is (negative)
The value is (true)	TP	TN
The value is (false)	FP	FN

Combing all the assessment indicators, we can compare these values with three classification methods. Furthermore, we can also compare one of these values between balance dataset and unbalance dataset.

4.2. Under Sampling

Undersampling is one way to solve the unbalance problem.
In general, the purpose of the correction method is to correct unbalanced data into balanced data. The correction method is to adjust the sample size of the original data set so that the proportion of different kinds of data is consistent
The undersampling method mainly deals with large classes, which will reduce the number of observations in large classes to balance the data set. This method is more suitable when the data set is large as a whole, and it can also reduce the computation time and storage cost by reducing the training sample size. However, the undersampling method has an obvious defect, because many observations need to be deleted, the use of this method will cause a lot of important information to be lost.
There are two types of under-sampling methods: Random and Informative.
The random undersampling method will randomly delete the large class of observations until the data set is balanced.
An under-sampling rule with information removes observations according to a predetermined rule.
Here we use the random deletion undersampling method to balance the data.

We check the 'Yes' and 'No' samples, then we random pick up 600 samples both from 'Yes' and 'No' samples. Then we combine them together and get a new balance dataset.

```
dfyesn = sample(which(df_except$Claim =='Yes'),600) # random select 600 samples from 'Yes'
dfnon = sample(which(df_except$Claim =='No'),600) # random select 600 samples from 'No'

df_new = df_except[c(dfyesn,dfnon), ] # we combine them together and get a new dataset
# which has 600 claim 'Yes' and 600 claim 'No' samples
summary(df_new)
```

```
Claim
No :600
Yes:600
```

According to the new dataset, we split it into training and testing dataset. By using training dataset, we build three classification models. Then we also compute three assessment indicator, recall, precision and accuracy.

```
which(df_except$Claim =='Yes')
dfyes = df_except[which(df_except$Claim =='Yes'), ]
str(dfyes) #906 obs.
dfno = df_except[which(df_except$Claim =='No'), ]
str(dfno) # 41973
summary(dfno)
```

So, we could use a balanced set of classified data for training.

4.3. Performance

For logistic regression, confusion matrix is:

```
      pred
         0   1
0 147  60
1  55 134
```

The error of prediction is:

```
> lg_error1
[1] 0.290404
```

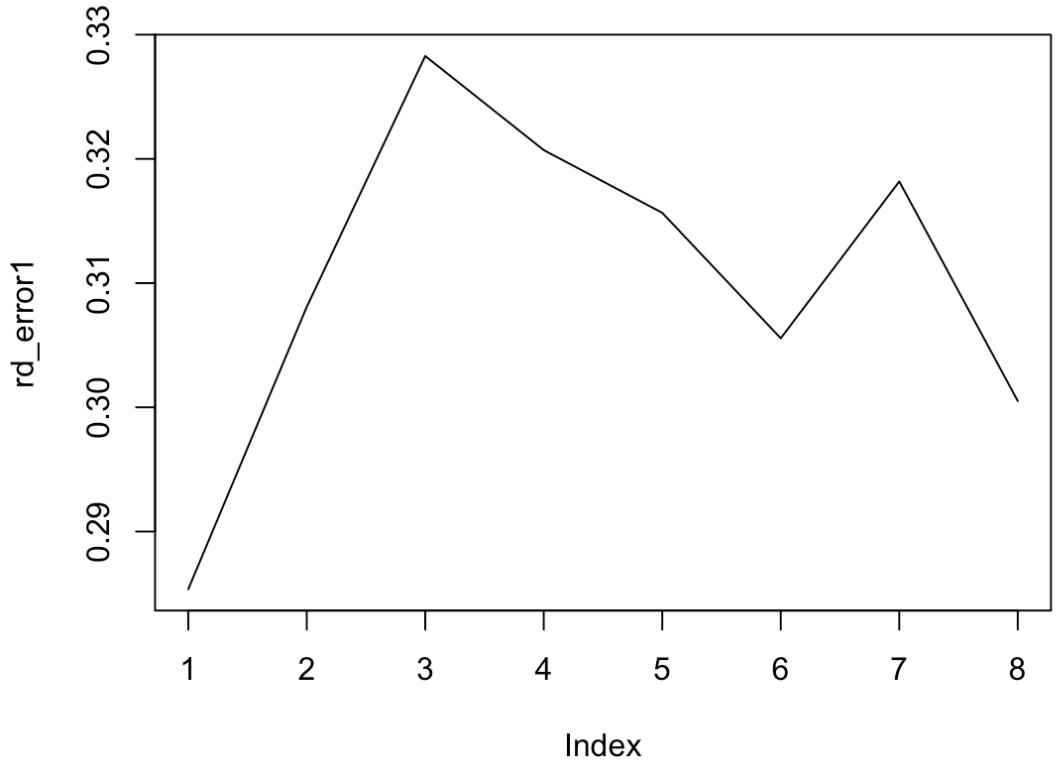
We can see the prediction value is more balance with Yes and No values. However, the error rate is higher than before, it is 0.2904. Because the more Yes we have, the more variance we have.
For Naïve Bayes, confusion matrix is:

```
nb_pred1  No Yes
     No  158  61
     Yes  49 128
```

The error of prediction is:

```
> nb_error1
[1] 0.2777778
```

We can see the result of prediction is also balance than before, and the error rate is 0.2778 which is higher than before.
For random forest, we use the following plot to determine number of features we choose to build decision tree.



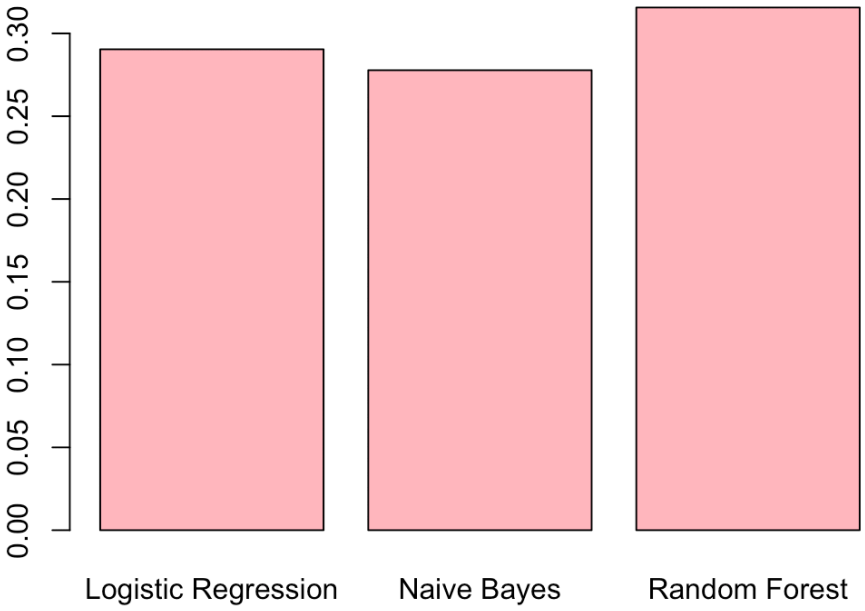
We decide to pick up 6 features to in our bagging. Because when the number is smaller and larger than 6, the error become higher which means number 6 is the best choice for this dataset.
After predicting, confusion matrix is:

```
baggingPred  No Yes
        No  130  48
        Yes  77 141
```

The error of prediction is:

```
> rd_error1
[1] 0.3156566
```

Then we get the confusion matrix of random forest. That is balance result than before. But the error rate is 0.3157 which is higher than 0.3.



In this time, three classification method has similar error rate in balance dataset. Random forest has the worst performance in balance dataset.
Now, we also compute the recall, precision and accuracy result between balance and unbalance dataset.

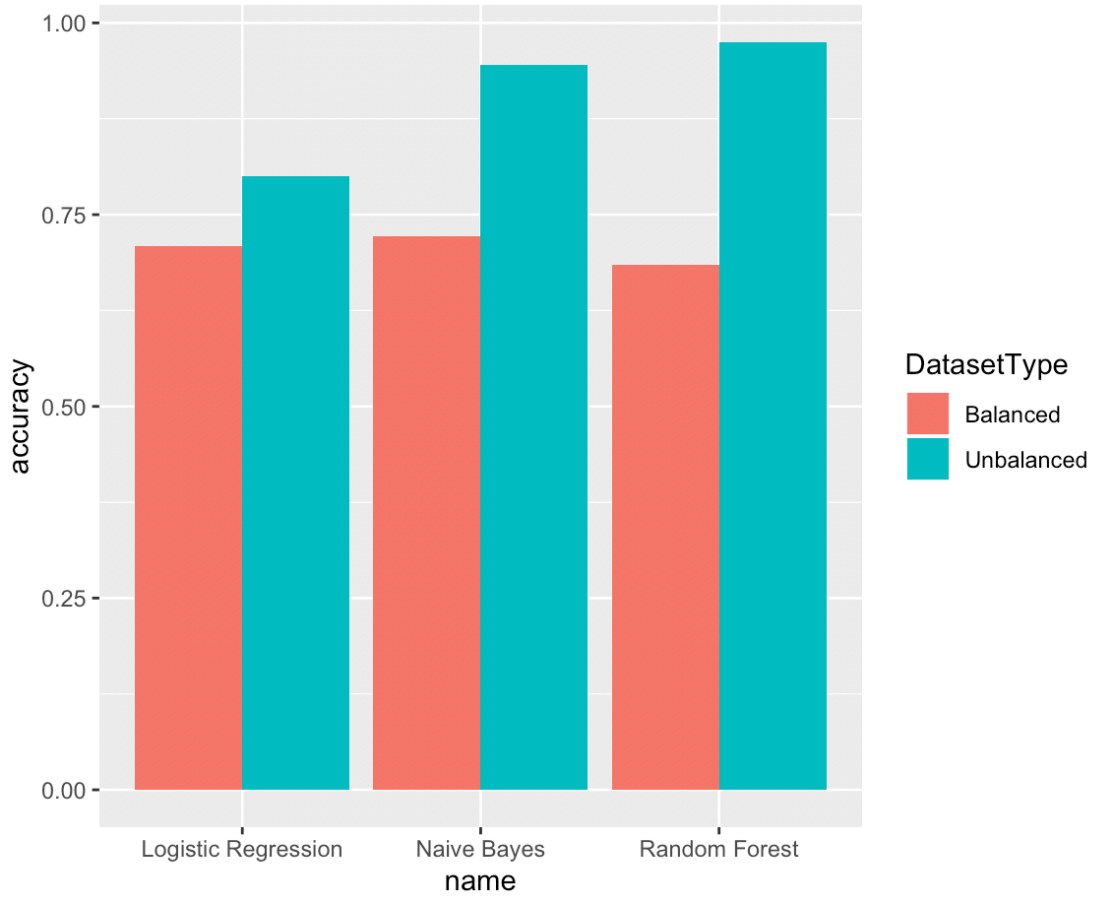
```
# recall = TP / (TP + FN)
# recall

# precision = TP / (TP + FP)
# precision
```

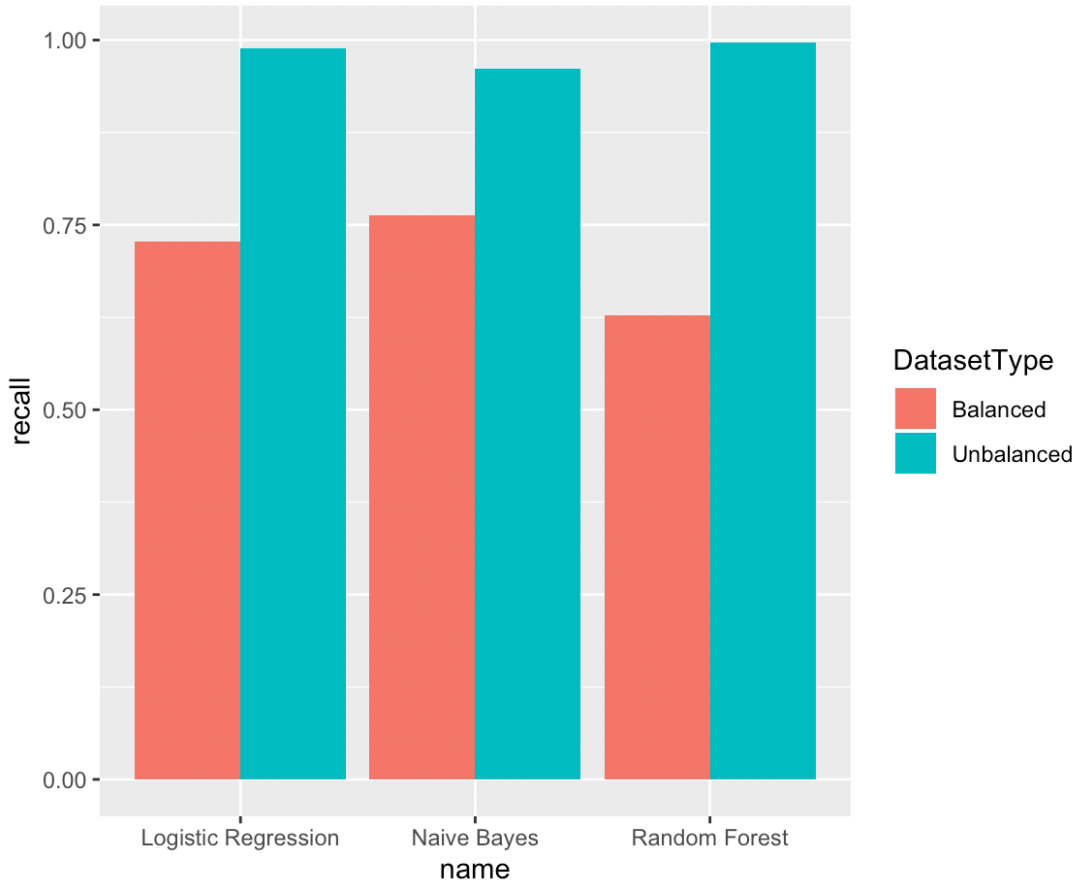
For Unbalance dataset:
Logistic regression: accuracy is 0.8007774, recall is 0.9890783 and precision is 0.8051319.
Naïve Bayes: accuracy is 0.9448763, recall is 0.960824 and precision is 0.9824108.
Random Forest: accuracy is 0.9753357, recall is 0.9972533 and precision is 0.9779558.

For Balance dataset:
Logistic regression: accuracy is 0.709596, recall is 0.7277228 and precision is 0.7101449.
Naïve Bayes: accuracy is 0.7222222, recall is 0.763285 and precision is 0.7214612.
Random Forest: accuracy is 0.6843434, recall is 0.6280193 and precision is 0.7303371.

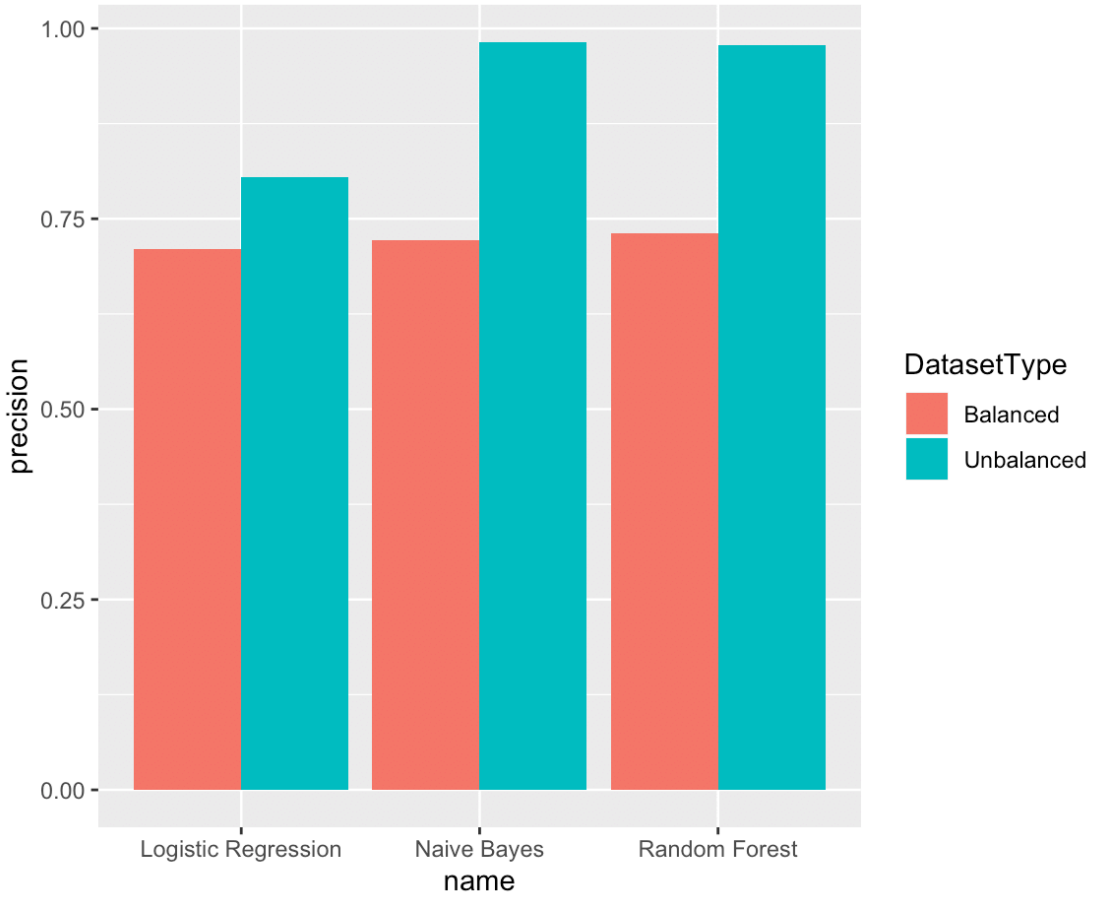
We also build up three plots, accuracy, recall and precision.



For accuracy, unbalance dataset is much higher than balance dataset. But in unbalance dataset, Random Forest has highest accuracy and in balance dataset, Naïve Bayes has highest accuracy which means the prediction values is the same as obvious value.



For recall indicator, unbalance dataset is also much higher than balance dataset. In unbalance dataset, Random Forest has highest recall rate and in balance dataset, Naïve Bayes has highest recall rate which means the rate of true positive values in whole real true value.



For precision indicator, unbalance dataset is still much higher than balance dataset. In unbalance dataset, Naïve Bayes has highest precision rate and in balance dataset, Random Forest has highest precision rate which means the rate of true positive values in whole positive values.

4.4. Conclusion

In summary, comparing with different types of datasets and assessment indicators, Logistic regression has the worst performance, no matter dataset type and indicator is. And Naïve Bayes has the best performance in recall and accuracy of balance dataset, best precision performance in unbalance dataset. For Random Forest, it has the best performance in accuracy and recall rate of unbalance dataset.

But compare with balance and unbalance dataset, in this case. According to three assessment indicator performance, unbalance data is better than balance data. So, we can conclude that it has a better model performance for unbalance data for the travel insurance dataset.

5. Reference

1. https://en.wikipedia.org/wiki/Oldest_people
2. https://zhuanlan.zhihu.com/p/32940093
3. https://www.cnblogs.com/leoo2sk/archive/2010/09/17/naive-bayesian-classifier.html
4. https://my.oschina.net/u/3121322/blog/806651
5. https://blog.csdn.net/mao_xiao_feng/article/details/52728164
6. https://en.wikipedia.org/wiki/Quadratic_programming
7. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
8. https://blog.csdn.net/asialee_bird/article/details/83714612#%EF%BC%882%EF%BC%89%E4%B8%8D%E5%B9%B3%E8%A1%A1%E6%95%B0%E6%8D%AE%E9%9B%86%E4%B8%BE%E4%BE%8B
9. https://www.cnblogs.com/caiyishuai/p/10122359.html
10. https://www.zhihu.com/question/65328356