STAT 4900 Final Project

Chris Taehwan Kim

2022 5 02

Introduction

The human body is more complex than what we think it is. Our body produces millions of new cells every second and proteins in our body interact and react to each others. Each organs takes specific roles for our body to function properly. Veins are the path to deliver essential sources all over the body parts to maintain other systems to function well. Our body is so intelligent that if there is any problem in our body, the body system tries to cure and protect our body from the harmful threats. Clearly, our body system knows more than what we know about ourselves. This means our body react almost immediately to the problems such as diseases, viruses and deficiency of essential sources. Our body contains many sources of energy and if any of these essential sources run into a deficiency, the body system will react to it and sometimes, it would cause second damage to our body. The data set we're going to discuss about will have the features that we've talked about and we will explore through it.

Background information of the data

The data set we're going to use for this project is "Breast Cancer Coimbra Dataset" from UCI Machine Learning Repository. The website link for this dataset is "https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra". To briefly describe this data set, the data set was donated on 2018-03-06 and the area of this study is sourced from Life science subject. This data set was made for the clinical purpose by observing and measuring 64 patients with breast cancer and 52 healthy controls, so in total of 116 samples. The data set has 10 attributes. The attribute characteristic are integers and categorical attribute are also recorded in numbers (1 = Healthy Control and 2 = Patients). The quantitative attributes are Age, BMI, Glucose, Insulin, HOMA, Leptin, Adiponectin, Resistin and MCP-1. The listed attributes are collected from patient's blood analysis and these attributes will indicate the presence or absence of breast cancer. To be more specific with our data set, the listed variables are going to be my 9 predictors (explanatory variables) and classification variable will be my binary dependent variable (response variable). The prediction models is based on these predictors. If our model is accurate enough, we can potentially be used as a biomarker of breast cancer. The data is not missing any values so no trimming process will be required. More detail interpretation of each variables will be provided in further study. As we can see, using our binary variable as our response variable, therefore, the logistic regression model can be used in this case.

Exploratory Data Analysis

Before we apply any statistic techniques, it is important to learn about the data set. When we're analyzing a multivariate data set, it is very important to check how much we can trust the statistic summary we obtained and to have strength to our inference. Now, let's import the data set and take a look.

```r
set.seed(10)
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.0.5
```

```r
dataset = read.csv("/Users/Chris Kim/Desktop/STAT 4900 Final Project/dataR2.csv")
dataset[,10] = dataset[,10] - 1 # To transform classes between 1 and 0.
```

```r
rows = sample(nrow(dataset))
dataset = dataset[rows,] # Randomly shuffled.
ex_data = dataset[,1:9]
kable(summary(ex_data))
```

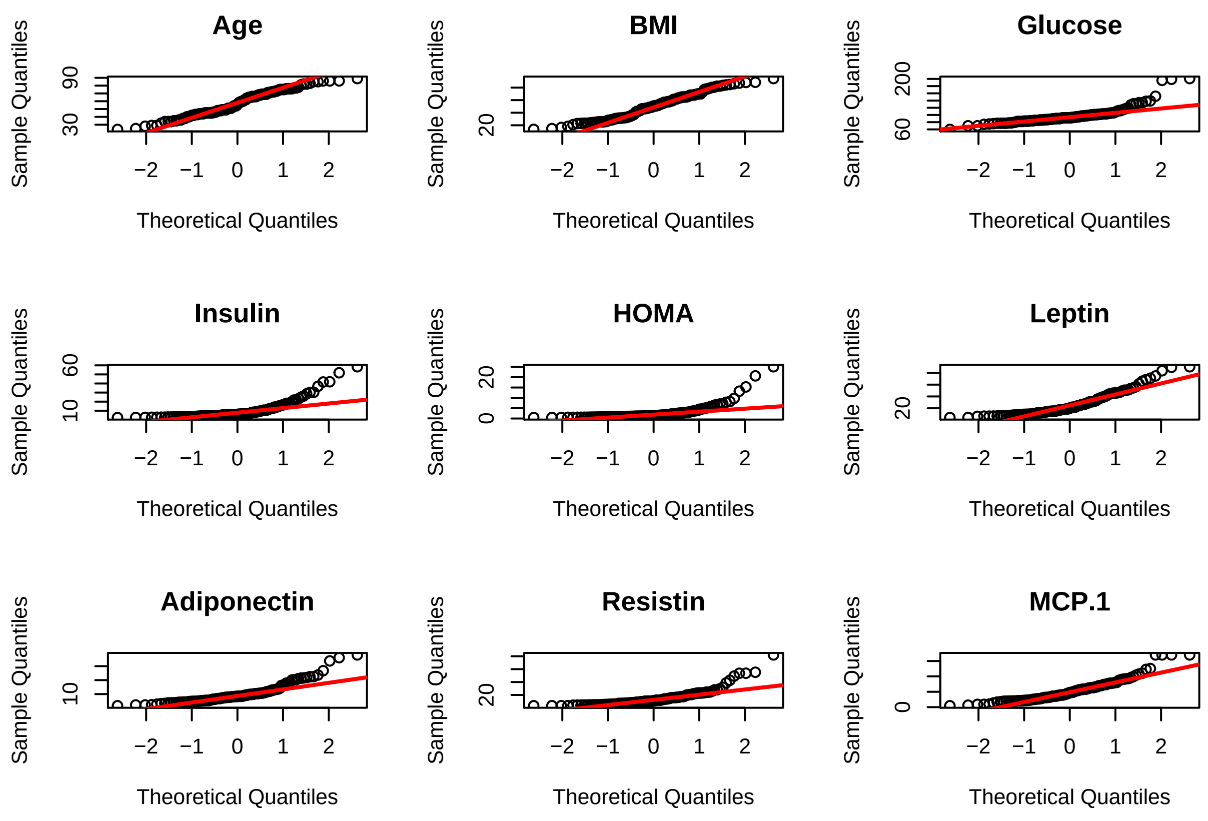
	Age	BMI	Glucose	Insulin	HOMA	Leptin	Adiponectin	Resistin	MCP.1
	Min. :24.0	Min. :18.37	Min. : 60.00	Min. : 2.432	Min. : 0.4674	Min. : 4.311	Min. : 1.656	Min. : 3.210	Min. : 45.84
	1st Qu.:45.0	1st Qu.:22.97	1st Qu.: 85.75	1st Qu.: 4.359	1st Qu.: 0.9180	1st Qu.:12.314	1st Qu.: 5.474	1st Qu.: 6.882	1st Qu.: 269.98
	Median :56.0	Median :27.66	Median : 92.00	Median : 5.925	Median : 1.3809	Median :20.271	Median : 8.353	Median :10.828	Median : 471.32
	Mean :57.3	Mean :27.58	Mean : 97.79	Mean :10.012	Mean : 2.6950	Mean :26.615	Mean :10.181	Mean :14.726	Mean : 534.65
	3rd Qu.:71.0	3rd Qu.:31.24	3rd Qu.:102.00	3rd Qu.:11.189	3rd Qu.: 2.8578	3rd Qu.:37.378	3rd Qu.:11.816	3rd Qu.:17.755	3rd Qu.: 700.09
	Max. :89.0	Max. :38.58	Max. :201.00	Max. :58.460	Max. :25.0503	Max. :90.280	Max. :38.040	Max. :82.100	Max. :1698.44

The imported data set seem very clean and well-organized. Now, let's create a table that contains mean, standard deviation and standard error of each category and plot them.

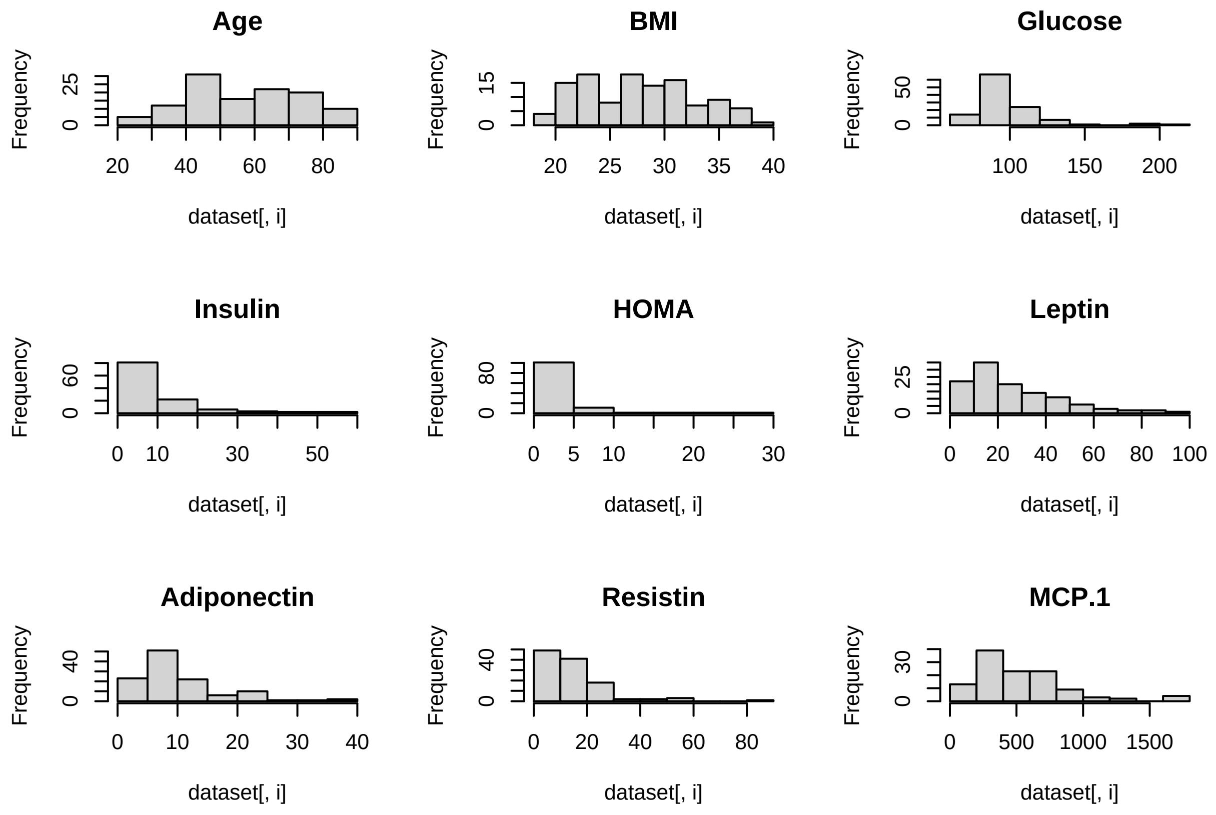
```r
library(knitr)
func <- function(x) {
  sd(x)/sqrt(nrow(dataset))
}

mean <- round(apply(dataset[1:9], 2, mean),3)
sd <- round(apply(dataset[1:9], 2, sd), 3)
se <- round(apply(dataset[1:9], 2, func), 3)

table = data.frame(Mean = mean,
                   SD = sd,
                   SE = se)
kable(table)
```

	Mean	SD	SE
Age	57.302	16.113	1.496
BMI	27.582	5.020	0.466
Glucose	97.793	22.525	2.091
Insulin	10.012	10.068	0.935
HOMA	2.695	3.642	0.338
Leptin	26.615	19.183	1.781
Adiponectin	10.181	6.843	0.635
Resistin	14.726	12.391	1.150
MCP.1	534.647	345.913	32.117

```r
col = names(dataset)

par(mfrow=c(3,3))
for(i in 1:9) {
  qqnorm(dataset[,i], main=paste(col[i], sep=""))
  qqline(dataset[,i], lwd=2, col = "red")
}
```



```r
for(i in 1:9) {
  hist(dataset[,i], main=paste(col[i], sep=""))
}
```

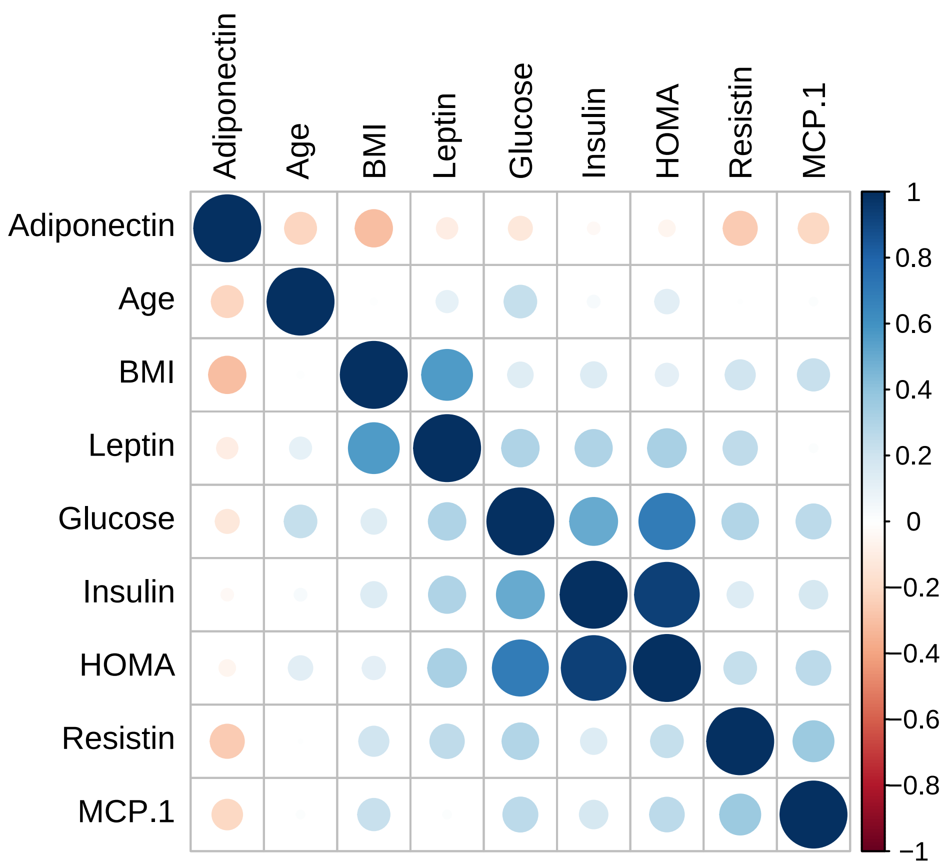


The table contains mean, standard deviation and standard error for each variable and we have 2 different type of plots, QQ-plot and histogram. The QQ-plot is a scatterplot created by plotting two set of quantiles(theoretical and sample) against one another and this plot tells us the normality of the samples and we can reflect our assumption to the histogram I've obtained. For example, the scatterplot for Age variable seems like the points are lining up the the red line. This means the distribution of age is fairly normal and we can check that from the histogram of Age variable. Other than Age and BMI variable, rest of the scatterplot seems like the right tail of the points are breaking away from the red line. This could be a sign that the distribution is skewed to the right and as we can see in the histogram, variables like Glucose and Insulin has a shape of gamma distribution. Let's find out the correlation between each variable by creating correlation matrix.

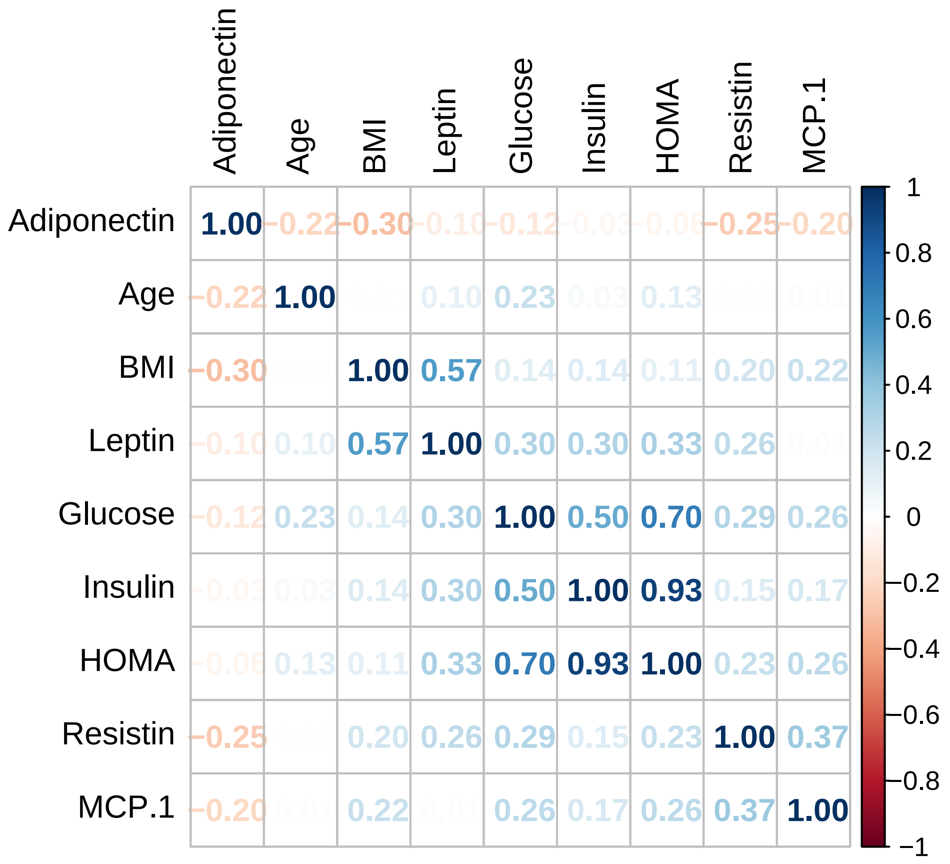
```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
corrplot(round(cor(dataset[,1:9]),3), order = "hclust", tl.col = "black", tl.srt = 90)
```

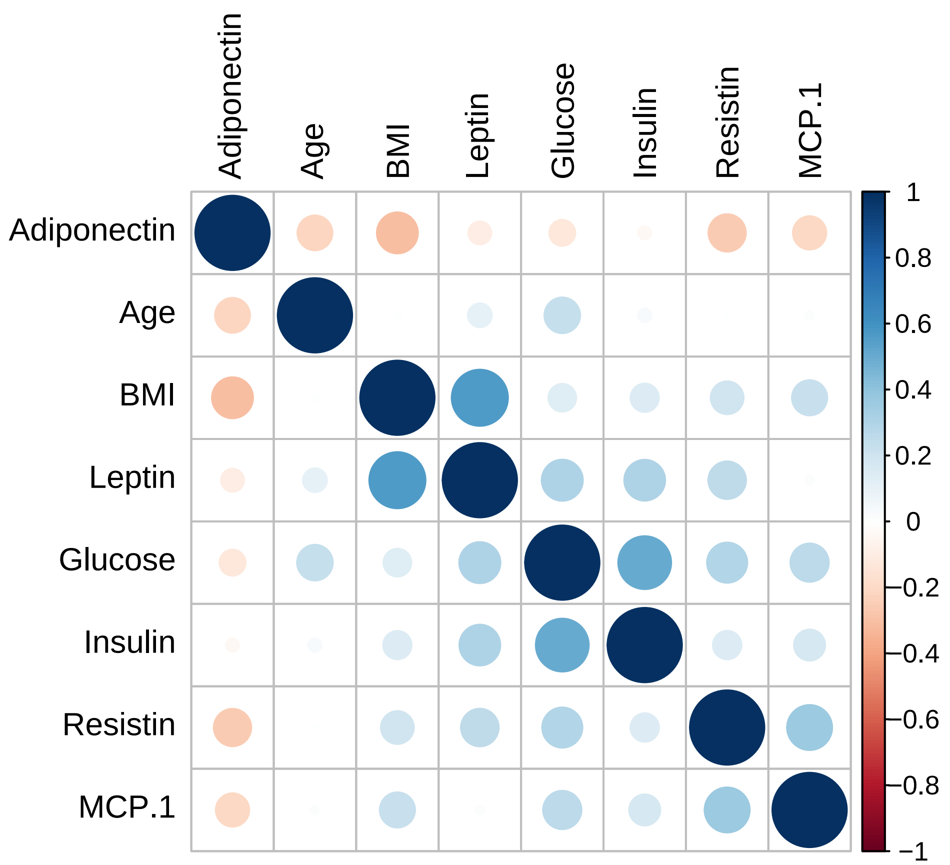


```r
corrplot(round(cor(dataset[,1:9]),3), method = "number", order = "hclust", tl.col = "black", tl.srt = 9
```

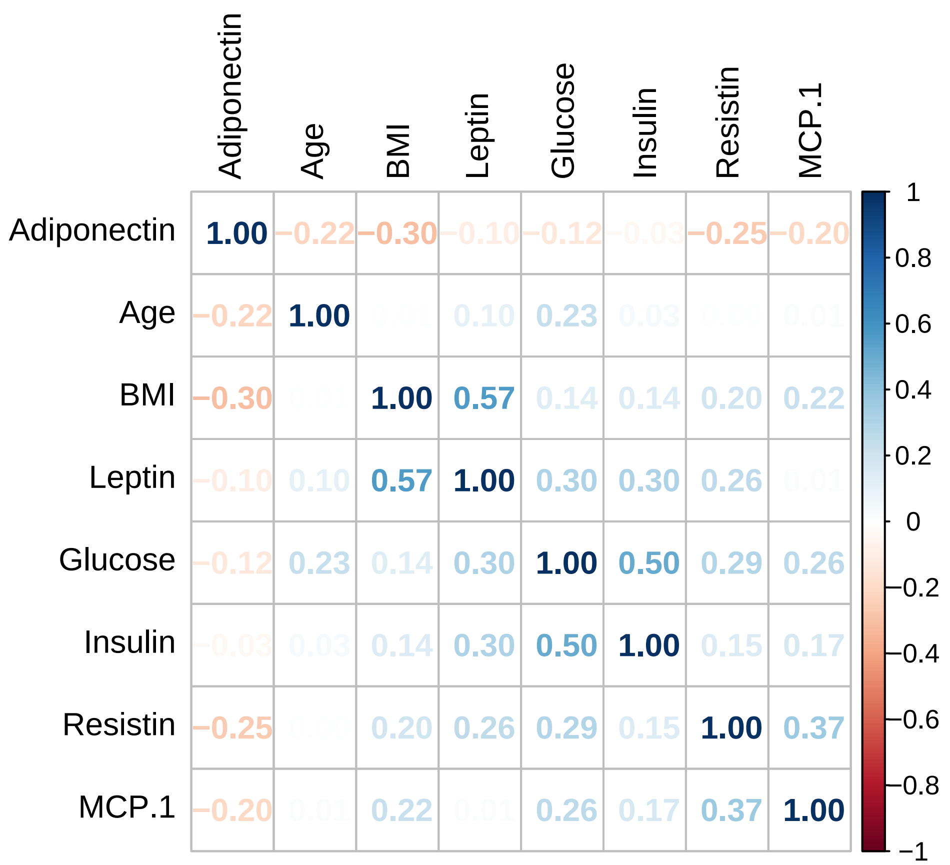


We have two identical correlation matrix but one with using only colors to visually represent the correlation between each variables and one with actual values of correlation between each variable. Most of variables are not significantly correlated to each other except for a few noticeable correlation. Insulin and HOMA has very close correlation and it is because there is a counteraction lying down between insulin and HOMA. Simply, high insulin level produces more HOMA to control blood sugar level. Same thing applies to glucose and HOMA. Because logistic regression model requires little or no data multicollinearity, It would be easier to remove HOMA from our analysis to fix data multicollinearity. This would help to predict more accurate coefficients and reduces potential of obtaining wrong p-values. After removal of HOMA variable, we will going to check variance inflation factor to determine data multicollinearity is in control in further study. Let's obtain a new correlation matrix after removal of HOMA variable.

```r
dataset = subset(dataset, select = -HOMA)
corrplot(round(cor(dataset[,1:8]),3), order = "hclust", tl.col = "black", tl.srt = 90)
```



```
corrplot(round(cor(dataset[,1:8]),3), method = "number", order = "hclust", tl.col = "black", tl.srt = 9
```



After removing HOMA variable, we reduced the number of significant correlation between each of our variable. The shape of our data seems more stable than before and we can perform further analysis with this version of data set.

Logistic Regression Model

As we've mentioned above, we have decided to use a logistic regression model to analyze through the data set. Logistic regression is a statistical model that can be used for a model with binary dependent variable. It is efficient to use logistic regression to predict our two possible outcomes in terms of probabilities. The logistic model uses a logistic function that we can assure to have estimated probabilities lie between 0 to 1 by converting log odds to probability and then we can find what combination of variables can be included in our best model to predict.

The probability of presence or absence of breast cancer can be shown as p (presence) and 1-p (absence). Most of the time, the probability and independent variables are non-linear and we can't guarantee to have probability between 0 and 1. In order to make a proper linear regression model, we have to find the ratio of two probabilities then we put a log function to it. This is called as log odds function and this will make a linear regression model and the parameter B are estimated by MLE. We can also derive p and 1-p from the equation and it can be shown as below.

$$log(Odds) = ln(p/(1-p)) = \beta_0 + \beta_1 x_1 + \ ... \ + \beta_n x_n$$

$$p = exp(\beta_0 + \beta_1 x_1 + \ ... \ + \beta_n x_n)/(1 + exp(\beta_0 + \beta_1 x_1 + \ ... \ + \beta_n x_n))$$

$$1 - p = 1/(1 + exp(\beta_0 + \beta_1 x_1 + \ ... \ + \beta_n x_n))$$

Now, we can easily fit a logistic regression model with glm() function in R since logistic regression model has a class of generalized linear model. Before we fit it, we want to split train set and test set for accuracy check later on. We will split train set for 70% of the data and 30% of the data for test set. Let's look at the summary of logistic regression model with all of independent variables in it.

```r
library(car)
```

```
## Warning: package 'car' was built under R version 4.0.5

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.0.5
```

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::recode() masks car::recode()
## x purrr::some()   masks car::some()
```

```r
split = round(nrow(dataset)*0.7)
train_set = dataset[1:split,]
test_set = dataset[split:nrow(dataset),]
nrow(train_set) # Number of samples for train set
```

```
## [1] 81
```

```r
nrow(test_set) # Number of samples for test set
```

```
## [1] 36
```

```r
model <- glm(Classification ~ ., data = train_set, family = "binomial")
summary(model)
```

```
##
## Call:
## glm(formula = Classification ~ ., family = "binomial", data = train_set)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.27885  -0.64769   0.09479   0.62198   1.83322
##
## Coefficients:
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.123e+00  3.424e+00  -2.080 0.037481 *
## Age         -1.498e-02  2.039e-02  -0.735 0.462349
## BMI         -1.566e-01  9.126e-02  -1.716 0.086227 .
## Glucose      1.085e-01  3.126e-02   3.471 0.000518 ***
## Insulin      4.959e-02  4.001e-02   1.239 0.215167
## Leptin      -5.036e-03  2.258e-02  -0.223 0.823522
## Adiponectin -1.350e-02  5.130e-02  -0.263 0.792475
## Resistin     1.689e-01  5.607e-02   3.012 0.002599 **
## MCP.1       -8.395e-05  9.397e-04  -0.089 0.928816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 110.194  on 80  degrees of freedom
## Residual deviance:  69.169  on 72  degrees of freedom
## AIC: 87.169
##
## Number of Fisher Scoring iterations: 6
```

```r
model$coefficients %>%
  exp() %>%
  round(3)
```

```
## (Intercept)         Age         BMI     Glucose     Insulin      Leptin
##       0.001       0.985       0.855       1.115       1.051       0.995
## Adiponectin    Resistin       MCP.1
##       0.987       1.184       1.000
```

For the training set, we have total of 81 samples and 36 samples for test set. In the summary, we obtained p-values for all independent variables and there are only 3 variables that are less than p-value of 0.1 which are BMI, Glucose and Resistin. Since they are less than p-value of 0.1, we can assume that they are significant variables to this logistic model. We can note on this when we perform variable selection later on. The last row of the summary is the values of odds ratio for each variables. In this way, we can interpret how much portions of each variables are going to affect the probability. The value of odds ratio indicates what relationship each variable can have with presence of breast cancer. For example, when odds ratio is greater than 1, it implies that there are positive relationship and if the odds ratio is less than 1, it means a negative relationship. In the aspect of numbers, we can see that Glucose has 1.115 odds ration that it implies there is a positive relationship with presence of breast cancer. Simply, we can assume that 1 unit increases of Glucose, can increase 1.115 times of being presence of breast cancer. Whereas BMI has 0.855 odds ratio, that means there is a negative relationship with presence of breast cancer. In other word, 1 unit of increaseS in BMI, increases of being presence of breast cancer by 0.855 times which leads to a decrease in probability. We can also check the variance inflation factor to measure of the amount of multicollinearity of each variable by using vif() function in car package.

```r
vif(model)
```

```
##         Age         BMI     Glucose     Insulin      Leptin Adiponectin
##    1.218087    2.855700    1.547442    1.058740    2.180682    1.362526
##    Resistin       MCP.1
##    1.558494    1.412847
```

It seems like the scores are all in acceptable level of multicollinearity. We want to keep them below 5 for better prediction of coefficient and this wouldn't be the case if we kept HOMA variable in the model. The variables with significant correlation will have larger score than what we obtained now.

Since this is not going to be our final model because there are non-significant variables are also in the model but, now we have some ideas of what variables are likely to be in our final model and we make sure to check with appropriate variable selection process. We can choose many different methods for variable selection so we will try a few effective methods to find out and compare them.

Model Selection

The performance of model can be vary depends on what combination of variables that we're putting into our model and the criteria of our best performing model is depends on least error and bias occur with high accuracy. AIC/BIC are the fundamental basis for model selection and we can evaluate how well our models fits the data set. Recall that we have 8 independent variables to select and there are many possible combinations we can use. We're not going to test out all of the combination so we will make 5 different models to test out AIC/BIC for each model to interpret which combinations of variables can perform the best.

Here are the list of models that we're going to test out.

- Model 1: Classification ~ BMI + Gloucse + Resistin
- Model 2: Classification ~ Age + BMI + Glucose + Insulin + Leptin + Adiponectin + Resistin + MCP-1
- Model 3: Classification ~ Resistin + Glucose + BMI + Age
- Model 4: Classification ~ Insulin + BMI + Resistin + Age
- Model 5: Classification ~ Insulin + BMI + Adiponectin + Age

```
model_1 = glm(Classification ~ BMI + Glucose + Resistin, data = train_set, family="binomial")
model_2 = glm(Classification ~ ., data = train_set, family = "binomial")
model_3 = glm(Classification ~ Resistin + Glucose + BMI + Age, data = train_set, family = "binomial")
model_4 = glm(Classification ~ Insulin + BMI + Resistin + Age, data = train_set, family = "binomial")
model_5 = glm(Classification ~ Insulin + BMI + Adiponectin + Age, data = train_set, family = "binomial"
```

Now, we're going to find AIC/BIC for each model with aictab()/bictab() function in AICcmodavg package.

```
library(AICcmodavg)
```

```
## Warning: package 'AICcmodavg' was built under R version 4.0.5
```

```
models <- list(model_1, model_2, model_3, model_4, model_5)
models.names <- c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5")

aictab(models, modnames = models.names)
```

```
##
## Model selection based on AICc:
##
##         K   AICc Delta_AICc AICcWt Cum.Wt     LL
## Model 1 4  80.32       0.00   0.69   0.69 -35.90
## Model 3 5  81.96       1.64   0.30   0.99 -35.58
## Model 2 9  89.70       9.38   0.01   1.00 -34.58
## Model 4 5 100.03      19.71   0.00   1.00 -44.62
## Model 5 5 112.09      31.76   0.00   1.00 -50.64
```

```
bictab(models, modnames = models.names)
```

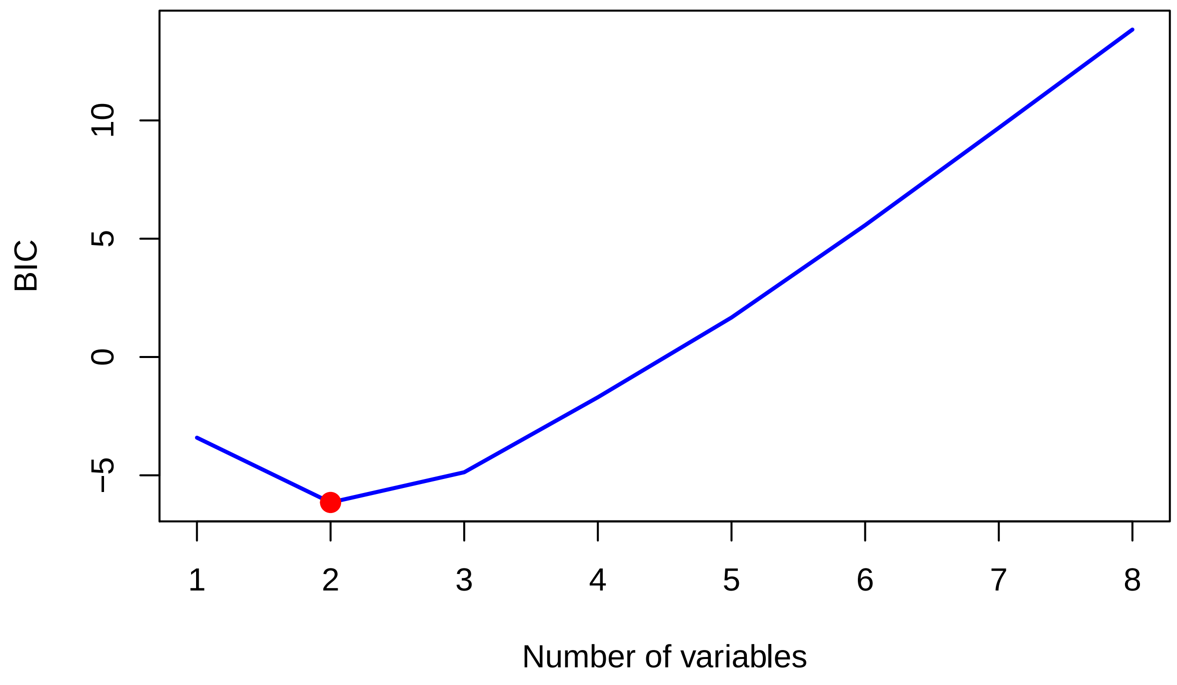
```
##
## Model selection based on BIC:
##
##         K    BIC Delta_BIC BICWt Cum.Wt     LL
## Model 1 4  89.38      0.00  0.87   0.87 -35.90
## Model 3 5  93.14      3.76  0.13   1.00 -35.58
## Model 2 9 108.72     19.34  0.00   1.00 -34.58
## Model 4 5 111.21     21.83  0.00   1.00 -44.62
## Model 5 5 123.26     33.88  0.00   1.00 -50.64
```

We have obtained 2 summaries of AIC and BIC for each model. These summaries indicates which model has lowest AIC/BIC score. We obtained model 1 and model 2 has similar AIC/BIC score but model 1 has slightly lower AIC/BIC than model 2. This can be mean that model 1 performs the best among those models. However, this is an early-assumption and we can use other model selection process to find better logistic regression model than model 1. This time, we can try to use regsubsets() function in the package leaps to analyze/visualize to determine how many variables are suitable for this model and what combination of variables can perform as good as model 1.

```
library(leaps)
```

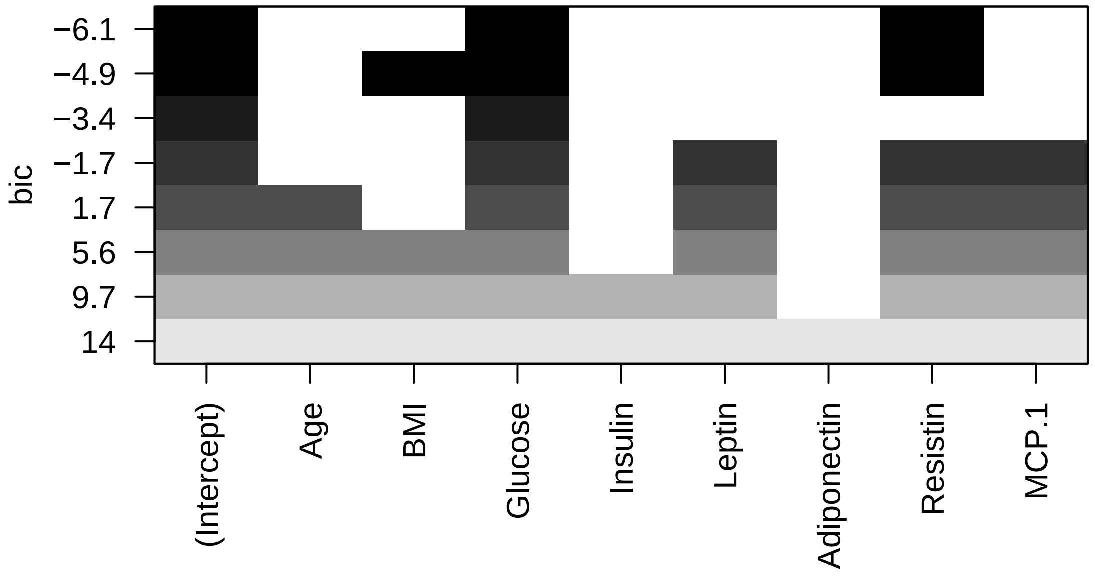
```
## Warning: package 'leaps' was built under R version 4.0.5
```

```
regsubset = regsubsets(as.matrix(train_set[,1:8]), y = train_set[,9], nvmax=8)
result = summary(regsubset)
min_bic = which.min(result$bic)
plot(result$bic, xlab = "Number of variables", ylab = "BIC", type = "l", col = "blue", lwd=2)
points(min_bic, result$bic[min_bic], col = "red", cex = 2, pch = 20)
```



This is a plot between the number of variables versus BIC score. We noticed that the red point indicates which number of variables can obtained the lowest BIC value. As we can see, the plot shows that the lowest BIC value can be obtained with 2 numbers of variables. Previously, the variables that we selected for model 1 was total of 3 variables, BMI, Glucose and Resistin but this plot shows that we can get an ideal logistic regression model with 2 selection of variables. Let's find out what those 2 variables should be.

```r
plot(regsubset, scale = "bic")
```



As we can see in the plot, there are only 2 variables that are reaching -6.1 bic score in the y-axis, Glucose and Resistin and they are also subset of model 1 variables. This model selection process recommends us to use only Glucose and Resistin to perform better than model 1 and this can be true because this process runs the whole combination of variables with identifying the ideal number of variables whereas we only used the 5 set of combinations in the first variable selection with randomly selected variables.

Lastly, we want to perform backward step-wise modeling to identify the best combination of variable set. We're going to set all of variables at first then the process will simulate through the predictors to determine what variables are not significant in the set by comparing deviance score. Every iteration removes non-significant variables from the model until it reaches to a point. We can perform this by using step() function.

```r
step <- step(model , direction = "backward")
```

```
## Start:  AIC=87.17
## Classification ~ Age + BMI + Glucose + Insulin + Leptin + Adiponectin +
##     Resistin + MCP.1
##
##               Df Deviance    AIC
## - MCP.1        1   69.177 85.177
## - Leptin       1   69.219 85.219
## - Adiponectin  1   69.239 85.239
## - Age          1   69.719 85.719
## - Insulin      1   71.042 87.042
## <none>             69.169 87.169
## - BMI          1   72.394 88.394
## - Resistin     1   83.176 99.176
## - Glucose      1   88.165 104.165
##
## Step:  AIC=85.18
## Classification ~ Age + BMI + Glucose + Insulin + Leptin + Adiponectin +
##     Resistin
##
##               Df Deviance     AIC
## - Leptin       1   69.220  83.220
## - Adiponectin  1   69.253  83.253
## - Age          1   69.736  83.736
## - Insulin      1   71.046  85.046
## <none>             69.177  85.177
## - BMI          1   73.080  87.080
## - Resistin     1   84.609  98.609
## - Glucose      1   89.035 103.035
##
## Step:  AIC=83.22
## Classification ~ Age + BMI + Glucose + Insulin + Adiponectin +
##     Resistin
##
##               Df Deviance     AIC
## - Adiponectin  1   69.308  81.308
## - Age          1   69.752  81.752
## - Insulin      1   71.078  83.078
## <none>             69.220  83.220
## - BMI          1   76.871  88.871
## - Resistin     1   84.618  96.618
## - Glucose      1   89.193 101.193
##
## Step:  AIC=81.31
## Classification ~ Age + BMI + Glucose + Insulin + Resistin
##
##            Df Deviance    AIC
## - Age       1   69.762 79.762
## - Insulin   1   71.164 81.164
## <none>          69.308 81.308
## - BMI       1   77.238 87.238
## - Resistin  1   86.442 96.442
## - Glucose   1   89.235 99.235
##
## Step:  AIC=79.76
## Classification ~ BMI + Glucose + Insulin + Resistin
##
##            Df Deviance    AIC
## <none>          69.762 79.762
## - Insulin   1   71.798 79.798
## - BMI       1   77.428 85.428
## - Resistin  1   87.513 95.513
## - Glucose   1   89.478 97.478
```

After 5 iterations of simulations, the process determined remaining variables are BMI, Glucose, Insulin and Resistin to perform the best model. Previously selected model in the first two method had less than 4 variables but this time, the backward stepwise method stopped simulation with 4 variables containing. Since we have selected 3 set of best model from 3 different methods, we want to compare which set of model can perform the best among them. Here's the list of candidate models.

- Model_S1 : Classification ~ BMI + Gloucse + Resistin
- Model_S2 : Classification ~ BMI + Gloucse
- Model_S3 : Classification ~ BMI + Gloucse + Insulin + Resistin

Now, Let's compare AIC/BIC score for these 3 models.

```
model_s1 = model_1
model_s2 = glm(Classification ~ BMI + Glucose, data = train_set, family = "binomial")
model_s3 = glm(Classification ~ BMI + Glucose + Insulin + Resistin, data = train_set, family = "binomia

models = list(model_s1, model_s2, model_s3)
models.names = c("Model_S1", "Model_S2", "Model_S3")
aictab(models, modnames = models.names)
```

```
##
## Model selection based on AICc:
##
##          K  AICc Delta_AICc AICcWt Cum.Wt     LL
## Model_S1 4 80.32       0.00   0.53   0.53 -35.90
## Model_S3 5 80.56       0.24   0.47   1.00 -34.88
## Model_S2 3 95.33      15.00   0.00   1.00 -44.51
```

```
bictab(models, modnames = models.names)
```

```
##
## Model selection based on BIC:
##
##          K    BIC Delta_BIC BICWt Cum.Wt     LL
## Model_S1 4  89.38      0.00  0.76   0.76 -35.90
## Model_S3 5  91.73      2.36  0.23   1.00 -34.88
## Model_S2 3 102.20     12.82  0.00   1.00 -44.51
```

The model_s1 has the lowest AIC/BIC compare to other two models. Therefore, we're going to use model_s1 as our best model and test the accuracy with our test set and analyze bias.

Accuracy of our Model

Before we run prediction with our test set, we want to make sure to interpret a few details of our model. In the coefficient section in the model summary, we can find estimate of coefficients for each variables along with standard errors. We can construct 95% confidence interval for each estimate betas to evaluate the length of each confidence intervals. We're going to use two different methods to construct 95% confidence intervals to compare which method has shorter length of confidence interval and we're going to construct 95% confidence interval for odds ratio for each estimate betas.

```
library(kableExtra)
```

```
## Warning: package 'kableExtra' was built under R version 4.0.5
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
best_model = model_s1
summary = summary(best_model)
```

```r
cf = c()
cf_odd = c()
cf_bon = c()
m = ncol(train_set) -1

alpha = 0.05
a1 = 1-alpha/2
a2 = 1-alpha/(2*m)

for (i in 1:4) {
  interval = c(summary$coefficients[,1][i] - qt(a1, df = summary$df[2])*summary$coefficients[,2][i],
               summary$coefficients[,1][i] + qt(a1, df = summary$df[2])*summary$coefficients[,2][i])
  cf = rbind(cf, interval)
}

for (i in 1:4) {
  interval_bon = c(summary$coefficients[,1][i] - qt(a2, df = summary$df[2])*summary$coefficients[,2][i]
                   summary$coefficients[,1][i] + qt(a2, df = summary$df[2])*summary$coefficients[,2][i]
  cf_bon = rbind(cf_bon, interval_bon)
}

for (i in 1:4) {
  interval_odd = c(exp(summary$coefficients[,1][i] -  qt(a1, df = summary$df[2])*summary$coefficients[,
                   exp(summary$coefficients[,1][i] +  qt(a1, df = summary$df[2])*summary$coefficients[,
  cf_odd = rbind(cf_odd, interval_odd)
}

length = cf[,2] - cf[,1]
row.names(cf) = c("B0", "B1", "B2", "B3")
colnames(cf) = c("Lower", "Upper")
names(length) = c("B0_Length","B1_Length","B2_Length","B3_Length")

length_bon = cf_bon[,2] - cf_bon[,1]
row.names(cf_bon) = c("B0", "B1", "B2", "B3")
colnames(cf_bon) = c("Lower", "Upper")
names(length_bon) = c("B0_Length","B1_Length","B2_Length","B3_Length")

length_odd = cf_odd[,2] - cf_odd[,1]
row.names(cf_odd) = c("Odd_ratio 1","Odd_ratio 2","Odd_ratio 3","Odd_ratio 4" )
colnames(cf_odd) = c("Lower", "Upper")
names(length_odd) = c("Odd1_Length","Odd2_Length","Odd3_Length","Odd4_Length")

result = cbind(summary$coefficients[,1], cf[,1], cf[,2], cf_bon[,1], cf_bon[,2])
colnames(result) = c("beta", "Lower", "Upper", "Lower", "Upper")
kable = kable(result, align="c", digits=3)
add_header_above(kable, header=c(" ", " ", "Student_t"= 2, "Bonferroni"=2), align="c")
```

		Student_t		Bonferroni	
	beta	Lower	Upper	Lower	Upper
(Intercept)	-7.863	-13.381	-2.344	-15.654	-0.071
BMI	-0.150	-0.272	-0.028	-0.322	0.022
Glucose	0.107	0.049	0.165	0.026	0.189
Resistin	0.167	0.065	0.268	0.024	0.310

The table above shows student_t and bonferroni 95% confidence intervals of estimate betas. Each estimate beta contains in the following confidence intervals but the length of two confidence interval are different. Let's compare the length of two confidence intervals.

```r
ints = cbind(length, length_bon, length_bon - length, qt(a2, df = summary$df[2])/qt(a1, df = summary$df
colnames(ints) = c("Student_t", "Bonferroni", "Difference", "Bonferroni/Student_t")
kable(ints, align="c", digits=3)
```

	Student_t	Bonferroni	Difference	Bonferroni/Student_t
B0_Length	11.037	15.583	4.547	1.412
B1_Length	0.243	0.343	0.100	1.412
B2_Length	0.116	0.163	0.048	1.412
B3_Length	0.203	0.287	0.084	1.412

We can observe the length of each confidence interval and as we can see in the table, the Bonferroni's confidence interval is wider than student_t's confidence interval by a factor of 1.412. We would choose narrower confidence interval because the wider confidence interval indicates instability in terms of predicting estimate coefficicents. The logistic regression model can also construct 95% confidence interval for odds ratio. The odds ratio are in an exponential form with power of lower bound and upper bound for each estimate beta. The equation can be shown as following way.

$$CI(Odds) = (exp(\beta_i - t_{1-a/2,df} * se(Bi)), exp(\beta_i + t_{1-a/2,df} * se(Bi))), i = 1, 2, 3, 4$$

```r
kable(cf_odd, digits=3, align="c")
```

	Lower	Upper
Odd_ratio 1	0.000	0.096
Odd_ratio 2	0.762	0.972
Odd_ratio 3	1.051	1.179
Odd_ratio 4	1.068	1.308

Prediction and Accuracy, Bias

We've obtained the best model and now we can use our training set to train our model then use our test set to compare our result to find accuracy and overall prediction error.

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 4.0.5

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
prediction = predict(best_model, newdata = data.frame(test_set), type="response")
pred_check = c()
p = 3
for (i in 1:length(prediction)) {
  if (round(prediction [i],0) == test_set[,9][i]) {
    pred_check[i] = 1
  }
  else {
    pred_check[i] = 0
  }
}

sigma = sd(test_set[,9])
res = sum((test_set[,9] - prediction)^2)
cp = (res +2*sigma^2*p)/nrow(test_set)

result_pred = cbind(prediction, round(prediction,0) ,test_set[,9], (test_set[,9]-prediction)^2, pred_che
colnames(result_pred) = c("Probability", "Prediction", "True", "Apparent error", "Prediction_check")
kable(result_pred, align="c")
```

	Probability	Prediction	True	Apparent error	Prediction_check
58	0.9009635	1	1	0.0098082	1
28	0.7258284	1	0	0.5268269	0
65	0.6439005	1	1	0.1268069	1
60	0.5964594	1	1	0.1628450	1
116	0.9730828	1	1	0.0007245	1
20	0.2393399	0	0	0.0572836	1
90	0.9933359	1	1	0.0000444	1
73	0.7648911	1	1	0.0552762	1
67	0.3668021	0	1	0.4009396	0
17	0.1674220	0	0	0.0280301	1
52	0.3391542	0	0	0.1150256	1
77	0.1684227	0	1	0.6915208	0
41	0.1189923	0	0	0.0141592	1
92	0.8187003	1	1	0.0328696	1
115	0.0861806	0	1	0.8350660	0
47	0.2919936	0	0	0.0852602	1
21	0.1980298	0	0	0.0392158	1
63	0.7550337	1	1	0.0600085	1
64	0.8427233	1	1	0.0247360	1
94	0.9310207	1	1	0.0047581	1
6	0.5739389	1	0	0.3294059	0
40	0.1023888	0	0	0.0104835	1
89	1.0000000	1	1	0.0000000	1
19	0.1498560	0	0	0.0224568	1
96	0.2046350	0	1	0.6326055	0
38	0.9999888	1	0	0.9999777	0
57	0.8590209	1	1	0.0198751	1
56	0.5941696	1	1	0.1646983	1
113	0.5144583	1	1	0.2357507	1
44	0.3106500	0	0	0.0965034	1
12	0.1473655	0	0	0.0217166	1
36	0.0419092	0	0	0.0017564	1
43	0.1711546	0	0	0.0292939	1
2	0.3963623	0	0	0.1571030	1
5	0.6462862	1	0	0.4176859	0
1	0.0726733	0	0	0.0052814	1

```
confusion_matrix = confusionMatrix(data = factor(c(round(prediction,0))), reference = factor(c(test_set
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 14  4
##          1  4 14
##
##                Accuracy : 0.7778
##                  95% CI : (0.6085, 0.8988)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0005966
##
##                   Kappa : 0.5556
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.7778
##             Specificity : 0.7778
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.7778
##              Prevalence : 0.5000
##          Detection Rate : 0.3889
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.7778
##
##        'Positive' Class : 0
##
```

```r
cat("Overall Prediction Error = ", cp)
```

```
## Overall Prediction Error =  0.2210738
```

The first column of table indicates the predicted probabilities of our test set using our best model along with our second column indicates the round up to one digit number of our predicted probabilities with using 0.5 as our cut-off value. The third column is the true value of our test set (1 = Presence, 0 = Absence) and the forth column shows apparent error of each row, and the equation of apparent error is equal to APE = (True - Probability)^2. The last column is for prediction check that when the prediction and true value are equal, the prediction check will indicate 1 if not, it will be 0. Our prediction are in a form of probability, if our prediction is greater 0.5, then we would consider the prediction as 1 which is presence of breast cancer. For example, the first row indicates probability equal to 0.9009635. Our best model predicted the patient number 58 has 90% of chance to have presence of breast cancer so we would suspect patient from presence of breast cancer and as we can see in the true column, the patient number 58 does have presence of breast cancer. The bias of our first prediction would be 1 - 0.9009635 = 0.0990365, approximately 10% of error rate. Therefore, our model predicted the true value correctly, so we would indicate 1 in prediction check column for patient number 58. We have total of 36 samples in our test set and not all of our prediction were correct. In order to interpret the overall result of our prediction, it would be easier to compute confusion matrix to review our prediction along with some details. We used confusionMatrix() function in caret package to compute confusion matrix. Before we look into confusion matrix, we need to remind that the type of errors. If we used this model as a biomarker to test presence or absence of breast cancer and if our model made wrong prediction, we would face two type of errors. Type I error would be our biomarker says a patience is not suspicious with presence of breast cancer, but patient actually do have presence of breast cancer. Type II error would be our biomarker says a patient is suspicious with presence of breast cancer, but patient actually don't have breast cancer. In this case, making type I error would be more critical than type II error because if our biomarker suspects a patient with presence of breast cancer, we can pursue further diagnosis to determine presence or absence of breast cancer but type I error wouldn't be the case. So it is important to know what predictions our model is making wrong. When we look at the confusion matrix along with some statistical results, our model made total of 28 predictions correctly, 4 type I error and 4 type II error. This means, we have accuracy of 77.78% with our model and we would focus on narrowing down high risk of making type I error. Also, we have 95% confidence interval for accuracy of the model. The lower bound says it could be down to 60.85% and the upper bound says the accuracy of the model can be up to 89.88%. For overall prediction error, we used Mallows' Cp estimate of prediction error which is equal to 0.2210738, approximately 22.1% overall prediction error.

If we chose a different model from our candidate models, we would have obtained different accuracy and overall prediction error. For specific comparison, let's try to use our model_S2 (BMI + Glucose) to fit our test set to compare results.

```r
prediction = predict(model_s2, newdata = data.frame(test_set), type="response")
pred_check = c()
p = 2
for (i in 1:length(prediction)) {
  if (round(prediction [i],0) == test_set[,9][i]) {
    pred_check[i] = 1
  }
  else {
    pred_check[i] = 0
  }
}

sigma = sd(test_set[,9])
res = sum((test_set[,9] - prediction)^2)
cp = (res +2*sigma^2*p)/nrow(test_set)

result_pred = cbind(prediction, round(prediction,0) ,test_set[,9], (test_set[,9]-prediction)^2, pred_che
colnames(result_pred) = c("Probability", "Prediction", "True", "Apparent error", "Prediction_check")
kable(result_pred, align="c")
```

	Probability	Prediction	True	Apparent error	Prediction_check
58	0.8252893	1	1	0.0305238	1
28	0.3991620	0	0	0.1593303	1
65	0.7415385	1	1	0.0668023	1
60	0.6367311	1	1	0.1319643	1
116	0.9828844	1	1	0.0002929	1
20	0.2451286	0	0	0.0600880	1
90	0.9812105	1	1	0.0003530	1
73	0.8863605	1	1	0.0129139	1
67	0.4775260	0	1	0.2729791	0
17	0.4849432	0	0	0.2351699	1
52	0.4187565	0	0	0.1753570	1
77	0.4286243	0	1	0.3264702	0
41	0.3100871	0	0	0.0961540	1
92	0.6130157	1	1	0.1497569	1
115	0.3649185	0	1	0.4033285	0
47	0.6115013	1	0	0.3739338	0
21	0.3806388	0	0	0.1448859	1
63	0.3754807	0	1	0.3900243	0
64	0.8134482	1	1	0.0348016	1
94	0.9619561	1	1	0.0014473	1
6	0.6325456	1	0	0.4001140	0
40	0.1220173	0	0	0.0148882	1
89	0.9998571	1	1	0.0000000	1
19	0.4339643	0	0	0.1883250	1
96	0.1242060	0	1	0.7670151	0
38	0.5478273	1	0	0.3001148	0
57	0.6822374	1	1	0.1009730	1
56	0.6027052	1	1	0.1578432	1
113	0.7023824	1	1	0.0885763	1
44	0.4377086	0	0	0.1915888	1
12	0.4208912	0	0	0.1771494	1
36	0.2360747	0	0	0.0557312	1
43	0.3880824	0	0	0.1506080	1
2	0.6770457	1	0	0.4583908	0
5	0.6686137	1	0	0.4470443	0
1	0.2007738	0	0	0.0403101	1

```r
confusion_matrix = confusionMatrix(data = factor(c(round(prediction,0))), reference = factor(c(test_set
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 13  5
##          1  5 13
##
##                Accuracy : 0.7222
##                  95% CI : (0.5481, 0.858)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.005665
##
##                   Kappa : 0.4444
```

```
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.7222
##             Specificity : 0.7222
##          Pos Pred Value : 0.7222
##          Neg Pred Value : 0.7222
##              Prevalence : 0.5000
##          Detection Rate : 0.3611
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.7222
##
##        'Positive' Class : 0
##
```

```r
cat("Overall Prediction Error = ", cp)
```

```
## Overall Prediction Error =  0.2120506
```

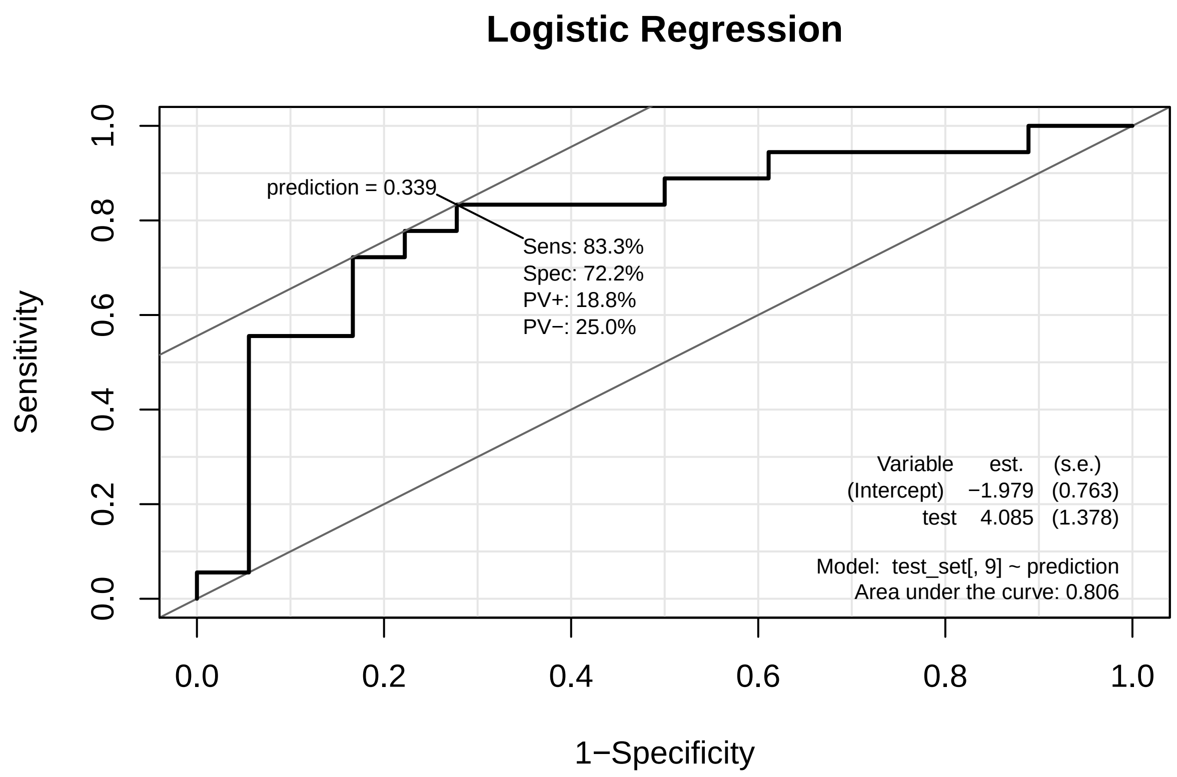
We obtained our model_s2's confusion matrix and it seems like model_s2 is making 2 more critical wrong predictions comparing to our previous model. The overall accuracy of model_s2 is 72.22% and the 95% confidence interval of model's accuracy is (0.5481, 0.858). Our previous model obtained 77.78% accuracy with (0.6085, 0.8988) confidence interval which is slightly performing better than model_s2 in general. However, the overall prediction error for model_s2 is slightly lower than our best model. This is a bit of odd result but as long as our best model has greater accuracy with less type I,II errors, we would say our best model is performing better than model_s2.

Previously, we determined our prediction by rounding up our probabilities to 0 decimal. In other word, our cut-off value was set as 0.5 that if our predicted probability is above 0.5, then we classified as 1 and if not, we classified it as 0. The main idea of cut-off value is that we can identify our best cut-off value to improve our model's accuracy since it can tremendously change our results. To find the ideal cut-off value, we can use ROC curve to visualize the relationship between sensitivity and specificity of our model to determine ideal cut-off value with AUC (Area under curve). To plot ROC curve, we will use ROC() function in Epi package.

```r
library(Epi)
```

```
## Warning: package 'Epi' was built under R version 4.0.5
```

```r
prediction = predict(best_model, newdata = data.frame(test_set), type="response")
ROC(prediction, test_set[,9], plot="ROC", AUC=T, main="Logistic Regression")
```



According to our result of ROC curve, it recommends to set our cut-off value equal to 0.339 (can round up to 0.34). This means, we would want to consider predicted probabilities that are greater than 0.339, we want to classify them as 1. The AUC represent the area under the curve and we obtained AUC equal to 0.806 which is in an acceptable range. Now, let's use our new cut-off value to re-classify our predictions and compare our results with our previous one.

```
cut_off = 0.34
new_prediction = c()
for (i in 1:nrow(test_set)) {
  new_prediction[i] = ifelse(prediction[i] >= cut_off, 1, 0)
}

result_pred = cbind(prediction, new_prediction ,test_set[,9], (test_set[,9]-prediction)^2, pred_check)
colnames(result_pred) = c("Probability", "New_Prediction", "True", "Apparent error", "Prediction_check"
kable(result_pred, align="c")
```

	Probability	New_Prediction	True	Apparent error	Prediction_check
58	0.9009635	1	1	0.0098082	1
28	0.7258284	1	0	0.5268269	1
65	0.6439005	1	1	0.1268069	1
60	0.5964594	1	1	0.1628450	1
116	0.9730828	1	1	0.0007245	1
20	0.2393399	0	0	0.0572836	1
90	0.9933359	1	1	0.0000444	1
73	0.7648911	1	1	0.0552762	1
67	0.3668021	1	1	0.4009396	0
17	0.1674220	0	0	0.0280301	1
52	0.3391542	0	0	0.1150256	1
77	0.1684227	0	1	0.6915208	0
41	0.1189923	0	0	0.0141592	1
92	0.8187003	1	1	0.0328696	1
115	0.0861806	0	1	0.8350660	0
47	0.2919936	0	0	0.0852602	0
21	0.1980298	0	0	0.0392158	1
63	0.7550337	1	1	0.0600085	0
64	0.8427233	1	1	0.0247360	1
94	0.9310207	1	1	0.0047581	1
6	0.5739389	1	0	0.3294059	0
40	0.1023888	0	0	0.0104835	1
89	1.0000000	1	1	0.0000000	1
19	0.1498560	0	0	0.0224568	1
96	0.2046350	0	1	0.6326055	0
38	0.9999888	1	0	0.9999777	0
57	0.8590209	1	1	0.0198751	1
56	0.5941696	1	1	0.1646983	1
113	0.5144583	1	1	0.2357507	1
44	0.3106500	0	0	0.0965034	1
12	0.1473655	0	0	0.0217166	1
36	0.0419092	0	0	0.0017564	1
43	0.1711546	0	0	0.0292939	1
2	0.3963623	1	0	0.1571030	0
5	0.6462862	1	0	0.4176859	0
1	0.0726733	0	0	0.0052814	1

```
confusion_matrix = confusionMatrix(data = factor(c(new_prediction)), reference = factor(c(test_set[,9])
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 13  3
##          1  5 15
##
##                Accuracy : 0.7778
##                  95% CI : (0.6085, 0.8988)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0005966
##
##                   Kappa : 0.5556
```

```
##
##  Mcnemar's Test P-Value : 0.7236736
##
##             Sensitivity : 0.7222
##             Specificity : 0.8333
##          Pos Pred Value : 0.8125
##          Neg Pred Value : 0.7500
##              Prevalence : 0.5000
##          Detection Rate : 0.3611
##    Detection Prevalence : 0.4444
##       Balanced Accuracy : 0.7778
##
##        'Positive' Class : 0
##
```

In overall, the number of errors and accuracy of two cut-off value are the same but the main difference is that we're making less type I error. Because making more type I error is more critical in our case, we want to reduce the number of type I error and it seems like setting our cut-off value as 0.34 can reduce the number of type I error. Overall, we would want to use ideal cut-off value rather than 0.5 for better predictions.

Further Analysis and Conclusion

At the end, we decided to remove HOMA variable from our analysis to prevent data multicollinearity from reducing the precision of the estimate coefficients and p-value due to high correlation of being affect by each other. We performed 3 different model selection process to determine which model can perform the best by evaluating AIC/BIC with following comparison of each models. We chose 3 candidate models from each process and we sorted them out in order of lowest AIC/BIC to select our best model and the best model we selected includes 3 predictors of BMI, Gloucse and Resistin. After the model selection, we constructed 95% confidence intervals for each estimated coefficients by using two different method and compared the length of each methods. Additionally, we constructed one more unique 95% confidence interval for odds-ratio for each estimated coefficient to interpret the length of each intervals. Next, we moved onto fitting a test set to our best model to compare our predictions to true value along with creating a confusion matrix to interpret how many type I error we made. The overall accuracy of our selected model was 77.79% with 0.2210738 overall prediction error by estimation of Mallows' Cp prediction error. To reduce the number of type I error, we plotted a ROC curve to identify an ideal cut-off value to classify our predictions and ended up using 0.34 as our cut-off value. By using our new cut-off value, we obtained 1 less type I error with the same accuracy and number of total errors as before.

For the further analysis, we want to point out a few worrisome aspects of our analysis. Because we are using a data set of limited number of samples (training set = 81, test set = 36, total of 117 samples), we would want to increase the number of sample sizes by collecting more information of both class of samples to increase the precision of our estimation and to eliminate uncertainty. In that way, we can split more samples to train our model efficiently and have more samples for test set to compare our predictions. Also, on the opposite door, there could be another suitable method for model selection that can perform better than the model we've selected, in additionally, there might be a better way to improve our model's accuracy with least bias. If we're planning to use this model as a biomarker and in order to do so, it might be possible to meet certain criterion such as higher accuracy with less error rate to put into practical uses. Therefore, additional adjustments will have to be applied to improve our model's accuracy.

Reference

Here are the list of reference website.

- https://www.macmillanlearning.com/studentresources/college/statistics/psbe5e/companion_chapters/22_psbe5e_10900_ch18_18-1_18-26.pdf
- https://stats.oarc.ucla.edu/r/dae/logit-regression/

- https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/
- https://rpubs.com/zau0hon/logistic_regression
- https://www.journaldev.com/46732/confusion-matrix-in-r