Binary Logistic Regression

Shyam BV

March 15, 2018

Contents

---

1 Binary Linear Regression Model : Predicting whether there is a crime or not

---

Deliverables:

1. A write-up submitted in PDF format. Your write-up should have four sections. Each one is described below. You may assume you are addressing me as a fellow data scientist, so do not need to shy away from technical details.
2. Assigned predictions (the number of wins for the team) for the evaluation data set.
3. Include your R statistical programming code in an Appendix.

```
library(dplyr)
library(ggplot2)
library(MASS)
library(faraway)
library(PerformanceAnalytics)
library(leaps)
library(bestglm)
library(rstanarm)
library(caret)
library(pROC)
```

1.1 Data Exploration

Describe the size and the variables in the moneyball training data set. Consider that too much detail will cause a manager to lose interest while too little detail will make the manager consider that you aren't doing your job. Some suggestions are given below. Please do NOT treat this as a check list of things to do to complete the assignment. You should have your own thoughts on what to tell the boss. These are just ideas.

a. Mean / Standard Deviation / Median
b. Bar Chart or Box Plot of the data and/or Histograms
c. Is the data correlated to the target variable (or to other variables?)
d. Are any of the variables missing and need to be imputed "fixed"?

```
df = read.csv('./data/crime-training-data.csv')
```

Below is the definition of all the predictors in the dataset.

. zn: proportion of residential land zoned for large lots (over 25000 square feet) (predictor variable) . indus: proportion of non-retail business acres per suburb (predictor variable) . chas: a dummy var. for whether the suburb borders the Charles River (1) or not (0) (predictor variable) . nox: nitrogen oxides concentration (parts per 10 million) (predictor variable) . rm: average number of rooms per dwelling (predictor variable) . age: proportion of owner-occupied units built prior to 1940 (predictor variable) . dis: weighted mean of distances to five Boston employment centers (predictor variable) . rad: index of accessibility to radial highways (predictor variable) . tax: full-value property-tax rate per $10,000 (predictor variable) . ptratio: pupil-teacher ratio by town (predictor variable) . black: 1000(Bk - 0.63)2 where Bk is the proportion of blacks by town (predictor variable) . lstat: lower status of the population (percent) (predictor variable) . medv: median value of owner-occupied homes in $1000s (predictor variable) . target: whether the crime rate is above the median crime rate (1) or not (0) (response variable)

1.1.1 Summary Stats and Imputations

Below is the summary of the dataset and some inference of it.

1. It seems there are no Null values in the predictor and response variables.
2. Each variables are in different scale.
3. Categorical variables are `chas` and `target`.
4. There are a total of 466 observations and 13 predictor variables.

```
head(df)
```

```
##   zn indus chas   nox    rm   age    dis rad tax ptratio  black lstat medv
## 1  0 19.58    0 0.605 7.929  96.2 2.0459   5 403    14.7 369.30  3.70 50.0
## 2  0 19.58    1 0.871 5.403 100.0 1.3216   5 403    14.7 396.90 26.82 13.4
## 3  0 18.10    0 0.740 6.485 100.0 1.9784  24 666    20.2 386.73 18.85 15.4
## 4 30  4.93    0 0.428 6.393   7.8 7.0355   6 300    16.6 374.71  5.19 23.7
## 5  0  2.46    0 0.488 7.155  92.2 2.7006   3 193    17.8 394.12  4.82 37.9
## 6  0  8.56    0 0.520 6.781  71.3 2.8561   5 384    20.9 395.58  7.67 26.5
##   target
## 1      1
## 2      1
## 3      1
## 4      0
## 5      0
## 6      0
```

```
print(paste0('Observation count: ',count(df)))
```

```
## [1] "Observation count: 466"
```

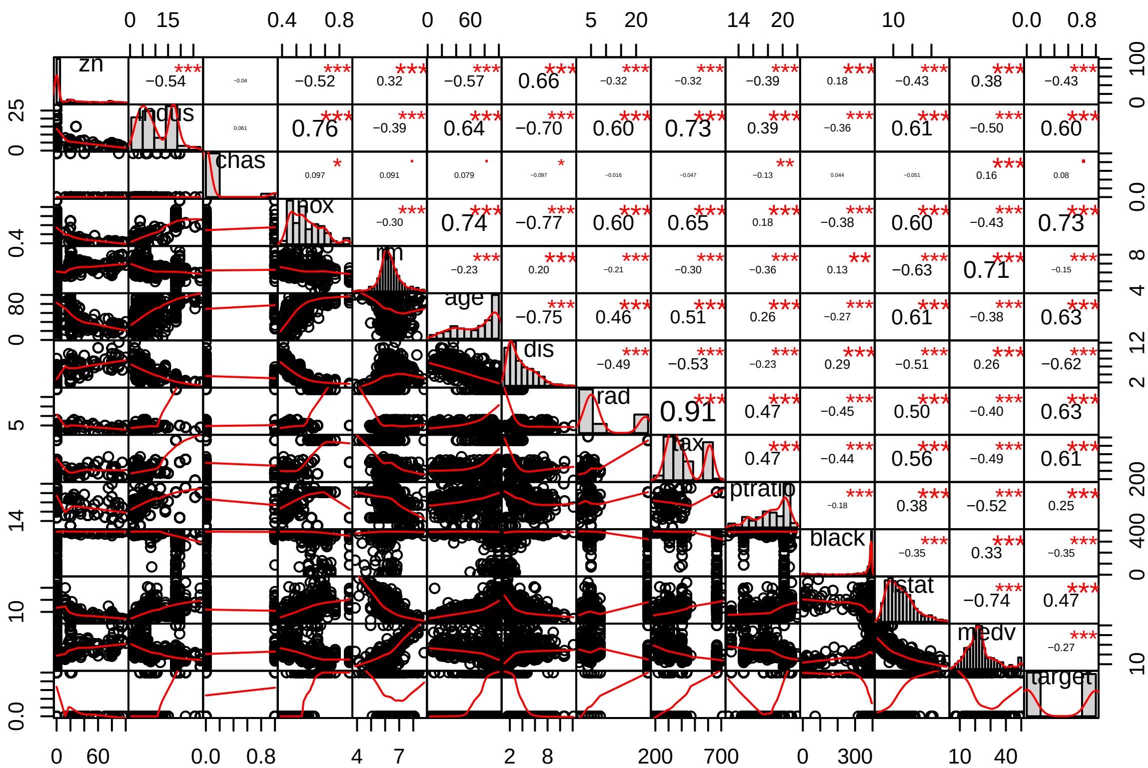
```
summary(df)
```

```
##        zn             indus            chas              nox
##  Min.   :  0.00   Min.   : 0.460   Min.   :0.00000   Min.   :0.3890
##  1st Qu.:  0.00   1st Qu.: 5.145   1st Qu.:0.00000   1st Qu.:0.4480
##  Median :  0.00   Median : 9.690   Median :0.00000   Median :0.5380
##  Mean   : 11.58   Mean   :11.105   Mean   :0.07082   Mean   :0.5543
##  3rd Qu.: 16.25   3rd Qu.:18.100   3rd Qu.:0.00000   3rd Qu.:0.6240
##  Max.   :100.00   Max.   :27.740   Max.   :1.00000   Max.   :0.8710
##        rm             age              dis              rad
##  Min.   :3.863   Min.   :  2.90   Min.   : 1.130   Min.   : 1.00
##  1st Qu.:5.887   1st Qu.: 43.88   1st Qu.: 2.101   1st Qu.: 4.00
##  Median :6.210   Median : 77.15   Median : 3.191   Median : 5.00
##  Mean   :6.291   Mean   : 68.37   Mean   : 3.796   Mean   : 9.53
##  3rd Qu.:6.630   3rd Qu.: 94.10   3rd Qu.: 5.215   3rd Qu.:24.00
##  Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.00
##       tax           ptratio         black            lstat
##  Min.   :187.0   Min.   :12.6   Min.   :  0.32   Min.   : 1.730
##  1st Qu.:281.0   1st Qu.:16.9   1st Qu.:375.61   1st Qu.: 7.043
##  Median :334.5   Median :18.9   Median :391.34   Median :11.350
##  Mean   :409.5   Mean   :18.4   Mean   :357.12   Mean   :12.631
##  3rd Qu.:666.0   3rd Qu.:20.2   3rd Qu.:396.24   3rd Qu.:16.930
##  Max.   :711.0   Max.   :22.0   Max.   :396.90   Max.   :37.970
##       medv           target
##  Min.   : 5.00   Min.   :0.0000
##  1st Qu.:17.02   1st Qu.:0.0000
##  Median :21.20   Median :0.0000
##  Mean   :22.59   Mean   :0.4914
##  3rd Qu.:25.00   3rd Qu.:1.0000
##  Max.   :50.00   Max.   :1.0000
```

1.1.2 Plots and Correlation

Below is the detailed plot of all the variables.

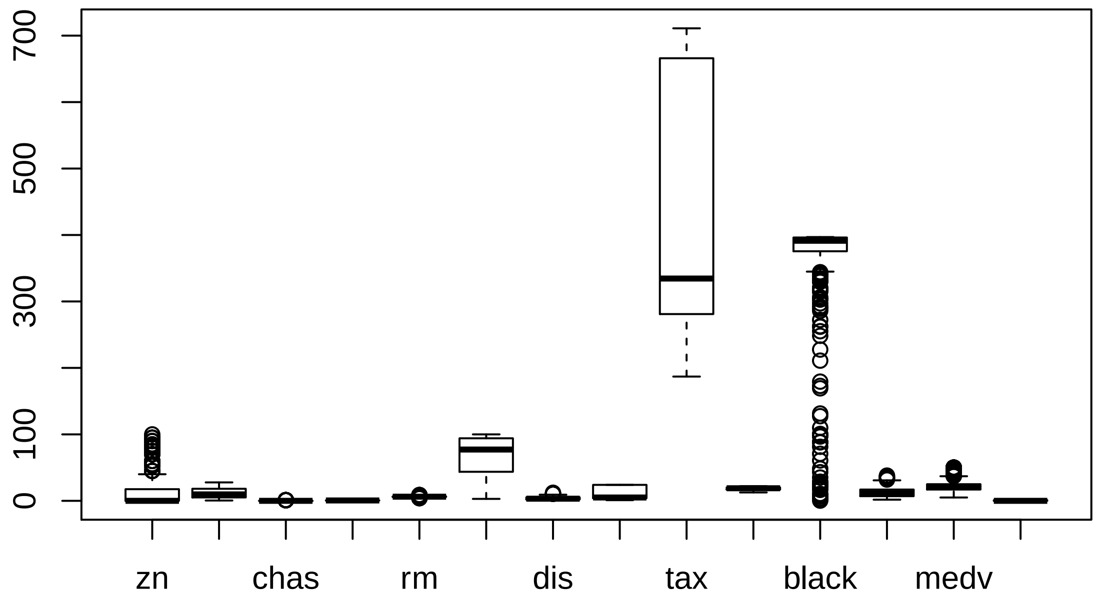
```
chart.Correlation(df,histogram=TRUE,pch=19)
```



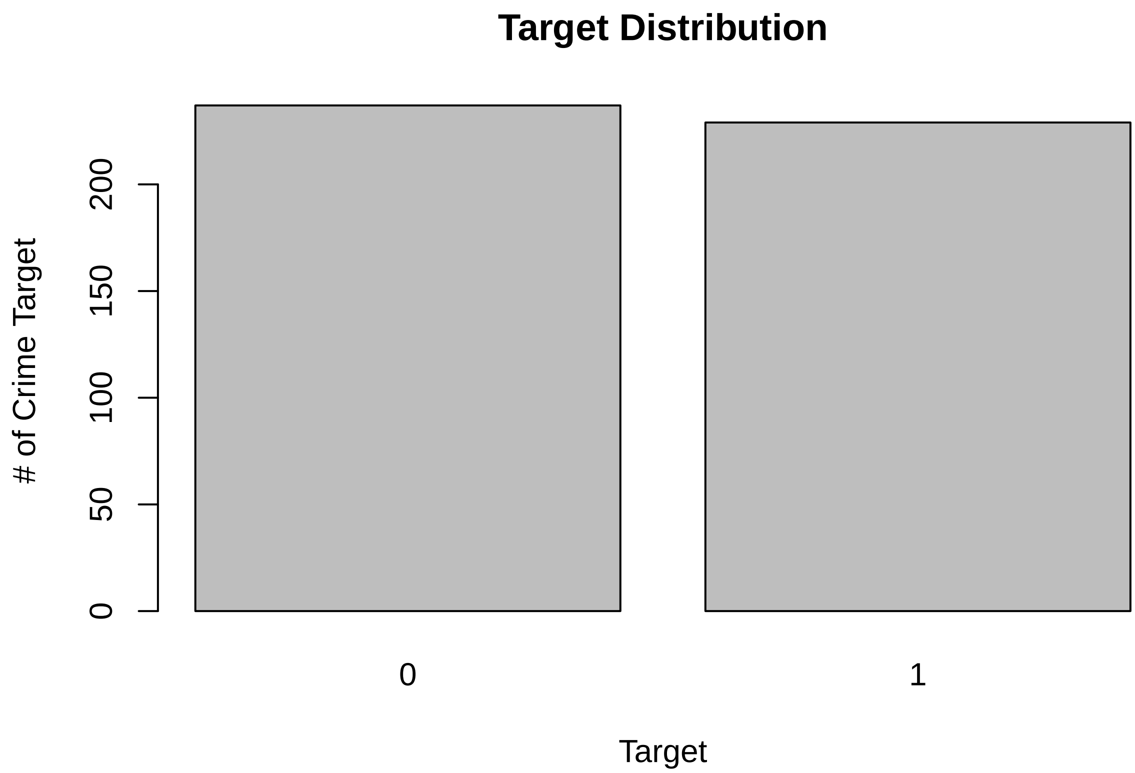
1. Some of the variables like `zn`, `dis`,`black`,`age` are heavily skewed.
2. `tax` and `rad` Variables are heavily correlated. `nox` and `indus`, `nox` and `age`, `medv` and `rm` are moderatly positive correlated.
3. `dis` and `nox`, `dis` and `indus` are negativly correlated.

Below shown are the box plot of all the variables and barplot of target variables.

```
boxplot(df,names = names(df))
```




```
barplot(table(df$target), main = 'Target Distribution', ylab='# of Crime Target',xlab='Target')
```




```r
data.frame(NA_count = sapply(df, function(x) sum(is.na(x))))
```

```
##         NA_count
## zn             0
## indus          0
## chas           0
## nox            0
## rm             0
## age            0
## dis            0
## rad            0
## tax            0
## ptratio        0
## black          0
## lstat          0
## medv           0
## target         0
```

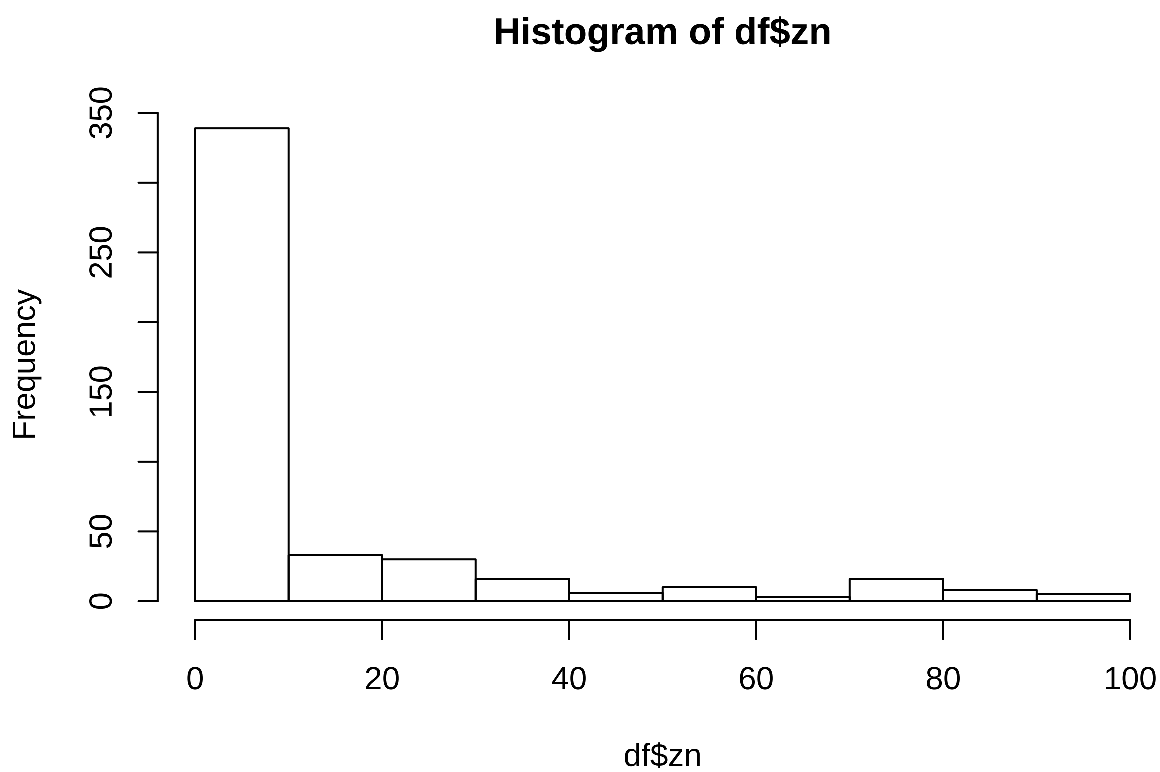
1.2 Data Preparation

Data preparation is an important step of this analysis. As some of the variables are heavily skewed, we need to transform the variables.

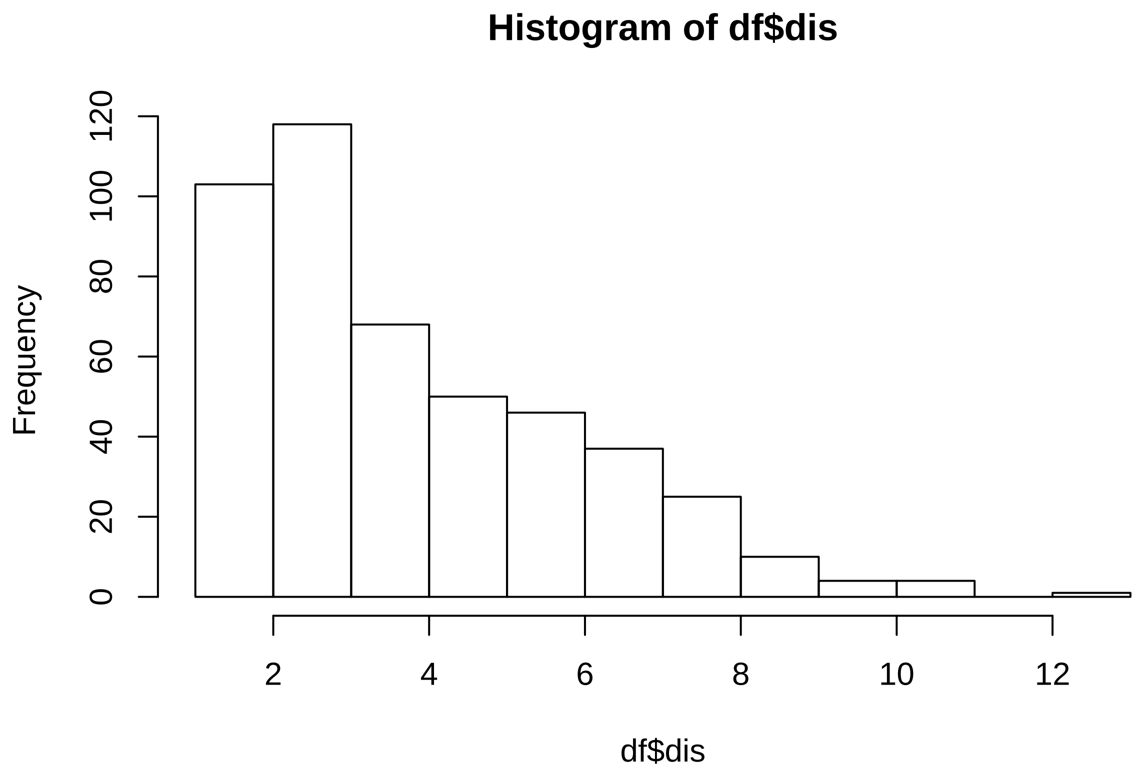
1.2.1 Data Transformations

As these variables `zn`, `dis`,`black`,`age` are skewed, we need to transform the variables.

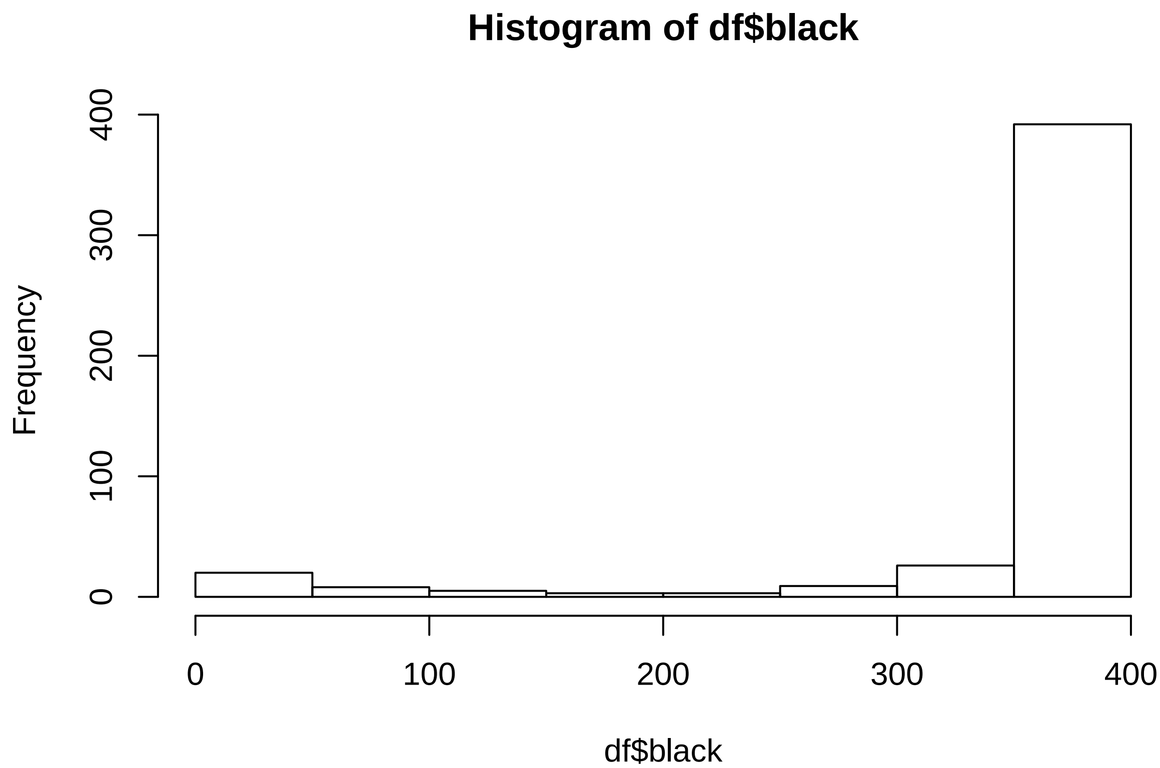
```
hist(df$zn)
```




```
hist(df$dis)
```

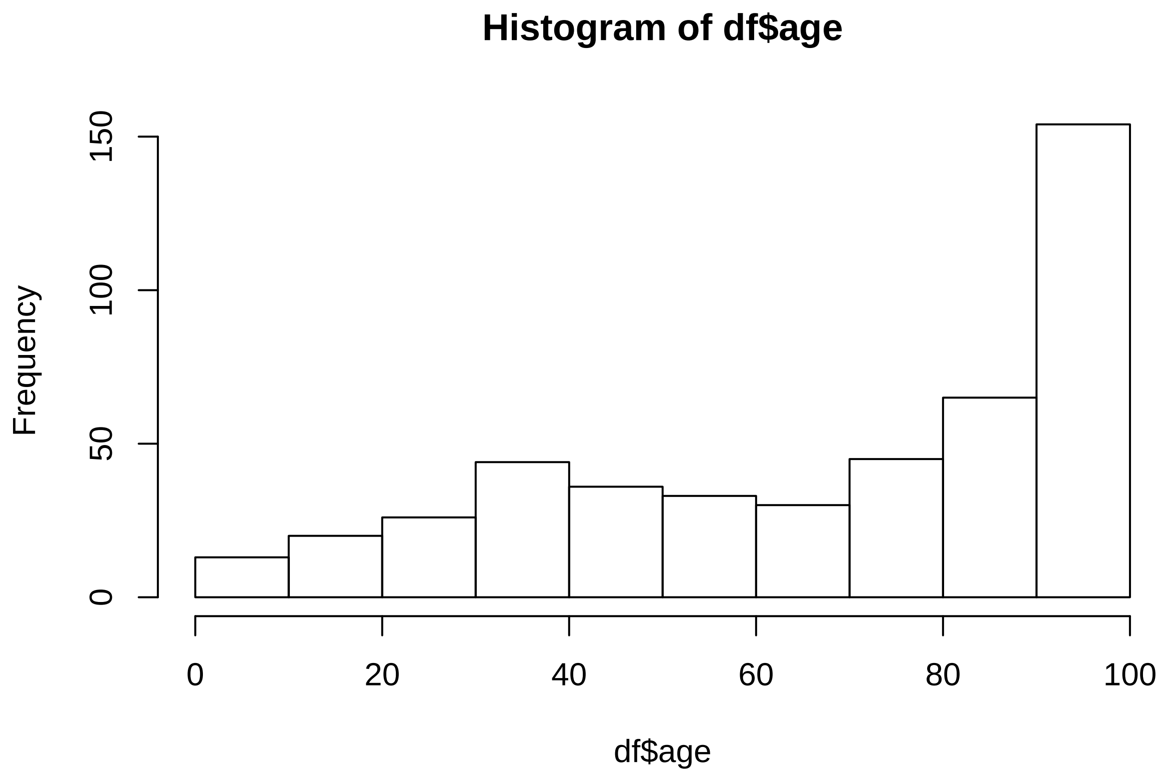



```
hist(df$black)
```

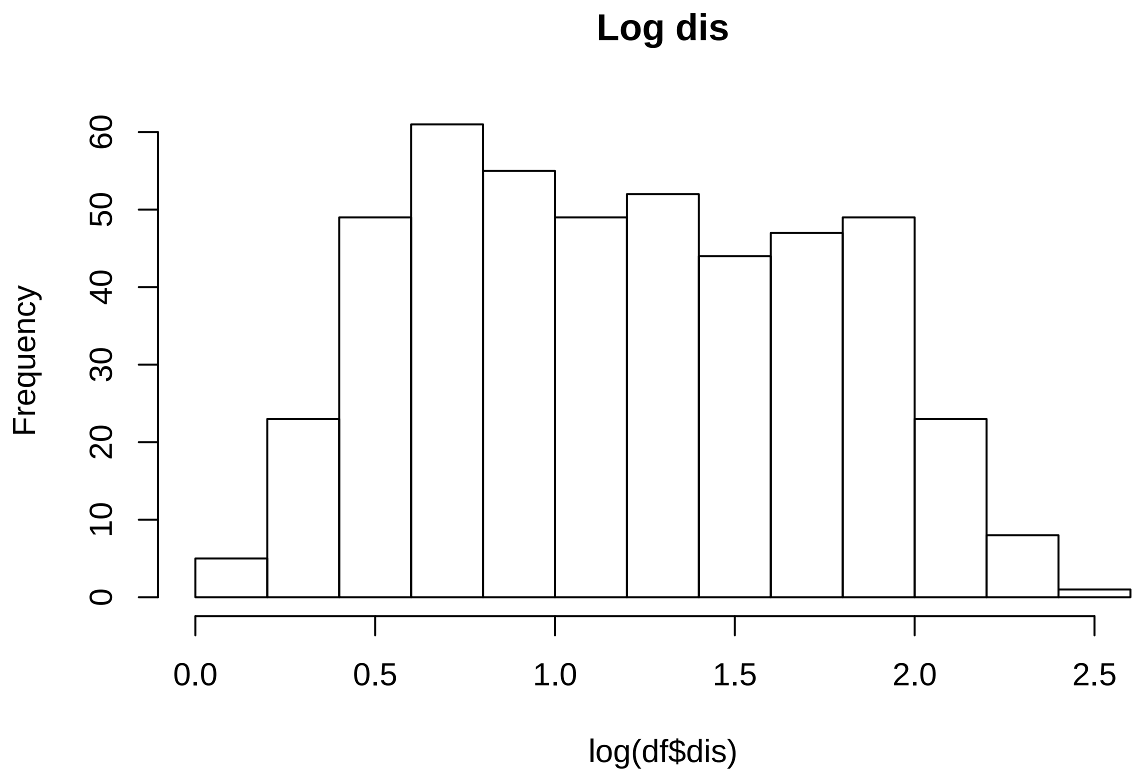


Histogram of df$black
Frequency
0
100
200
300
400
0
100
200
300
400
df$black


```
hist(df$age)
```




```r
hist(log(df$dis),main = 'Log dis')
```

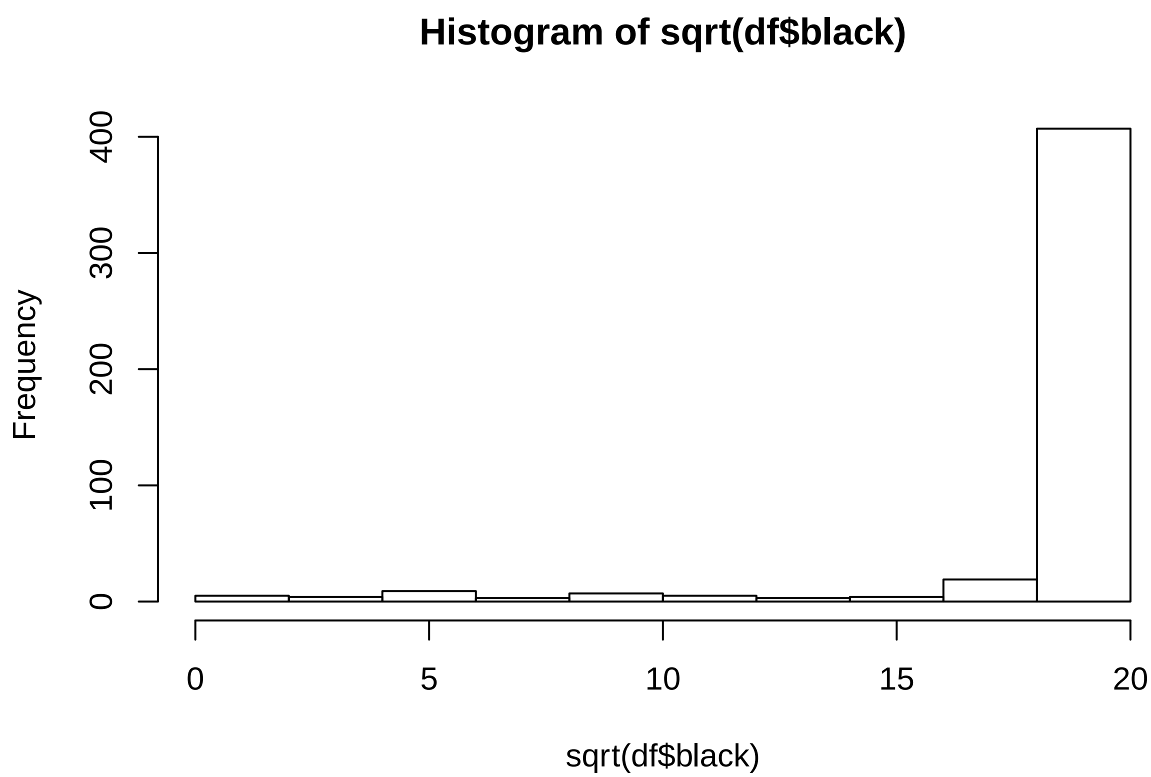



```r
df_transformed = df

summary(df_transformed)
```

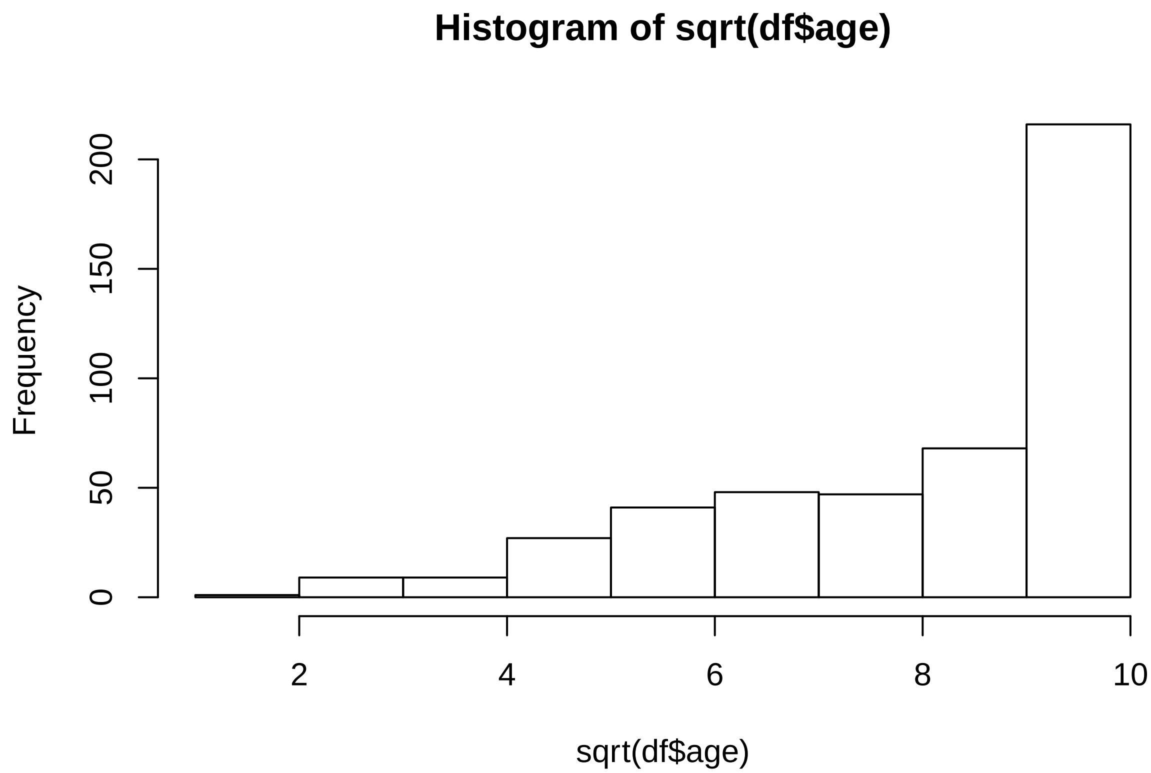
```
##        zn             indus            chas              nox
##  Min.   :  0.00   Min.   : 0.460   Min.   :0.00000   Min.   :0.3890
##  1st Qu.:  0.00   1st Qu.: 5.145   1st Qu.:0.00000   1st Qu.:0.4480
##  Median :  0.00   Median : 9.690   Median :0.00000   Median :0.5380
##  Mean   : 11.58   Mean   :11.105   Mean   :0.07082   Mean   :0.5543
##  3rd Qu.: 16.25   3rd Qu.:18.100   3rd Qu.:0.00000   3rd Qu.:0.6240
##  Max.   :100.00   Max.   :27.740   Max.   :1.00000   Max.   :0.8710
##        rm             age              dis              rad
##  Min.   :3.863   Min.   :  2.90   Min.   : 1.130   Min.   : 1.00
##  1st Qu.:5.887   1st Qu.: 43.88   1st Qu.: 2.101   1st Qu.: 4.00
##  Median :6.210   Median : 77.15   Median : 3.191   Median : 5.00
##  Mean   :6.291   Mean   : 68.37   Mean   : 3.796   Mean   : 9.53
##  3rd Qu.:6.630   3rd Qu.: 94.10   3rd Qu.: 5.215   3rd Qu.:24.00
##  Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.00
##       tax           ptratio          black            lstat
##  Min.   :187.0   Min.   :12.6   Min.   :  0.32   Min.   : 1.730
##  1st Qu.:281.0   1st Qu.:16.9   1st Qu.:375.61   1st Qu.: 7.043
##  Median :334.5   Median :18.9   Median :391.34   Median :11.350
##  Mean   :409.5   Mean   :18.4   Mean   :357.12   Mean   :12.631
##  3rd Qu.:666.0   3rd Qu.:20.2   3rd Qu.:396.24   3rd Qu.:16.930
##  Max.   :711.0   Max.   :22.0   Max.   :396.90   Max.   :37.970
##       medv           target
```

```
##  Min.   : 5.00   Min.   :0.0000
##  1st Qu.:17.02   1st Qu.:0.0000
##  Median :21.20   Median :0.0000
##  Mean   :22.59   Mean   :0.4914
##  3rd Qu.:25.00   3rd Qu.:1.0000
##  Max.   :50.00   Max.   :1.0000
```

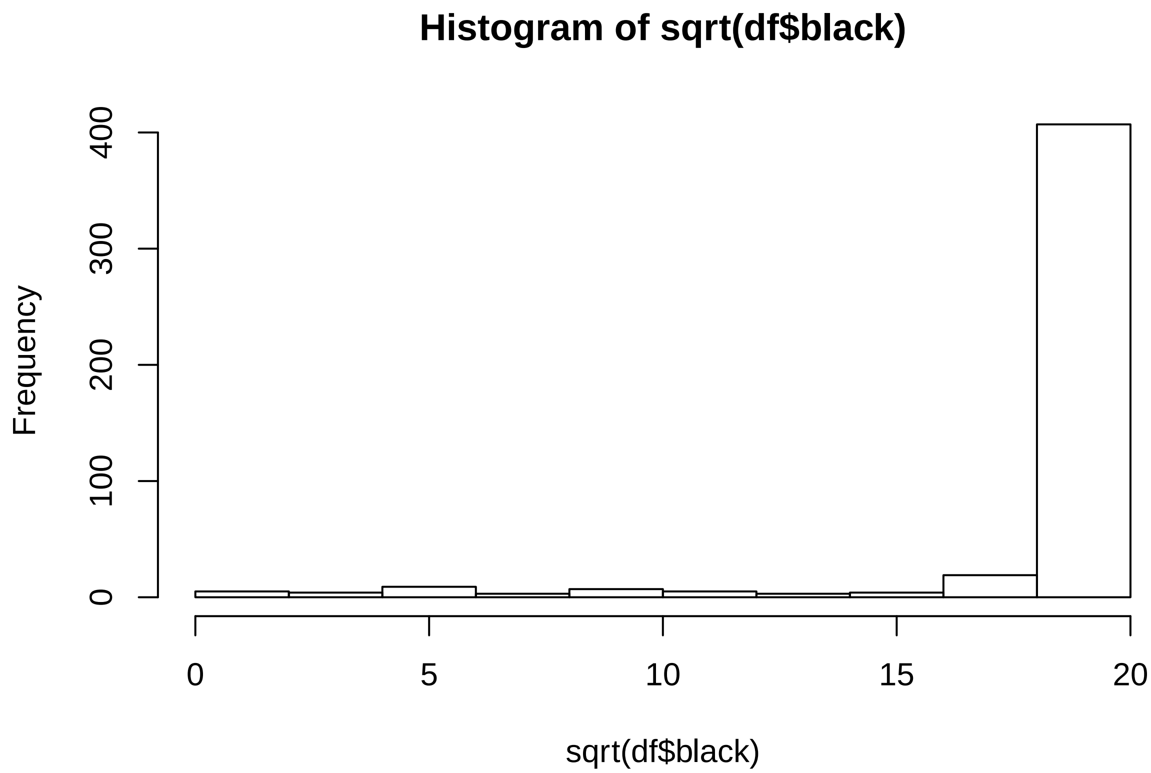
```r
df_transformed$dis = log(df$dis)


df_transformed$chas = factor(df$chas)
#df_transformed$target = factor(df$target)

# Need to be transformed
hist(sqrt(df$black))
```



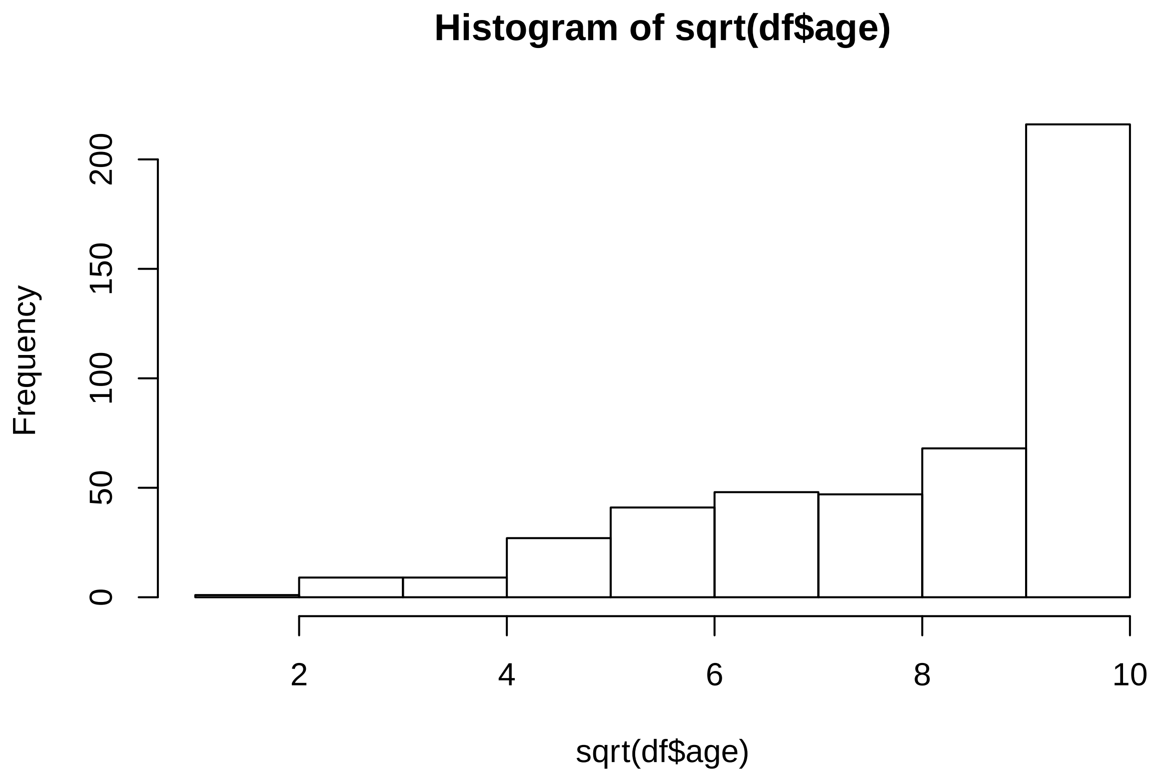

```r
hist(sqrt(df$age))
```




```r
#box_cox_transformation = function(df,var){
#  print(c(df[var]))
#boxcox_detail = boxcox(c(df[var]) ~ .,data = df, lambda= seq(-0.25,.25,length=10))
#lambda = boxcox_detail$x[which.max(boxcox_detail$y)]

#return((df[var] ^ lambda - 1)/lambda)
#}


#box_cox_transformation(df,'dis')



#log(df$black)

hist(sqrt(df$black))
```




```
hist(sqrt(df$age))
```




```
#boxcox(black ~ .,data = df, lambda= seq(-0.25,.25,length=10))
#boxcox(age ~ .,data = df, lambda= seq(-0.25,.25,length=10))

#qqnorm((df$dis ^ lambda - 1)/lambda)
```

For other predictor variables, the transformations did not change the skweness in it. SO we will leave as it is.

1.3 Build Models

As a next step we will build different models and evaluate the metrics.

1.3.1 Model 1 - Stepwise elimination - Logit model

As a first step we will build a `logit` model with backward elimination. In this model, we will remove the predictors which are not statistically significant.

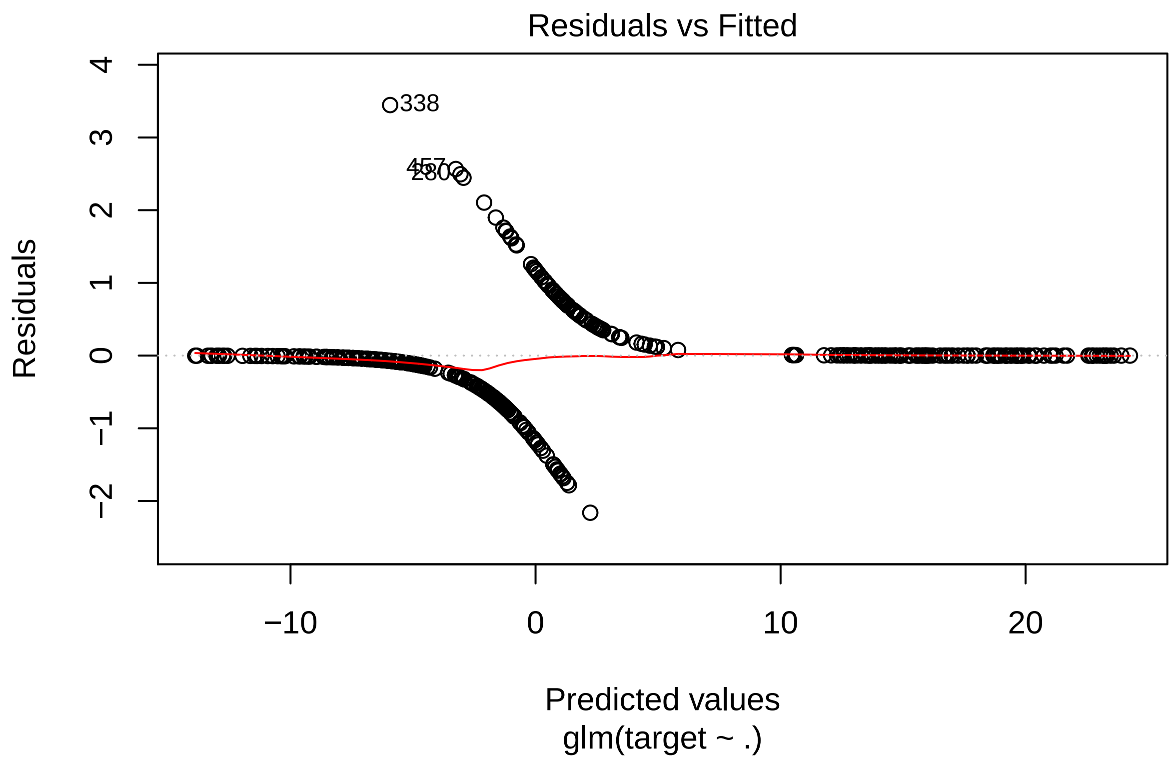
```
# Function for printing all the analysis
analysis <- function(df,model){
  print(summary(model))
  print(paste0("BIC: ",BIC(model)))
  print(paste0("VIF: ",vif(model)))

  plot(model)
  plot(model,which = c(4))
  n = length(df$target)
```

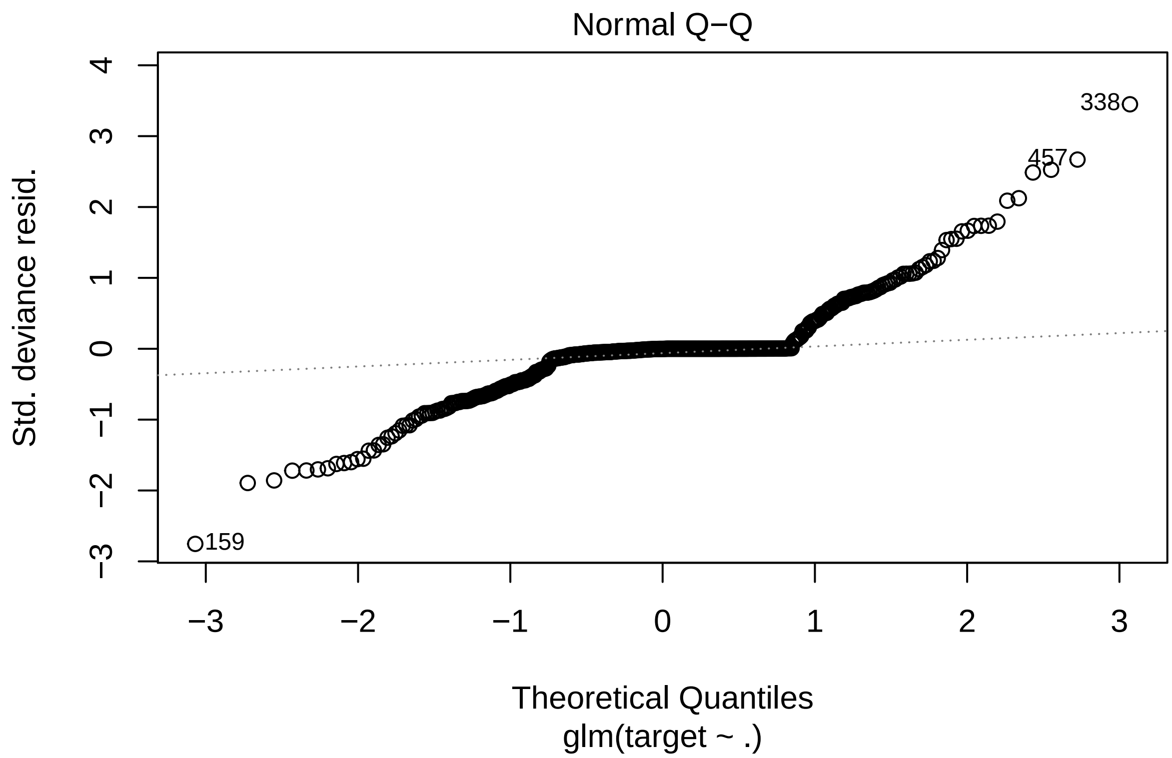
```r
  print(paste0("Naglekerke-pseudo-R2:",(1-exp((model$dev-model$null)/n))/(1-exp(-model$null/n))))
}

# Function for printing confusion matrix
confusion_analysis <- function(df,model){
  # Threshold value is 0.5, positive class is 1

  predicted = if_else(predict(model,df,type='response')>=0.5, 1,0)
  confusionMatrix(data = predicted,
                  reference = df$target,
                  positive = "1")
}

# Function for calculating evaluation metrics
summary_analysis <- function(df,model){
  print(summary(model))
  print(paste0("BIC: ",BIC(model)))
  print(paste0("VIF: ",vif(model)))

  n = length(df$target)
  print(paste0("Naglekerke-pseudo-R2:",(1-exp((model$dev-model$null)/n))/(1-exp(-model$null/n))))
  print("Confusion Matrix:")
  confusion_analysis(df,model)

}

model_11_base = glm(target ~ ., df_transformed, family=binomial(link = 'logit'))
```

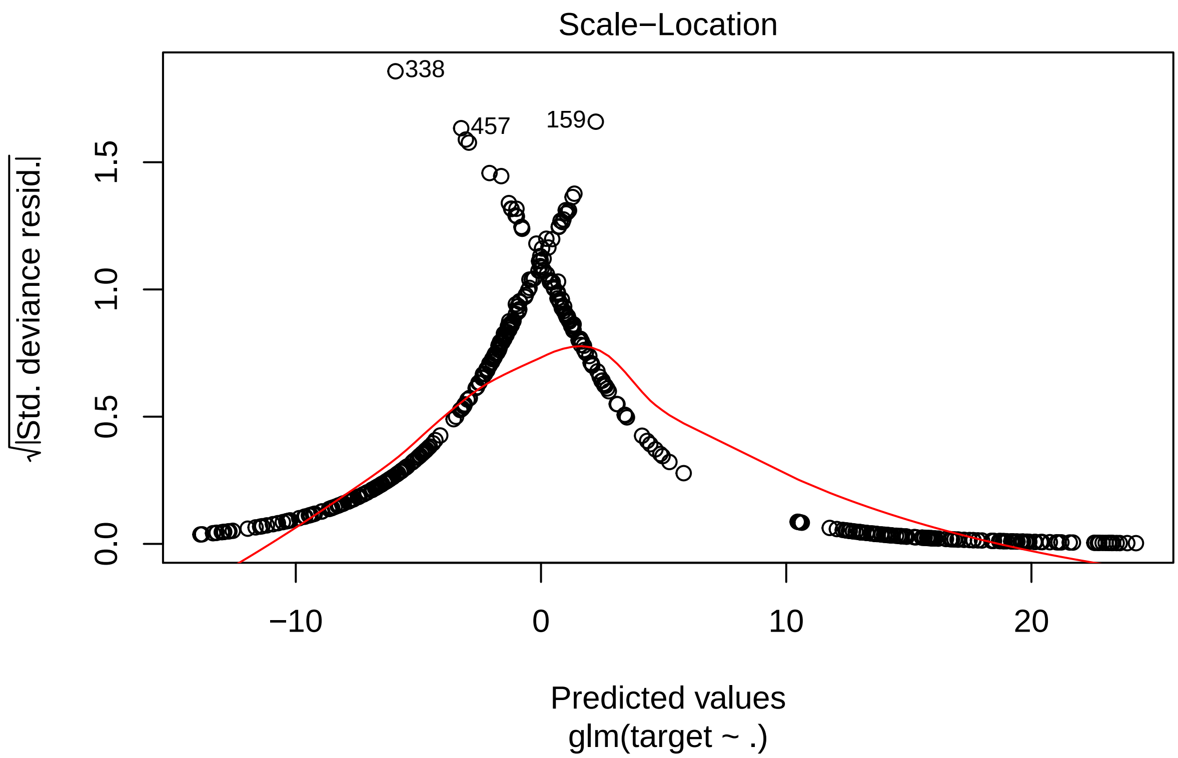
```r
analysis(df_transformed,model_11_base)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##     data = df_transformed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1614  -0.1247  -0.0019   0.0018   3.4458
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -42.300350   7.670450  -5.515 3.49e-08 ***
## zn           -0.046282   0.031448  -1.472 0.141108
## indus        -0.041424   0.049401  -0.839 0.401733
## chas1         0.942922   0.746613   1.263 0.206614
## nox          53.832586   8.257517   6.519 7.07e-11 ***
## rm           -0.873032   0.767264  -1.138 0.255182
## age           0.039262   0.014537   2.701 0.006919 **
## dis           3.812466   0.985131   3.870 0.000109 ***
## rad           0.679686   0.170564   3.985 6.75e-05 ***
## tax          -0.005607   0.003049  -1.839 0.065895 .
## ptratio       0.486181   0.137740   3.530 0.000416 ***
## black        -0.012760   0.006536  -1.952 0.050920 .
## lstat         0.045381   0.054272   0.836 0.403057
```

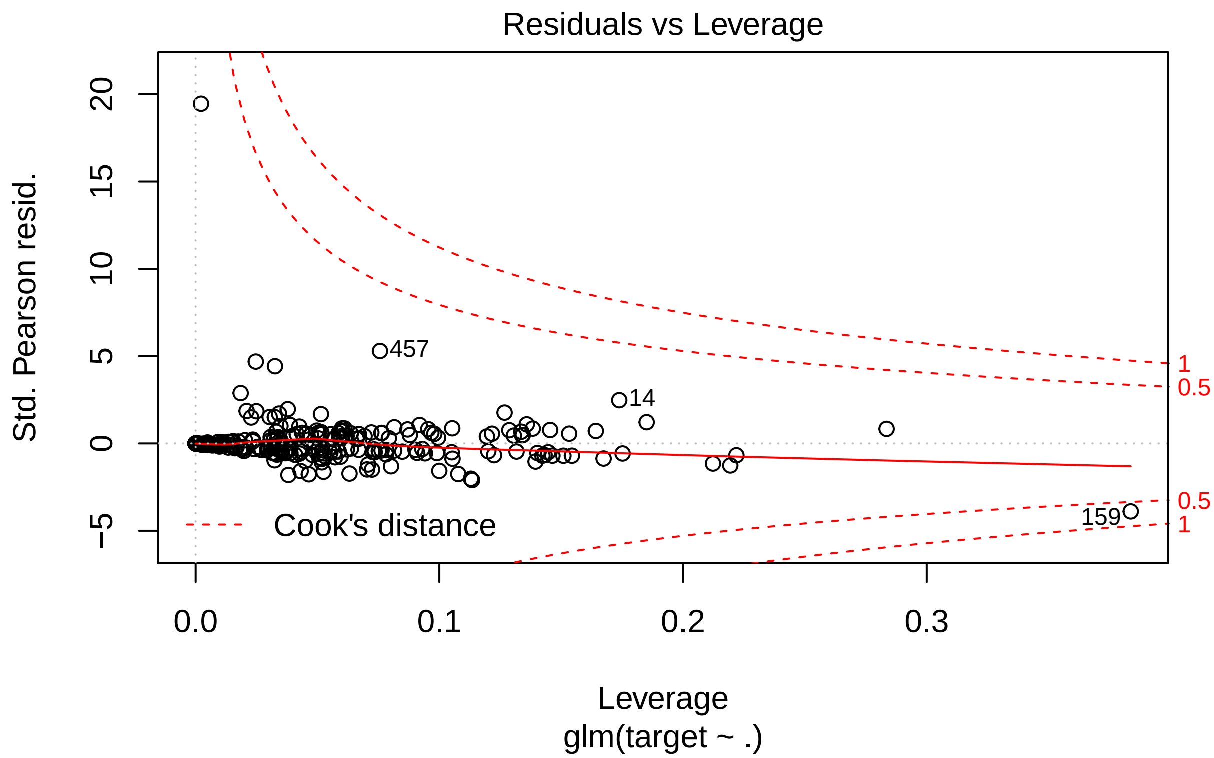
```
## medv          0.233438   0.075060   3.110 0.001871 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 180.63  on 452  degrees of freedom
## AIC: 208.63
##
## Number of Fisher Scoring iterations: 9
##
## [1] "BIC: 266.644653528456"
##  [1] "VIF: 251.051356635014" "VIF: 53.1845764390557"
##  [3] "VIF: 17.0925695791237" "VIF: 431.564256604095"
##  [5] "VIF: 135.999527198491" "VIF: 78.8240831188305"
##  [7] "VIF: 132.191658109089" "VIF: 1020.61111458064"
##  [9] "VIF: 121.834222269219" "VIF: 42.5765453941568"
## [11] "VIF: 165.685993730264" "VIF: 69.0798375377885"
## [13] "VIF: 223.658965927382"
```



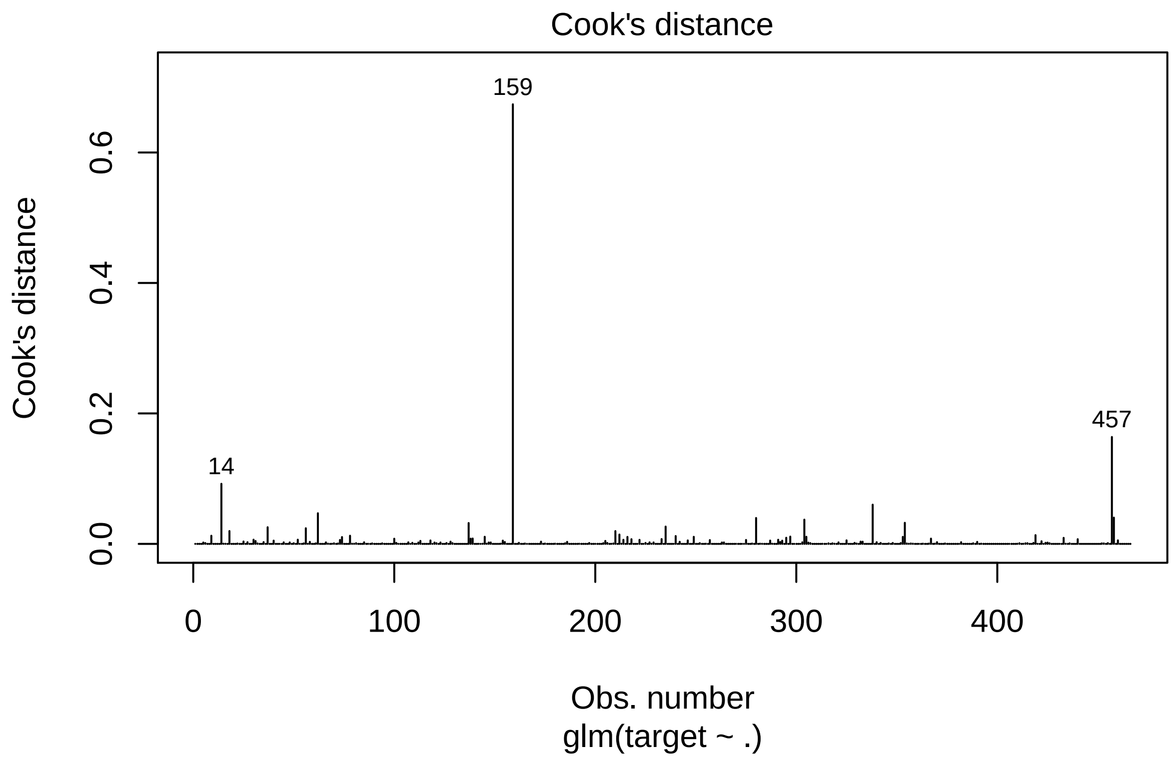

Normal Q−Q
Std. deviance resid.
Theoretical Quantiles
glm(target ~ .)
338
457
159
−3
−2
−1
0
1
2
3
4



Scale−Location
338
457
159
1.5
1.0
0.5
0.0
√|Std. deviance resid.|
−10
0
10
20
Predicted values
glm(target ~ .)



Residuals vs Leverage
Std. Pearson resid.
20
15
10
5
0
−5
0.0
0.1
0.2
0.3
Leverage
glm(target ~ .)
457
14
159
1
0.5
0.5
1
Cook's distance




```
## [1] "Naglekerke-pseudo-R2:0.84211984056351"
```

```
summary_analysis(df_transformed,model_11_base)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##     data = df_transformed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1614  -0.1247  -0.0019   0.0018   3.4458
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -42.300350   7.670450  -5.515 3.49e-08 ***
## zn           -0.046282   0.031448  -1.472 0.141108
## indus        -0.041424   0.049401  -0.839 0.401733
## chas1         0.942922   0.746613   1.263 0.206614
## nox          53.832586   8.257517   6.519 7.07e-11 ***
## rm           -0.873032   0.767264  -1.138 0.255182
## age           0.039262   0.014537   2.701 0.006919 **
## dis           3.812466   0.985131   3.870 0.000109 ***
## rad           0.679686   0.170564   3.985 6.75e-05 ***
## tax          -0.005607   0.003049  -1.839 0.065895 .
## ptratio       0.486181   0.137740   3.530 0.000416 ***
## black        -0.012760   0.006536  -1.952 0.050920 .
```

```
## lstat         0.045381   0.054272   0.836 0.403057
## medv          0.233438   0.075060   3.110 0.001871 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 180.63  on 452  degrees of freedom
## AIC: 208.63
##
## Number of Fisher Scoring iterations: 9
##
## [1] "BIC: 266.644653528456"
##  [1] "VIF: 251.051356635014" "VIF: 53.1845764390557"
##  [3] "VIF: 17.0925695791237" "VIF: 431.564256604095"
##  [5] "VIF: 135.999527198491" "VIF: 78.8240831188305"
##  [7] "VIF: 132.191658109089" "VIF: 1020.61111458064"
##  [9] "VIF: 121.834222269219" "VIF: 42.5765453941568"
## [11] "VIF: 165.685993730264" "VIF: 69.0798375377885"
## [13] "VIF: 223.658965927382"
## [1] "Naglekerke-pseudo-R2:0.84211984056351"
## [1] "Confusion Matrix:"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 220  19
##          1  17 210
##
##                Accuracy : 0.9227
##                  95% CI : (0.8947, 0.9453)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8454
##  Mcnemar's Test P-Value : 0.8676
##
##             Sensitivity : 0.9170
##             Specificity : 0.9283
##          Pos Pred Value : 0.9251
##          Neg Pred Value : 0.9205
##              Prevalence : 0.4914
##          Detection Rate : 0.4506
##    Detection Prevalence : 0.4871
##       Balanced Accuracy : 0.9227
##
##        'Positive' Class : 1
##
```

There are some outliers in the dataset. However, those are not influential points. So we will not remove any data points for now.

But there are some variables which are not statically significant. We will remove those variables one by one and try again.

```r
#Indus is highly correlated with other variables and not statiscially significant. So we will remove th

model_12_base_removed = update(model_11_base,.~.-indus)

#summary_analysis(df_transformed, model_12_base_removed)

# Removing Lstat
model_13_base_removed = update(model_12_base_removed,.~.-lstat)

#summary_analysis(df_transformed, model_13_base_removed)

# Removing rm
model_14_base_removed = update(model_13_base_removed,.~.-rm)

# summary_analysis(df_transformed, model_14_base_removed)

# Converges after removing the above predictor
model_15_base_removed = update(model_14_base_removed,.~.-chas )
```

```r
summary_analysis(df_transformed, model_15_base_removed)
```

```
##
## Call:
## glm(formula = target ~ zn + nox + age + dis + rad + tax + ptratio +
##     black + medv, family = binomial(link = "logit"), data = df_transformed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2083  -0.1513  -0.0022   0.0018   3.4244
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -39.842899   7.338342  -5.429 5.65e-08 ***
## zn           -0.054577   0.029568  -1.846 0.064922 .
## nox          48.429151   7.407408   6.538 6.24e-11 ***
## age           0.035834   0.011378   3.149 0.001636 **
## dis           3.424998   0.898420   3.812 0.000138 ***
## rad           0.707663   0.153639   4.606 4.10e-06 ***
## tax          -0.006718   0.002773  -2.422 0.015423 *
## ptratio       0.375534   0.117734   3.190 0.001424 **
## black        -0.011553   0.006531  -1.769 0.076874 .
## medv          0.136373   0.038277   3.563 0.000367 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 186.44  on 456  degrees of freedom
## AIC: 206.44
##
## Number of Fisher Scoring iterations: 9
##
```

```
## [1] "BIC: 247.880825678868"
## [1] "VIF: 221.933282237831" "VIF: 347.279446218199" "VIF: 48.2857767818802"
## [4] "VIF: 109.944972926948" "VIF: 828.106863816859" "VIF: 100.818637975386"
## [7] "VIF: 31.1070735327358" "VIF: 165.387781205124" "VIF: 58.163772444473"
## [1] "Naglekerke-pseudo-R2:0.835952390075459"
## [1] "Confusion Matrix:"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 222  19
##          1  15 210
##
##                Accuracy : 0.927
##                  95% CI : (0.8995, 0.9489)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.854
##  Mcnemar's Test P-Value : 0.6069
##
##             Sensitivity : 0.9170
##             Specificity : 0.9367
##          Pos Pred Value : 0.9333
##          Neg Pred Value : 0.9212
##              Prevalence : 0.4914
##          Detection Rate : 0.4506
##    Detection Prevalence : 0.4828
##       Balanced Accuracy : 0.9269
##
##        'Positive' Class : 1
##
```

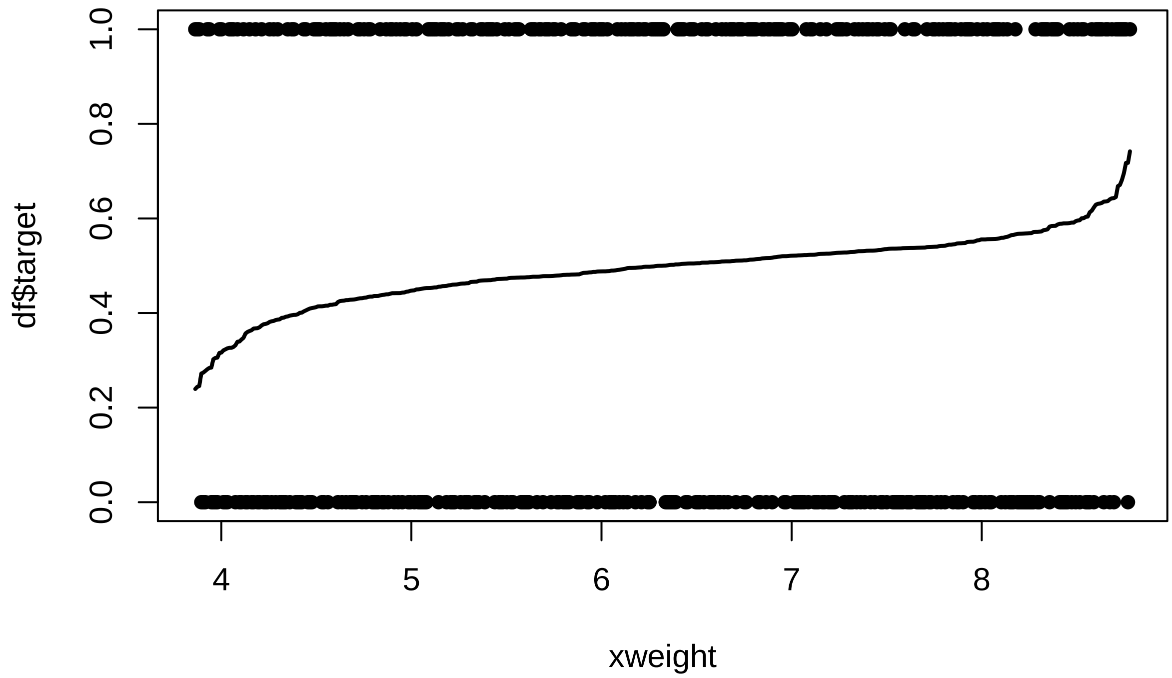
It seems the best model which we can get is the above model with AIC score of ~`206.9`. But still it seems VIF is large for the predictor variables and `Naglekerke-pseudo-R2` is around `0.83`.

1.3.1.1 Individual variable analysis

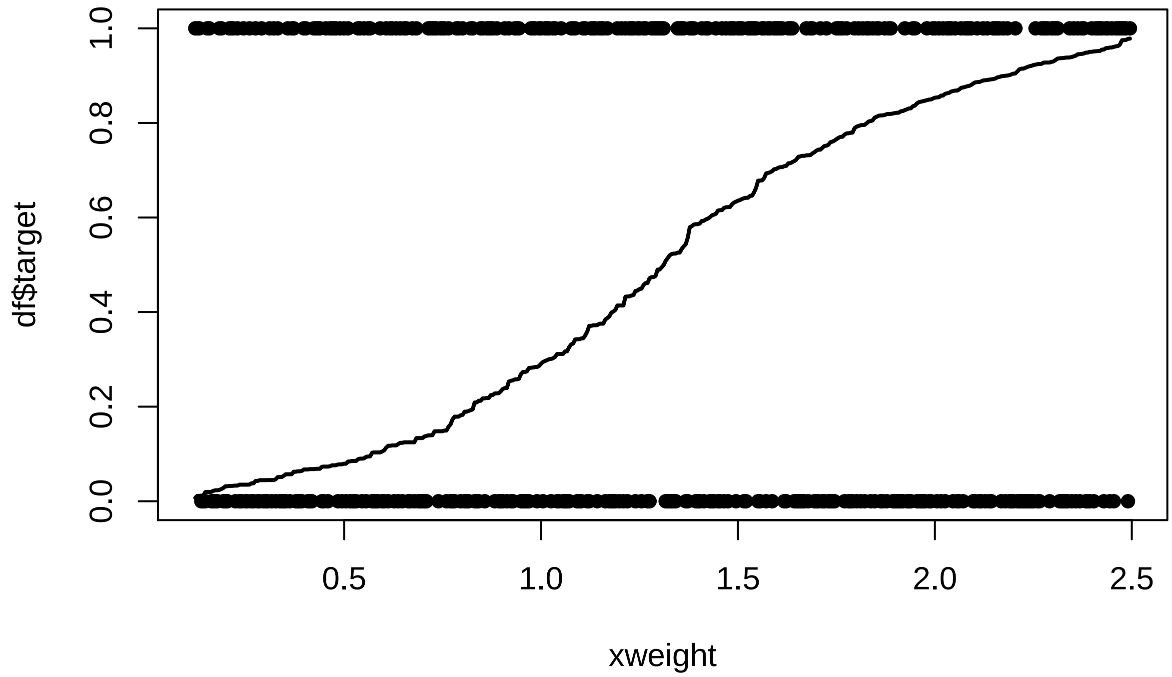
Analyzing individual predictor and the target response will provide the strength of the predictor. Below function will calculate the probablities of individual predictors and then plot it.

```r
variable_analysis = function(df,variable){


temp_mdl = glm(target ~ variable, df, family=binomial(link = 'logit'))


xweight <- seq(range(variable)[1],range(variable)[2],length.out = length(variable))

yweight = predict(temp_mdl,new_data=list(xweight),type = 'response')

plot(xweight, df$target, pch = 16)

lines(xweight,sort(yweight),lwd=2)
```

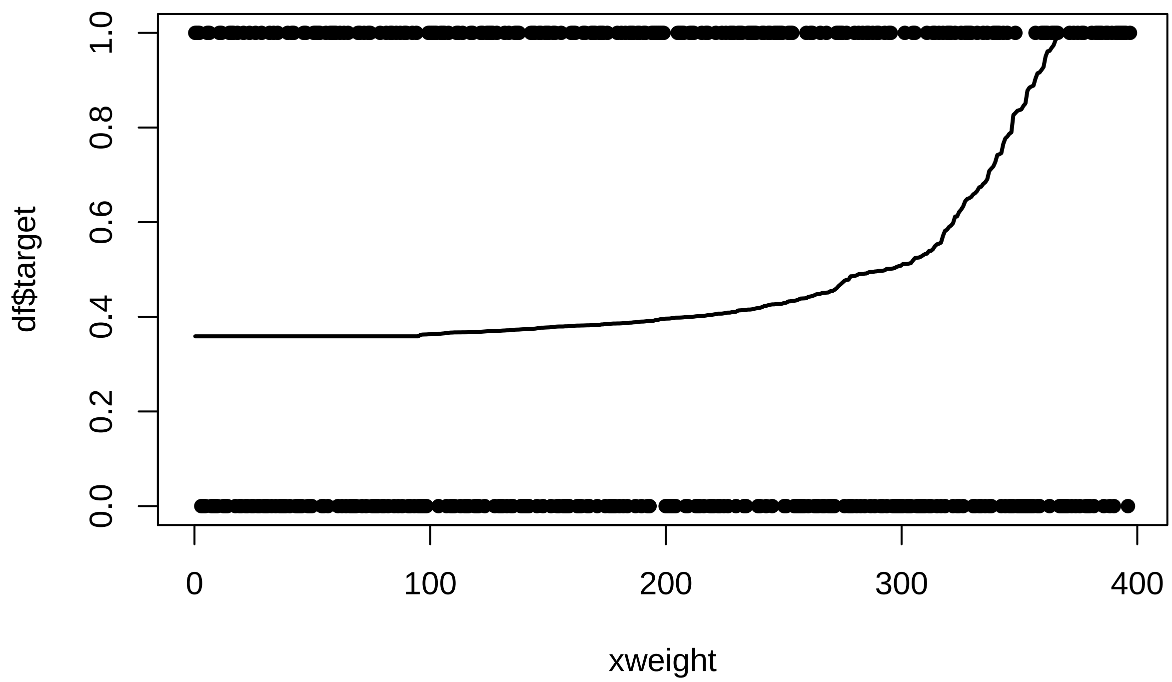
```
}

variable_analysis(df_transformed,df_transformed$rm)
```



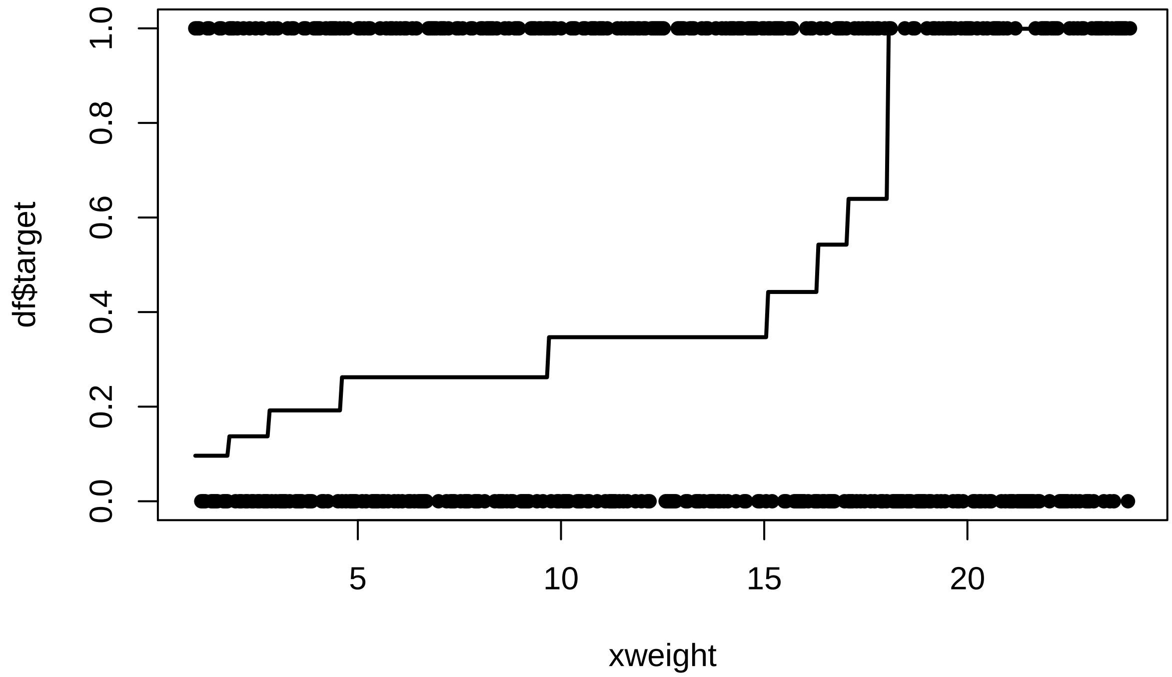

```
variable_analysis(df_transformed,df_transformed$dis)
```



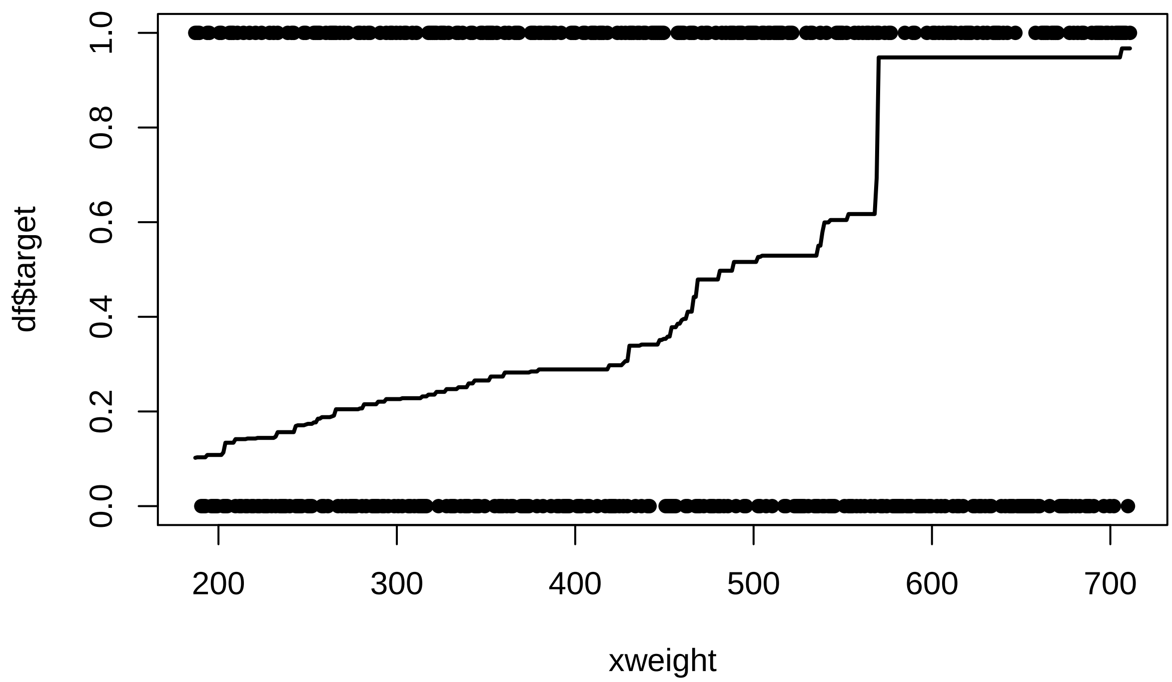

```
variable_analysis(df_transformed,df_transformed$black)
```



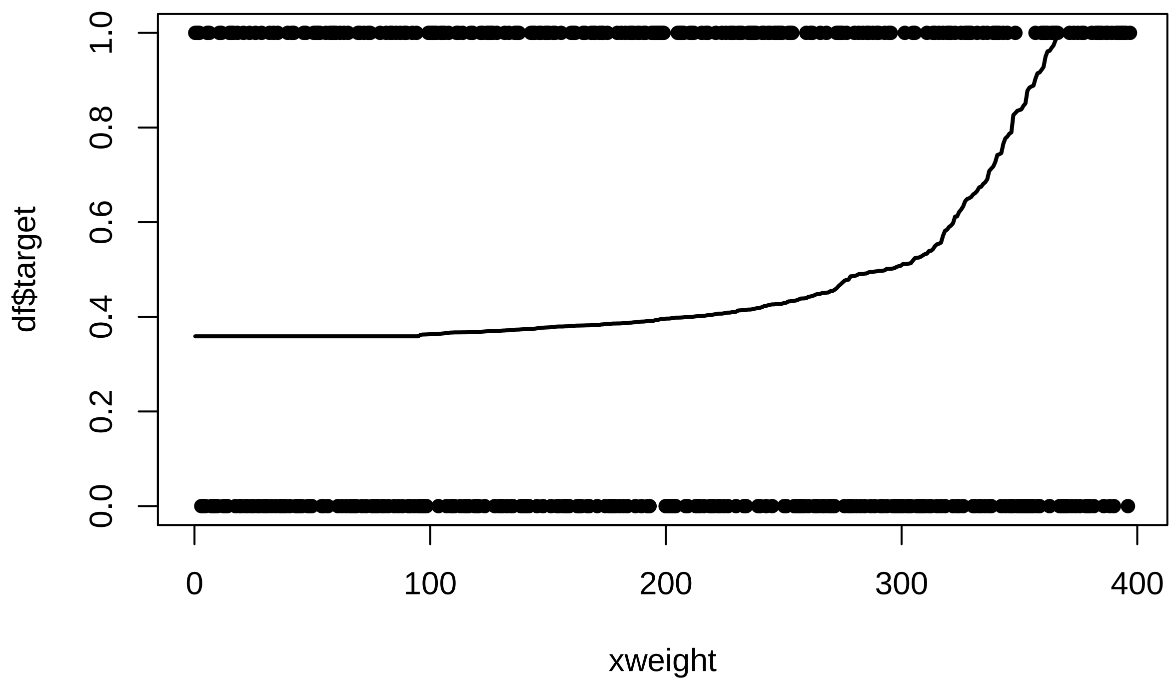

```r
variable_analysis(df_transformed,df_transformed$rad)
```



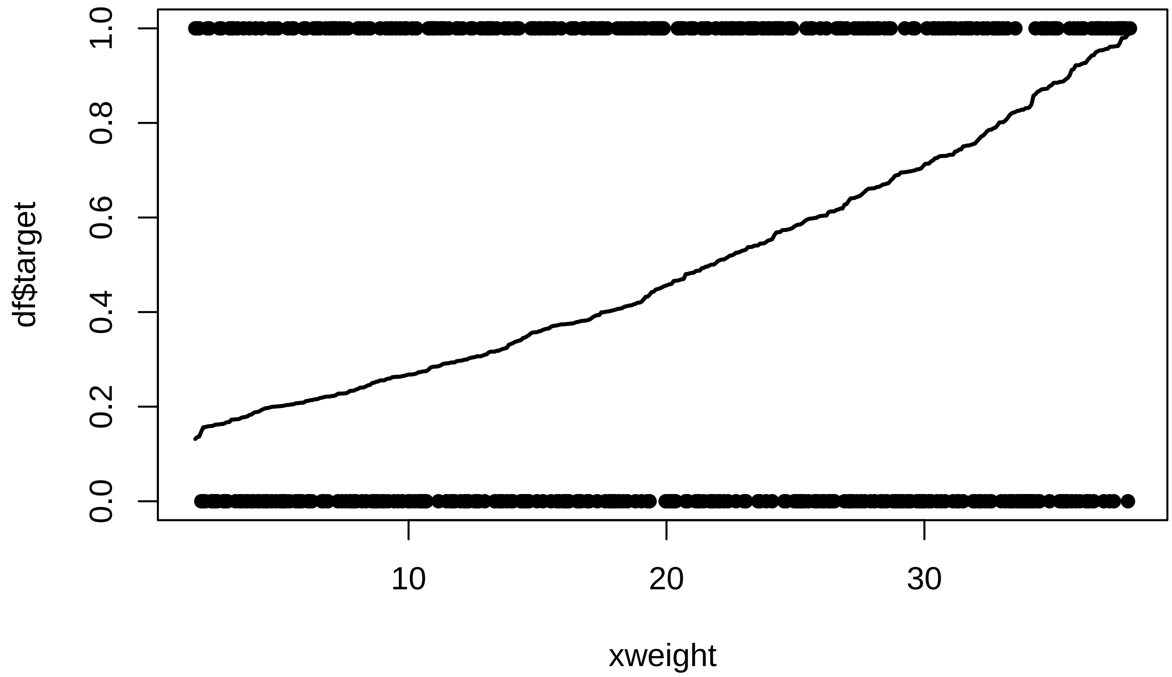

```r
variable_analysis(df_transformed,df_transformed$tax)
```



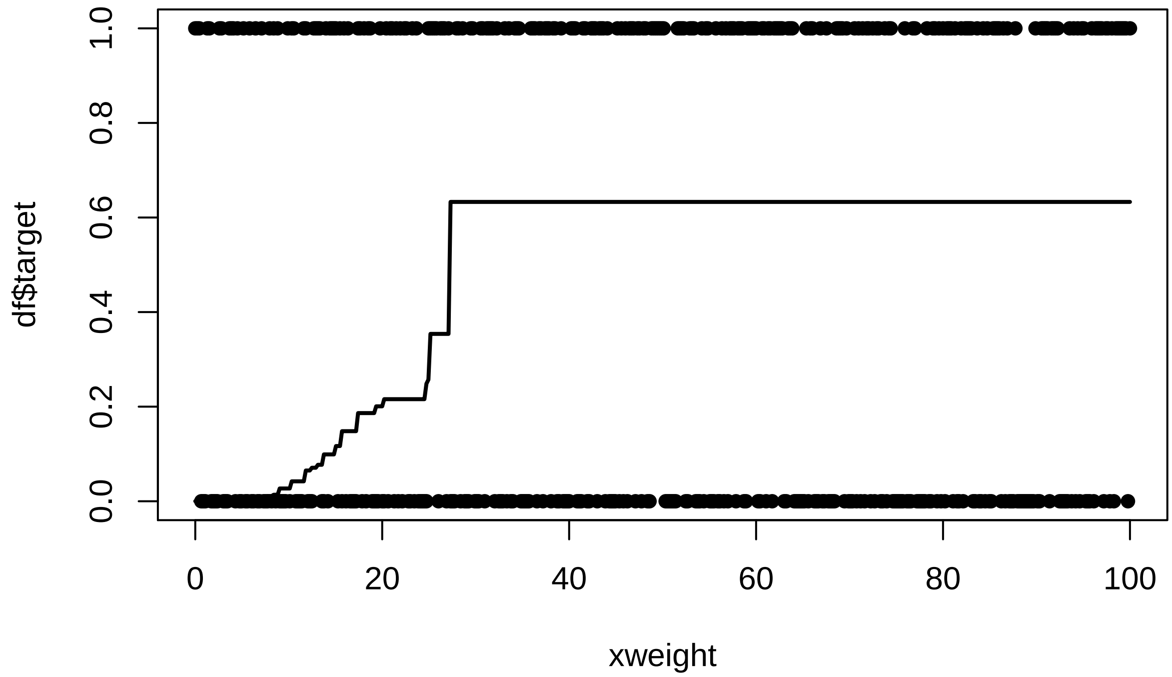

```
variable_analysis(df_transformed,df_transformed$black)
```



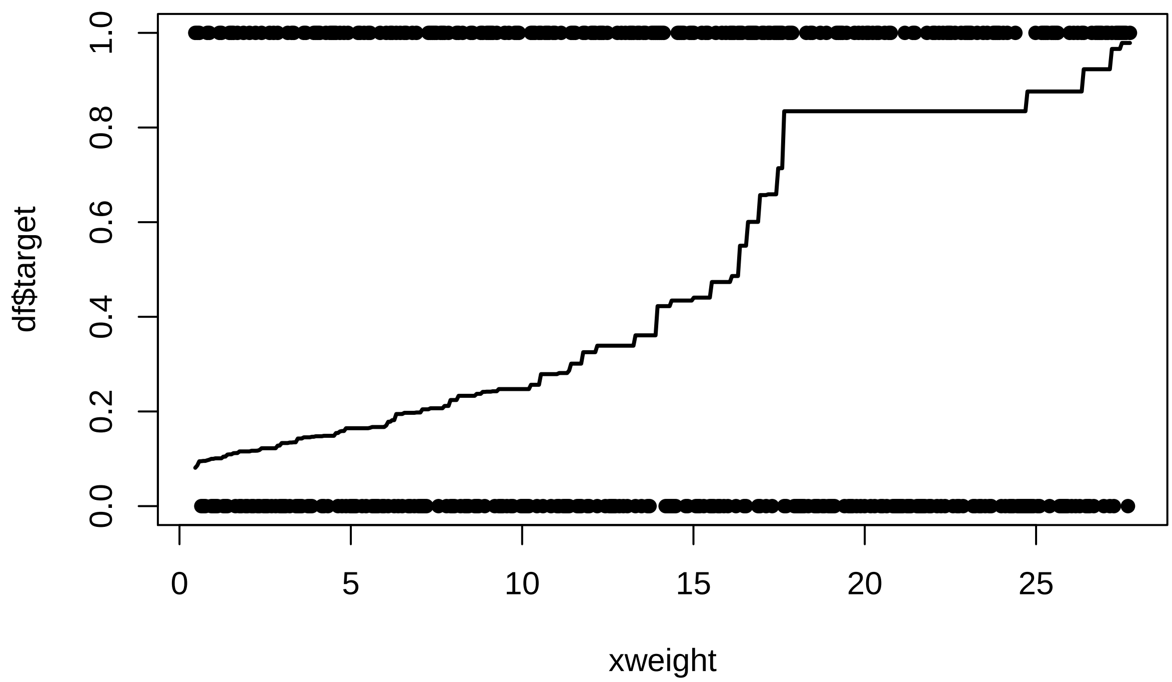

```
variable_analysis(df_transformed,df_transformed$lstat)
```



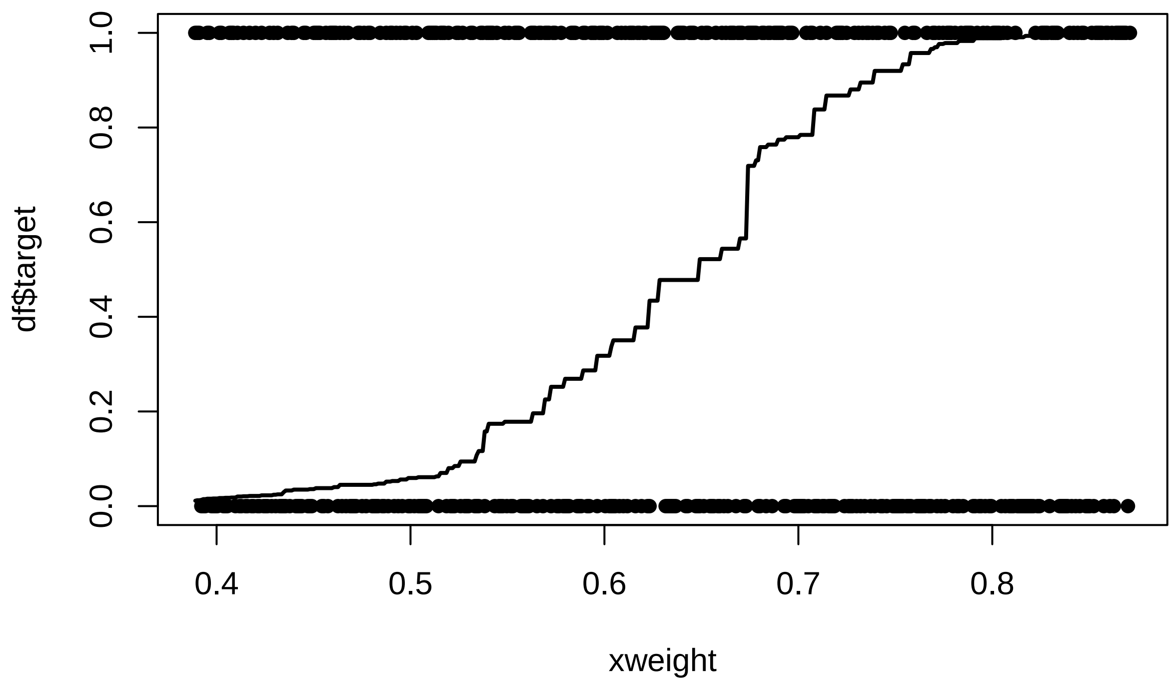

```
# ZN
variable_analysis(df_transformed,df_transformed$zn)
```



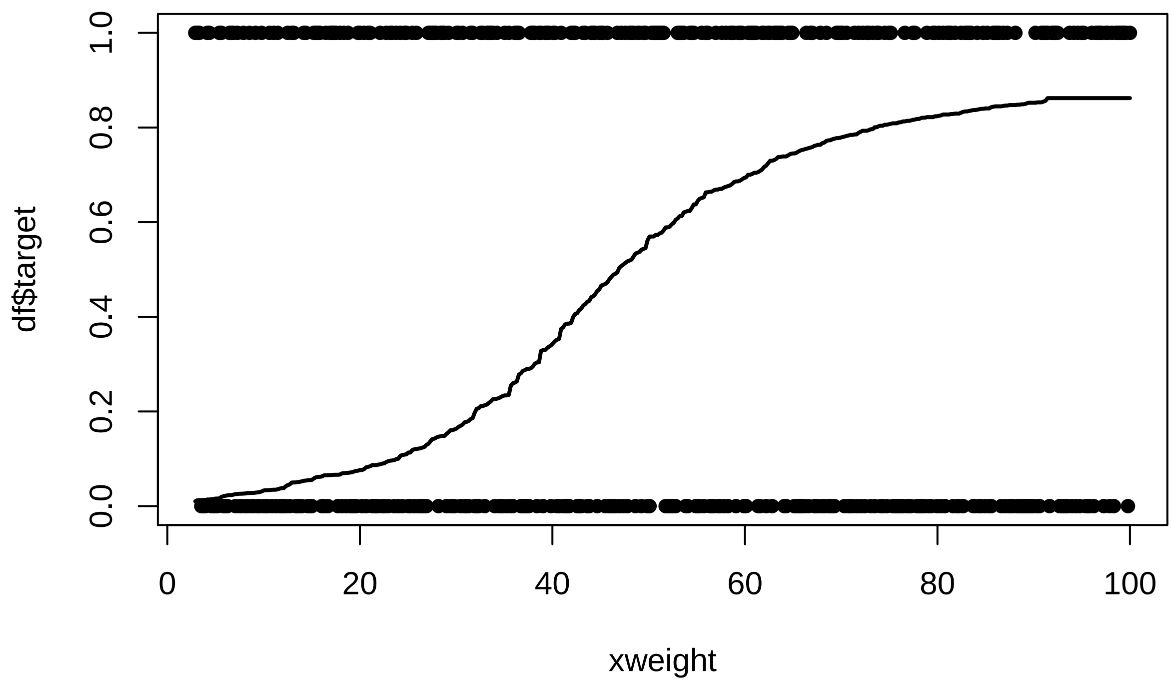

```
# Indus
variable_analysis(df_transformed,df_transformed$indus)
```




```
#Nox
variable_analysis(df_transformed,df_transformed$nox)
```




```
#Age
variable_analysis(df_transformed,df_transformed$age)
```



1.3.2 Model 2 - Stepwise elimination:Probit model

In this model, we are going to use `probit` as our link function using `glm` method. We will run the evaluation metrics on the model. Also remove the predictors which are not statistically significant.

```
model_21_base = glm(target ~ ., df_transformed, family=binomial(link = 'probit'))

model_22_base_removed = update(model_21_base,. ~ .-indus-lstat-zn-chas)
```

```
summary_analysis(df_transformed,update(model_21_base,. ~ .-indus-lstat-zn-chas))
```

```
##
## Call:
## glm(formula = target ~ nox + rm + age + dis + rad + tax + ptratio +
##     black + medv, family = binomial(link = "probit"), data = df_transformed)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1084  -0.1312  -0.0001   0.0000   3.3858
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.402954   3.781409  -5.396 6.83e-08 ***
## nox          28.232988   4.046495   6.977 3.01e-12 ***
## rm           -0.765989   0.379998  -2.016 0.043824 *
## age           0.022593   0.006913   3.268 0.001082 **
```

```
## dis           1.802020   0.493142   3.654 0.000258 ***
## rad           0.404838   0.083989   4.820 1.43e-06 ***
## tax          -0.004181   0.001529  -2.735 0.006233 **
## ptratio       0.276636   0.070773   3.909 9.28e-05 ***
## black        -0.007097   0.003428  -2.071 0.038395 *
## medv          0.130997   0.039630   3.306 0.000948 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 189.87  on 456  degrees of freedom
## AIC: 209.87
##
## Number of Fisher Scoring iterations: 10
##
## [1] "BIC: 251.312164199847"
## [1] "VIF: 103.634563970155" "VIF: 33.3588621314685" "VIF: 17.8242035861392"
## [4] "VIF: 33.1253918975355" "VIF: 247.471745935775" "VIF: 30.6308715939034"
## [7] "VIF: 11.2406723645207" "VIF: 45.5599669849325" "VIF: 62.345886194892"
## [1] "Naglekerke-pseudo-R2:0.832275496142205"
## [1] "Confusion Matrix:"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 217  20
##          1  20 209
##
##                Accuracy : 0.9142
##                  95% CI : (0.8849, 0.938)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8283
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9127
##             Specificity : 0.9156
##          Pos Pred Value : 0.9127
##          Neg Pred Value : 0.9156
##              Prevalence : 0.4914
##          Detection Rate : 0.4485
##    Detection Prevalence : 0.4914
##       Balanced Accuracy : 0.9141
##
##        'Positive' Class : 1
##
```

After removing the statistically insignificant predictors, we can see that the AIC, BIC, and classfication is bad compared to `logit` model.

1.3.3 Model 3 - Automatic Variable selection

Lets try automatic variables selection using step method. It uses `AIC` to select the best parameters.

```
step (model_11_base, trace=F)
```

```
##
## Call:  glm(formula = target ~ zn + nox + rm + age + dis + rad + tax +
##     ptratio + black + medv, family = binomial(link = "logit"),
##     data = df_transformed)
##
## Coefficients:
## (Intercept)           zn          nox           rm          age
##  -39.328293    -0.049571    50.993946    -1.194859     0.046076
##         dis          rad          tax      ptratio        black
##    3.854724     0.761433    -0.006575     0.466897    -0.011917
##        medv
##    0.241430
##
## Degrees of Freedom: 465 Total (i.e. Null);  455 Residual
## Null Deviance:       645.9
## Residual Deviance: 183.5     AIC: 205.5
```

Using automatic selection methods, the best AIC we can get is ~205.9 which is better than manual stepwise selection.

Now lets try to develop the model using AIC and BIC metrics using `bestglm` package.

```
# Best model using AIC
bestglm(df_transformed,IC='AIC')
```

```
## Note: binary categorical variables converted to 0-1 so 'leaps' could be used.

## AIC
## BICq equivalent for q in (0.878203863021405, 0.903175001615091)
## Best Model:
##                Estimate    Std. Error   t value     Pr(>|t|)
## (Intercept) -1.412836094 0.2249300576 -6.281224 7.790512e-10
## nox          1.956694224 0.2157623073  9.068749 3.504319e-18
## age          0.003531713 0.0007664319  4.607993 5.272540e-06
## rad          0.017106647 0.0023402175  7.309854 1.193318e-12
## ptratio      0.012716341 0.0086324347  1.473089 1.414111e-01
## medv         0.008021190 0.0019934004  4.023873 6.692468e-05
```

```
# Best model using BIC
bestglm(df_transformed,IC='BIC')
```

```
## Note: binary categorical variables converted to 0-1 so 'leaps' could be used.

## BIC
## BICq equivalent for q in (0.0184033929221794, 0.878203863021405)
## Best Model:
##                Estimate    Std. Error    t value     Pr(>|t|)
## (Intercept) -1.118239456 0.1030832442 -10.847927 1.424136e-24
## nox          1.853194962 0.2042609148   9.072685 3.376310e-18
## age          0.003720475 0.0007566023   4.917345 1.222144e-06
## rad          0.018598826 0.0021123035   8.804997 2.659972e-17
## medv         0.006675859 0.0017741326   3.762886 1.896311e-04
```

bestglm package converts the target variable as an regression variable and then performs the predition. This might not be the best approach to create a binary logistic regression.

1.3.4 Model 4 - Bayesian Logistic Regression

In this model, we will run Bayesian type logistic regression. Bayesian model calculates the prior and posterior probability using Markov Chain Monte Carlo(MCMC) method.

rstanarm package provides functions to run Bayesian type models.

```
#Reference: https://www.kaggle.com/avehtari/bayesian-logistic-regression-with-rstanarm

df_transformed$target=factor(df_transformed$target)

t_prior =student_t(df=14, location = 0, scale = 2.5)

post1 <- stan_glm(target ~ . ,data =df_transformed,family=binomial(link='logit'),
                  prior = t_prior, prior_intercept = t_prior, seed =1)
```

```
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 8.263 seconds (Warm-up)
##                7.055 seconds (Sampling)
##                15.318 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
## Rejecting initial value:
##   Log probability evaluates to log(0), i.e. negative infinity.
##   Stan can't start sampling from this initial value.
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
```

```
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 7.931 seconds (Warm-up)
##                7.04 seconds (Sampling)
##                14.971 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 8.351 seconds (Warm-up)
##                7.269 seconds (Sampling)
##                15.62 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 7.996 seconds (Warm-up)
##                7.019 seconds (Sampling)
##                15.015 seconds (Total)
```

```
model_41_bayesian = post1
```

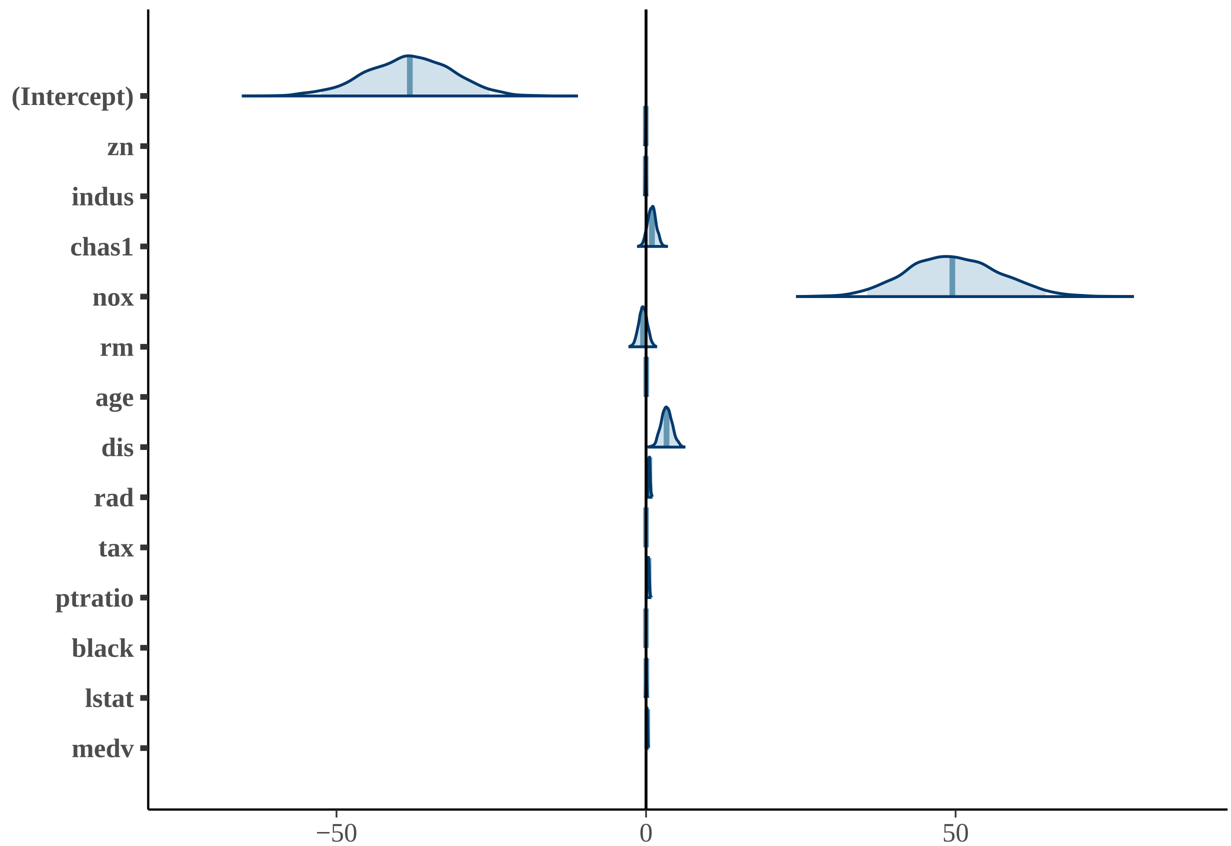
Below is the summary of the bayesian model. It runs for various iterations and the provides coefficients.

```
summary(post1)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      target ~ .
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 466
##  predictors:   14
##
## Estimates:
##                 mean   sd     2.5%   25%    50%    75%    97.5%
## (Intercept)    -38.3    6.9  -52.5  -43.0  -38.2  -33.4  -25.3
## zn              -0.1    0.0   -0.1   -0.1    0.0    0.0    0.0
## indus           -0.1    0.0   -0.2   -0.1   -0.1    0.0    0.0
## chas1            0.9    0.7   -0.4    0.5    1.0    1.4    2.3
## nox             49.7    7.5   35.7   44.5   49.5   54.6   64.6
## rm              -0.5    0.7   -1.8   -1.0   -0.5    0.0    0.9
## age              0.0    0.0    0.0    0.0    0.0    0.0    0.1
## dis              3.3    0.9    1.7    2.7    3.3    3.9    5.1
## rad              0.5    0.1    0.3    0.4    0.5    0.6    0.8
## tax              0.0    0.0    0.0    0.0    0.0    0.0    0.0
## ptratio          0.4    0.1    0.2    0.3    0.4    0.5    0.7
## black            0.0    0.0    0.0    0.0    0.0    0.0    0.0
## lstat            0.0    0.1   -0.1    0.0    0.0    0.1    0.1
## medv             0.2    0.1    0.1    0.1    0.2    0.2    0.3
## mean_PPD         0.5    0.0    0.5    0.5    0.5    0.5    0.5
## log-posterior -116.7    2.7 -122.7 -118.3 -116.4 -114.8 -112.5
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.1  1.0  4000
## zn            0.0  1.0  4000
```

```
## indus         0.0  1.0  4000
## chas1         0.0  1.0  4000
## nox           0.1  1.0  2954
## rm            0.0  1.0  2440
## age           0.0  1.0  2421
## dis           0.0  1.0  2807
## rad           0.0  1.0  3174
## tax           0.0  1.0  4000
## ptratio       0.0  1.0  2580
## black         0.0  1.0  4000
## lstat         0.0  1.0  4000
## medv          0.0  1.0  2354
## mean_PPD      0.0  1.0  3965
## log-posterior 0.1  1.0  1663
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```r
pplot<-plot(post1, "areas", prob = 0.95, prob_outer = 1)
pplot+ geom_vline(xintercept = 0)
```



Coefficients of predictor variables and its confidence intervals.

```r
round(coef(post1), 2)
```

```
## (Intercept)          zn       indus       chas1         nox          rm
##      -38.15       -0.05       -0.05        0.96       49.47       -0.48
##         age         dis         rad         tax     ptratio       black
##        0.03        3.31        0.53        0.00        0.40       -0.01
```

```
##       lstat        medv
##        0.05        0.19
```

```r
round(posterior_interval(post1, prob = 0.9), 2)
```

```
##                 5%    95%
## (Intercept) -49.93 -27.23
## zn           -0.10  -0.01
## indus        -0.14   0.02
## chas1        -0.17   2.11
## nox          37.77  62.18
## rm           -1.64   0.65
## age           0.01   0.06
## dis           1.88   4.82
## rad           0.33   0.77
## tax          -0.01   0.00
## ptratio       0.19   0.61
## black        -0.02   0.00
## lstat        -0.04   0.13
## medv          0.09   0.30
```

```r
# Predicted probabilities
linpred <- posterior_linpred(post1)
preds <- posterior_linpred(post1, transform=TRUE)
pred <- colMeans(preds)
pr <- as.integer(pred >= 0.5)
```

Evaluation metrics of Bayesian models.

```r
confusionMatrix(data = pr,
                reference = df$target,
                positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 221  20
##          1  16 209
##
##                Accuracy : 0.9227
##                  95% CI : (0.8947, 0.9453)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8454
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9127
##             Specificity : 0.9325
##          Pos Pred Value : 0.9289
##          Neg Pred Value : 0.9170
##              Prevalence : 0.4914
##          Detection Rate : 0.4485
##    Detection Prevalence : 0.4828
##       Balanced Accuracy : 0.9226
```

```
##
##        'Positive' Class : 1
##
```

1.3.5 Model 5 - Scaled Basyesian/logit approach

In this model, we will evaluate if the scaling makes any difference in our model. We will scale the predictor variables and comeup with a solution.

1.3.5.1 Logit approach

In the previous model, we have not scaled the data. In this model, we will to scale the predictors and remove the outliers.

```
##  [1] "zn"      "indus"   "chas"    "nox"     "rm"      "age"     "dis"
##  [8] "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"    "target"
```

```
##
## Call:
## glm(formula = target ~ zn + nox + age + dis + rad + tax + ptratio +
##     black + medv, family = binomial(link = "logit"), data = df_scale)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2083  -0.1513  -0.0022   0.0018   3.4244
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.7444     0.7181   3.822 0.000132 ***
## zn           -1.2752     0.6909  -1.846 0.064922 .
## nox           5.6501     0.8642   6.538 6.24e-11 ***
## age           1.0149     0.3222   3.149 0.001636 **
## dis           1.8537     0.4863   3.812 0.000138 ***
## rad           6.1467     1.3345   4.606 4.10e-06 ***
## tax          -1.1279     0.4656  -2.422 0.015423 *
## ptratio       0.8250     0.2586   3.190 0.001424 **
## black        -1.0551     0.5964  -1.769 0.076874 .
## medv          1.2600     0.3537   3.563 0.000367 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 186.44  on 456  degrees of freedom
## AIC: 206.44
##
## Number of Fisher Scoring iterations: 9
##
## [1] "BIC: 247.880825678868"
## [1] "VIF: 221.933282237832" "VIF: 347.279446218245" "VIF: 48.2857767818799"
## [4] "VIF: 109.944972926947" "VIF: 828.106863816864" "VIF: 100.818637975386"
## [7] "VIF: 31.107073532732"  "VIF: 165.387781205109" "VIF: 58.163772444474"
## [1] "Naglekerke-pseudo-R2:0.835952390075459"
```

```
## [1] "Confusion Matrix:"

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 222  19
##          1  15 210
##
##                Accuracy : 0.927
##                  95% CI : (0.8995, 0.9489)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.854
##  Mcnemar's Test P-Value : 0.6069
##
##             Sensitivity : 0.9170
##             Specificity : 0.9367
##          Pos Pred Value : 0.9333
##          Neg Pred Value : 0.9212
##              Prevalence : 0.4914
##          Detection Rate : 0.4506
##    Detection Prevalence : 0.4828
##       Balanced Accuracy : 0.9269
##
##        'Positive' Class : 1
##
```

It seems scaling the predictor variables did improve the model but it is very little.

1.3.5.2 Bayesian approach

```r
df_transformed$target=factor(df_scale$target)

t_prior =student_t(df=14, location = 0, scale = 2.5)

model_61_bayseian_scaled <- stan_glm(target ~ . ,data =df_scale,family=binomial(link='logit'),
                  prior = t_prior, prior_intercept = t_prior, seed =1)
```

```
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).
##
## Gradient evaluation took 0.001 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
```

```
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 7.998 seconds (Warm-up)
##                6.909 seconds (Sampling)
##                14.907 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 2).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 8.031 seconds (Warm-up)
##                6.9 seconds (Sampling)
##                14.931 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 3).
##
## Gradient evaluation took 0.001 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
```

```
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 7.754 seconds (Warm-up)
##                7.156 seconds (Sampling)
##                14.91 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 4).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 7.538 seconds (Warm-up)
##                7.443 seconds (Sampling)
##                14.981 seconds (Total)
```

```r
linpred <- posterior_linpred(model_61_bayseian_scaled)
preds <- posterior_linpred(model_61_bayseian_scaled, transform=TRUE)
pred <- colMeans(preds)
pr <- as.integer(pred >= 0.5)

confusionMatrix(data = pr,
                reference = df_scale$target,
                positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 221  20
##          1  16 209
##
##                Accuracy : 0.9227
##                  95% CI : (0.8947, 0.9453)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8454
##  Mcnemar's Test P-Value : 0.6171
##
```

```
##          Sensitivity : 0.9127
##          Specificity : 0.9325
##       Pos Pred Value : 0.9289
##       Neg Pred Value : 0.9170
##           Prevalence : 0.4914
##       Detection Rate : 0.4485
## Detection Prevalence : 0.4828
##    Balanced Accuracy : 0.9226
##
##     'Positive' Class : 1
##
```

Seems scaling has no effect on Bayesian logistic regression. So we will use it as a final model and evaluate the test dataset.

1.4 Select Models

We have evaluated all the models using our training data. Now we need to select the best model for further predications. We will select three different models from the above analysis. `logit`, `probit` and `Bayesian model`.

1.4.1 Predictions

```r
pr_model1 = if_else(predict(model_15_base_removed,newdata = df_test,type='response')>=0.5, 1,0)
pr_model2 = if_else(predict(model_22_base_removed,newdata =df_test,type='response')>=0.5, 1,0)

pr_model4 = c(posterior_predict(model_41_bayesian,df_test)[4000,])
```

Below is the predictions of all the three models.

```r
data.frame(logit_model = pr_model1,probit_model = pr_model2,bayes=pr_model4)
```

```
##    logit_model probit_model bayes
## 1            0            0     0
## 2            1            1     0
## 3            1            1     0
## 4            1            1     1
## 5            0            0     0
## 6            0            1     0
## 7            0            1     0
## 8            0            0     0
## 9            0            0     0
## 10           0            0     0
## 11           0            0     0
## 12           0            0     0
## 13           1            1     1
## 14           1            1     1
## 15           1            1     1
## 16           0            0     0
## 17           0            0     1
## 18           1            1     1
## 19           0            0     0
## 20           0            0     0
```

```
## 21            0             0      0
## 22            0             0      0
## 23            0             0      0
## 24            0             0      0
## 25            0             0      0
## 26            1             1      0
## 27            0             0      0
## 28            1             1      1
## 29            1             1      1
## 30            1             1      1
## 31            1             1      1
## 32            1             1      1
## 33            1             1      1
## 34            1             1      1
## 35            1             1      1
## 36            1             1      1
## 37            1             1      1
## 38            1             1      1
## 39            1             1      1
## 40            0             0      0
```

Metric	Logit	Probit	Bayes	Automatic
AIC	206.44	212.34	-	205.5
BIC	247.88	270.36	-	-
Naglekerke-pseudo-R2	0.835	0.835	-	-
Accuracy	0.927	0.920	0.922	-

Based on the above metrics, automatic selection is performing best. After than logit model is performing well in this dataset. Bayesian method works well, but we need to calcuate other metrics for proper validation. As automatic variable selection cannot be explained, we will choose logit model for this dataset.

```
rocCurve <- roc(response = df_transformed$target,
                predictor = predict(model_15_base_removed,newdata = df_transformed,type='response'),
                  levels = c(1,0))
```

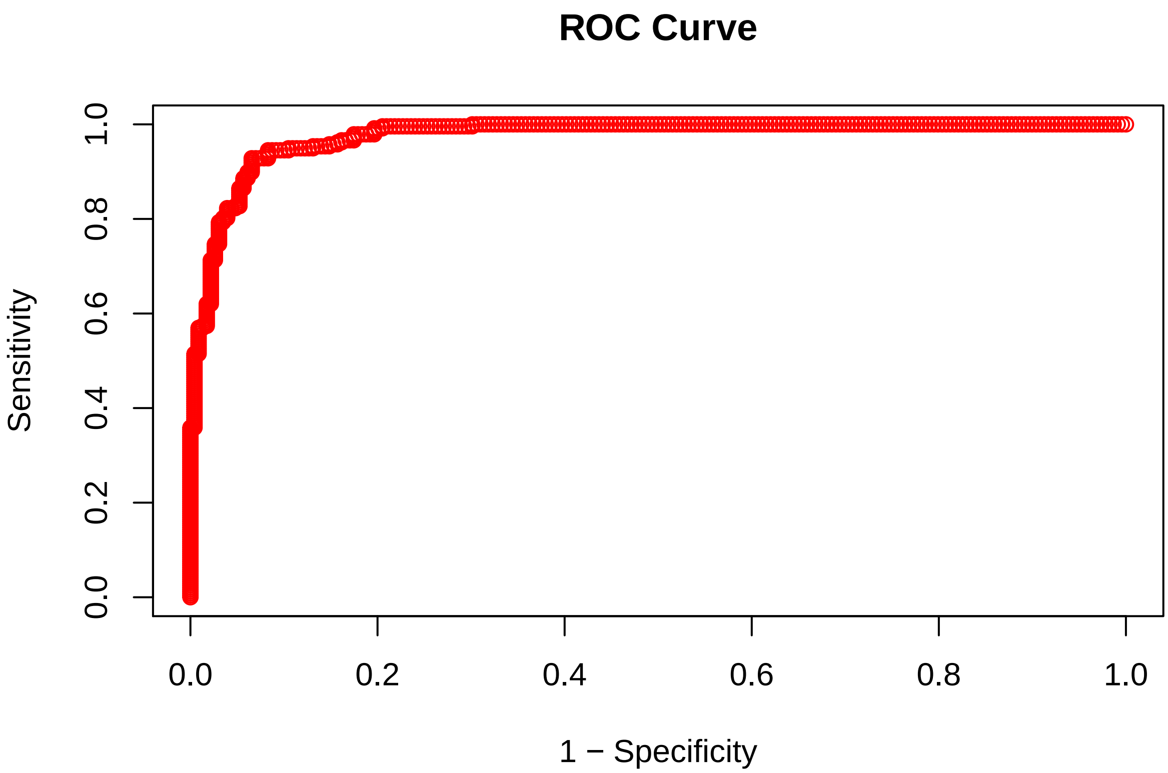
```
print('ROC Curve:')
```

```
## [1] "ROC Curve:"
```

```
auc(rocCurve)
```

```
## Area under the curve: 0.9752
```

```
plot(y = rocCurve$sensitivities, ylab = "Sensitivity", x = 1 - rocCurve$specificities, xlab = "1 - Spec
main = "ROC Curve", col = "red")
```


ROC Curve
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
1 − Specificity