Data 621 - HW3

Habib Khan

10/11/2020

Contents

Overview

In this homework assignment, you will explore, analyze and model a data set containing information on crime for various neighborhoods of a major city. Each record has a response variable indicating whether or not the crime rate is above the median crime rate (1) or not(0). Your objective is to build a binary logistic regression model on the training data set to predict whether the neighborhood will be at risk for high crime levels. You will provide classifications and probabilities for the evaluation data set using your binary logistic regression model. You can only use the variables given to you (or variables that you derive from the variables provided). Below is a short description of the variables of interest in the data set:

• 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:	$1000(B_k - 0.63)^2$ where B_k 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 - Data Exploration

In this section, we are going to explore the data to see the data type and data structure, We will also check the correlation among the variables and most importantly to see if there are missing values in the data but first let's read both training and test datasets.

```
# Reading multiple libraries together with pacman's p_load function
#Sys.setenv(JAVA_HOME= "C:\\Program Files\\Java\\jre1.8.0_261")
pacman::p_load(naniar, tidyverse, knitr, kableExtra, skimr, Amelia, reshape2, stats, corrplot,caret, e1
```

```
training <- read.csv('C:/Users/hukha/Desktop/MS - Data Science/Data 621/HW3/crime-training-data_modifie
training2 <- training # for melting and boxploting
evaluation <- read.csv('C:/Users/hukha/Desktop/MS - Data Science/Data 621/HW3/crime-evaluation-data_mod

training %>% head() %>% kable() %>% kableExtra::kable_styling()
```

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

```
# Converting into factor variables
var <- c("chas","target")
training[,var] <- lapply(training[,var], as.factor)
evaluation$chas <- as.factor(evaluation$chas)
```

Both training and evaluation datasets have been read using read.csv function and above table is a sample of training dataset. Before jumping in model building, we have to explore what kind of dataset we have, is it usable at this moment or we have to do data preparation which is the case usually. According to variable's documentation, target and chas seem to be factor variables that's why we converted them into factors to be able to explore the data in a right way. Now let's explore the data structure using skim function from skimr package. This is an efficient function which not only produces the statistics summary but also builds histogram for each numberic variable, show number of missing values and quantiles. This gives a bird eye view of the training dataset. We double checked the total number of missing values with colSums and missmap functions as well to verify if we have any missing value and seems like we do not have missing values.

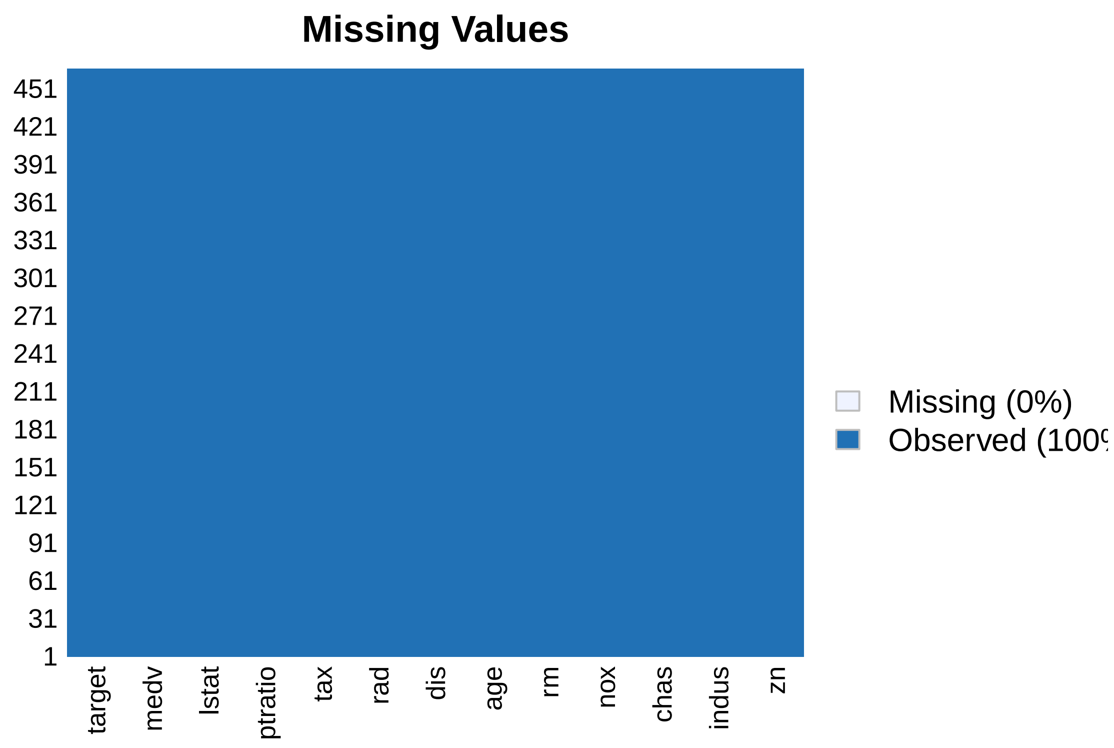
```
Sys.setlocale("LC_CTYPE", "Chinese") # Don't remove this line otherwise histogram will not show up corr
```

```
## [1] "Chinese (Simplified)_China.936"
```

```
skim(training)  %>% kable() %>% kable_styling(full_width=FALSE) # Using skimr package
```

skim_type	skim_variable	n_missing	complete_rate	factor.ordered	factor.n_unique	factor.top_counts	nume
factor	chas	0	1	FALSE	2	0: 433, 1: 33	
factor	target	0	1	FALSE	2	0: 237, 1: 229	
numeric	zn	0	1	NA	NA	NA	11
numeric	indus	0	1	NA	NA	NA	11
numeric	nox	0	1	NA	NA	NA	0
numeric	rm	0	1	NA	NA	NA	6
numeric	age	0	1	NA	NA	NA	68
numeric	dis	0	1	NA	NA	NA	3
numeric	rad	0	1	NA	NA	NA	9
numeric	tax	0	1	NA	NA	NA	409
numeric	ptratio	0	1	NA	NA	NA	18
numeric	lstat	0	1	NA	NA	NA	12
numeric	medv	0	1	NA	NA	NA	22

```
missmap(training, main="Missing Values") # using Amelia package
```

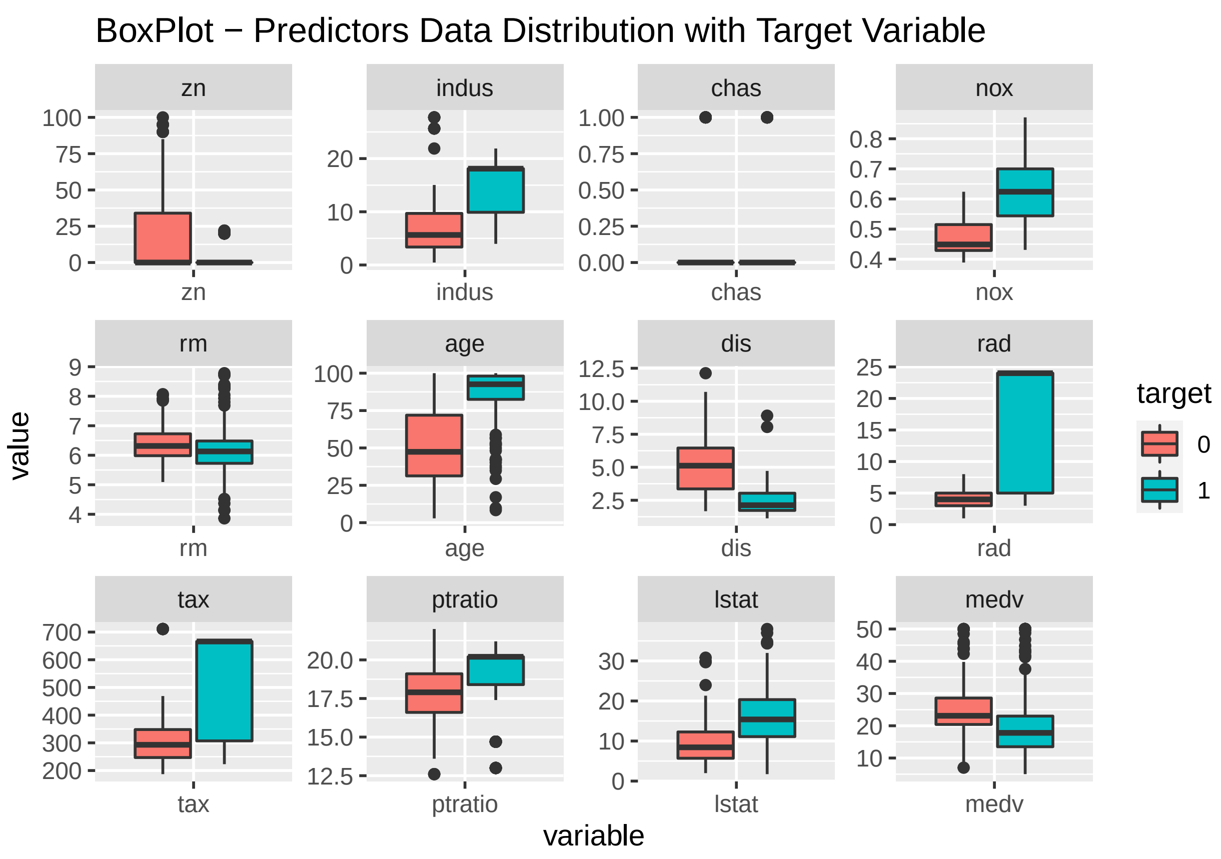


```
colSums(is.na(training))
```

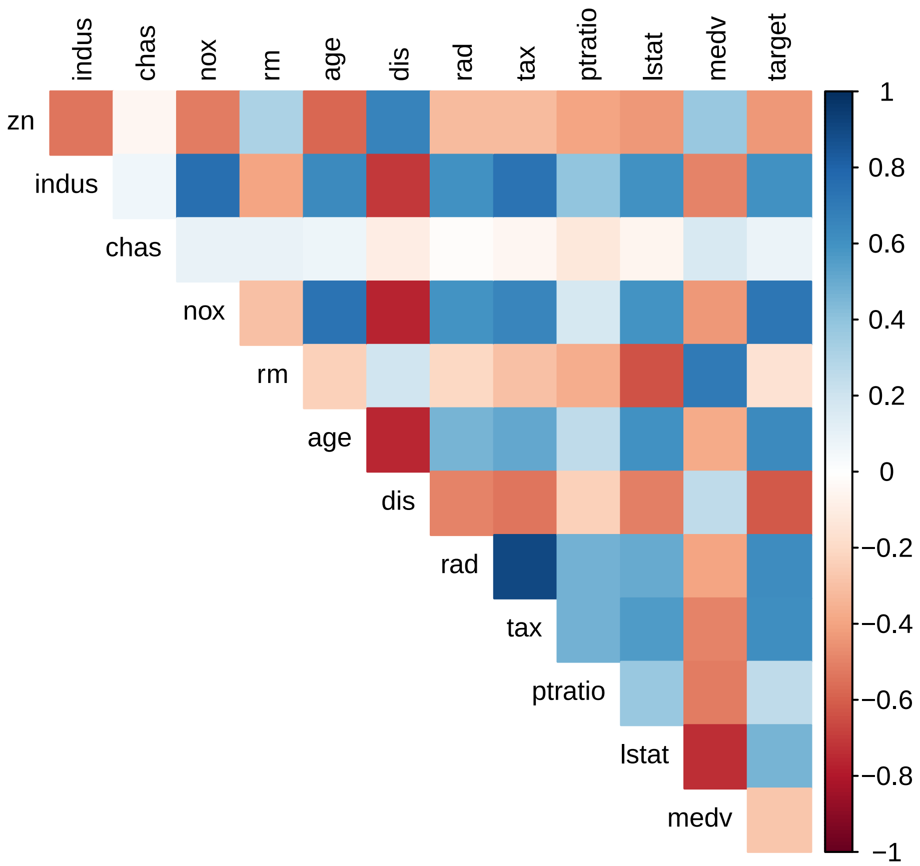
```
##      zn   indus    chas     nox      rm     age     dis     rad     tax ptratio
##       0       0       0       0       0       0       0       0       0       0
##   lstat    medv  target
##       0       0       0
```

It's important to check the correlation of predictors among themselves and especially with the target variable. Now we are going to create boxplot and correlations to explore more.

```
# Boxplot to see distributions with target variable
melt(training2, id.vars='target') %>% mutate(target = as.factor(target)) %>%
  ggplot(., aes(x=variable, y=value))+geom_boxplot(aes(fill=target))+facet_wrap(~variable, dir='h',scal
```



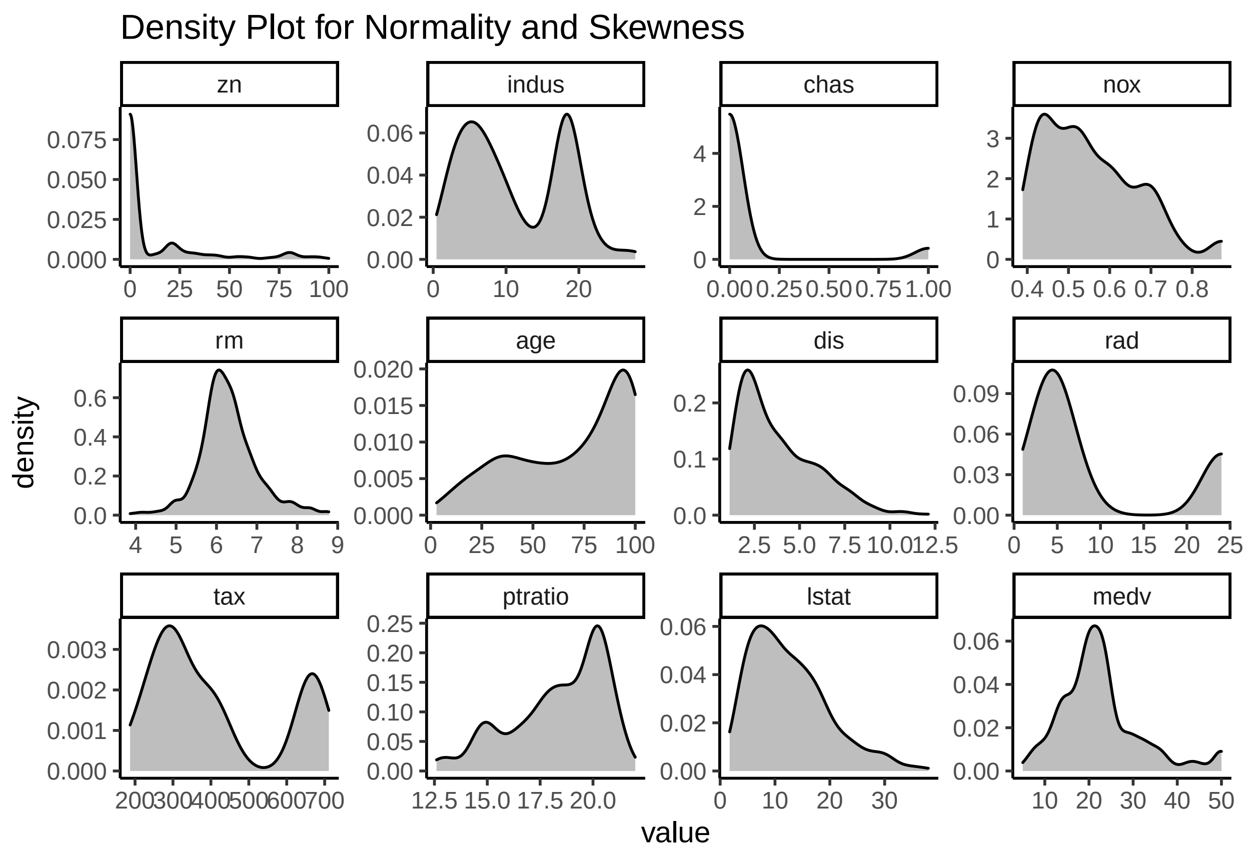
```r
# Correlation matrix among variables
training2 %>%
  cor(., use = "complete.obs") %>%
  corrplot(., method = "color", type = "upper", tl.col = "black", tl.cex=.8, diag = FALSE)
```



```
# Correlation table
correlation <- training2 %>%
  cor(., use = "complete.obs") %>%
  as.data.frame() %>%
  rownames_to_column()%>%
  gather(Variable, Correlation, -rowname)

correlation %>%
  filter(Variable == "target") %>%
     arrange(desc(Correlation)) %>%
  kable() %>%
  kable_styling(full_width = FALSE)
```

rowname	Variable	Correlation
target	target	1.0000000
nox	target	0.7261062
age	target	0.6301062
rad	target	0.6281049
tax	target	0.6111133
indus	target	0.6048507
lstat	target	0.4691270
ptratio	target	0.2508489
chas	target	0.0800419
rm	target	-0.1525533
medv	target	-0.2705507
zn	target	-0.4316818
dis	target	-0.6186731

```r
# Density plot to check normality
melt(training2, id.vars='target') %>% mutate(target = as.factor(target)) %>%
  ggplot(., aes(x=value))+geom_density(fill='gray')+facet_wrap(~variable, scales='free')+
  labs(title="Density Plot for Normality and Skewness") +
  theme_classic()
```



```r
# Skewness and outliers
sapply(training2, skewness, function(x) skewness(x))
```

```
##          zn       indus        chas         nox          rm         age
##  2.17681518  0.28854503  3.33548988  0.74632807  0.47932023 -0.57770755
##         dis         rad         tax     ptratio       lstat        medv
##  0.99889262  1.01027875  0.65931363 -0.75426808  0.90558642  1.07669198
##      target
##  0.03422935
```

According to correlation plot and matrix, nox, age, rad, tax and indus are positively correlated with target. lstat, ptratio and chas have weak correlations with target variable. dis have good negative correlation followed by zn medv and rm which do not seem to have strong correlation with target varible. It is also important to discuss that the predictors are mostly not normally distributed with skewed on both sides other than rm. We might have to go through data preparation step to make the data usable for model building. In last, we are going to split the training set into two subsets i.e. train and test which will be helpful for accuracy checking before predicting the target variable in evaluation dataset.

2 - Data Preparation

There are few issues in the data. Although there are no missing values but most of the variables seem to be skewed and not normally distributed. Outliers have also seen in some variables which need to be checked either the values make sense or not. If not then some of the high outliers will be replaced with either median or knn function. We will also use log transformation before going ahead and we will explore if log transformation is enough.

Data Splitting

Let's start with splitting the training dataset into train and test datasets which will help us to check the accuracy of our model in the next step. Keep in mind that evaluation data does not have target variable because we will be predicting the values but how are we going to make sure the predicted values are accurate? The answer is data splitting i.e. training data. We will be using createDataPartition function from caret library to split the datasets with 70 percent train and 30 percent test data. It's important to check out the skewness as well as previously in boxplot we saw that there are some outliers but are they problematic?

```r
# Data splitting into train and test datasets out of training2
set.seed(1003)
training_partition <- createDataPartition(training2$target, p=0.7, list = FALSE, times=1)
train2 <- training2[training_partition, ]
test2 <- training2[-training_partition, ]

sapply(training2, skewness, function(x) skewness(x))
```

```
##          zn       indus        chas         nox          rm         age
##  2.17681518  0.28854503  3.33548988  0.74632807  0.47932023 -0.57770755
##         dis         rad         tax     ptratio       lstat        medv
##  0.99889262  1.01027875  0.65931363 -0.75426808  0.90558642  1.07669198
##      target
##  0.03422935
```

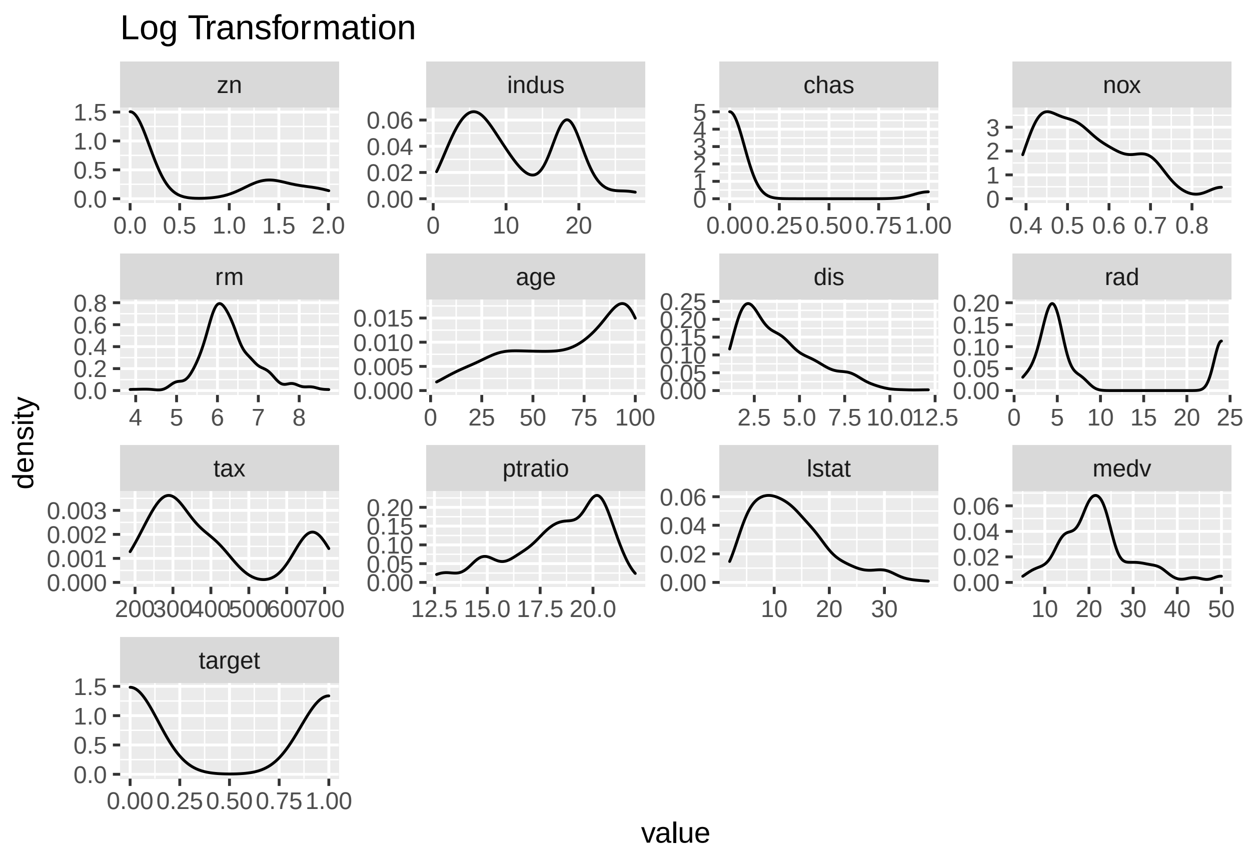
Skewness function from e1071 was used with sapply function to create the skewness of all the variables. If the values are positive then those variables are positively skewed and vice versa. If skewness is 0 then the data is perfectly symmetric but it's very unlikely in the real world scenario. Statisticians suggest anything under +1 and -1 to be in a safe zone. Another statistican has also considered anything under +2 and -2 are fine too. According to the above results, zn and chas are not symmetric and hence we have to use log transformation to make them symmetric but in this case we won't consider transforming chas because it is a categorical data. It will leave us using log transformation only on zn. Let's do this:

log transformation

```
train_log <- train2 # copy of basic model for log transformation
test_log <- test2


train_log$zn <- log10(train_log$zn + 1)
test_log$zn <- log10(test_log$zn + 1)

# Plot and check skewness
sapply(train_log, skewness, function(x) skewness(x))
```

```
##         zn      indus       chas        nox         rm        age        dis
##  1.1954317  0.3708873  3.2567321  0.8108244  0.4843153 -0.5322325  0.9721716
##        rad        tax    ptratio      lstat       medv     target
##  1.1317640  0.7385414 -0.8218609  0.9700911  0.9700927  0.1036391
```

```
ggplot(melt(train_log), aes(x=value))+geom_density()+facet_wrap(~variable, scales='free') + labs(title=
```

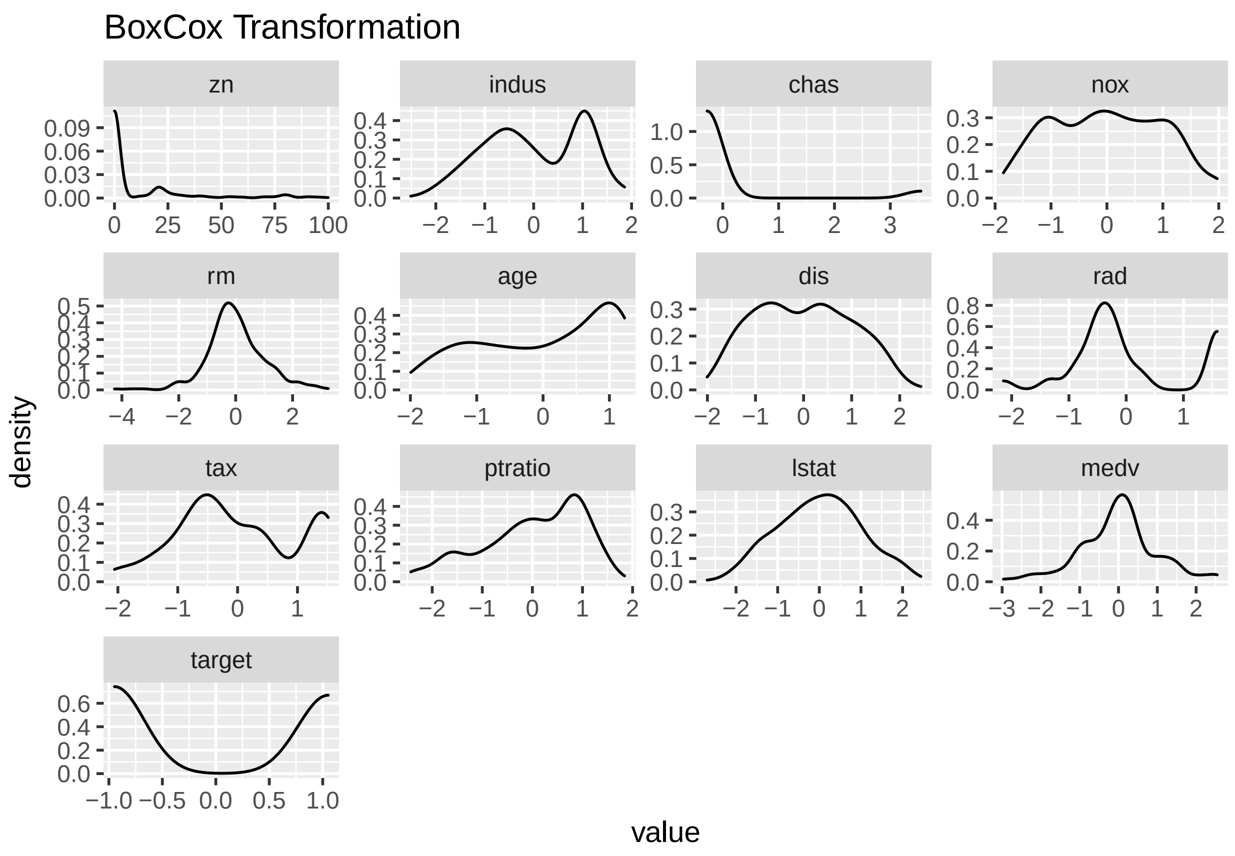


Now it's very close to 1 which is still not ideal but will consider it safer than before for model building. We were planning to remove the extreme outliers but seems like we had only problem with zn which is resolved after transformation that's why we will continue working ahead. Plot has also slightly improved as compared with before.

BoxCox Transformation

Although log transformation has already made the data symmetrical but we will also boxcox transformation to see if using model based on boxcox transformation will give better results in terms of accuracy as compared with log transformed data or not. We will apply boxcox transformation both on train and test datasets.

```
# Copy of train and test
train_boxcox <- train2
test_boxcox <- test2

# Preprocessing
preproc_value <- preProcess(train2[,-1] , c("BoxCox", "center", "scale"))

# Transformation on both train and test datasets
train_boxcox_transformed <- predict(preproc_value, train_boxcox)
test_boxcox_transformed <- predict(preproc_value, test_boxcox)

ggplot(melt(train_boxcox_transformed), aes(x=value))+geom_density()+facet_wrap(~variable, scales='free'
```



```
sapply(train_boxcox_transformed, function(x) skewness(x))
```

```
##              zn          indus            chas             nox             rm             age
##  2.353729370 -0.120195838  3.256732134  0.068418489  0.026658478 -0.366340589
##          dis          rad          tax      ptratio        lstat         medv
##  0.106811243  0.402503618  0.069764777 -0.613098264 -0.002592159 -0.039697233
##       target
##  0.103639097
```

Although the skewness did not improve much for zn but overall it seems to be symmetrical as compared with the results of log transformation. At this point we are not sure which transformation will give better predictions until later. We will keep both transformed datasets and will check them both as seperate models. The data is ready to be used for model building. We are going to use backward elimination model and another model that will be result of cumulative variables that have collinearity. We will discuss about it in detail in the next step.

3 - Build Models

Model 1 - Backward elimination on Log Transformed data

In this model, we are going to use log transformed data and remove the least insignificant variables one-by-one until the model becomes completely significant. Let's dive in and see how this model is doing. We had checked the skewness for the variables and only zn seemed to be skewed for which we adjusted it to log + 1 which not only handled transformation but also handles missing value if exists. After creating a model, we had to remove chas, lstat, rm, indus, ptratio, tax and dis one by one to keep only the significant variables in the model and seems like only age, nox, rad and medv have significant impact on target with adjusted r-square of 0.59. R-square was not initially 0.59 which did not improve at all after eliminating the insignificant variables one by one and it makes sense as they have least to no impact on target variable. There were some collinearity between tax and rad but since tax has been excluded from the model that's why there is no more any multicollinearity among the variables as the values of VIF are less than 10. Important point is here that only zn was adjusted to log but since it was removed that's we can say model is normal and not transformed.

```
# creating model1
model1 <- lm(target ~ nox + age+ rad+ medv, family= binomial, data= train_log)
summ(model1)
```

Observations	327
Dependent variable	target
Type	OLS linear regression

F(4,322)	118.76
R2	0.60
Adj. R2	0.59

```
check_collinearity(model1) %>% kable(caption="Multicollinearity") %>% kable_styling(full_width = FALSE)
```

Model 2 - Backward elimination on BoxCox Transformed data

In this model, we will use the same concept to eliminate the insignificant variables one-by-one that have highest p-value but on boxcox transformed data to see if this model will make any better result or not. The

	Est.	S.E.	t val.	p
(Intercept)	-1.12	0.13	-8.62	0.00
nox	1.81	0.24	7.50	0.00
age	0.00	0.00	3.90	0.00
rad	0.02	0.00	7.57	0.00
medv	0.01	0.00	2.90	0.00

Standard errors: OLS

Table 1: Multicollinearity

Parameter	VIF	SE_factor
nox	2.760468	1.661466
age	2.344391	1.531141
rad	1.623970	1.274351
medv	1.433166	1.197149

model seems slightly better as compared with Model1. R2 is improved from 0.60 to 0.63 and adjusted r-sq is improved from 0.59 to 0.61. Also, dis came to be significant which was insignificant in Model1. We had to remove lstat, rm, indus, chas, ptratio, zn and tax from the model which are consistently insignificant. There is no multicollinearity as the values of VIF are less than 10. Statisticians suggest anything less than 10 is good which shows consistency with our previous model. Overall, the model is significant as per the p-value of F-stat.

```
model2 <- lm(target ~ nox + age + dis + rad +  medv, family= binomial, data= train_boxcox_transformed)
summ(model2)
```

Observations	327
Dependent variable	target
Type	OLS linear regression

F(5,321)	104.74
R2	0.62
Adj. R2	0.61

	Est.	S.E.	t val.	p
(Intercept)	-0.00	0.03	-0.00	1.00
nox	0.65	0.09	7.61	0.00
age	0.20	0.06	3.37	0.00
dis	0.23	0.08	2.93	0.00
rad	0.33	0.04	7.55	0.00
medv	0.12	0.04	2.75	0.01

Standard errors: OLS

```
check_collinearity(model2) %>% kable(caption="Multicollinearity") %>% kable_styling(full_width = FALSE)
```

Table 2: Multicollinearity

Parameter	VIF	SE_factor
nox	6.140573	2.478018
age	3.082054	1.755578
dis	5.293362	2.300731
rad	1.592763	1.262047
medv	1.508302	1.228129

Model 3 - Using Stepwise Regression

Although we have used backward elimination in which we eliminated insignificant variables one by one from the model as discussed before. We can use step() function which is more robust and it is used for stepwise regression. Basically, it eliminates all the insignificant variables one-by-one under the hood and brings the significant variables. This model is used only to verify the result of Model2 using step-wise regression.

```
model3 <- step(model2)
```

```
## Start:  AIC=-305.39
## target ~ nox + age + dis + rad + medv
##
##        Df Sum of Sq    RSS     AIC
## <none>              123.88 -305.39
## - medv  1    2.9275 126.81 -299.75
## - dis   1    3.3095 127.19 -298.77
## - age   1    4.3841 128.27 -296.02
## - rad   1   22.0257 145.91 -253.88
## - nox   1   22.3745 146.26 -253.10
```

```
summ(model3)
```

Observations	327
Dependent variable	target
Type	OLS linear regression

F(5,321)	104.74
R2	0.62
Adj. R2	0.61

	Est.	S.E.	t val.	p
(Intercept)	-0.00	0.03	-0.00	1.00
nox	0.65	0.09	7.61	0.00
age	0.20	0.06	3.37	0.00
dis	0.23	0.08	2.93	0.00
rad	0.33	0.04	7.55	0.00
medv	0.12	0.04	2.75	0.01

Standard errors: OLS

Model 4 - Using glmulti

```r
summary(model4@objects[[1]])
```

```
##
## Call:
## fitfunc(formula = as.formula(x), family = ..1, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7621  -0.2870  -0.0080   0.0057   3.2397
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -36.411735   6.701072  -5.434 5.52e-08 ***
## zn           -0.052983   0.034824  -1.521  0.12815
## indus        -0.069985   0.048870  -1.432  0.15213
## nox          44.712077   8.392153   5.328 9.94e-08 ***
## age           0.032535   0.012388   2.626  0.00863 **
## dis           0.749006   0.262842   2.850  0.00438 **
## rad           0.569547   0.159955   3.561  0.00037 ***
## tax          -0.005851   0.003072  -1.905  0.05683 .
## ptratio       0.268150   0.129009   2.079  0.03766 *
## medv          0.089608   0.039255   2.283  0.02245 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 452.43  on 326  degrees of freedom
## Residual deviance: 152.03  on 317  degrees of freedom
## AIC: 172.03
##
## Number of Fisher Scoring iterations: 8
```

glmulti() function from glmulti package is one of the best automated function which optimizes the best performing model through simulating all possible models under the hood and finds the best performing model. It takes few time to optimize the model though. We will see which model performed best in terms of performance and accuracy in the next section.

4 - Select Models

Model Performance

In this section, we are going to select the best model out of all through using compare_performance and model_performance functions from performance package. It calculates AIC, BIC, R2 & adjusted r-sq, RMSE, BF and Performance_Score. If we take a look at first three models, model1 is doing great as the values of AIC and BIC both are lower in first three models. Even RMSE is lower as compared with model2 and model3. Model4 was calculated through glmulti() package which optimizes the model and gets the best. The value of AIC and BIC is lower than model1 which is good and r2 has also increased to 0.71 but RMSE has increased slightly. Overall, RMSEs in all models are very low so we won't worry much about that. We can say that Model4 is the best performing model in terms of AIC, BIC and R2 and hence we will select Model4.

```r
compare_performance(model1, model2, model3, rank = TRUE) %>% kable() %>% kable_styling()
```

Model	Type	AIC	BIC	R2	R2_adjusted	RMSE	BF	Performance_Score
model1	lm	189.4090	212.1487	0.5960003	0.5909817	0.3173751	1	0.6666667
model2	lm	624.5961	651.1258	0.6199863	0.6140671	0.6155092	0	0.3333333
model3	lm	624.5961	651.1258	0.6199863	0.6140671	0.6155092	0	0.3333333

```r
model_performance(model4@objects[[1]]) %>% kable() %>% kable_styling()
```

AIC	BIC	R2_Tjur	RMSE	LOGLOSS	SCORE_LOG	SCORE_SPHERICAL	PCP
172.0349	209.9345	0.7092821	0.681864	0.2324693	-Inf	0.024527	0.8550339

Prediction Accuracy

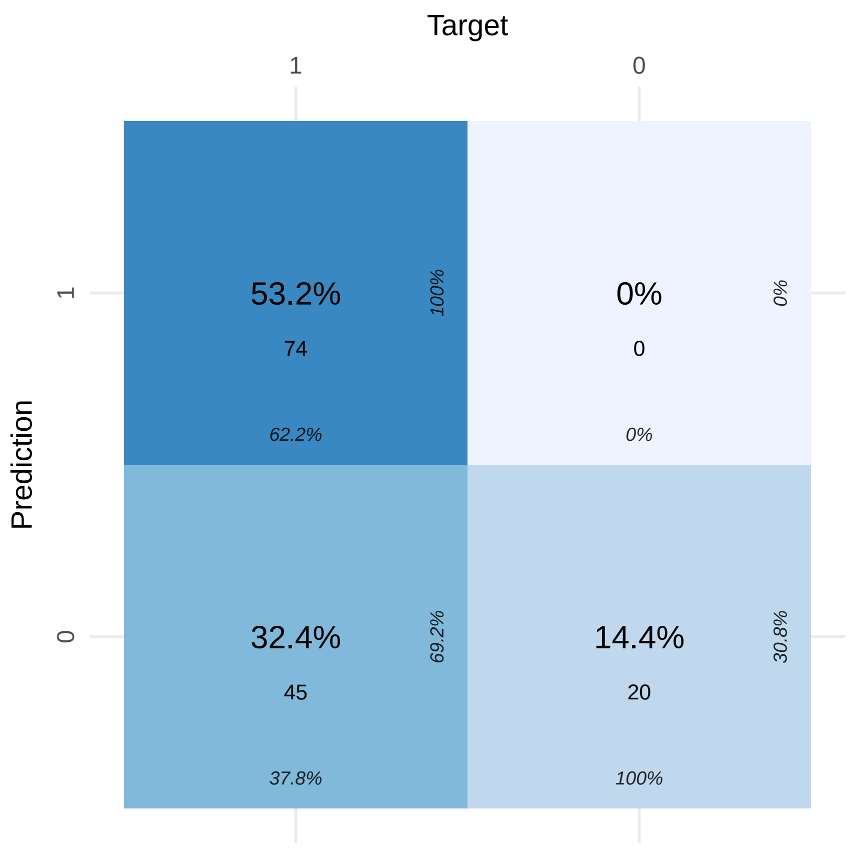
```r
test3 <- test2 # copy of test dataset
test3$target <- as.factor(test3$target)

# Calculating confusion matrix for model1
preds1 <- predict(model1, newdata = test3)
preds1[preds1 > 0.05] = 1
preds1[preds1 < 0.05] = 0
preds1 <- as.factor(preds1)
model1_cm <- confusionMatrix(preds1, test3$target,mode="everything")
tidy1 <- tidy(model1_cm[[2]])
model1_cm[[2]]
```

```
##             Reference
## Prediction   0  1
##            0 20  0
##            1 45 74
```

```r
round(model1_cm[[3]],3)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##          0.676          0.321          0.592          0.753          0.532
## AccuracyPValue  McnemarPValue
##          0.000          0.000
```

```r
plot_confusion_matrix(tidy1, targets_col="Prediction", predictions_col = "Reference",counts_col = "n")
```



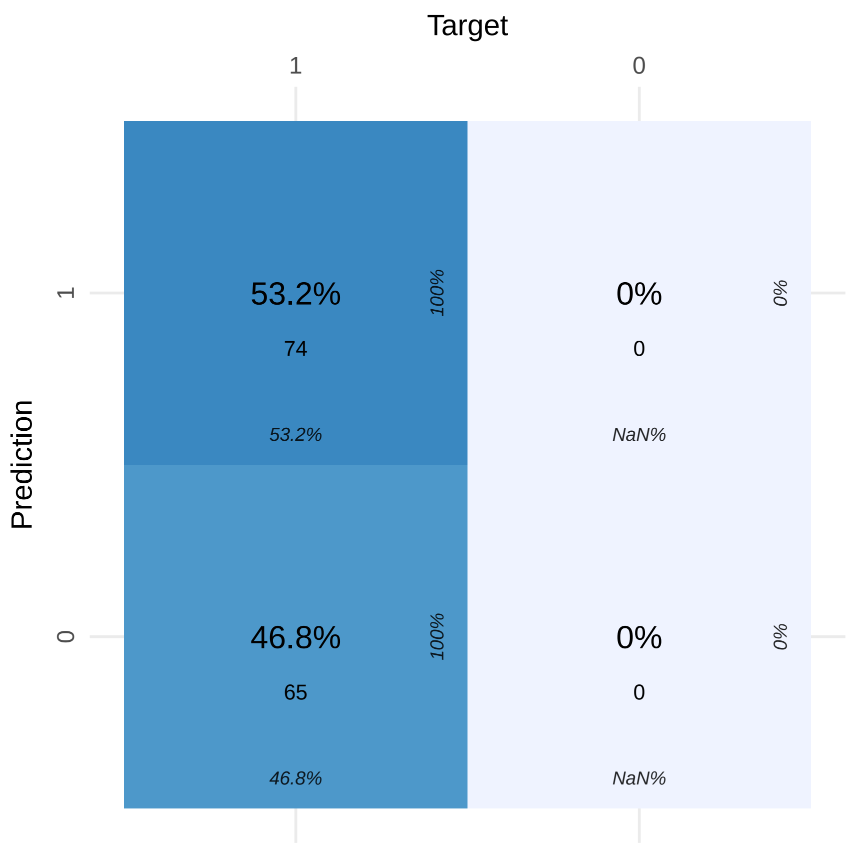
```r
# Calculating confusion matrix for model2
preds2 <- predict(model2, newdata = test3)
preds2[preds2 > 0.05] = 1
preds2[preds2 < 0.05] = 0
preds2 <- as.factor(preds2)
model2_cm <- confusionMatrix(preds2, test3$target,mode="everything")
tidy2 <- tidy(model2_cm[[2]])
model2_cm[[2]]
```

```
##           Reference
## Prediction  0  1
##          0  0  0
##          1 65 74
```

```r
round(model2_cm[[3]],3)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##          0.532          0.000          0.446          0.617          0.532
## AccuracyPValue  McnemarPValue
##          0.535          0.000
```

```r
plot_confusion_matrix(tidy2, targets_col="Prediction", predictions_col = "Reference",counts_col = "n")
```



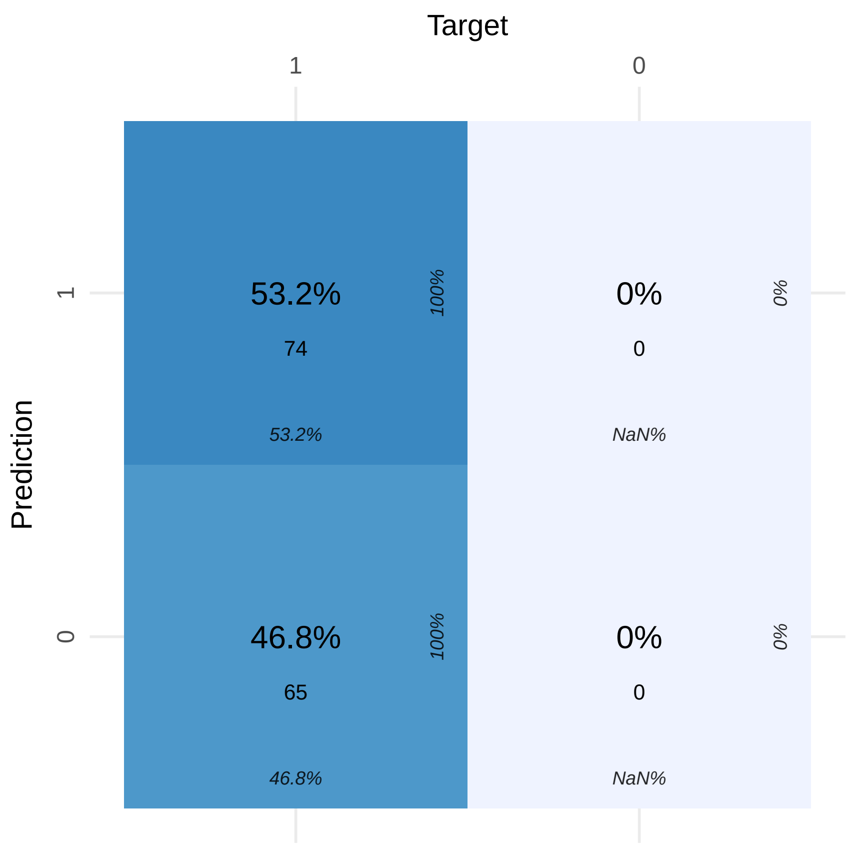
```r
# Calculating confusion matrix for model3
preds3 <- predict(model3, newdata = test3)
preds3[preds3 > 0.05] = 1
preds3[preds3 < 0.05] = 0
preds3 <- as.factor(preds3)
model3_cm <- confusionMatrix(preds3, test3$target,mode="everything")
tidy3 <- tidy(model3_cm[[2]])
model3_cm[[2]]
```

```
##           Reference
## Prediction  0  1
##          0  0  0
##          1 65 74
```

```r
round(model3_cm[[3]],3)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##          0.532          0.000          0.446          0.617          0.532
## AccuracyPValue  McnemarPValue
##          0.535          0.000
```

```r
plot_confusion_matrix(tidy3, targets_col="Prediction", predictions_col = "Reference",counts_col = "n")
```



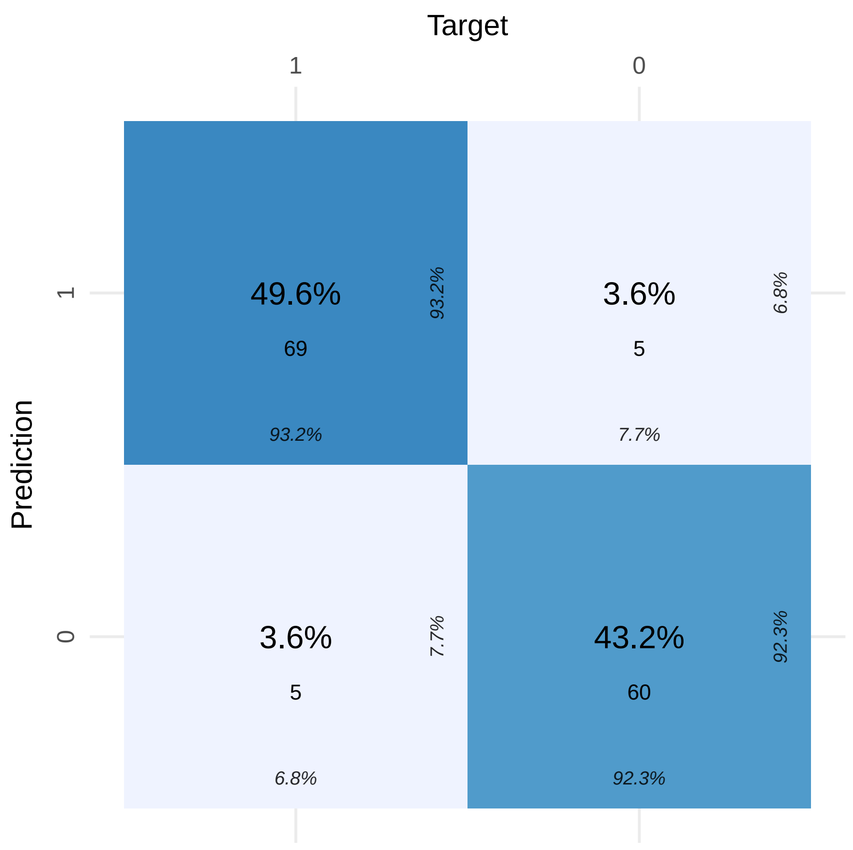
```r
# Calculating confusion matrix for model4
preds4 <- predict(model4@objects[[1]], newdata = test3)
preds4[preds4 > 0.05]  = 1
preds4[preds4 < 0.05]  = 0
preds4 <- as.factor(preds4)
model4_cm <- confusionMatrix(preds4, test3$target,mode="everything")
tidy4 <- tidy(model4_cm[[2]])
model4_cm[[2]]
```

```
##           Reference
## Prediction  0  1
##          0 60  5
##          1  5 69
```

```r
round(model4_cm[[3]],3)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##          0.928          0.856          0.872          0.965          0.532
## AccuracyPValue  McnemarPValue
##          0.000          1.000
```

```r
plot_confusion_matrix(tidy4, targets_col="Prediction", predictions_col = "Reference",counts_col = "n")
```



Model 1, 2 and 3 are the ones which we made out of some transformation and eliminating insignificant variables and seems like model 1 is the best as compared with Model 2 and Model 3. Model 1 has accuracy of 67.6 % as compared to Model 2 and 3 which have 53.2 percent. It makes sense because model2 and model3 are same other than using different technique i.e. step-wise regression. Model4 has accuracy of 92.8 percent which was achieved using glmulti function. It means that the prediction accuracy in model4 is 92.8%. 69 were truly identified as true positives while 60 were identified as true negative. Only 10 cases were identified as false positive and negative.

We will use model4 to predict the test set.

Predicting the test set

```
evaluation$target <- round(predict(model4@objects[[1]], evaluation),3)
evaluation <- evaluation %>% mutate(target = if_else(evaluation$target < 0.5, 0,1))
evaluation %>% kable(caption="Prediction based on Model4") %>% kable_styling(full_width = FALSE)
```

Table 3: Prediction based on Model4

zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	lstat	medv	target
0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	34.7	0
0	8.14	0	0.538	6.096	84.5	4.4619	4	307	21.0	10.26	18.2	1
0	8.14	0	0.538	6.495	94.4	4.4547	4	307	21.0	12.80	18.4	1
0	8.14	0	0.538	5.950	82.0	3.9900	4	307	21.0	27.71	13.2	0
0	5.96	0	0.499	5.850	41.5	3.9342	5	279	19.2	8.77	21.0	0
25	5.13	0	0.453	5.741	66.2	7.2254	8	284	19.7	13.15	18.7	0
25	5.13	0	0.453	5.966	93.4	6.8185	8	284	19.7	14.44	16.0	0
0	4.49	0	0.449	6.630	56.1	4.4377	3	247	18.5	6.53	26.6	0
0	4.49	0	0.449	6.121	56.8	3.7476	3	247	18.5	8.44	22.2	0
0	2.89	0	0.445	6.163	69.6	3.4952	2	276	18.0	11.34	21.4	0
0	25.65	0	0.581	5.856	97.0	1.9444	2	188	19.1	25.41	17.3	0
0	25.65	0	0.581	5.613	95.6	1.7572	2	188	19.1	27.26	15.7	0
0	21.89	0	0.624	5.637	94.7	1.9799	4	437	21.2	18.34	14.3	1
0	19.58	0	0.605	6.101	93.0	2.2834	5	403	14.7	9.81	25.0	1
0	19.58	0	0.605	5.880	97.3	2.3887	5	403	14.7	12.03	19.1	0
0	10.59	1	0.489	5.960	92.1	3.8771	4	277	18.6	17.27	21.7	0
0	6.20	0	0.504	6.552	21.4	3.3751	8	307	17.4	3.76	31.5	0
0	6.20	0	0.507	8.247	70.4	3.6519	8	307	17.4	3.95	48.3	1
22	5.86	0	0.431	6.957	6.8	8.9067	7	330	19.1	3.53	29.6	0
90	2.97	0	0.400	7.088	20.8	7.3073	1	285	15.3	7.85	32.2	0
80	1.76	0	0.385	6.230	31.5	9.0892	1	241	18.2	12.93	20.1	0
33	2.18	0	0.472	6.616	58.1	3.3700	7	222	18.4	8.93	28.4	0
0	9.90	0	0.544	6.122	52.8	2.6403	4	304	18.4	5.98	22.1	0
0	7.38	0	0.493	6.415	40.1	4.7211	5	287	19.6	6.12	25.0	0
0	7.38	0	0.493	6.312	28.9	5.4159	5	287	19.6	6.15	23.0	0
0	5.19	0	0.515	5.895	59.6	5.6150	5	224	20.2	10.56	18.5	1
80	2.01	0	0.435	6.635	29.7	8.3440	4	280	17.0	5.99	24.5	0
0	18.10	0	0.718	3.561	87.9	1.6132	24	666	20.2	7.12	27.5	1
0	18.10	1	0.631	7.016	97.5	1.2024	24	666	20.2	2.96	50.0	1
0	18.10	0	0.584	6.348	86.1	2.0527	24	666	20.2	17.64	14.5	1
0	18.10	0	0.740	5.935	87.9	1.8206	24	666	20.2	34.02	8.4	1
0	18.10	0	0.740	5.627	93.9	1.8172	24	666	20.2	22.88	12.8	1
0	18.10	0	0.740	5.818	92.4	1.8662	24	666	20.2	22.11	10.5	1
0	18.10	0	0.740	6.219	100.0	2.0048	24	666	20.2	16.59	18.4	1
0	18.10	0	0.740	5.854	96.6	1.8956	24	666	20.2	23.79	10.8	1
0	18.10	0	0.713	6.525	86.5	2.4358	24	666	20.2	18.13	14.1	1
0	18.10	0	0.713	6.376	88.4	2.5671	24	666	20.2	14.65	17.7	1
0	18.10	0	0.655	6.209	65.4	2.9634	24	666	20.2	13.22	21.4	1
0	9.69	0	0.585	5.794	70.6	2.8927	6	391	19.2	14.10	18.3	1
0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	5.64	23.9	0