DATA_621_HW3

Chi Pong, Euclid Zhang, Jie Zou, Joseph Connolly, LeTicia Cancel

3/8/2022

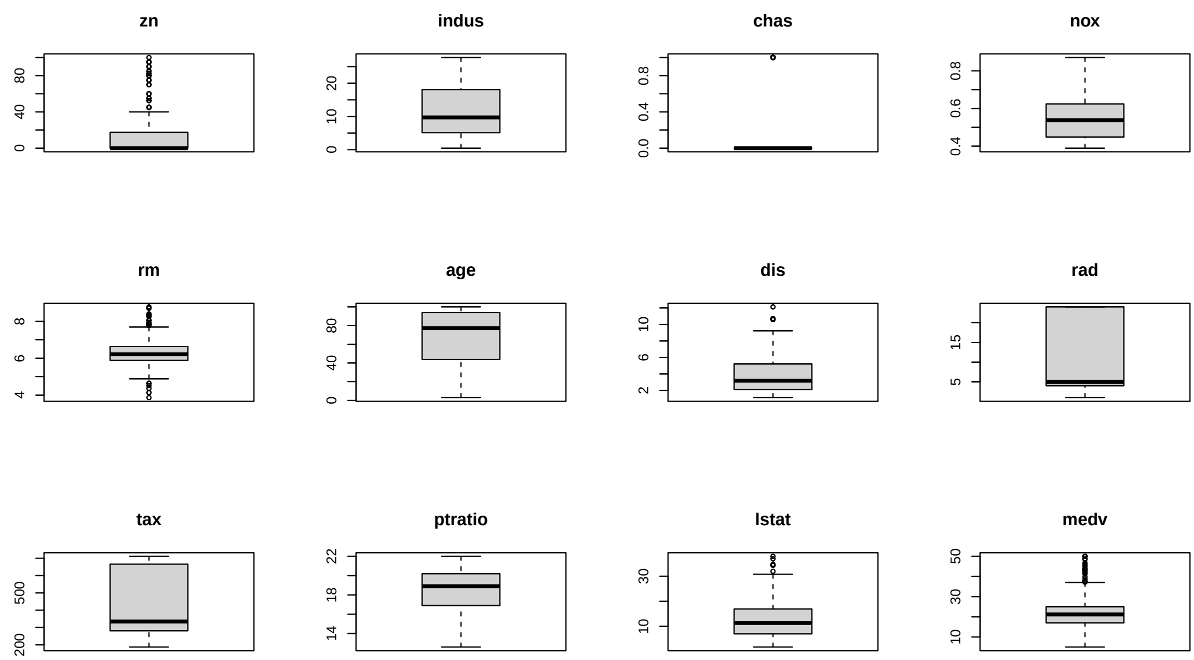
```
train_df <- read.csv("https://raw.githubusercontent.com/ezaccountz/DATA_621/main/HW3/crime-training-data
test_df <- read.csv("https://raw.githubusercontent.com/ezaccountz/DATA_621/main/HW3/crime-evaluation-da
```

DATA EXPLORATION

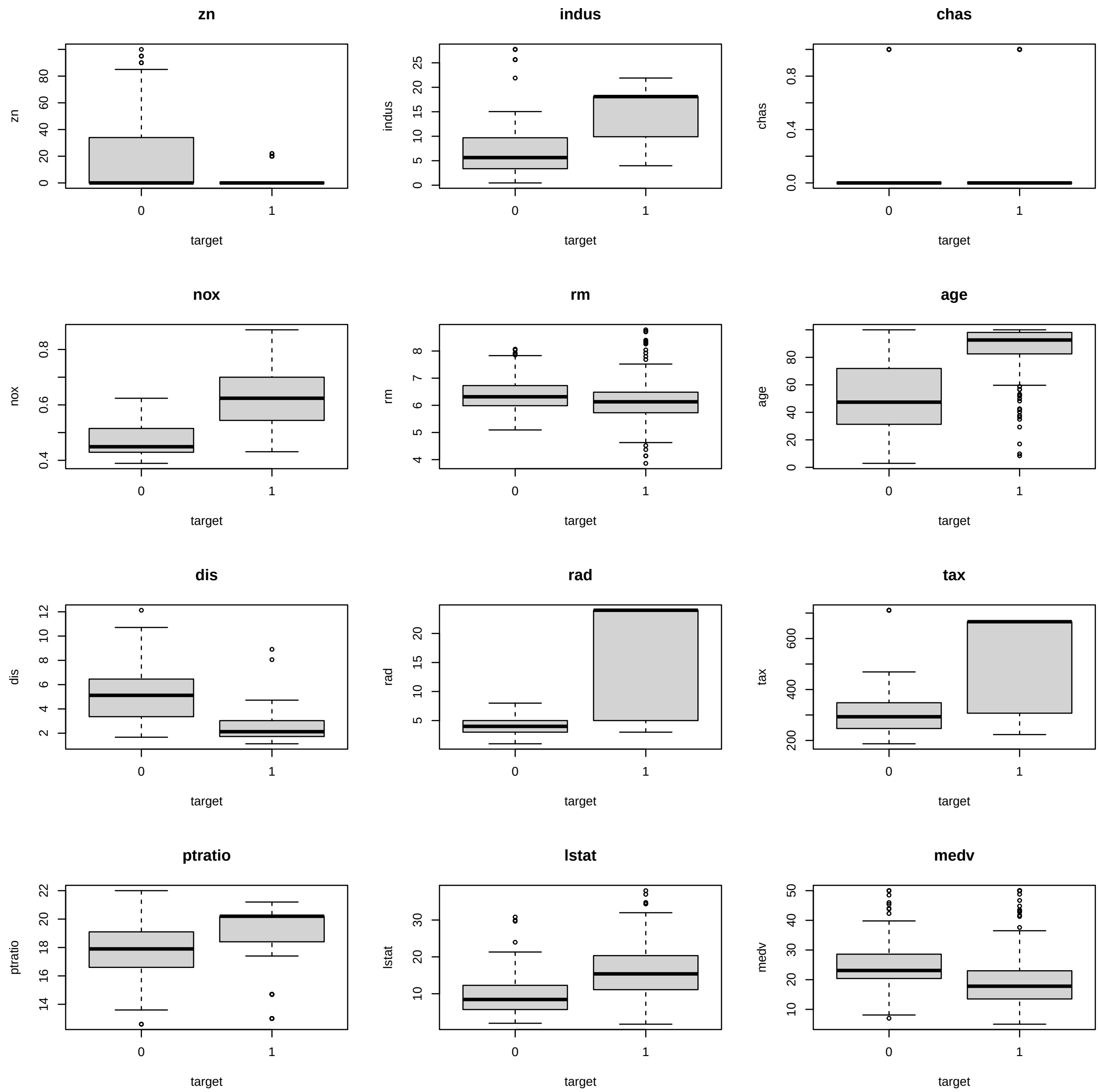
```
summary(train_df)
```

```
##        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         lstat             medv
##  Min.   :187.0   Min.   :12.6   Min.   : 1.730   Min.   : 5.00
##  1st Qu.:281.0   1st Qu.:16.9   1st Qu.: 7.043   1st Qu.:17.02
##  Median :334.5   Median :18.9   Median :11.350   Median :21.20
##  Mean   :409.5   Mean   :18.4   Mean   :12.631   Mean   :22.59
##  3rd Qu.:666.0   3rd Qu.:20.2   3rd Qu.:16.930   3rd Qu.:25.00
##  Max.   :711.0   Max.   :22.0   Max.   :37.970   Max.   :50.00
##      target
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.4914
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

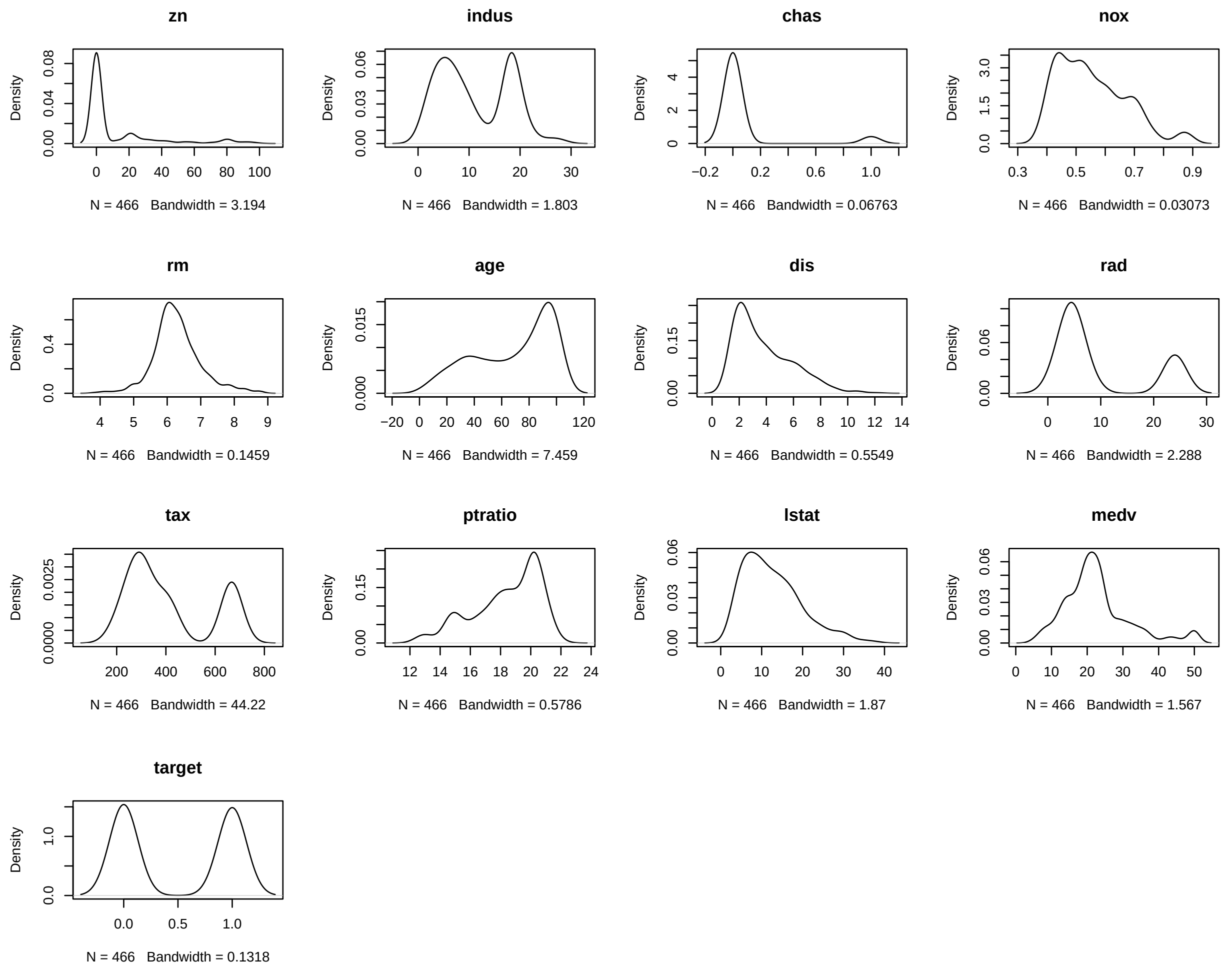
```r
par(mfrow=c(3,4))
predictors <- colnames(train_df)
predictors <- predictors[!predictors %in% c("target")]

for(preditor in predictors) {
  boxplot(train_df[,preditor],main=preditor)
}
```




```r
par(mfrow=c(4,3))
predictors <- colnames(train_df)
predictors <- predictors[!predictors %in% c("target")]

for(preditor in predictors) {
  boxplot(train_df[,preditor]~train_df$target,main=preditor,
          xlab = "target", ylab = preditor)
}
```



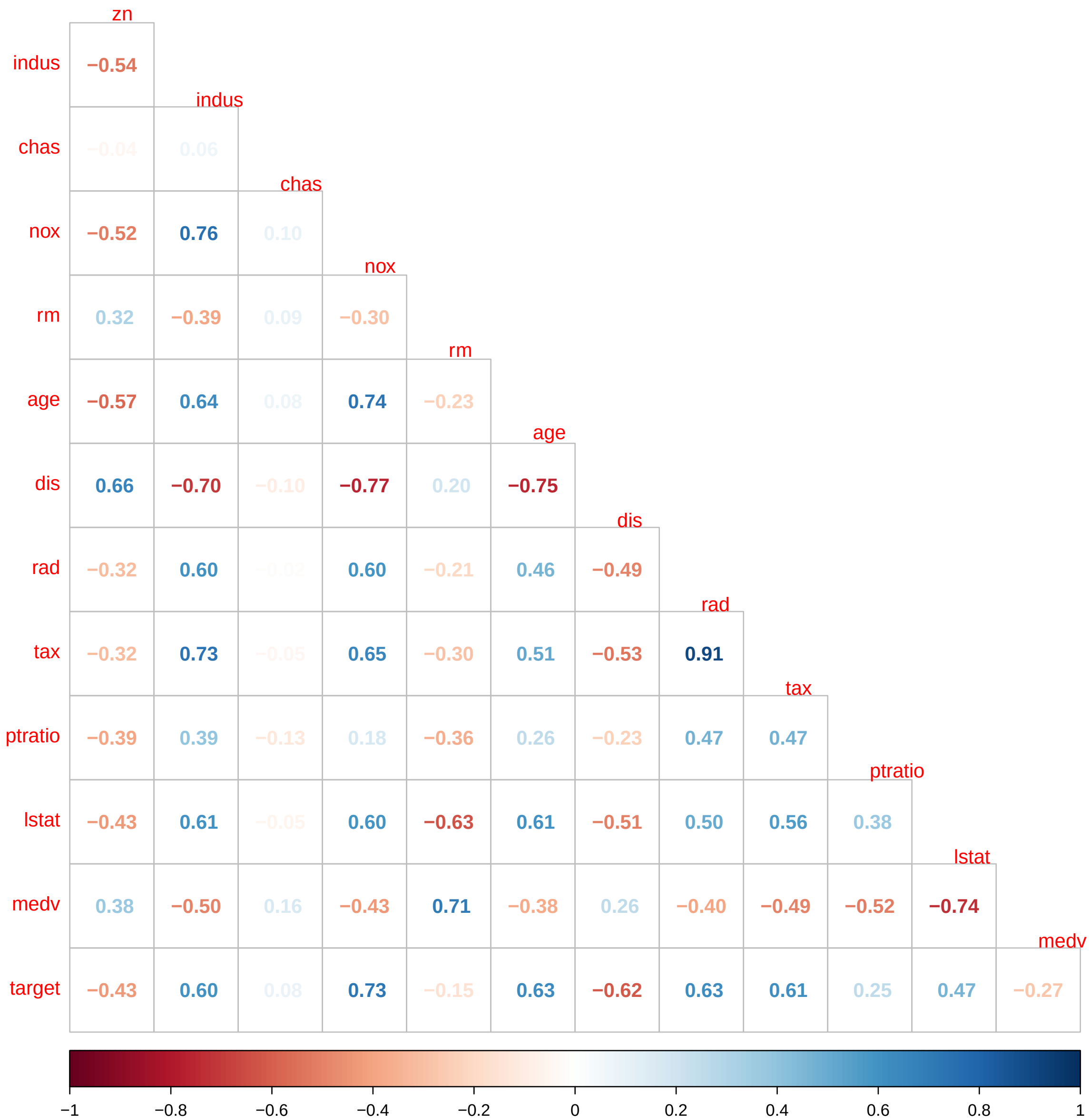

```r
par(mfrow=c(3,4))

predictors <- colnames(train_df)
#predictors <- predictors[!predictors %in% c("target")]

for(preditor in predictors) {
  plot(density(train_df[,preditor],na.rm=TRUE),main=preditor)
}
```



* Correlations

Now let's look at the correlations between the variables

```r
corrplot(cor(train_df, use = "na.or.complete"), method = 'number', type = 'lower', diag = FALSE, tl.srt
```



DATA PREPARATION

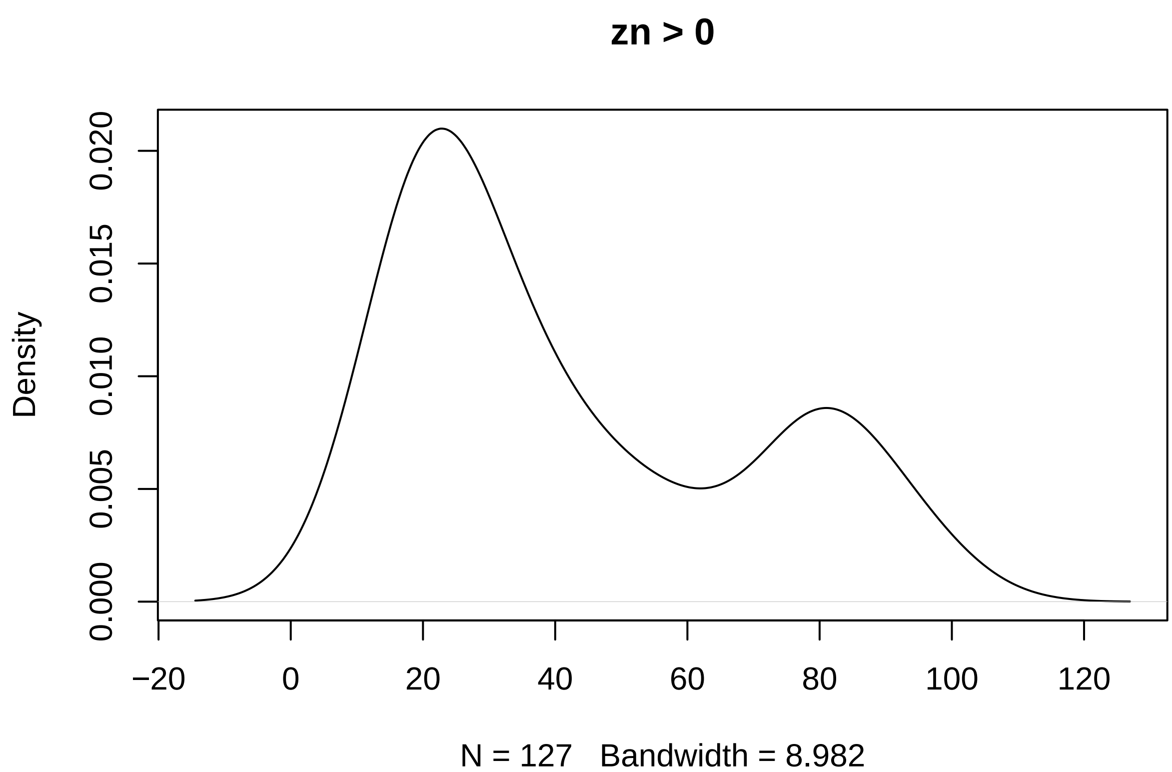
From the density plot of **zn**, we know that the variable is zero-inflated. The percentage of 0 values is

```r
nrow(train_df[train_df$zn==0,])/nrow(train_df)
```

```
## [1] 0.7274678
```

Let's check the distribution of the **zn** without the 0 values

```
plot(density(train_df[train_df$zn>0,]$zn,na.rm=TRUE), main = "zn > 0")
```



The distribution looks a lot better.
We will add a new dummy variable zn_y indicating if **zn** is >0. The interaction **zn x zn_y = zn** so we don't need to do anything to it. If **zn_y** is deemed to be insignificant by our models, then we can simply drop it.

```
train_df$zn_y <- 0
train_df$zn_y[train_df$zn>0] <- 1
```

According to the text book *A Modern Approach To Regression With R*, "when the predictor variable X has a Poisson distribution, the log odds are a linear function of x". Let's check if any of the predictors follows a Poisson distribution

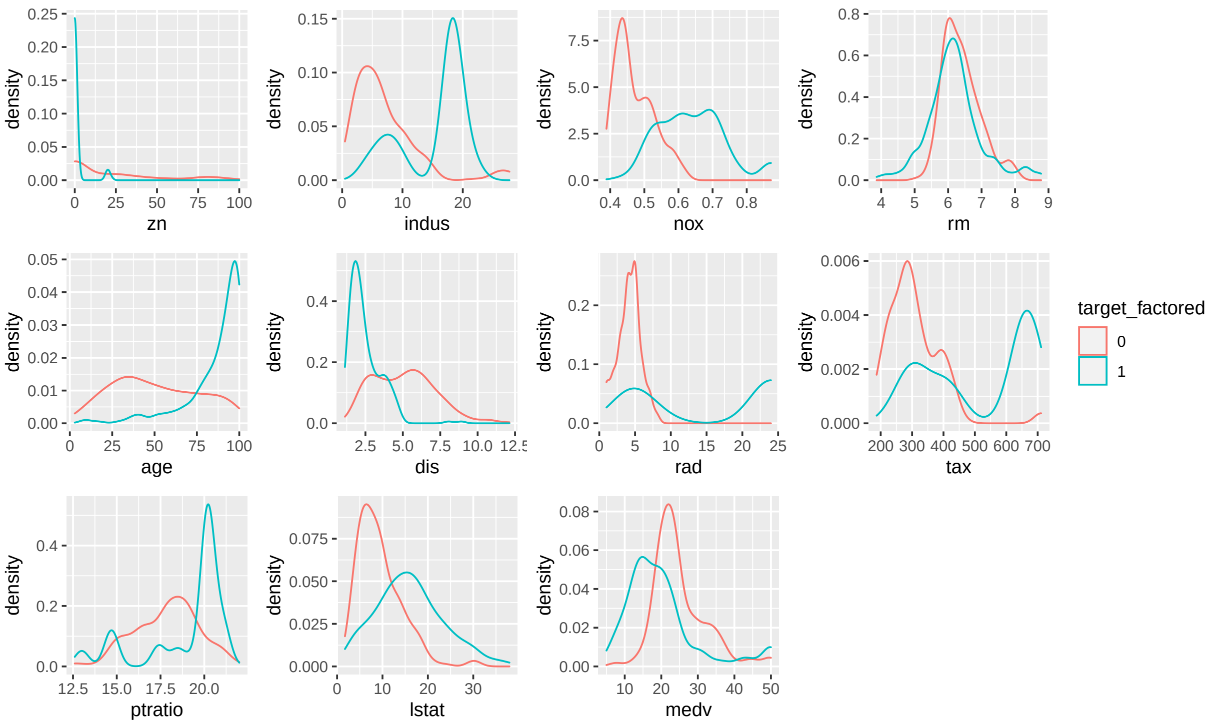
```
#Method of possion distribution test is from https://stackoverflow.com/questions/59809960/how-do-i-know

#two tail test
p_poisson <- function(x) {
  return (1-2 * abs((1 - pchisq((sum((x - mean(x))^2)/mean(x)), length(x) - 1))-0.5))
}

predictors <- colnames(train_df)
predictors <- predictors[!predictors %in% c("target","chas","zn_y")]

data.frame(mean = round(apply(train_df[,predictors],2,mean),2),
```

```
          variance = round(apply(train_df[,predictors],2,var),2),
          probability_of_poisson = round(apply(train_df[,predictors],2,p_poisson),2))
```

```
##            mean variance probability_of_poisson
## zn        11.58   545.91                   0.00
## indus     11.11    46.87                   0.00
## nox        0.55     0.01                   0.00
## rm         6.29     0.50                   0.00
## age       68.37   802.10                   0.00
## dis        3.80     4.44                   0.01
## rad        9.53    75.45                   0.00
## tax      409.50 28190.44                   0.00
## ptratio   18.40     4.83                   0.00
## lstat     12.63    50.44                   0.00
## medv      22.59    85.37                   0.00
```

None of the predictors follows a poisson distribution

```
target_factored <- as.factor(train_df$target)

plot_zn <- ggplot(train_df, aes(x=zn, color=target_factored)) + geom_density()
plot_indus <- ggplot(train_df, aes(x=indus, color=target_factored)) + geom_density()
plot_nox <- ggplot(train_df, aes(x=nox, color=target_factored)) + geom_density()
plot_rm <- ggplot(train_df, aes(x=rm, color=target_factored)) + geom_density()
plot_age <- ggplot(train_df, aes(x=age, color=target_factored)) + geom_density()
plot_dis <- ggplot(train_df, aes(x=dis, color=target_factored)) + geom_density()
plot_rad <- ggplot(train_df, aes(x=rad, color=target_factored)) + geom_density()
plot_tax <- ggplot(train_df, aes(x=tax, color=target_factored)) + geom_density()
plot_prtatio <- ggplot(train_df, aes(x=ptratio, color=target_factored)) + geom_density()
plot_lstat <- ggplot(train_df, aes(x=lstat, color=target_factored)) + geom_density()
plots_medv <- ggplot(train_df, aes(x=medv, color=target_factored)) + geom_density()

plot_zn+plot_indus+plot_nox+plot_rm+plot_age+plot_dis+plot_rad+plot_tax+
  plot_prtatio+plot_lstat+plots_medv+plot_layout(ncol = 4, guides = "collect")
```



The distributions for rm with target = 0 and target = 1 are approximately normal with the same variance. Hence we don't need to transform the variable The distributions for lstat and medv are skewed for both target = 0 and target = 1, we will add a log-transformed variable for each of them

The distributions for indus, nox, age, dis, tax, ptratio look significantly different for the target values. Let perform a anova tests on the single preditor models to see if adding a log transformed or a quadratic transformed variable will improve the performance.

```
predictors <- c("indus", "nox", "age", "dis", "tax", "ptratio")

n <- length(predictors)

model_compare <- data.frame(
    model_1 = paste0("target~",predictors),
    model_2 = paste0("target~",predictors,"+I(",predictors,"^2)"),
    Diff_DF = rep(0,n),
    Diff_Deviance = rep(0.0000,n),
    Pr_Gt_Chi = rep(0.0000,n)
)

for (i in (1:n)) {
    test_model_1 <- glm(target~train_df[,predictors[i]],family = binomial, train_df)
    test_model_2 <- glm(target~train_df[,predictors[i]]+I(train_df[,predictors[i]]^2),family = binomial
    anova_test <- anova(test_model_1,test_model_2,test="Chi")
    model_compare[i,3] <- anova_test$Df[2]
    model_compare[i,4] <- round(anova_test$Deviance[2],2)
    model_compare[i,5] <- round(anova_test$`Pr(>Chi)`[2],6)
}

model_compare
```

```
##          model_1                         model_2 Diff_DF Diff_Deviance Pr_Gt_Chi
## 1   target~indus     target~indus+I(indus^2)       1         31.13  0.000000
## 2     target~nox         target~nox+I(nox^2)       1          0.29  0.587879
## 3     target~age         target~age+I(age^2)       1          7.63  0.005748
## 4     target~dis         target~dis+I(dis^2)       1          3.64  0.056401
## 5     target~tax         target~tax+I(tax^2)       1          0.44  0.505781
## 6 target~ptratio target~ptratio+I(ptratio^2)       1         98.71  0.000000
```

```
predictors <- c("indus", "nox", "age", "dis", "tax", "ptratio")

n <- length(predictors)

model_compare <- data.frame(
    model_1 = paste0("target~",predictors),
    model_2 = paste0("target~",predictors,"+I(log(",predictors,"))"),
    Diff_DF = rep(0,n),
    Diff_Deviance = rep(0.0000,n),
    Pr_Gt_Chi = rep(0.0000,n)
)

for (i in (1:n)) {
    test_model_1 <- glm(target~train_df[,predictors[i]],family = binomial, train_df)
    test_model_2 <- glm(target~train_df[,predictors[i]]+I(log(train_df[,predictors[i]])),family = binomi
    anova_test <- anova(test_model_1,test_model_2,test="Chi")
    model_compare[i,3] <- anova_test$Df[2]
    model_compare[i,4] <- round(anova_test$Deviance[2],2)
    model_compare[i,5] <- round(anova_test$`Pr(>Chi)`[2],6)
}

model_compare
```

```
##          model_1                            model_2 Diff_DF Diff_Deviance Pr_Gt_Chi
## 1   target~indus     target~indus+I(log(indus))       1         13.91  0.000192
## 2     target~nox         target~nox+I(log(nox))       1          0.63  0.427570
## 3     target~age         target~age+I(log(age))       1          6.37  0.011603
## 4     target~dis         target~dis+I(log(dis))       1          5.15  0.023182
## 5     target~tax         target~tax+I(log(tax))       1          1.00  0.317895
## 6 target~ptratio target~ptratio+I(log(ptratio))       1         98.09  0.000000
```

For indus, the improvement is bigger by adding the squared term. For ptratio, since the distribution is left-skewed, it may be better to add the squared term. For other variables, no transformation is needed.

```
train_df$log_lstat <- log(train_df$lstat)
train_df$log_medv <- log(train_df$medv)
train_df$indus_squared <- train_df$indus^2
train_df$ptratio_squared <- train_df$ptratio^2
```

```
predictors <- colnames(train_df)
predictors <- predictors[!predictors %in% c("target","chas","zn_y","log_lstat","log_medv","indus_squared

interaction_test <- data.frame(matrix(ncol = 3, nrow = 0))
colnames(interaction_test) <- c("Preditor","Interaction","Pr_Gt_Chi")
```

```r
class(interaction_test$Pr_Gt_Chi) = "Numeric"

for (predictor in predictors) {
  interaction_test[nrow(interaction_test) + 1,] <-
    c(predictor, paste0(predictor, ":chas"),
      round(anova(glm(target ~ train_df[,predictor]*chas,data = train_df, family = "binomial"),test="Chi
}
```

```r
interaction_test
```

```
##    Preditor  Interaction Pr_Gt_Chi
## 1        zn      zn:chas    0.0645
## 2     indus   indus:chas     0.954
## 3       nox     nox:chas    0.6638
## 4        rm      rm:chas    0.6647
## 5       age     age:chas    0.0719
## 6       dis     dis:chas    0.0681
## 7       rad     rad:chas    0.0191
## 8       tax     tax:chas         0
## 9   ptratio ptratio:chas    0.3917
## 10    lstat   lstat:chas    0.1006
## 11     medv    medv:chas    0.1555
```

We will add an interaction between **tax** and **chas** and an interaction between **rad** and **chas** to our preditor candidates.

```r
train_df$tax_chas <- train_df$tax * train_df$chas
train_df$rad_chas <- train_df$rad * train_df$chas
```

```r
full_model <- glm(target~.,family = binomial, train_df)
```

```r
summary(full_model)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial, data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6227  -0.0932   0.0000   0.0001   3.7950
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      5.352e+01  2.666e+01   2.008 0.044688 *
## zn               1.010e-02  3.979e-02   0.254 0.799658
## indus            1.342e+00  4.258e-01   3.152 0.001623 **
## chas            -7.255e+02  5.172e+04  -0.014 0.988806
## nox              3.688e+01  8.993e+00   4.100 4.13e-05 ***
## rm              -1.801e+00  9.893e-01  -1.820 0.068699 .
## age              3.642e-02  1.562e-02   2.332 0.019712 *
## dis              6.235e-01  2.946e-01   2.116 0.034323 *
```

```
## rad               1.219e+00  3.188e-01   3.824 0.000131 ***
## tax              -2.145e-02  7.854e-03  -2.731 0.006311 **
## ptratio          -6.900e+00  2.303e+00  -2.996 0.002732 **
## lstat             2.391e-01  1.681e-01   1.422 0.154981
## medv              5.679e-01  2.286e-01   2.484 0.012980 *
## zn_y             -2.085e+00  1.398e+00  -1.491 0.136052
## log_lstat        -3.186e+00  2.269e+00  -1.404 0.160181
## log_medv         -8.051e+00  4.937e+00  -1.631 0.102965
## indus_squared    -4.242e-02  1.329e-02  -3.191 0.001418 **
## ptratio_squared   2.027e-01  6.390e-02   3.172 0.001516 **
## tax_chas          2.868e+00  2.052e+02   0.014 0.988849
## rad_chas         -1.707e+01  1.429e+03  -0.012 0.990469
## ---
## 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: 148.14  on 446  degrees of freedom
## AIC: 188.14
##
## Number of Fisher Scoring iterations: 22
```

```r
model_AIC <- step(full_model, trace=0)
```

```r
summary(model_AIC)
```

```
##
## Call:
## glm(formula = target ~ indus + chas + nox + rm + age + dis +
##     rad + tax + ptratio + lstat + medv + zn_y + log_lstat + log_medv +
##     indus_squared + ptratio_squared + tax_chas, family = binomial,
##     data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6018  -0.0948   0.0000   0.0001   3.7829
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       5.453e+01  2.645e+01   2.061 0.039262 *
## indus             1.325e+00  4.199e-01   3.156 0.001601 **
## chas             -5.397e+03  3.651e+05  -0.015 0.988205
## nox               3.637e+01  8.744e+00   4.160 3.18e-05 ***
## rm               -1.813e+00  9.855e-01  -1.840 0.065813 .
## age               3.602e-02  1.550e-02   2.324 0.020109 *
## dis               6.136e-01  2.945e-01   2.083 0.037218 *
## rad               1.210e+00  3.177e-01   3.810 0.000139 ***
## tax              -2.117e-02  7.793e-03  -2.716 0.006603 **
## ptratio          -6.908e+00  2.298e+00  -3.006 0.002649 **
## lstat             2.396e-01  1.684e-01   1.423 0.154768
## medv              5.748e-01  2.267e-01   2.535 0.011244 *
## zn_y             -1.864e+00  1.097e+00  -1.699 0.089327 .
```

```
## log_lstat       -3.227e+00  2.267e+00  -1.424 0.154519
## log_medv        -8.226e+00  4.900e+00  -1.679 0.093179 .
## indus_squared   -4.183e-02  1.306e-02  -3.202 0.001365 **
## ptratio_squared  2.029e-01  6.378e-02   3.181 0.001468 **
## tax_chas         1.949e+01  1.318e+03   0.015 0.988204
## ---
## 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: 148.20  on 448  degrees of freedom
## AIC: 184.2
##
## Number of Fisher Scoring iterations: 25
```

```
drop1(glm(target ~ .-rad_chas-zn-lstat-log_lstat, family=binomial, train_df), test="Chi")
```

```
## Single term deletions
##
## Model:
## target ~ (zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + lstat + medv + zn_y + log_lstat + log_medv + indus_squared +
##     ptratio_squared + tax_chas + rad_chas) - rad_chas - zn -
##     lstat - log_lstat
##                 Df Deviance    AIC    LRT  Pr(>Chi)
## <none>               150.3  182.3
## indus            1    169.0  199.0   18.7 1.525e-05 ***
## chas             1    169.3  199.3   18.9 1.347e-05 ***
## nox              1    173.7  203.7   23.4 1.349e-06 ***
## rm               1   4829.8 4859.8 4679.5 < 2.2e-16 ***
## age              1    156.5  186.5    6.2 0.0130611 *
## dis              1    156.1  186.1    5.8 0.0164442 *
## rad              1    186.3  216.3   36.0 1.951e-09 ***
## tax              1    161.9  191.9   11.6 0.0006679 ***
## ptratio          1    159.7  189.7    9.4 0.0021868 **
## medv             1    163.4  193.4   13.1 0.0002903 ***
## zn_y             1    153.7  183.7    3.4 0.0668683 .
## log_medv         1    156.7  186.7    6.4 0.0116846 *
## indus_squared    1    170.0  200.0   19.6 9.343e-06 ***
## ptratio_squared  1    161.0  191.0   10.7 0.0010693 **
## tax_chas         1    169.4  199.4   19.0 1.274e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model_chi <- glm(target ~ .-rad_chas-zn-lstat-log_lstat, family=binomial, train_df)
```

```
summary(model_chi)
```

```
##
## Call:
## glm(formula = target ~ . - rad_chas - zn - lstat - log_lstat,
```

```
##     family = binomial, data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7406  -0.0828   0.0000   0.0001   3.7127
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      5.201e+01  2.542e+01   2.046 0.040793 *
## indus            1.359e+00  4.286e-01   3.171 0.001521 **
## chas            -5.510e+03  3.890e+05  -0.014 0.988699
## nox              3.631e+01  8.729e+00   4.160 3.18e-05 ***
## rm              -1.622e+00  8.247e-01  -1.966 0.049273 *
## age              3.496e-02  1.462e-02   2.391 0.016786 *
## dis              6.704e-01  2.935e-01   2.284 0.022385 *
## rad              1.224e+00  3.178e-01   3.851 0.000118 ***
## tax             -2.186e-02  7.821e-03  -2.795 0.005195 **
## ptratio         -6.725e+00  2.261e+00  -2.974 0.002935 **
## medv             7.044e-01  2.064e-01   3.412 0.000645 ***
## zn_y            -1.858e+00  1.078e+00  -1.724 0.084774 .
## log_medv        -1.090e+01  4.489e+00  -2.428 0.015167 *
## indus_squared   -4.276e-02  1.335e-02  -3.203 0.001358 **
## ptratio_squared  1.978e-01  6.285e-02   3.148 0.001643 **
## tax_chas         1.989e+01  1.404e+03   0.014 0.988698
## ---
## 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: 150.31  on 450  degrees of freedom
## AIC: 182.31
##
## Number of Fisher Scoring iterations: 25
```

```
model_p <- glm(target~.-rad_chas-tax_chas-chas-zn-log_medv-rm,family = binomial, train_df)
summary(model_p)
```

```
##
## Call:
## glm(formula = target ~ . - rad_chas - tax_chas - chas - zn -
##     log_medv - rm, family = binomial, data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0390  -0.1319  -0.0040   0.0006   3.7725
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     27.763802  21.866625   1.270  0.20420
## indus            0.703113   0.276722   2.541  0.01106 *
## nox             32.957730   8.394101   3.926 8.63e-05 ***
## age              0.024737   0.011576   2.137  0.03260 *
## dis              0.547057   0.273186   2.003  0.04523 *
```

```
## rad               0.966955   0.223057   4.335 1.46e-05 ***
## tax              -0.013892   0.005486  -2.532  0.01133 *
## ptratio          -6.231168   2.098641  -2.969  0.00299 **
## lstat             0.326216   0.141837   2.300  0.02145 *
## medv              0.120303   0.061489   1.957  0.05041 .
## zn_y             -2.467359   1.060157  -2.327  0.01995 *
## log_lstat        -3.272514   1.912560  -1.711  0.08707 .
## indus_squared    -0.023234   0.008872  -2.619  0.00883 **
## ptratio_squared   0.181452   0.057790   3.140  0.00169 **
## ---
## 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: 169.63  on 452  degrees of freedom
## AIC: 197.63
##
## Number of Fisher Scoring iterations: 9
```

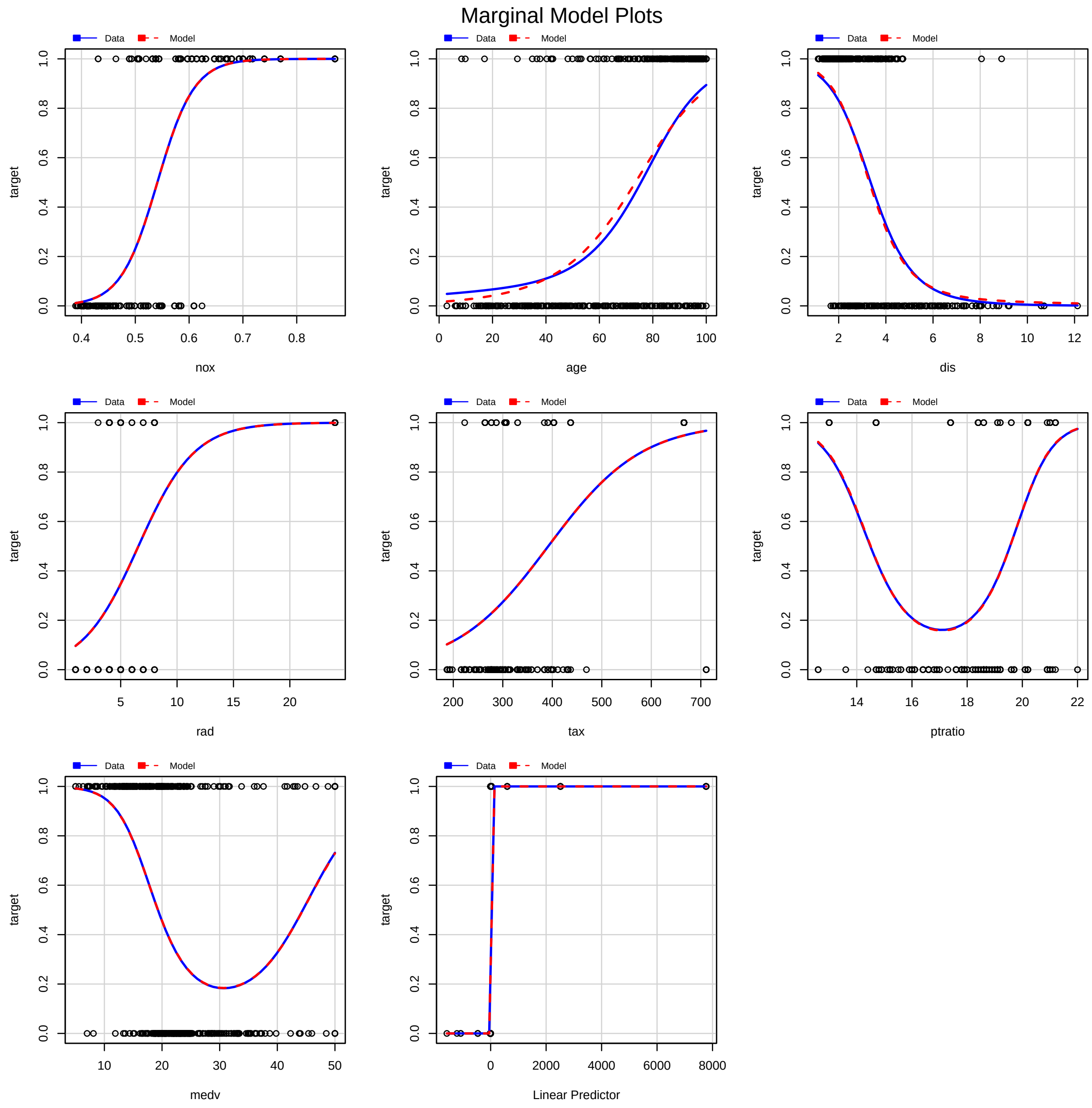
```r
model_compare <- data.frame(
    model = c("full_model","model_AIC","model_chi","model_p"),
    Deviance = rep(0.0000,4),
    AIC = rep(0.0000,4),
    Accurarcy = rep(0.0000,4),
    Sensitivity = rep(0.0000,4),
    Specificity = rep(0.0000,4),
    Precision = rep(0.0000,4),
    F1 = rep(0.0000,4),
    AUC = rep(0.0000,4),
    Nagelkerke_R_squared = rep(0.0000,4)
)
```

```r
models <- list(full_model, model_AIC, model_chi, model_p)

for (i in c(1:4)) {
  predicted_class <- ifelse(models[[i]]$fitted.values>0.5,1,0)
  confusion_matrix <- confusionMatrix(as.factor(predicted_class),
                                      as.factor(train_df$target),positive = "1")

  model_compare[i,] <- c(model_compare[i,1],round(models[[i]]$deviance,4), models[[i]]$aic,
                            confusion_matrix$overall[1],
                            confusion_matrix$byClass[1],
                            confusion_matrix$byClass[2],
                            confusion_matrix$byClass[3],
                            2*confusion_matrix$byClass[1]*confusion_matrix$byClass[3]/
                              (confusion_matrix$byClass[1]+confusion_matrix$byClass[3]),
                            auc(roc(train_df$target, models[[i]]$fitted.values)),
                            (1-exp((models[[i]]$dev-models[[i]]$null)/
                                     length(models[[i]]$residuals)))/
                              (1-exp(-models[[i]]$null/length(models[[i]]$residuals)))
                            )
}
```

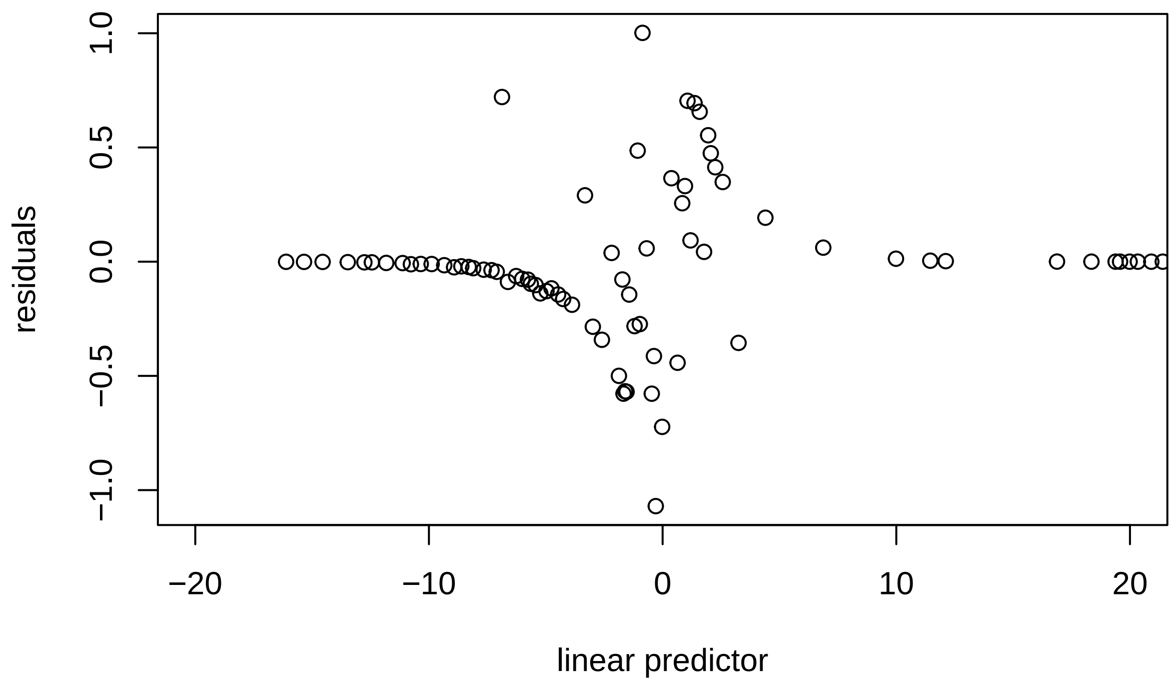
```r
model_compare[,c(2:10)] <- sapply(model_compare[,c(2:10)],as.numeric)
model_compare
```

```
##        model Deviance      AIC Accurarcy Sensitivity Specificity Precision
## 1 full_model 148.1421 188.1421 0.9356223   0.9257642   0.9451477 0.9422222
## 2  model_AIC 148.1999 184.1999 0.9356223   0.9257642   0.9451477 0.9422222
## 3  model_chi 150.3125 182.3125 0.9442060   0.9301310   0.9578059 0.9551570
## 4    model_p 169.6323 197.6323 0.9206009   0.9170306   0.9240506 0.9210526
##          F1       AUC Nagelkerke_R_squared
## 1 0.9339207 0.9854808            0.8752040
## 2 0.9339207 0.9856466            0.8751471
## 3 0.9424779 0.9844306            0.8730647
## 4 0.9190372 0.9794004            0.8535759
```

```r
marginalModelPlots(model_chi,~nox+age+dis+rad+tax+ptratio+medv,layout =c(3,3))
```

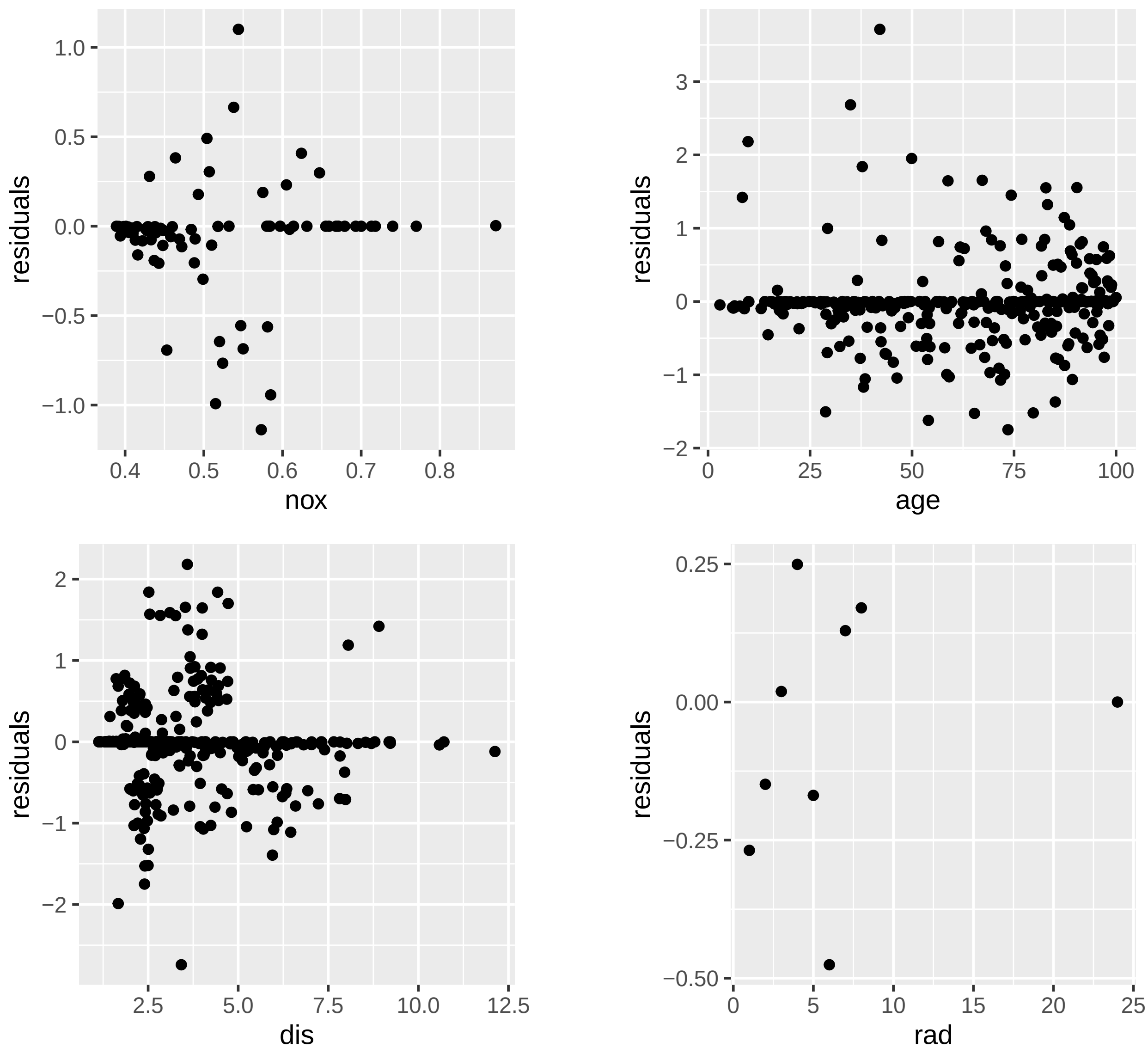


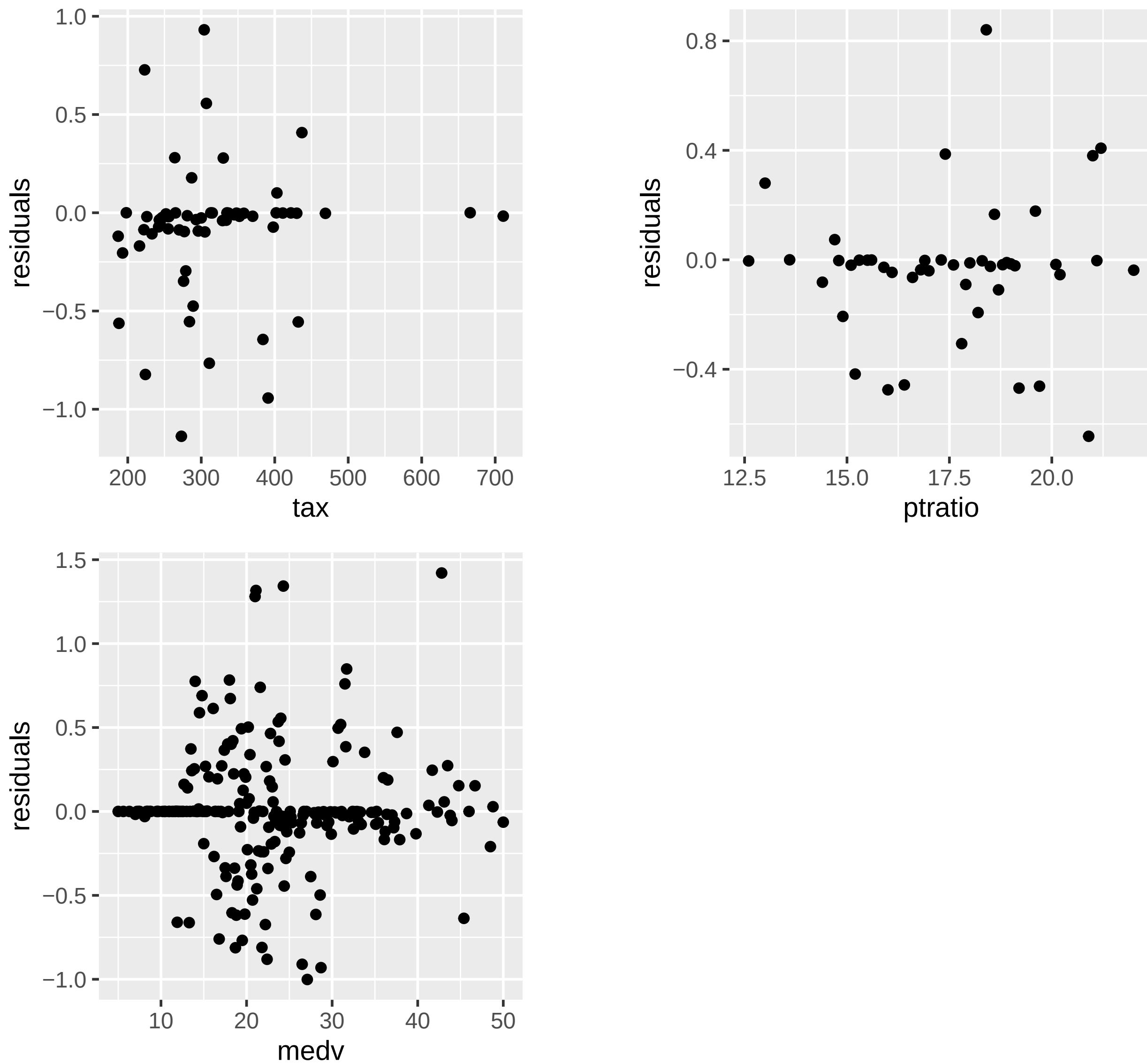

```
residual_df <- mutate(train_df, residuals=residuals(model_AIC,type="deviance"), linpred=predict(model_c
gdf <- group_by(residual_df, cut(linpred, breaks=unique(quantile(linpred,(1:100)/101))))
diagdf <- summarise(gdf, residuals=mean(residuals), linpred=mean(linpred))
plot(residuals ~ linpred, diagdf, xlab="linear predictor",xlim=c(-20,20))
```



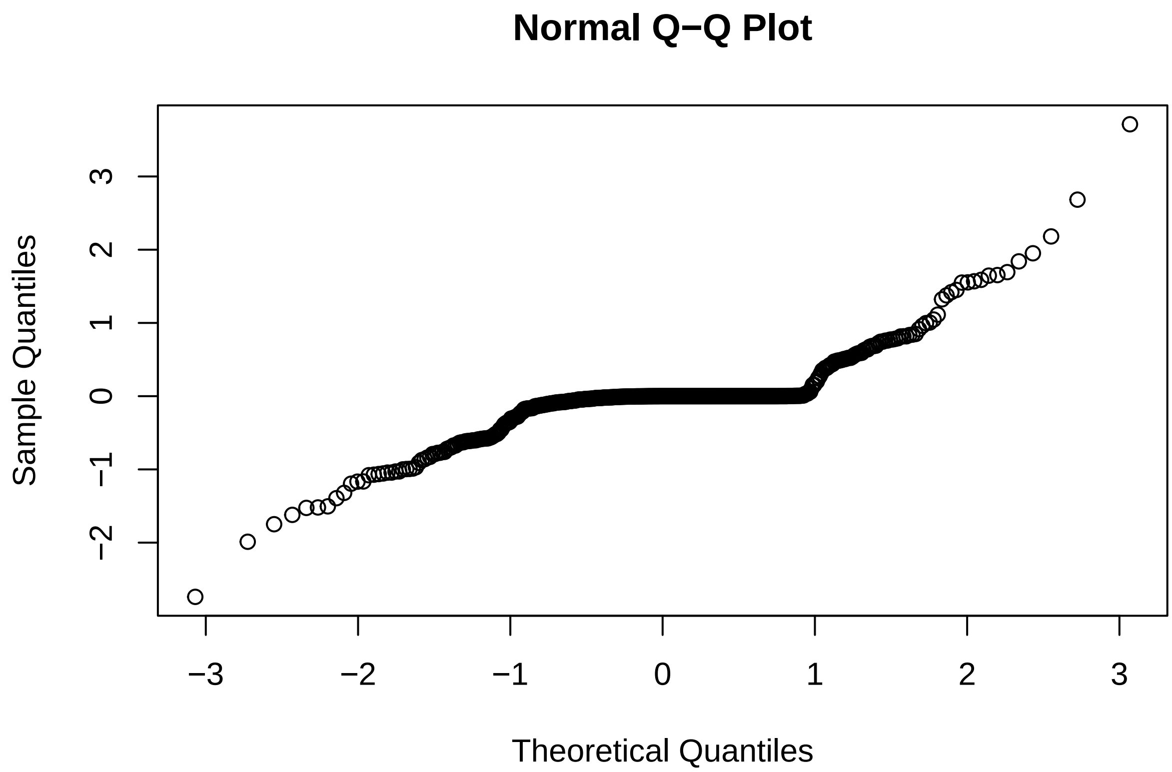

```r
predictors <- c("nox","age","dis","rad","tax","ptratio","medv")

residual_df <- mutate(train_df, residuals=residuals(model_chi,type="deviance"))
gg_plots <- list()

for (i in c(1:length(predictors))) {
    gdf <- group_by(residual_df, .dots = predictors[i])
    diagdf <- summarise(gdf, residuals=mean(residuals))
    print(ggplot(diagdf, aes_string(x=predictors[i],y="residuals")) + geom_point())
}
```

```
## Warning: The `.dots` argument of `group_by()` is deprecated as of dplyr 1.0.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

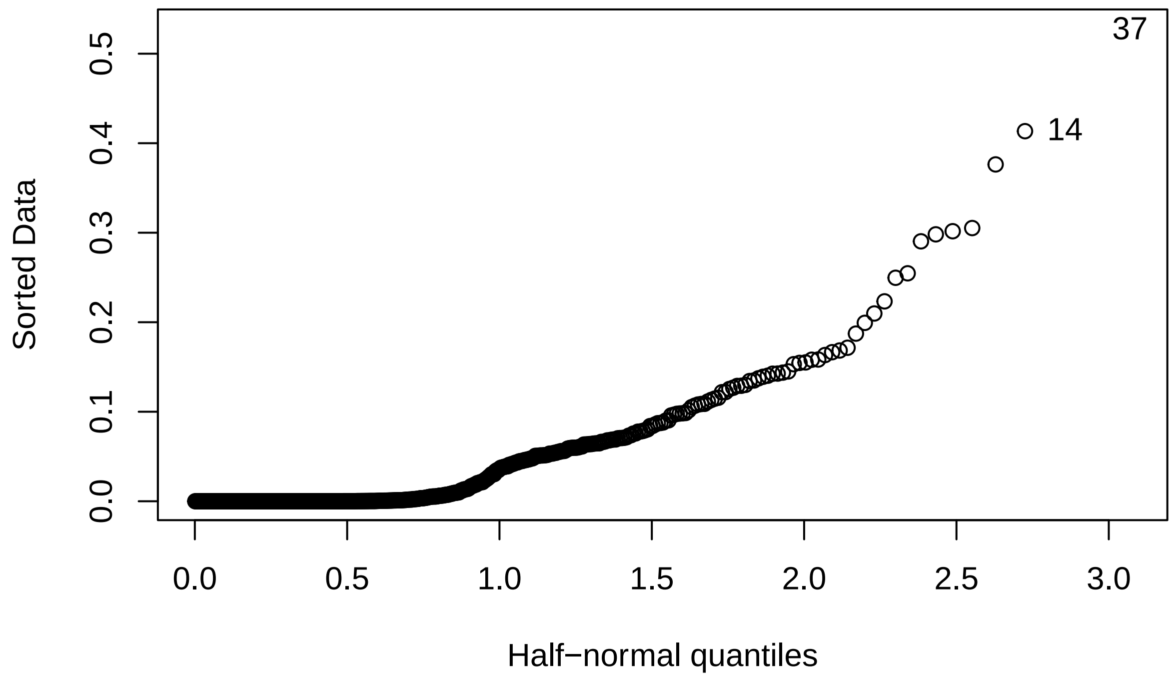

residuals
nox
0.4
0.5
0.6
0.7
0.8
1.0
0.5
0.0
−0.5
−1.0
residuals
age
0
25
50
75
100
3
2
1
0
−1
−2
residuals
dis
2.5
5.0
7.5
10.0
12.5
2
1
0
−1
−2
residuals
rad
0
5
10
15
20
25
0.25
0.00
−0.25
−0.50




```
qqnorm(residuals(model_chi))
```




```
halfnorm(hatvalues(model_chi))
```




```
train_df[c(14,37),]
```

```
##    zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv target zn_y
## 14 22  5.86    0 0.431 8.259  8.4 8.9067   7 330    19.1  3.54 42.8      1    1
## 37  0  2.46    0 0.488 7.831 53.6 3.1992   3 193    17.8  4.45 50.0      0    0
##    log_lstat log_medv indus_squared ptratio_squared tax_chas rad_chas
## 14  1.264127 3.756538       34.3396          364.81        0        0
## 37  1.492904 3.912023        6.0516          316.84        0        0
```

```
predict(model_chi,train_df[c(14,37),], type="link")
```

```
##         14         37
## -0.5557560 -0.8629056
```

```
test_df$zn_y <- 0
test_df$zn_y[test_df$zn>0] <- 1

test_df$indus_squared <- test_df$indus^2
test_df$ptratio_squared <- test_df$ptratio^2

test_df$log_lstat <- log(test_df$lstat)
test_df$log_medv <- log(test_df$medv)

test_df$tax_chas <- test_df$tax * test_df$chas
test_df$rad_chas <- test_df$rad * test_df$chas
```

```
test_df$predicted_class <- ifelse(predict(model_chi,test_df, type = "response") >0.5,1,0)
```

```
hist(test_df$predicted_class, main = "model_AIC prediction", xlab="predicted value")
```