DATA_621_HW3

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

3/31/2022

DATA EXPLORATION

Data Summary

```
summary(train_df)
```

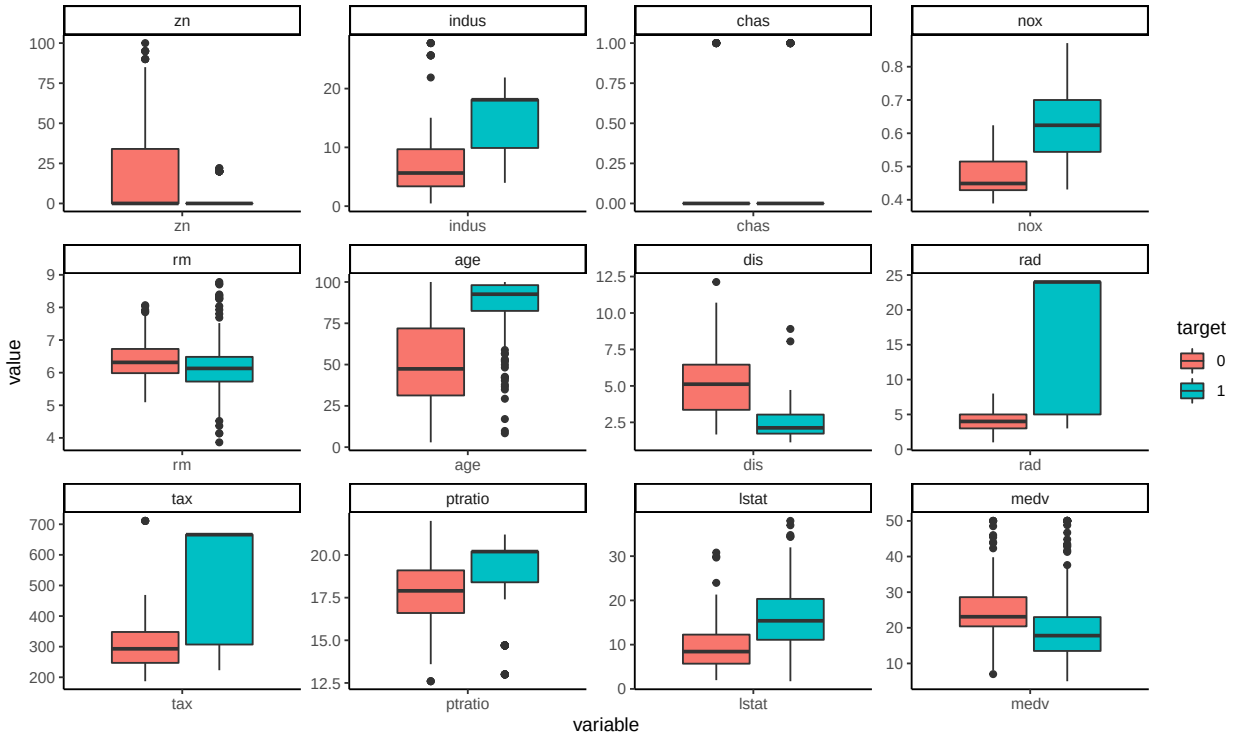
```
##        zn              indus             chas              nox        
##  Min.   :  0.00   Min.   : 0.460   Min.   :0.00000   Min.   :0.3890  
##  1st Qu.:  0.00   1st Qu.: 5.145   1st Qu.:0.00000   1st Qu.:0.4480  
##  Median :  0.00   Median : 9.690   Median :0.00000   Median :0.5380  
##  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
```

From the summary, two things stand out:

1. There are **no missing value**
2. All columns are numeric variables except **zn**. The max value for **zn** is not significantly off from the third quartile. This indicates that they have no extreme outliers. We will need to look at the distribution of **zn** to check the outliers.

Box Plots

```
data.m <- melt(train_df, id.vars = 'target')%>% mutate(target = as.factor(target))
ggplot(data.m, aes(x = variable, y = value, fill = target)) + geom_boxplot() +
  facet_wrap(~ variable, scales = 'free') + theme_classic()
```



From the boxplots, the following predictors seem to be good candidates differentiating target = 0 and target = 1:

- **zn**
- **indus**
- **nox**
- **age**
- **dis**
- **rad**
- **tax**

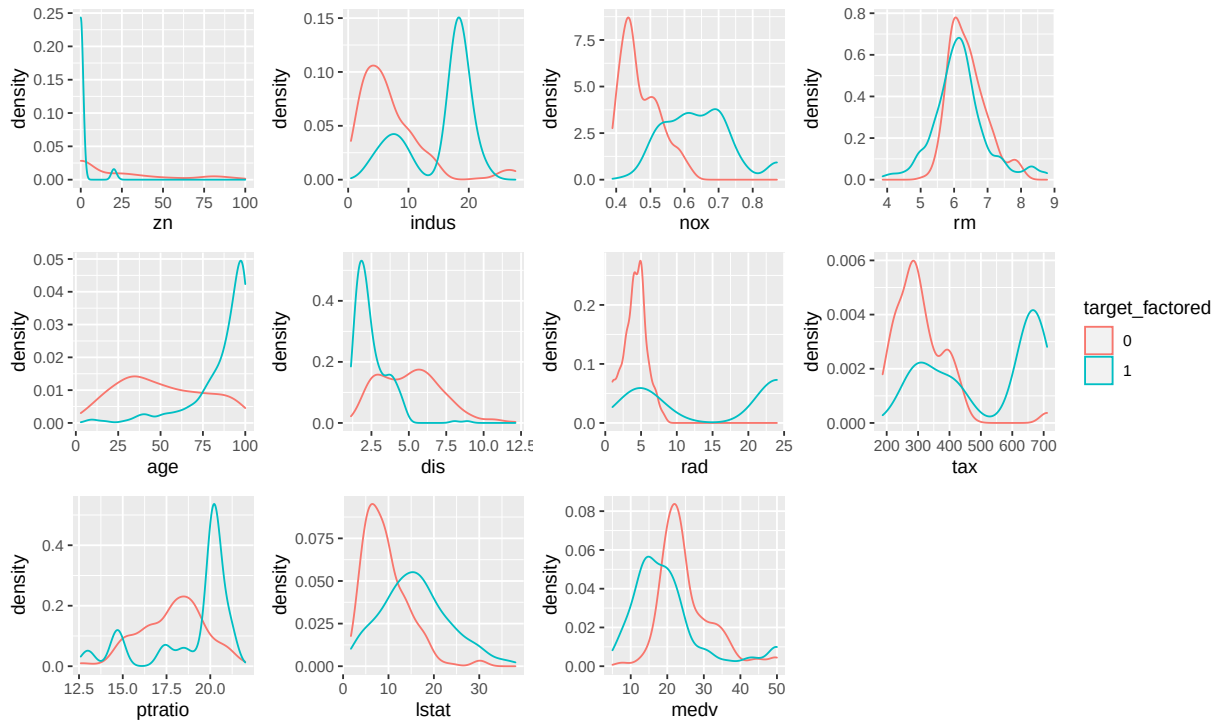
Most of the predictors seem to have a different distribution for target = 0 and target = 1.

Let's check the distributions of the predictors.

Distribution plots

```
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")
```



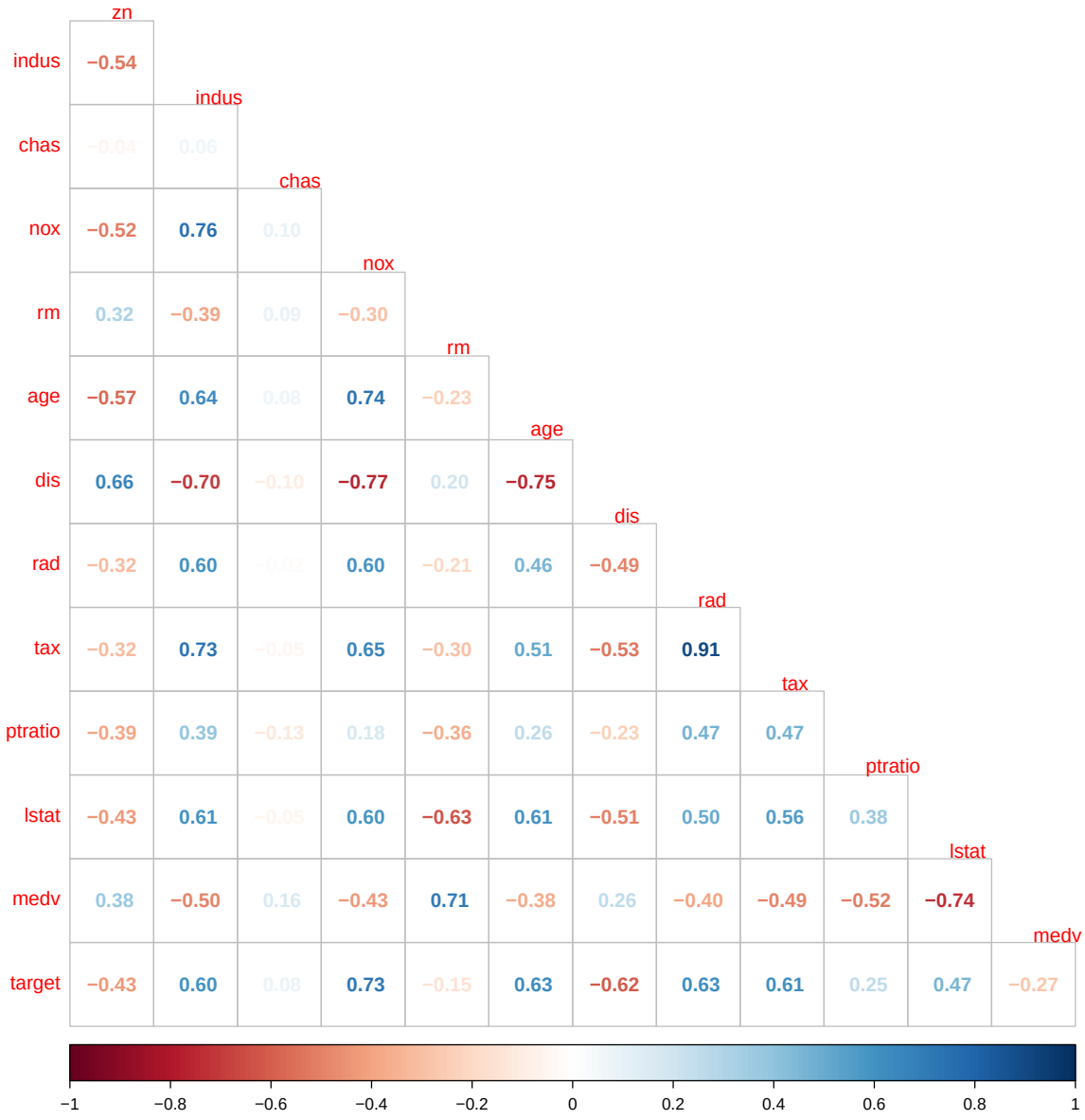
zn is zero-inflated for target = 1 (blue line). We may want to add or transform it into a dummy variable that will indicate if **zn** is greater than 0 or not.

lstat and **medv**, last two graphs, are right-skewed for both target = 0 and target = 1, we may consider a log-transformation. For other numeric variables, we may check later if transformations are needed.

Correlations

Now let's look at the correlations between the variables. We see that **rad** and **tax** have strong correlation.

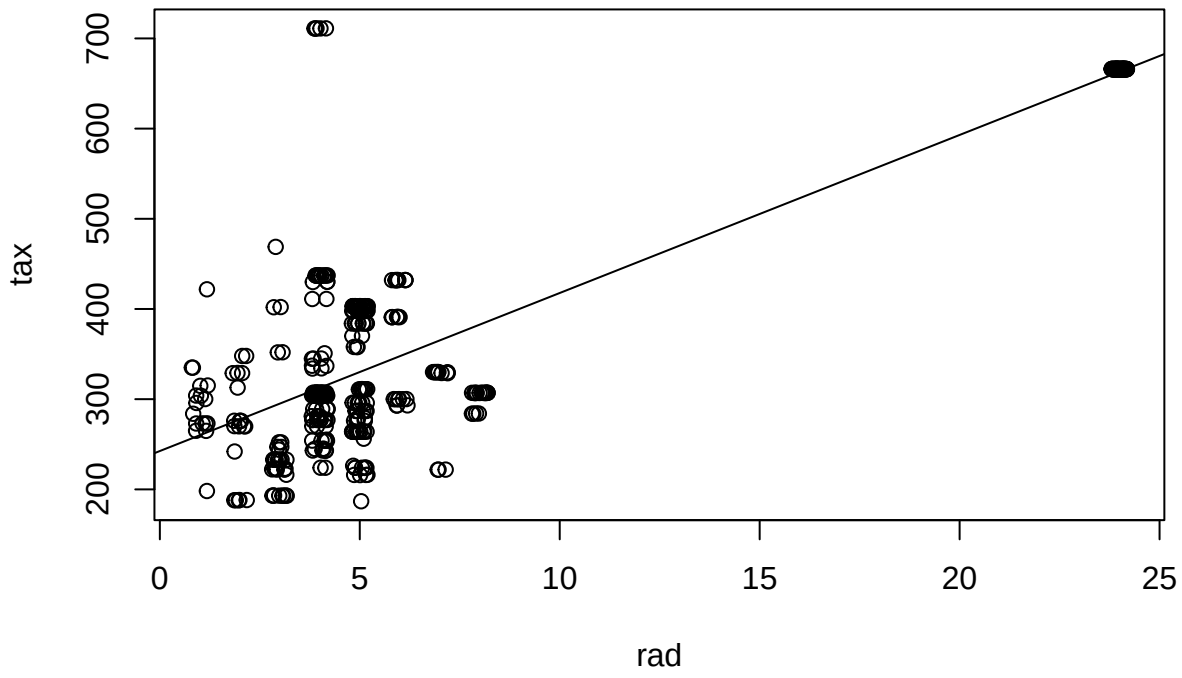
```
corrplot(cor(train_df, use = "na.or.complete"), method = 'number',
         type = 'lower', diag = FALSE, tl.srt = 0.1)
```



Linear Plot

Let's take a look at the linear plot of the two variables.

```
plot(jitter(train_df$rad),jitter(train_df$tax), xlab="rad", ylab="tax")
abline(lm (train_df$tax ~ train_df$rad))
```



We can see that the correlation is strongly influenced by one point, where **rad** = 24 (upper right corner). Without **rad**, the correlation is only 0.1799913.

```
cor(train_df[train_df$rad < 20,"rad"],train_df[train_df$rad < 20,"tax"])
```

```
## [1] 0.1799913
```

In this case, we will not remove **rad** or **tax** from our model, but we will need to be cautious of the t-statistics of the two in our models.

Data Preparation

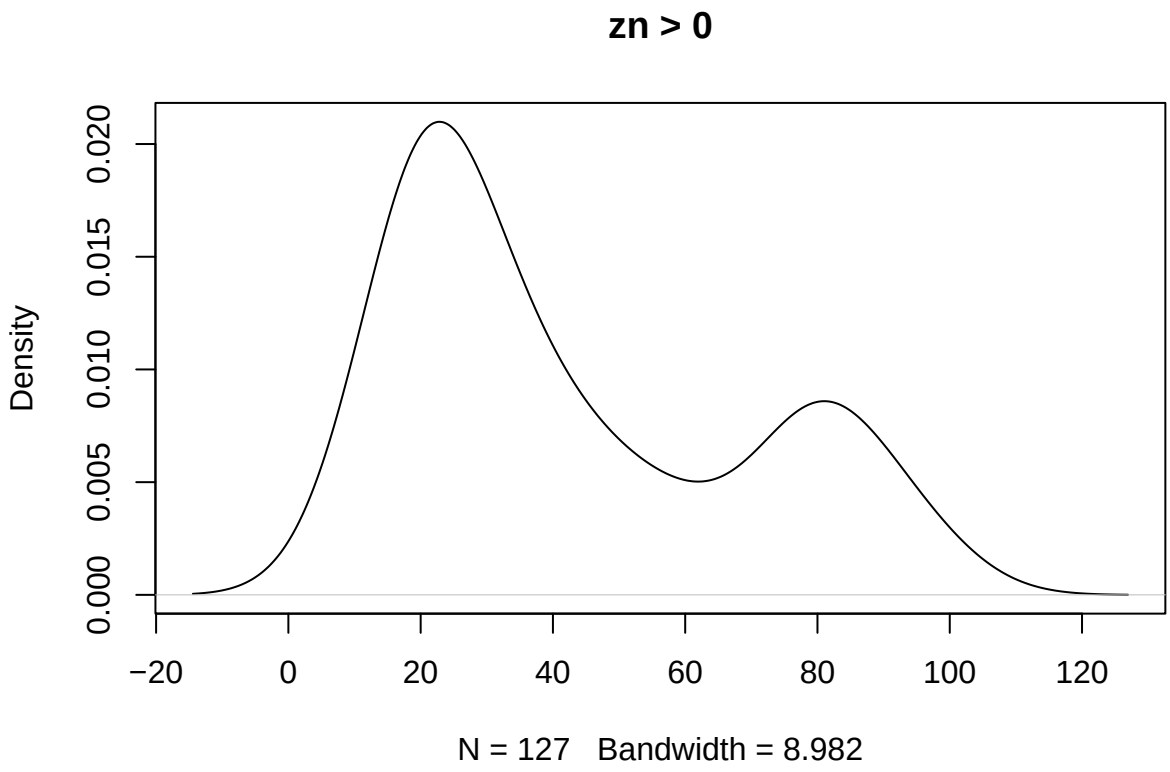
From the density plot of **zn**, we know that the variable is zero-inflated. The percentage of 0 values is 0.7274678.

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

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

If we plot the distribution of **zn** without the 0 values, we can see that the distribution looks much better.

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



We will add a new dummy variable **zn_y** indicating if **zn** is > 0. The interaction **zn * 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_target0 = round(apply(train_df[train_df$target==0,predictors],2,mean),2),
    variance_target0 = round(apply(train_df[train_df$target==0,predictors],2,var),2),
```

```
    p_poisson_target0 = round(apply(train_df[train_df$target==0,predictors],2,p_poisson),2),
    mean_target1 = round(apply(train_df[train_df$target==1,predictors],2,mean),2),
    variance_target1 = round(apply(train_df[train_df$target==1,predictors],2,var),2),
    p_poisson_target1 = round(apply(train_df[train_df$target==1,predictors],2,p_poisson),2))
```

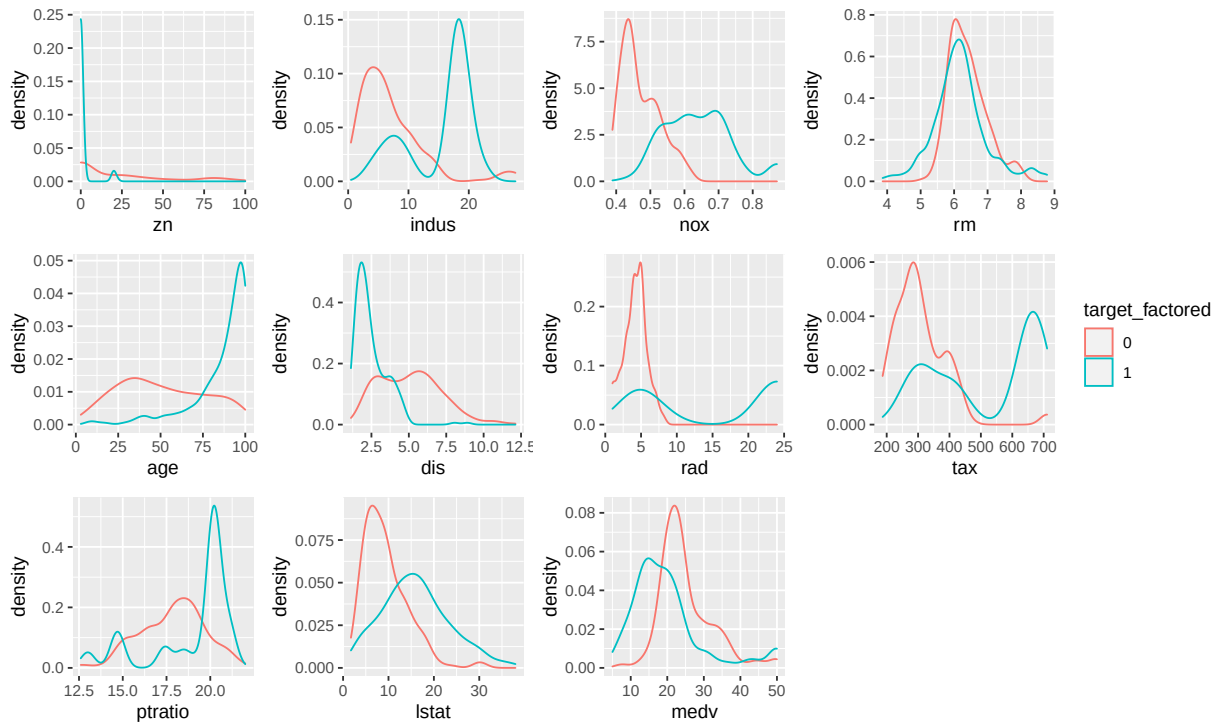
```
##         mean_target0 variance_target0 p_poisson_target0 mean_target1
## zn             21.48           850.75              0.00         1.33
## indus           7.04            30.27              0.00        15.31
## nox             0.47             0.00              0.00         0.64
## rm              6.40             0.31              0.00         6.18
## age            50.84           665.13              0.00        86.50
## dis             5.08             4.27              0.07         2.47
## rad             4.17             2.54              0.00        15.07
## tax           308.75          7956.24              0.00       513.77
## ptratio        17.86             3.35              0.00        18.96
## lstat           9.36            23.86              0.00        16.02
## medv           25.04            53.83              0.00        20.05
##         variance_target1 p_poisson_target1
## zn                 25.28                 0
## indus              29.28                 0
## nox                 0.01                 0
## rm                  0.66                 0
## age               297.90                 0
## dis                 1.16                 0
## rad                90.53                 0
## tax             27786.70                 0
## ptratio             5.75                 0
## lstat              55.52                 0
## medv              105.65                 0
```

The null hypothesis in the tests is that the variable follows a poision distribution.

Based on the p-values, we reject the null hypothesis for all predictors. None of the predictors follows a poisson distribution for both target = 0 and target = 1.

Let's take a second look at the distributions plots.

```
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 would not 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.

```
train_df$log_lstat <- log(train_df$lstat)
train_df$log_medv <- log(train_df$medv)
```

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

Quadratic transformation test

```
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, train_df)
    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
```

Log transformation test

```
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 = binomial, train_df)
    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 added.

```
train_df$indus_squared <- train_df$indus^2
train_df$ptratio_squared <- train_df$ptratio^2
```

Interaction term test

chas is a dummy variable. We will perform a anova tests on the single predictor models to see if adding an interaction between **chas** and a predictor will improve the model.

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

interaction_test <- data.frame(matrix(ncol = 3, nrow = 0))
colnames(interaction_test) <- c("Preditor","Interaction","Pr_Gt_Chi")
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")[4,5],4))
}
```

```
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
```

From the result, we will add an interaction between **tax** and **chas** and an interaction between **rad** and **chas** to our predictor candidates.

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

BUILD MODELS

1. Full model

The full model includes all original and the transformed version of the predictors.

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

#store the model formulas for buidling models for cross validation
model_formulas <- c(paste(deparse(formula(full_model), width.cutoff = 500), collapse=""))

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
```

2. Backward Elimination by AIC

Starting with our full model we perform a backward elimination by comparing the AIC of the models. The result model is:

```
model_AIC <- step(full_model, trace=0)
model_formulas <- c(model_formulas, paste(deparse(formula(model_AIC),
                                                  width.cutoff = 500), collapse=""))

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
```

3. Backward Elimination with Chi-square test

Starting with our full model we perform backward elimination with Chi-square test.

```
#Define a function to perform backward elimination with Chi-square test
#using the significancy / alpha as one of the parameters

backward_chi <- function (train_df, significancy) {
  glm_string <- "target~."
  glm_formula <- as.formula(glm_string)

  repeat{
    drop1_chi <- drop1(glm(glm_formula, family=binomial, train_df), test="Chi")

    chi_result <- data.frame(preditors = rownames(drop1_chi)[-1],
             p_value = drop1_chi[-1,5])
    chi_result <- chi_result[order(chi_result$p_value,decreasing=TRUE),]

    if(chi_result[1,2] < significancy){
        break
    }
    else {
        glm_string <- paste0(glm_string,"-",chi_result[1,1])
        glm_formula <- as.formula(glm_string)
    }
  }

  return(glm_formula)
}
```

model with alpha 0.1 (based on Chi-square test)

```
model_chi_0.1 <- backward_chi(train_df, 0.1)
model_formulas <- c(model_formulas, model_chi_0.1)
model_chi_0.1 <- glm(model_chi_0.1, family=binomial, train_df)

summary(model_chi_0.1)
```

```
##
## Call:
## glm(formula = model_chi_0.1, 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 with alpha 0.05 (based on Chi-square test)

```
model_chi_0.05 <- backward_chi(train_df, 0.05)
model_formulas <- c(model_formulas, model_chi_0.05)
model_chi_0.05 <- glm(model_chi_0.05, family=binomial, train_df)
summary(model_chi_0.05)
```

```
##
## Call:
## glm(formula = model_chi_0.05, family = binomial, data = train_df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.36990  -0.09297   0.00000   0.00003   3.05262
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      6.020e+01  2.251e+01   2.674 0.007490 **
## indus            1.407e+00  3.722e-01   3.780 0.000157 ***
## chas            -5.411e+03  2.712e+05  -0.020 0.984082
## nox              3.217e+01  6.979e+00   4.610 4.02e-06 ***
## rm              -6.657e-01  6.996e-01  -0.952 0.341302
## rad              1.311e+00  2.903e-01   4.517 6.26e-06 ***
## tax             -2.429e-02  6.625e-03  -3.667 0.000246 ***
## ptratio         -6.620e+00  2.173e+00  -3.046 0.002319 **
## medv             6.507e-01  2.025e-01   3.212 0.001316 **
## log_medv        -1.315e+01  4.484e+00  -2.933 0.003358 **
## indus_squared   -4.418e-02  1.160e-02  -3.808 0.000140 ***
## ptratio_squared  1.950e-01  6.093e-02   3.200 0.001375 **
```

```
## tax_chas         1.954e+01  9.790e+02   0.020 0.984078
## ---
## 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: 160.19  on 453  degrees of freedom
## AIC: 186.19
##
## Number of Fisher Scoring iterations: 24
```

model with alpha 0.001 (based on Chi-square test)

```
model_chi_0.001 <- backward_chi(train_df, 0.001)
model_formulas <- c(model_formulas, model_chi_0.001)
model_chi_0.001 <- glm(model_chi_0.001, family=binomial, train_df)
summary(model_chi_0.001)
```

```
##
## Call:
## glm(formula = model_chi_0.001, family = binomial, data = train_df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.30930  -0.10815   0.00000   0.00013   2.94594
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      3.571e+01  2.051e+01   1.741 0.081663 .
## indus            1.188e+00  3.649e-01   3.256 0.001132 **
## chas            -5.464e+03  2.914e+05  -0.019 0.985041
## nox              3.285e+01  7.007e+00   4.688 2.76e-06 ***
## rm              -1.518e-01  6.562e-01  -0.231 0.817100
## rad              1.165e+00  2.953e-01   3.947 7.92e-05 ***
## tax             -2.195e-02  6.597e-03  -3.327 0.000877 ***
## ptratio         -7.218e+00  2.213e+00  -3.262 0.001107 **
## medv             8.703e-02  6.717e-02   1.295 0.195148
## indus_squared   -3.794e-02  1.132e-02  -3.353 0.000801 ***
## ptratio_squared  2.110e-01  6.166e-02   3.421 0.000623 ***
## tax_chas         1.973e+01  1.052e+03   0.019 0.985036
## ---
## 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: 169.78  on 454  degrees of freedom
## AIC: 193.78
##
## Number of Fisher Scoring iterations: 24
```

4. Backward Elimination based on t-values of the coefficients

Starting with our full model we perform backward elimination based on the t-values of the coefficients.

```
#Define a function to perform backward elimination based on the t-values of the coefficients
#using the significancy / alpha as one of the parameters

backward_p <- function (train_df, significancy) {
  glm_string <- "target~."
  glm_formula <- as.formula(glm_string)

  repeat{
    model_p <- glm(glm_formula, family=binomial, train_df)

    p_result <- data.frame(preditors = rownames(summary(model_p)$coefficients)[-1],
              p_value = summary(model_p)$coefficients[-1,4])
    p_result <- p_result[order(p_result$p_value,decreasing=TRUE),]


    if(p_result[1,2] < significancy){
        break
    }
    else {
        glm_string <- paste0(glm_string,"-",p_result[1,1])
        glm_formula <- as.formula(glm_string)
    }
  }

  return(glm_formula)
}
```

model with alpha 0.05 (based on the t-values of the coefficients)

alpha = 0.1 produces the same model as alpha = 0.05 so alpha = 0.1 is not used here.

```
model_p_0.05 <- backward_p(train_df, 0.05)
model_formulas <- c(model_formulas, model_p_0.05)
model_p_0.05 <- glm(model_p_0.05, family=binomial, train_df)
summary(model_p_0.05)
```

```
##
## Call:
## glm(formula = model_p_0.05, family = binomial, data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1811  -0.1348  -0.0043   0.0012   3.7040
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     21.565494  21.025464   1.026 0.305040
## indus            0.655387   0.270470   2.423 0.015387 *
## nox             32.968725   8.347166   3.950 7.83e-05 ***
```

```
## age              0.025691   0.011714   2.193 0.028298 *
## dis              0.666502   0.266970   2.497 0.012541 *
## rad              0.900580   0.211886   4.250 2.13e-05 ***
## tax             -0.013018   0.005438  -2.394 0.016672 *
## ptratio         -6.272631   2.063883  -3.039 0.002372 **
## lstat            0.100858   0.049742   2.028 0.042600 *
## medv             0.176336   0.053238   3.312 0.000926 ***
## zn_y            -2.617454   1.063343  -2.462 0.013834 *
## indus_squared   -0.021730   0.008692  -2.500 0.012414 *
## ptratio_squared  0.182663   0.056883   3.211 0.001322 **
## ---
## 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: 172.68  on 453  degrees of freedom
## AIC: 198.68
##
## Number of Fisher Scoring iterations: 9
```

model with alpha 0.01 (based on the t-values of the coefficients)

```
model_p_0.01 <- backward_p(train_df, 0.01)
model_formulas <- c(model_formulas, model_p_0.01)
model_p_0.01 <- glm(model_p_0.01, family=binomial, train_df)
summary(model_p_0.01)
```

```
##
## Call:
## glm(formula = model_p_0.01, family = binomial, data = train_df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.88094  -0.18124  -0.00327   0.00014   2.85404
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     34.400490  17.446746   1.972 0.048639 *
## indus            0.831323   0.282877   2.939 0.003295 **
## nox             29.226546   6.347383   4.605 4.13e-06 ***
## rad              1.132391   0.224553   5.043 4.59e-07 ***
## tax             -0.017648   0.005160  -3.420 0.000626 ***
## ptratio         -6.515603   1.893531  -3.441 0.000580 ***
## indus_squared   -0.026394   0.008849  -2.983 0.002857 **
## ptratio_squared  0.188472   0.052980   3.557 0.000375 ***
## ---
## 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: 193.89  on 458  degrees of freedom
## AIC: 209.89
##
## Number of Fisher Scoring iterations: 9
```

model with alpha 0.001 (based on the t-values of the coefficients)

```
model_p_0.001 <- backward_p(train_df, 0.001)
model_formulas <- c(model_formulas, model_p_0.001)
model_p_0.001 <- glm(model_p_0.001, family=binomial, train_df)
summary(model_p_0.001)
```

```
##
## Call:
## glm(formula = model_p_0.001, family = binomial, data = train_df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.89721  -0.27798  -0.03997   0.00557   2.55954
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -19.867422   2.368325  -8.389  < 2e-16 ***
## nox          35.633515   4.523677   7.877 3.35e-15 ***
## rad           0.637643   0.119444   5.338 9.38e-08 ***
## tax          -0.008146   0.002332  -3.493 0.000478 ***
## ---
## 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: 224.47  on 462  degrees of freedom
## AIC: 232.47
##
## Number of Fisher Scoring iterations: 8
```

SELECT MODELS

First, let's compare different metrics of all models we have built.

```
models <- list(full_model, model_AIC, model_chi_0.1, model_chi_0.05, model_chi_0.001,
               model_p_0.05, model_p_0.01, model_p_0.001)
model_names <- list("full_model", "model_AIC", "model_chi_0.1", "model_chi_0.05",
                    "model_chi_0.001","model_p_0.05", "model_p_0.01", "model_p_0.001")

model_compare <- data.frame(
    model = rep("",length(models)),
    Deviance = rep(0.0000,length(models)),
    AIC = rep(0.0000,length(models)),
```

```
    Accuracy = rep(0.0000,length(models)),
    Sensitivity = rep(0.0000,length(models)),
    Specificity = rep(0.0000,length(models)),
    Precision = rep(0.0000,length(models)),
    F1 = rep(0.0000,length(models)),
    AUC = rep(0.0000,length(models)),
    Nagelkerke_R_squared = rep(0.0000,length(models))
)
```

```
for (i in c(1:length(models))) {
  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,1] <- model_names[i]
  model_compare[i,2] <- round(models[[i]]$deviance,4)
  model_compare[i,3] <- models[[i]]$aic
  model_compare[i,4] <- confusion_matrix$overall[1]
  model_compare[i,5] <- confusion_matrix$byClass[1]
  model_compare[i,6] <- confusion_matrix$byClass[2]
  model_compare[i,7] <- confusion_matrix$byClass[3]
  model_compare[i,8] <- 2*confusion_matrix$byClass[1]*confusion_matrix$byClass[3]/
                        (confusion_matrix$byClass[1]+confusion_matrix$byClass[3])
  model_compare[i,9] <- auc(roc(train_df$target, models[[i]]$fitted.values))
  model_compare[i,10] <- (1-exp((models[[i]]$dev-models[[i]]$null)/
                                  length(models[[i]]$residuals)))/
                               (1-exp(-models[[i]]$null/length(models[[i]]$residuals)))


}
model_compare
```

```
##             model Deviance      AIC  Accuracy 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_0.1 150.3125 182.3125 0.9442060   0.9301310   0.9578059 0.9551570
## 4  model_chi_0.05 160.1878 186.1878 0.9420601   0.9257642   0.9578059 0.9549550
## 5 model_chi_0.001 169.7788 193.7788 0.9291845   0.9213974   0.9367089 0.9336283
## 6    model_p_0.05 172.6756 198.6756 0.9227468   0.9170306   0.9282700 0.9251101
## 7    model_p_0.01 193.8917 209.8917 0.9163090   0.9213974   0.9113924 0.9094828
## 8   model_p_0.001 224.4719 232.4719 0.8884120   0.8384279   0.9367089 0.9275362
##          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 0.9401330 0.9814457            0.8632040
## 5 0.9274725 0.9794926            0.8534249
## 6 0.9210526 0.9789030            0.8504315
## 7 0.9154013 0.9747572            0.8279318
## 8 0.8807339 0.9594458            0.7936446
```

Since this is **logistic regression with binary data**, Deviance shouldn't be used to judge a model's goodness of fit. **We will mainly use AIC and the accuracy**. Depending on the business objective, we may use

other metrics such as sensitivity and specificity to compare the models' performance. However, the business objective is not defined here so we simply use the accuracy. The Nagelkerke R squared is a pseudo version of the R squared, since R squared cannot be used for generalized linear regression. The Nagelkerke R squared should not be used to judge the goodness of fit of a single model. It can be used to compare the fit of different models.

The result shows that model_chi_0.1 has the lowest AIC and best performance in predicting the using the training data. The models produced based the t-values of the coefficients are not doing so well. It is reasonable since some of the predictors have high correlation with each other.

The following is the confusion matrix for our best model **model_chi_0.1**:

```
predicted_class <- ifelse(model_chi_0.1$fitted.values>0.5,1,0)
confusion_matrix <- confusionMatrix(as.factor(predicted_class),
                                    as.factor(train_df$target),positive = "1")
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 227  16
##          1  10 213
##
##                Accuracy : 0.9442
##                  95% CI : (0.9193, 0.9632)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8883
##
##  Mcnemar's Test P-Value : 0.3268
##
##             Sensitivity : 0.9301
##             Specificity : 0.9578
##          Pos Pred Value : 0.9552
##          Neg Pred Value : 0.9342
##              Prevalence : 0.4914
##          Detection Rate : 0.4571
##    Detection Prevalence : 0.4785
##       Balanced Accuracy : 0.9440
##
##        'Positive' Class : 1
##
```

Cross Validation (5 fold)

Let's perform a cross validation on all the models we have to check if they are doing well with unseen data.

```
models <- list(full_model, model_AIC, model_chi_0.1, model_chi_0.05, model_chi_0.001,
               model_p_0.05, model_p_0.01, model_p_0.001)
model_names <- list("full_model", "model_AIC", "model_chi_0.1", "model_chi_0.05",
                    "model_chi_0.001","model_p_0.05", "model_p_0.01", "model_p_0.001")
```

```
model_compare <- data.frame(
    model = rep("",length(models)),
    Accuracy_1 = rep(0.0000,length(models)),
    Accuracy_2 = rep(0.0000,length(models)),
    Accuracy_3 = rep(0.0000,length(models)),
    Accuracy_4 = rep(0.0000,length(models)),
    Accuracy_5 = rep(0.0000,length(models)),
    Accuracy_average = rep(0.0000,length(models)),
    AIC_1 = rep(0.0000,length(models)),
    AIC_2 = rep(0.0000,length(models)),
    AIC_3 = rep(0.0000,length(models)),
    AIC_4 = rep(0.0000,length(models)),
    AIC_5 = rep(0.0000,length(models)),
    AIC_average = rep(0.0000,length(models))
)
```

```
set.seed(14)
cv_df<-train_df[sample(nrow(train_df)),]
folds <- cut(seq(1,nrow(cv_df)),breaks=5,labels=FALSE)

#Perform 5 fold cross validation
for(i in 1:5){

    testIndexes <- which(folds==i,arr.ind=TRUE)
    testData <- cv_df[testIndexes, ]
    trainData <- cv_df[-testIndexes, ]

    for (j in c(1:length(models))) {
        test_model <- glm(model_formulas[[j]], family=binomial, trainData)
        predicted_class <- ifelse(predict(test_model,testData,type="response")>0.5,1,0)
        confusion_matrix <- confusionMatrix(as.factor(predicted_class),
                                            as.factor(testData$target),positive = "1")
        model_compare[j,1+i] <- confusion_matrix$overall[1]
        model_compare[j,7+i] <- test_model$aic
    }


}

model_compare$model <- unlist(model_names)
model_compare$Accuracy_average <- apply(model_compare[,c(2:6)],1,mean)
model_compare$AIC_average <- apply(model_compare[,c(8:12)],1,mean)
```

The following table shows the accuracy of predictions with the test data and the AICs of the trained model.

```
model_compare
```

```
##             model Accuracy_1 Accuracy_2 Accuracy_3 Accuracy_4 Accuracy_5
## 1      full_model  0.9042553  0.8924731  0.9247312  0.9462366  0.9139785
## 2       model_AIC  0.9042553  0.8924731  0.9247312  0.9462366  0.9139785
## 3   model_chi_0.1  0.9042553  0.8817204  0.9247312  0.9462366  0.9139785
```

```
## 4  model_chi_0.05  0.8936170  0.9247312  0.9462366  0.9354839  0.9139785
## 5 model_chi_0.001  0.8404255  0.9032258  0.9247312  0.9354839  0.8924731
## 6    model_p_0.05  0.8723404  0.8924731  0.9139785  0.9354839  0.8924731
## 7    model_p_0.01  0.8936170  0.8817204  0.9032258  0.9462366  0.8817204
## 8   model_p_0.001  0.7978723  0.9032258  0.9032258  0.9032258  0.8494624
##   Accuracy_average    AIC_1    AIC_2    AIC_3    AIC_4    AIC_5 AIC_average
## 1        0.9163349 138.2636 153.1110 154.1078 165.8600 162.2430    154.7171
## 2        0.9163349 134.2647 149.1812 150.1884 161.9548 158.3747    150.7928
## 3        0.9141844 131.6432 151.1524 146.5266 160.1229 156.9649    149.2820
## 4        0.9228094 136.2853 150.3605 153.8125 159.8476 160.0786    152.0769
## 5        0.8992679 141.4014 157.6030 159.1907 165.3722 164.1969    157.5529
## 6        0.9013498 147.3185 158.6745 157.4345 172.7835 166.4235    160.5269
## 7        0.9013040 156.1461 170.6614 169.5861 178.6296 172.1746    169.4396
## 8        0.8714024 168.8905 192.8978 188.4387 193.8342 190.5635    186.9249
```

model_AIC and **model_chi_0.1** have the best performance.

Since **model_chi_0.1** is a simpler model with 15 coefficients, we select model_chi_0.1 to be our best model as it is a more **parsimonious** model.

```
length(model_chi_0.1$coefficients) - 1  # -1 for the intercept
```

```
## [1] 15
```

```
length(model_AIC$coefficients) - 1      # -1 for the intercept
```

```
## [1] 17
```

Let's check our final model again.

```
summary(model_chi_0.1)
```

```
##
## Call:
## glm(formula = model_chi_0.1, 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
```

zn_y is not so significant. However, from the distribution plots, **zn** is has strong ability to differentiate target = 0 and target = 1 when **zn** is 0. It does poorly when **zn** is 1. We should keep this in our model.

For **chas** and **tax_chas**, they are highly correlated since **tax_chas** = 0 when **chas** = 0.

The percentage of 0 in **chas** is 0.9291845.

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

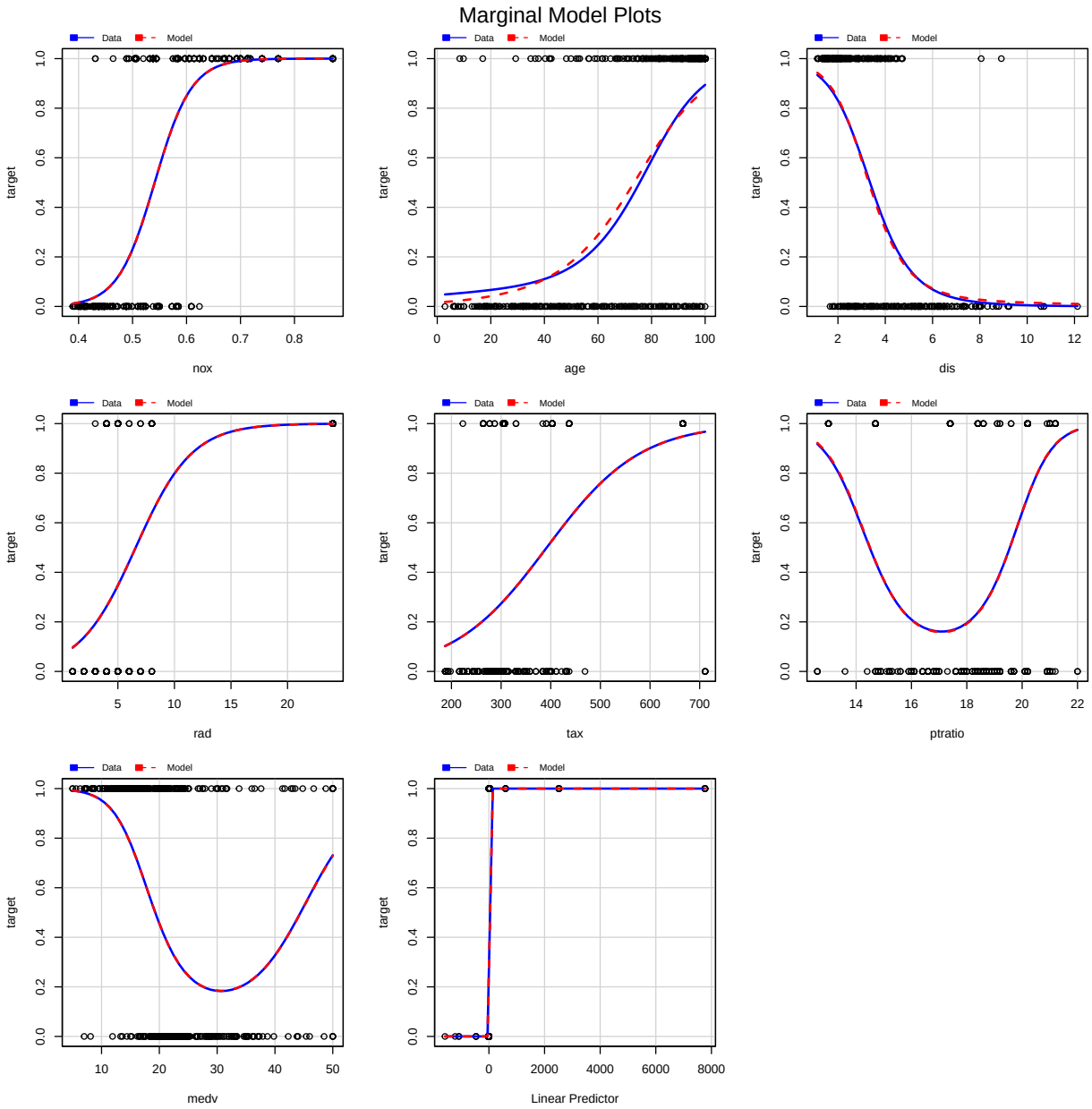
```
## [1] 0.9291845
```

Since they are correlated, we would not judge the two coefficients by the t-value. The Chi-square tests told us that these two variables are important to the model's performance. We will keep **chas** and **tax_chas** in our model.

Model Diagnostics

Now let's look at the marginal plots to see if our model is fitting well to the training data.

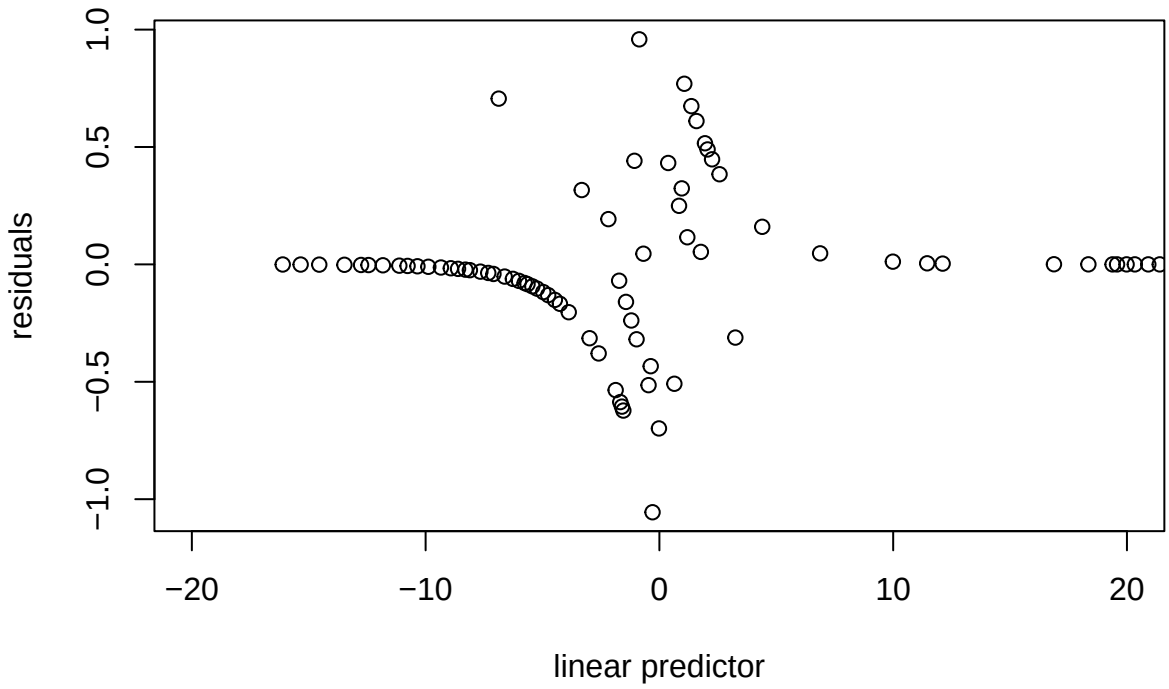
```
marginalModelPlots(model_chi_0.1,~nox+age+dis+rad+tax+ptratio+medv,layout =c(3,3))
```



All variables follow nearly the same as the nonparametric estimations. Our model is fitting well to the data.

Residual Plots

```
residual_df <- mutate(train_df, residuals=residuals(model_chi_0.1,type="deviance"),
                      linpred=predict(model_chi_0.1,type = "link"))
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))
```



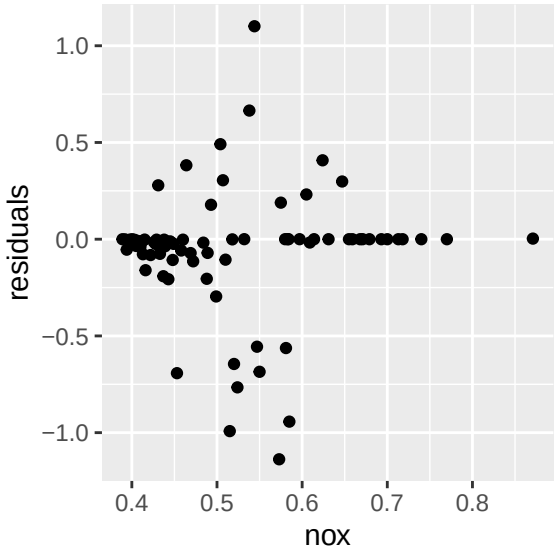
The deviance residual vs linear predictor plot shows that our model is valid.

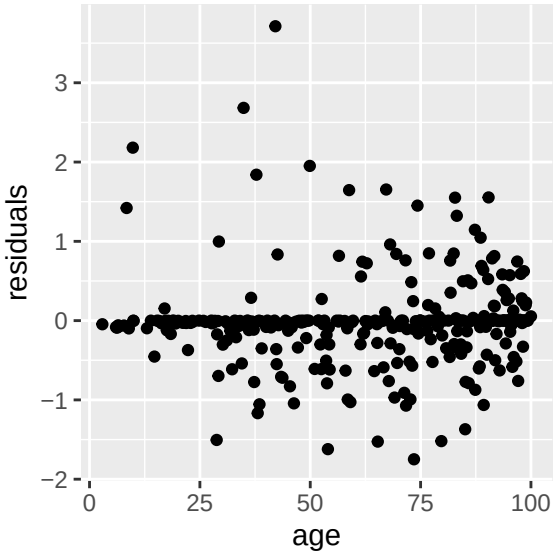
The model is producing accurate predictions at the two ends.

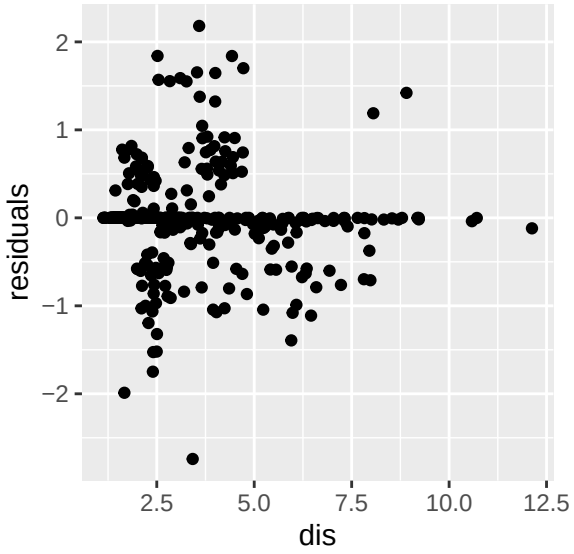
The errors around the match point 0 are independent and random.

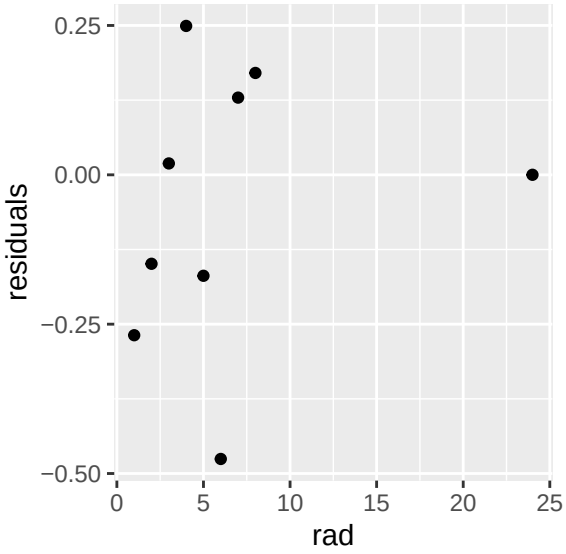
Let's also check the residual plots with individual predictors.

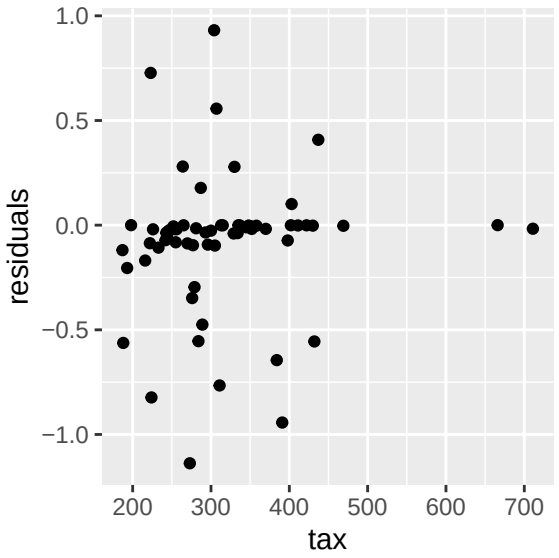
```
predictors <- c("nox","age","dis","rad","tax","ptratio","medv")

residual_df <- mutate(train_df, residuals=residuals(model_chi_0.1,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())
}
```

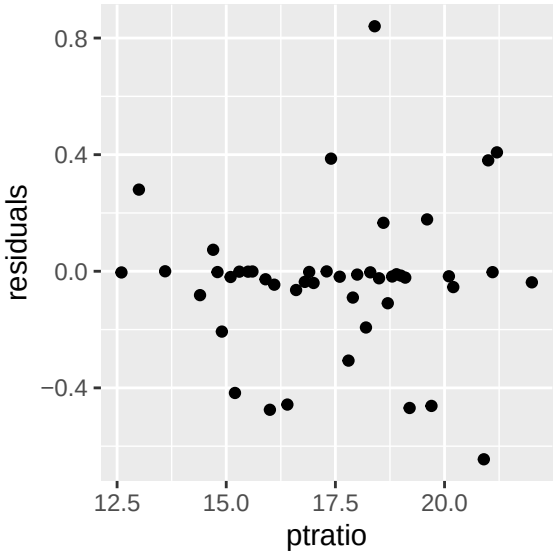

residuals
1.0
0.5
0.0
−0.5
−1.0
0.4
0.5
0.6
0.7
0.8
nox

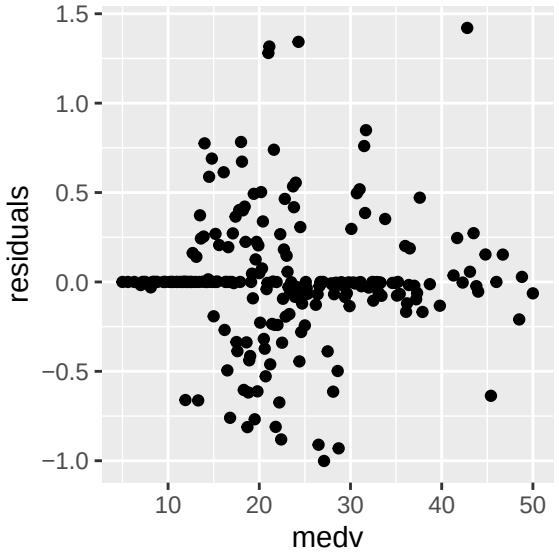

residuals
3
2
1
0
−1
−2
0
25
50
75
100
age


residuals
2
1
0
−1
−2
2.5
5.0
7.5
10.0
12.5
dis


residuals
0.25
0.00
−0.25
−0.50
0
5
10
15
20
25
rad



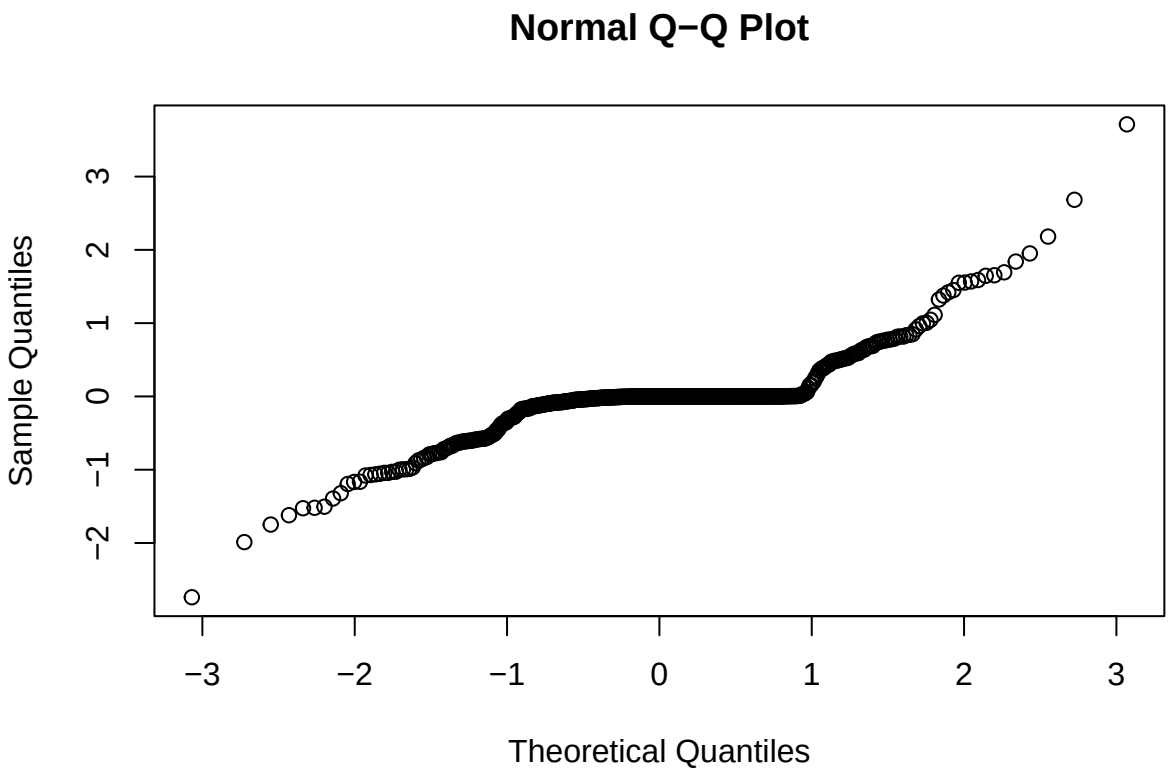




The residuals in each plots are centered at 0, mostly independent and with roughly the same variance, except a few outliers. We conclude that our model does not have notable violation against its validity. The residuals for *nox* and *dis* seems to be heteroscedastic. Given this is a logistic regression with binary data, this phenomena is acceptable.

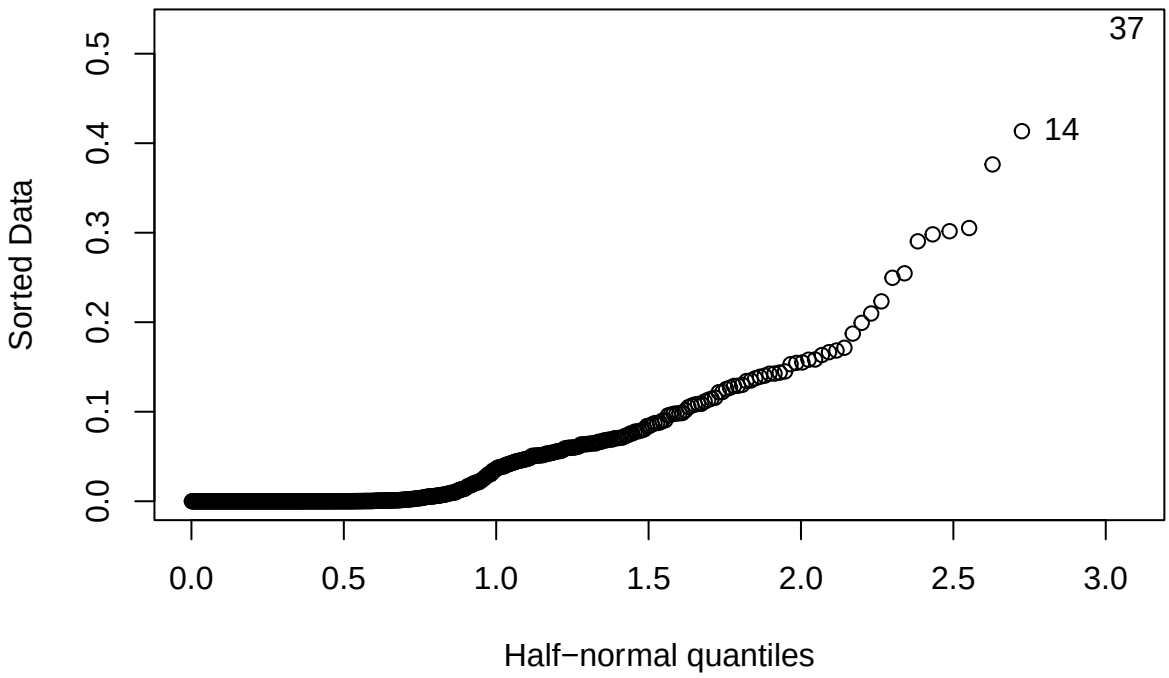
Q-Q Plot and half normal plot

```
qqnorm(residuals(model_chi_0.1))
```



The Q-Q plot seems to be fine given it's a logistic regression with binary data.

```
halfnorm(hatvalues(model_chi_0.1))
```



The half normal plots shows case 14 and 37 have high leverage.

By looking at the details of the cases, there is nothing extreme in the values.

```
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
```

Additionally, the predicted link values are close to 0, which confirmed they are not outliers. We would keep them in our model training.

```
predict(model_chi_0.1,train_df[c(14,37),], type="link")
```

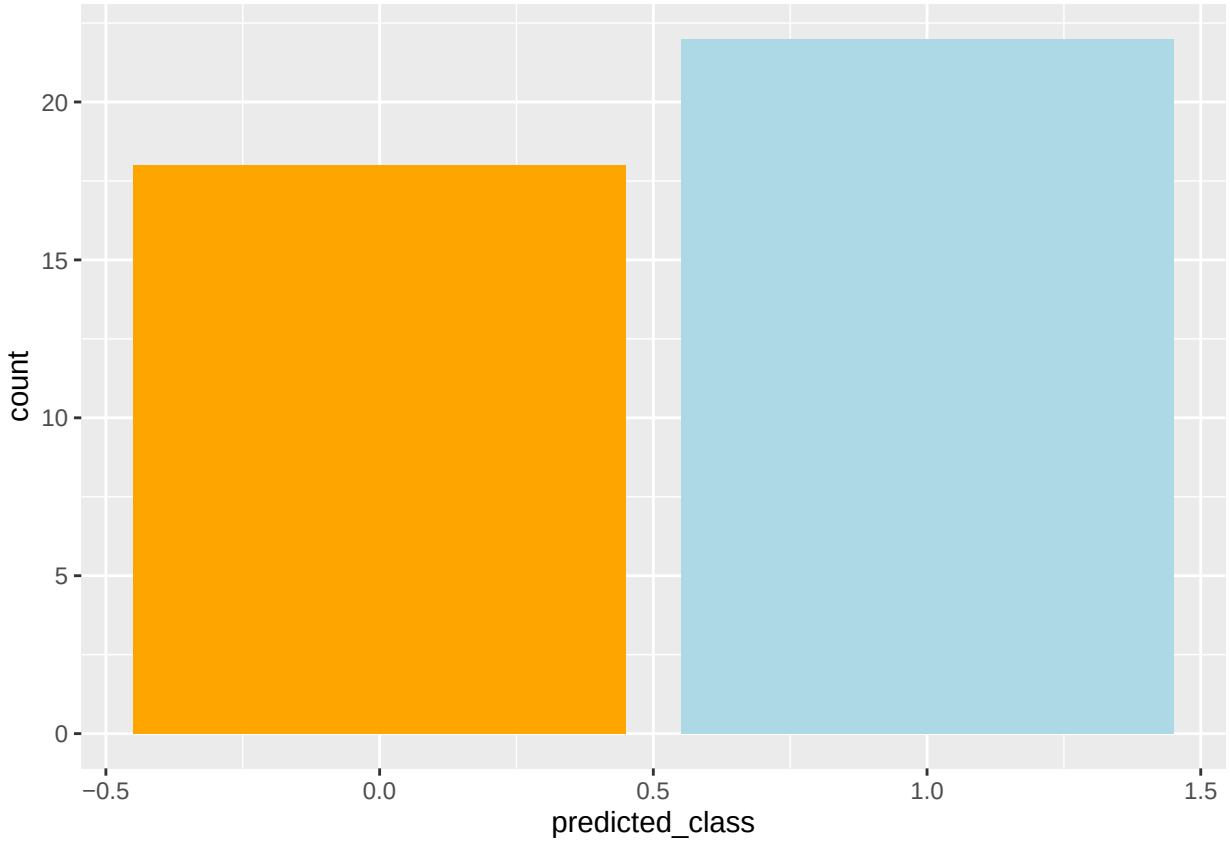
```
##         14         37
## -0.5557560 -0.8629056
```

Evaluation Data Prediction

Finally, let's see how our model will predict using the evaluation data set.

```
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_0.1,test_df, type = "response") >0.5,1,0)
ggplot(test_df, aes(predicted_class)) + geom_bar(fill=c("orange","lightblue"))
```



Both target = 0 and target = 1 are close to 50%, which is a very plausible outcomes.

This is the same as we expected since there are 50% of the cases above the median crime rate and 50% of the cases below the median crime rate.