

# D621 - Assignment 3

Marco Castro

2025-03-07

## DATA EXPLORATION

### Summary Stats

```
column_types <- sapply(df_training, class)
print(column_types)
```

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

The following three columns were imported as numerical but should be factors: - chas: binomial - rad: ordinal - target: binomial

```
# convert to factor
df_training <- df_training |>
  mutate(
    chas = as.factor(chas),
    rad = as.factor(rad),
    target = as.factor(target),
  )

numeric_cols <- c('zn', 'indus', 'nox', 'rm', 'age', 'dis', 'tax', 'ptratio', 'lstat', 'medv')

factor_cols <- c('chas', 'rad', 'target')

glimpse(df_training)
```

```
## Rows: 466
## Columns: 13
## $ zn      <dbl> 0, 0, 0, 30, 0, 0, 0, 0, 0, 80, 22, 0, 0, 22, 0, 0, 100, 20, 0~
## $ indus   <dbl> 19.58, 19.58, 18.10, 4.93, 2.46, 8.56, 18.10, 18.10, 5.19, 3.6~
## $ chas    <fct> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ nox     <dbl> 0.605, 0.871, 0.740, 0.428, 0.488, 0.520, 0.693, 0.693, 0.515, ~
## $ rm      <dbl> 7.929, 5.403, 6.485, 6.393, 7.155, 6.781, 5.453, 4.519, 6.316, ~
## $ age     <dbl> 96.2, 100.0, 100.0, 7.8, 92.2, 71.3, 100.0, 100.0, 38.1, 19.1, ~
## $ dis     <dbl> 2.0459, 1.3216, 1.9784, 7.0355, 2.7006, 2.8561, 1.4896, 1.6582~
## $ rad     <fct> 5, 5, 24, 6, 3, 5, 24, 24, 5, 1, 7, 5, 24, 7, 3, 3, 5, 5, 24, ~
## $ tax     <dbl> 403, 403, 666, 300, 193, 384, 666, 666, 224, 315, 330, 398, 66~
## $ ptratio <dbl> 14.7, 14.7, 20.2, 16.6, 17.8, 20.9, 20.2, 20.2, 20.2, 16.4, 19~
## $ lstat   <dbl> 3.70, 26.82, 18.85, 5.19, 4.82, 7.67, 30.59, 36.98, 5.68, 9.25~
## $ medv    <dbl> 50.0, 13.4, 15.4, 23.7, 37.9, 26.5, 5.0, 7.0, 22.2, 20.9, 24.8~
## $ target  <fct> 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, ~
```

A closer examination of the `rad` data shows that that our observations have a `rad` index value of 1-8 or 24 in this column. Below are the counts:

```
##
##  1  2  3  4  5  6  7  8 24
## 17 20 36 103 109 25 15 20 121
```

**Means** We can now calculate the summary statistics for the numeric parameters in our dataframe, including our mean, median, min/max, and standard deviations.

```
# only show summary stats for numeric values
for (param in numeric_cols) {
  cat("\nSummary for", param, ":\n")
  print(describe(df_training[[param]]))
}

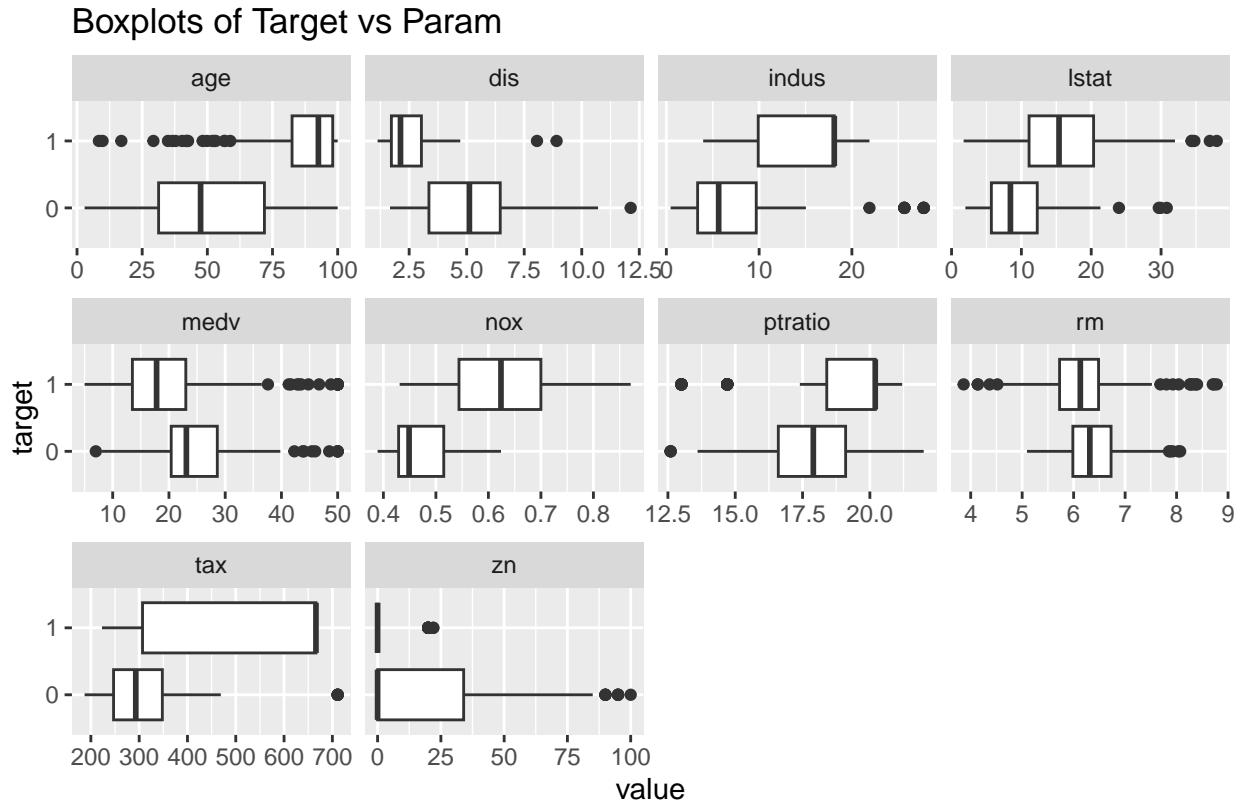
##
## Summary for zn :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
## X1    1 466 11.58 23.36      0   5.35  0  0 100   100 2.18    3.81 1.08
##
## Summary for indus :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
## X1    1 466 11.11 6.85   9.69   10.91 9.34 0.46 27.74 27.28 0.29   -1.24 0.32
##
## Summary for nox :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
## X1    1 466 0.55 0.12   0.54    0.54 0.13 0.39 0.87  0.48 0.75   -0.04 0.01
##
## Summary for rm :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
## X1    1 466 6.29 0.7   6.21    6.26 0.52 3.86 8.78  4.92 0.48    1.54 0.03
##
## Summary for age :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
## X1    1 466 68.37 28.32 77.15   70.96 30.02 2.9 100  97.1 -0.58   -1.01 1.31
##
## Summary for dis :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
## X1    1 466  3.8 2.11   3.19    3.54 1.91 1.13 12.13   11  1    0.47 0.1
##
## Summary for tax :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
## X1    1 466 409.5 167.9 334.5  401.51 104.52 187 711   524 0.66   -1.15 7.78
##
## Summary for ptratio :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
## X1    1 466 18.4 2.2   18.9   18.6 1.93 12.6  22   9.4 -0.75   -0.4 0.1
##
## Summary for lstat :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
## X1    1 466 12.63 7.1  11.35   11.88 7.07 1.73 37.97 36.24 0.91    0.5 0.33
##
## Summary for medv :
##   vars  n mean   sd median trimmed mad min max range skew kurtosis  se
```

```
## X1      1 466 22.59 9.24    21.2    21.63    6    5    50    45 1.08    1.37 0.43
```

## Plots of data

**Boxplots** Below is series of boxplots for all numeric parameters where target is our dependent variable.

```
plot_boxplot(df_training, by = "target", title="Boxplots of Target vs Param")
```



```
df_training_hi_crime <- df_training |>
  filter(target == 1) |>
  subset(select = -c(chas, rad, target))

df_training_lo_crime <- df_training |>
  filter(target == 0) |>
  subset(select = -c(chas, rad, target))

cat("\nIQR for High Crime Neighborhoods\n")
```

```
##
## IQR for High Crime Neighborhoods
```

```
summary(df_training_hi_crime)
```

```
##          zn          indus          nox          rm
##  Min.   : 0.000   Min.   : 3.97   Min.   :0.4310   Min.   :3.863
## 1st Qu.: 0.000   1st Qu.: 9.90   1st Qu.:0.5440   1st Qu.:5.727
## Median : 0.000   Median :18.10   Median :0.6240   Median :6.130
## Mean   : 1.328   Mean   :15.31   Mean   :0.6404   Mean   :6.181
## 3rd Qu.: 0.000   3rd Qu.:18.10   3rd Qu.:0.7000   3rd Qu.:6.484
## Max.   :22.000   Max.   :21.89   Max.   :0.8710   Max.   :8.780
```

```
##      age      dis      tax      ptratio
## Min.   : 8.4   Min.   :1.130  Min.   :223.0  Min.   :13.00
## 1st Qu.: 82.5  1st Qu.:1.728  1st Qu.:307.0  1st Qu.:18.40
## Median : 92.6  Median :2.125  Median :666.0  Median :20.20
## Mean   : 86.5  Mean   :2.471  Mean   :513.8  Mean   :18.96
## 3rd Qu.: 98.1  3rd Qu.:3.033  3rd Qu.:666.0  3rd Qu.:20.20
## Max.   :100.0  Max.   :8.907  Max.   :666.0  Max.   :21.20
##      lstat      medv
## Min.   : 1.73   Min.   : 5.00
## 1st Qu.:11.10   1st Qu.:13.50
## Median :15.39   Median :17.80
## Mean   :16.02   Mean   :20.05
## 3rd Qu.:20.34   3rd Qu.:23.00
## Max.   :37.97   Max.   :50.00
```

```
cat("\nIQR for Low Crime Neighborhoods\n")
```

```
##
## IQR for Low Crime Neighborhoods
```

```
summary(df_training_lo_crime)
```

```
##      zn      indus      nox      rm
## Min.   : 0.00   Min.   : 0.460  Min.   :0.3890  Min.   :5.093
## 1st Qu.: 0.00   1st Qu.: 3.370  1st Qu.:0.4290  1st Qu.:5.985
## Median : 0.00   Median : 5.640  Median :0.4490  Median :6.315
## Mean   : 21.48   Mean   : 7.039  Mean   :0.4711  Mean   :6.396
## 3rd Qu.: 34.00   3rd Qu.: 9.690  3rd Qu.:0.5150  3rd Qu.:6.727
## Max.   :100.00   Max.   :27.740  Max.   :0.6240  Max.   :8.069
##      age      dis      tax      ptratio
## Min.   : 2.90   Min.   : 1.669  Min.   :187.0  Min.   :12.60
## 1st Qu.: 31.30   1st Qu.: 3.360  1st Qu.:247.0  1st Qu.:16.60
## Median : 47.40   Median : 5.118  Median :293.0  Median :17.90
## Mean   : 50.84   Mean   : 5.076  Mean   :308.8  Mean   :17.86
## 3rd Qu.: 71.90   3rd Qu.: 6.458  3rd Qu.:348.0  3rd Qu.:19.10
## Max.   :100.00   Max.   :12.127  Max.   :711.0  Max.   :22.00
##      lstat      medv
## Min.   : 1.98   Min.   : 7.00
## 1st Qu.: 5.70   1st Qu.:20.40
## Median : 8.43   Median :23.10
## Mean   : 9.36   Mean   :25.04
## 3rd Qu.:12.27   3rd Qu.:28.60
## Max.   :30.81   Max.   :50.00
```

The boxplots show the distribution numerical parameters grouped by the dependent variable `target`. The plots are useful for getting a sense as to which parameters may be good predictors based on how different the parameter's IQRs are. Conversely, similar IQRs may provide insight into which may not add much information to our model. Based on these box plots, we see that the IQR for `rm` are very similar where `target` is 0 and 1 and should be flagged for potential removal of our plot. `ptratio` and `medv` have some overlap All other variables appear somewhat

Further, we see that param `zn` has a median value around zero, suggesting that few neighborhoods have residential areas zoned for large plots as shown below. We should also consider omitting this variable from our model down the line

```
count_zeros <- sum(df_training_hi_crime$zn == 0)
cat("\nAbove Median Crime Rate Neighborhoods have ", count_zeros, " rows with a value of 0 for param zn")
```

```
(count_zeros / nrow(df_training_hi_crime)), "%)\n")
```

```
##
```

```
## Above Median Crime Rate Neighborhoods have 214 rows with a value of 0 for param zn out of 229 obs
```

```
count_zeros <- sum(df_training_lo_crime$zn == 0)
```

```
cat("\nBelow Median Crime Rate Neighborhoods have ", count_zeros, " rows with a value of 0 for param zn\n",  
(count_zeros / nrow(df_training_lo_crime)), "%)\n")
```

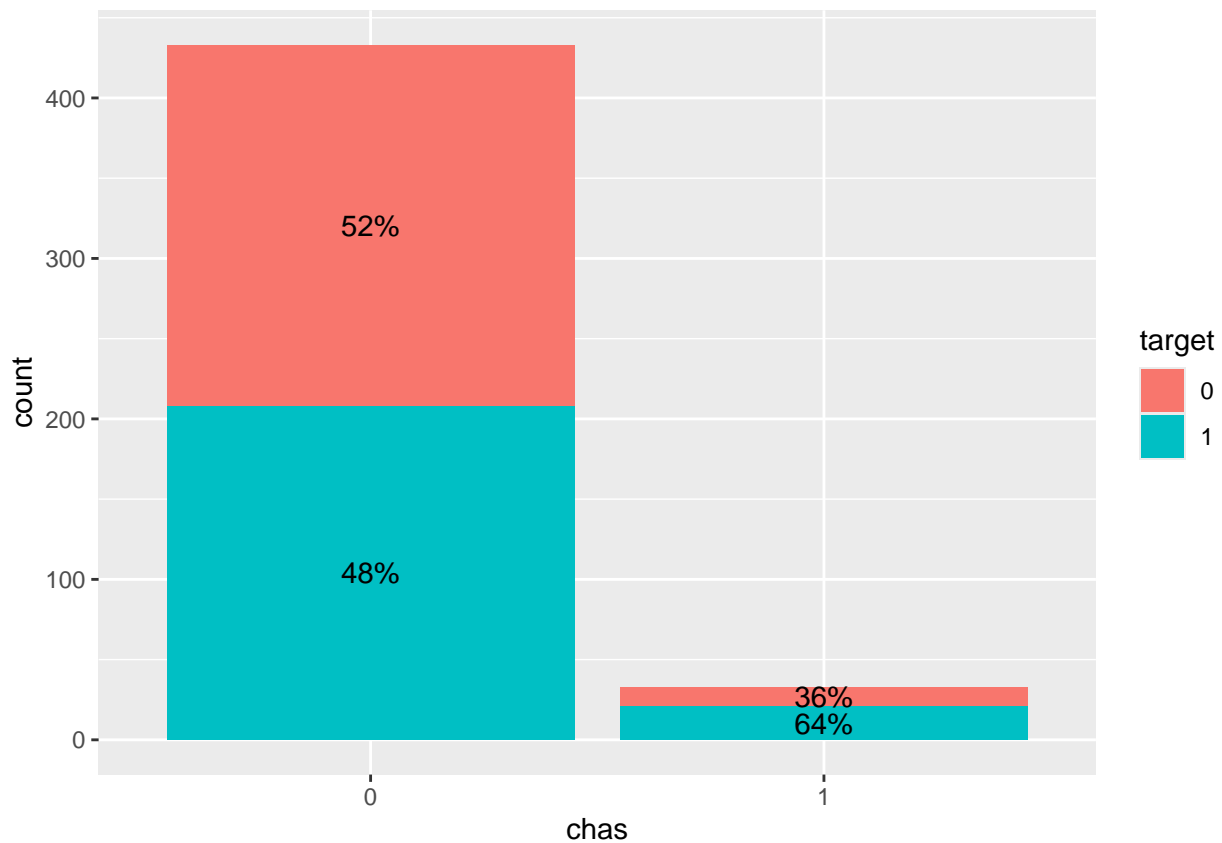
```
##
```

```
## Below Median Crime Rate Neighborhoods have 125 rows with a value of 0 for param zn out of 237 obs
```

### *Categorical Variables*

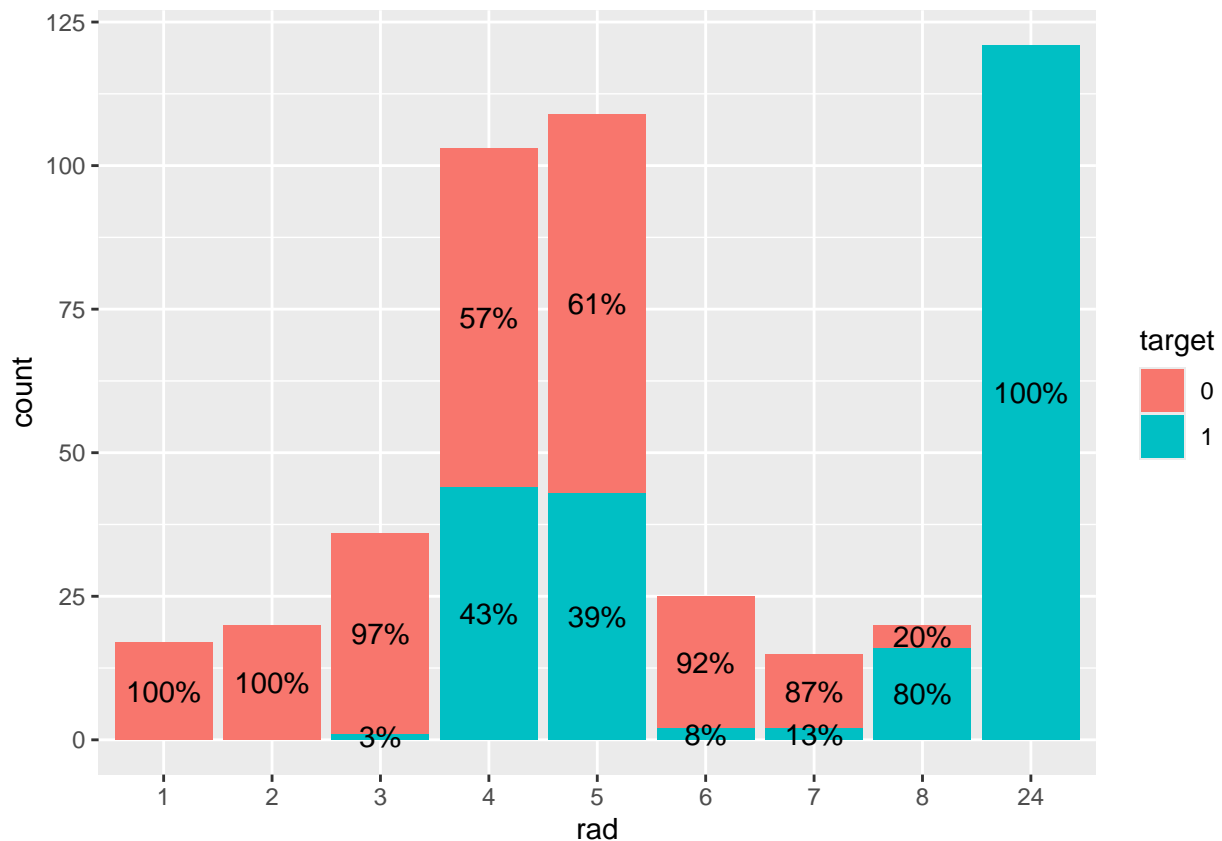
For our categorical variables, we can use bargraphs to get a sense of the parameter's impact on **target**.

```
df_training |>  
  group_by(  
    target, chas  
  ) |>  
  dplyr::summarise(  
    count = n()  
  ) |>  
  ungroup() |>  
  group_by(chas) |>  
  mutate(  
    percent = 100 * count / sum(count),  
    label = paste0(round(percent), "%")  
  ) |>  
  ggplot() +  
  aes(x = chas, y = count, label = label, fill=target) +  
  geom_col() +  
  geom_text(position = position_stack(0.5))
```



The bargraph for `chas` shows fairly equal values for 0 and 1 across the `chas` values. This suggests that the variable may have low impact on our model and we should consider removing it.

```
### rad
df_training |>
  group_by(
    target, rad
  ) |>
  dplyr::summarise(
    count = n()
  ) |>
  ungroup() |>
  group_by(rad) |>
  mutate(
    percent = 100 * count / sum(count),
    label = paste0(round(percent), "%")
  ) |>
  ggplot() +
  aes(x = rad, y = count, label = label, fill=target) +
  geom_col() +
  geom_text(position = position_stack(0.5))
```



The bargraphs for **rad** are somewhat more revealing. They suggest a strong relationship between low **rad** index values of 1-3 and below median crime rate, while an index value of 24 (the highest **rad** index) has a strong relationship with above median crime rate.

**Pairs** Using the **pair** function, we can print scatterplots comparing each of the variables to the others.

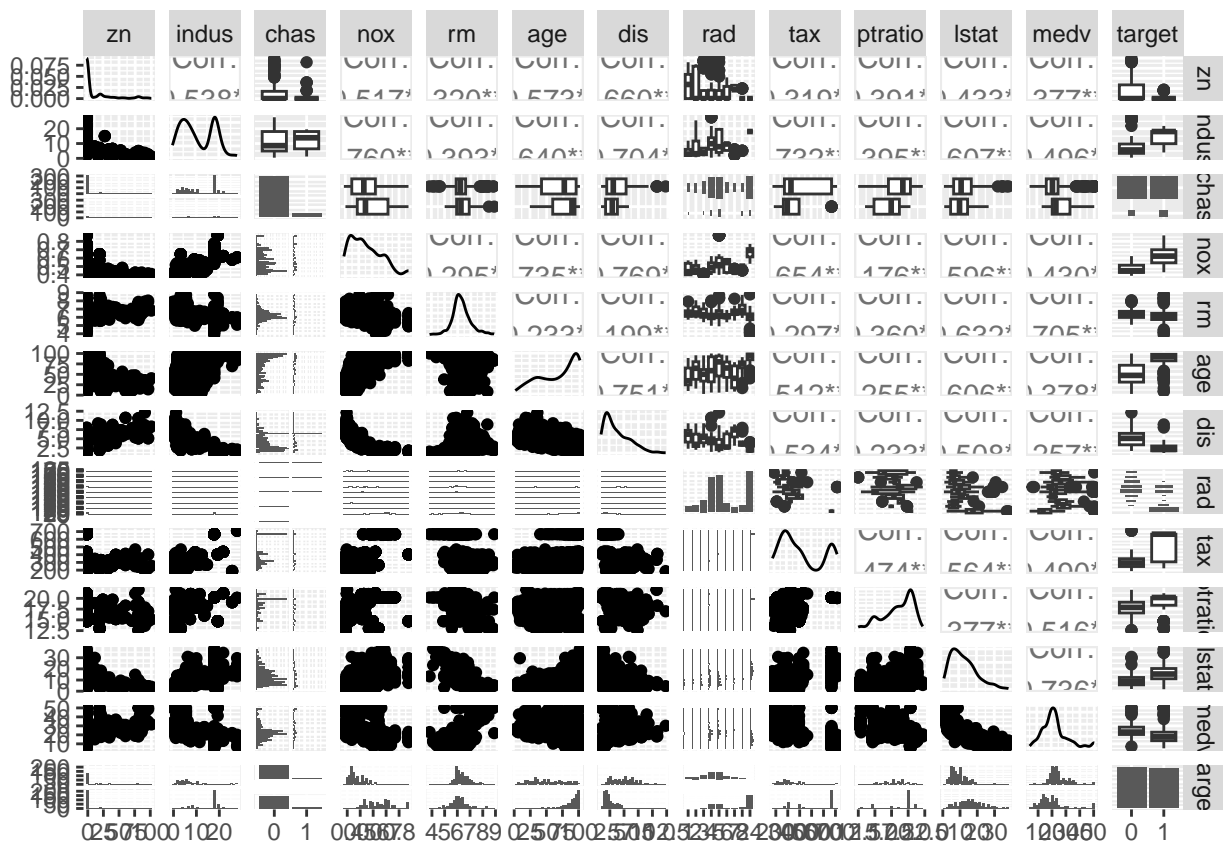
```
png("scatterplot_matrix.png", width = 800, height = 800)
```

```
pairs(df_training, main="")
dev.off()
```

```
## pdf
## 2
```

GGpairs plots take this a step further and show normal distribution and boxplots to get a fuller sense of how the data parameters relate to one another.

```
ggpairs(df_training)
```



## Missing data

The training set contains no missing values.

```
introduce(df_training)
```

```
## # A tibble: 1 x 9
##   rows columns discrete_columns continuous_columns all_missing_columns
##   <int>   <int>           <int>             <int>               <int>
## 1    466     13             3              10                 0
## # i 4 more variables: total_missing_values <int>, complete_rows <int>,
## #   total_observations <int>, memory_usage <dbl>
```

```
missing_values_count <- sapply(data, function(x) sum(is.na(x)))
print(missing_values_count)
```

```
##           ...      list  package  lib.loc  verbose    envir overwrite
##           0         0       0        0       0        0         0         0
```

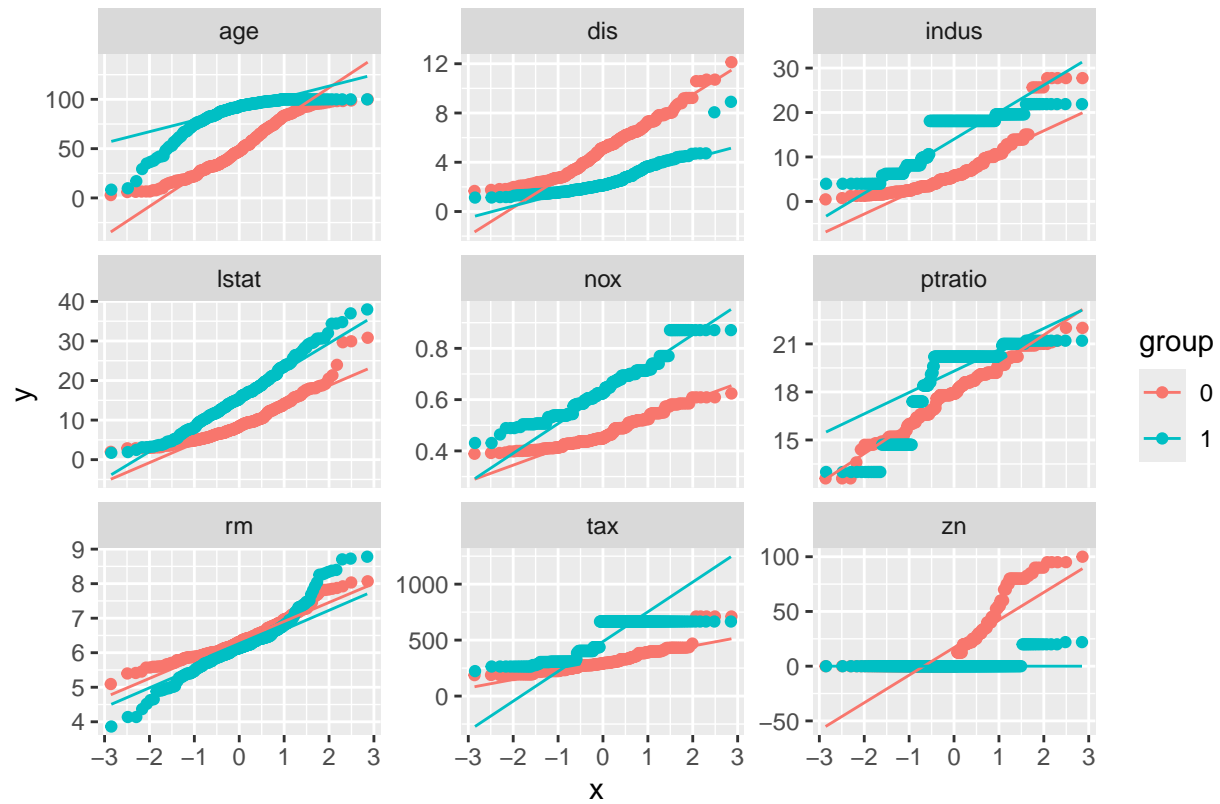
## Distribution

Scatterplots of  $y=target$  plotted against each of the parameters confirm that the dependent variable is binomial. Therefore, linear regression is not the best fit for this data and we should explore logistic regression such as logit and probit.

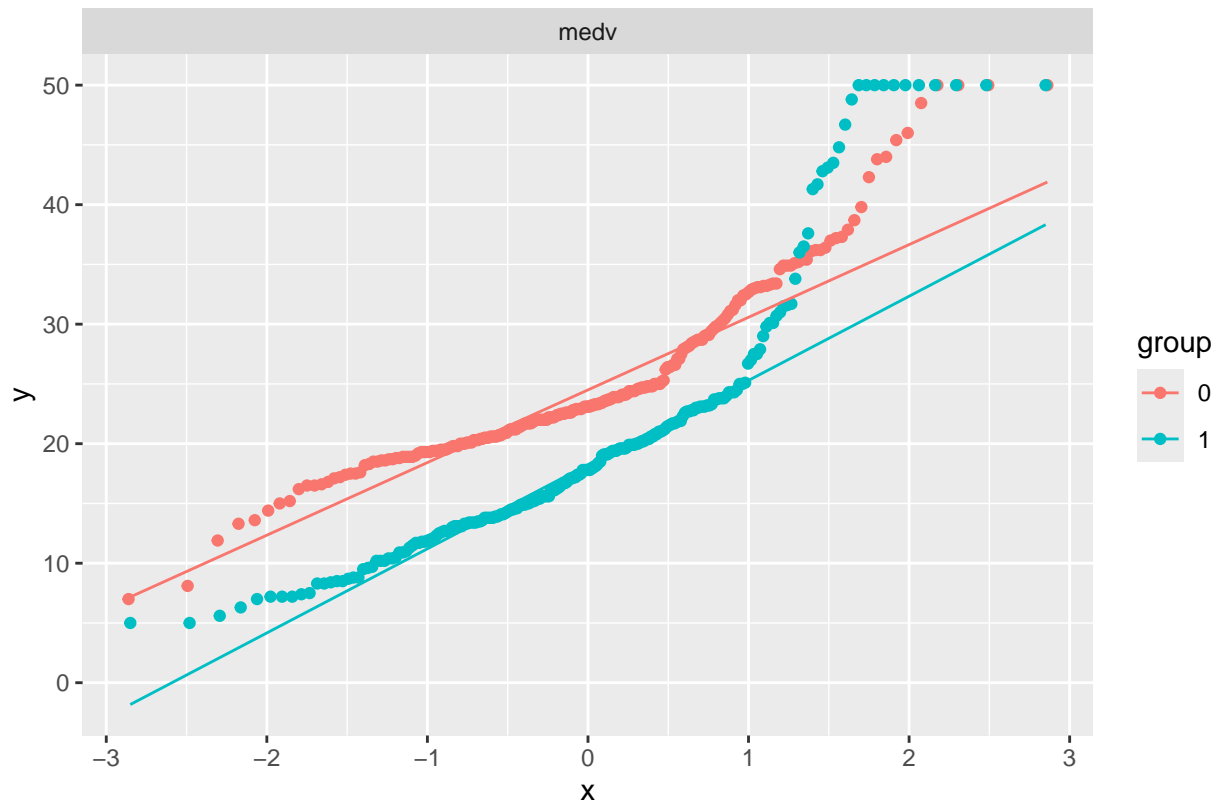
```
# scatter plot doesn't show much
# plot_scatterplot(df_training, by = "target")
# plot_qq(df_training, sampled_rows = 1000L)
```



```
plot_qq(df_training, by="target", sampled_rows = 1000L)
```



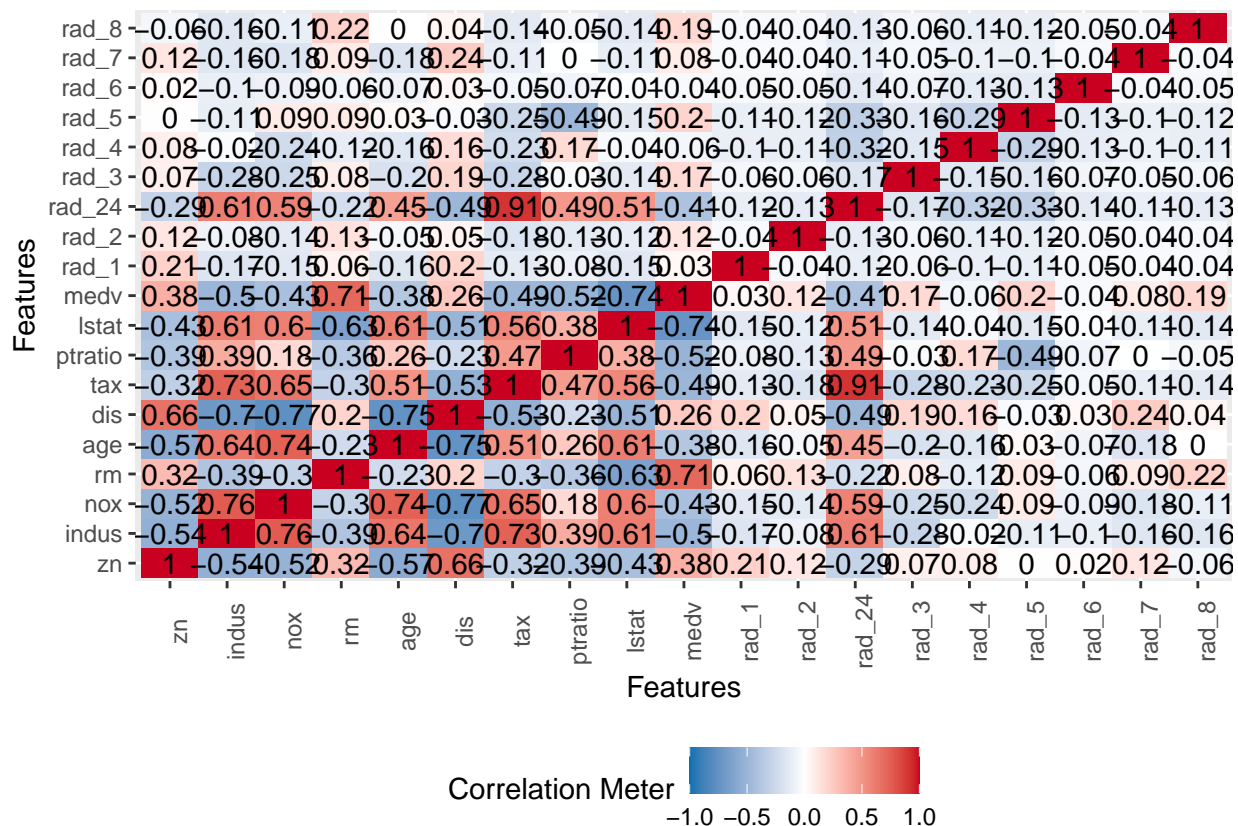
Page 1



Page 2

## Correlation

```
df_training |>
  subset(select=-c(target, chas)) |>
  plot_correlation(type = "all")
```



## DATA PREPARATION

### Fixing missing values

Luckily, there are no missing values in the training set.

### Transforming data by bucketing and combining variables

The variable `rad` contains an ordinal factor that represents an index of accessibility to radial highways with values ranging from 1-24. A count of the `rad` values reveals that the `rad` column contains only values 1-8 and 24. This data set does not include any rows with a `rad` value of 9-23.

Since this column contains values for an index value where 1 is assigned to neighborhoods with the poorest accessibility to a highway and 24 is assigned to neighborhoods with the most accessibility, we can simplify our variables by binning our `rad` values. Here we are using quantiles to bin the values into three buckets of nearly equal sizes for low, moderate and high accessibility. This method ensures a more balanced distribution of rows across the bins over using equal sized bins (1-8, 9-16, 17-24). This is especially useful when the data is not uniformly distributed across the range such as in our case where we do not have any `rad` values of 1-23.

```
rad_counts

##
##  1  2  3  4  5  6  7  8  24
## 17 20 36 103 109 25 15 20 121

quantile_breaks <- quantile(as.numeric(df_training$rad), probs = c(0, 1/3, 2/3, 1))

df_training$radq <- cut(as.numeric(df_training$rad),
                        breaks = quantile_breaks,
                        labels = c('_low', '_mid', '_hi'),
```

```

        include.lowest = TRUE,
        right = TRUE)

```

```
table(df_training$radq)
```

```
##
## _low _mid _hi
## 176 149 141
```

While the `glm` function should automatically perform one-hot encoding to factors, we should consider one-hot encoding on the `rad_quantile` parameter to perform other operations, such as calculating correlation using the spearman test.

We will drop one of the one-hot encoded params as the presence of this additional param will result in correlation issues down the line. `radq_mid` was selected, as it seemed to have the most mixed results in our plots above.

```

# one-hot encode rad values
rad_one_hot <- model.matrix(~ radq - 1, data = df_training)

# combine new columns
df_training_one_hot <- cbind(df_training[, !names(df_training) %in% "rad"], rad_one_hot) |>
  subset(select=c(radq, radq_mid))

glimpse(df_training_one_hot)

```

```

## Rows: 466
## Columns: 14
## $ zn      <dbl> 0, 0, 0, 30, 0, 0, 0, 0, 0, 80, 22, 0, 0, 22, 0, 0, 100, 20, ~
## $ indus   <dbl> 19.58, 19.58, 18.10, 4.93, 2.46, 8.56, 18.10, 18.10, 5.19, 3.~
## $ chas    <fct> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ nox     <dbl> 0.605, 0.871, 0.740, 0.428, 0.488, 0.520, 0.693, 0.693, 0.515~
## $ rm      <dbl> 7.929, 5.403, 6.485, 6.393, 7.155, 6.781, 5.453, 4.519, 6.316~
## $ age     <dbl> 96.2, 100.0, 100.0, 7.8, 92.2, 71.3, 100.0, 100.0, 38.1, 19.1~
## $ dis     <dbl> 2.0459, 1.3216, 1.9784, 7.0355, 2.7006, 2.8561, 1.4896, 1.658~
## $ tax     <dbl> 403, 403, 666, 300, 193, 384, 666, 666, 224, 315, 330, 398, 6~
## $ ptratio <dbl> 14.7, 14.7, 20.2, 16.6, 17.8, 20.9, 20.2, 20.2, 20.2, 16.4, 1~
## $ lstat   <dbl> 3.70, 26.82, 18.85, 5.19, 4.82, 7.67, 30.59, 36.98, 5.68, 9.2~
## $ medv    <dbl> 50.0, 13.4, 15.4, 23.7, 37.9, 26.5, 5.0, 7.0, 22.2, 20.9, 24.~
## $ target  <fct> 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0~
## $ radq_low <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1~
## $ radq_hi  <dbl> 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0~

```

We will use the one-hot encoded dataframe to diagnose a preliminary model with all of the predictors.

```

model_full <- glm(target ~ ., binomial(link = "logit"), data=df_training_one_hot)
summary(model_full)

```

```

##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##      data = df_training_one_hot)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.736e+01  6.617e+00 -5.646 1.65e-08 ***
## zn          -8.026e-02  4.105e-02 -1.955 0.050551 .

```

```
## indus      -1.724e-01  4.986e-02  -3.457 0.000547 ***
## chas1      1.179e+00  8.097e-01   1.456 0.145311
## nox       5.946e+01  9.038e+00   6.579 4.74e-11 ***
## rm       -9.564e-01  6.992e-01  -1.368 0.171345
## age       2.030e-02  1.322e-02   1.536 0.124585
## dis       8.131e-01  2.456e-01   3.310 0.000931 ***
## tax       9.619e-04  2.291e-03   0.420 0.674583
## ptratio   1.190e-01  1.333e-01   0.893 0.372038
## lstat     5.447e-02  5.231e-02   1.041 0.297705
## medv     1.760e-01  5.907e-02   2.980 0.002887 **
## radq_low  1.563e+00  5.254e-01   2.975 0.002931 **
## radq_hi   5.118e+00  9.416e-01   5.436 5.46e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 180.97  on 452  degrees of freedom
## AIC: 208.97
##
## Number of Fisher Scoring iterations: 8
```

Reviewing the summary statistics for full model indicates that the variable `indus`, `nox`, `dis`, and `radq_hi` has very strong statistical significance. Two additional variables, `medv`, `dis` and `radq_mid`, have high statistical significance while `zn` has weak statistical significance. `chas1`, `rm`, `age`, `tax`, `ptratio` and `lstat` have weak statistical significance values.

**Note:** had we one-hot encoded all of the values for `rad` instead of binning them first, all `rad` params would have very weak statistical significance, as their p-values are nearly 1.0.

## Multicollinearity

To test if correlation exists between the dependent and independent variables, we used a Pearson's Correlation test. The function below loops through each of our columns and prints out the correlation of the dependent variable `target` with each of the predictors. For predictors where Pearson's Correlation coefficient is close to zero, we can determine that collinearity does not exist.

```
# is above .7 would be too highly correlated
cor_results <- data.frame(name = character(0), value = numeric(0))
for (param in colnames(df_training_one_hot)) {
  cat("\nPearson Test score for", param, ":\n")
  x <- as.numeric(df_training_one_hot$target)
  y <- as.numeric(df_training_one_hot[[param]])
  pears <- cor.test(x, y, method = "pearson")
  print(pears)
  # calc pearson cor value only
  cor_object <- data.frame(name = param, value = cor(x, y))
  assign("cor_results", rbind(cor_results, cor_object), envir = .GlobalEnv)
}
```

```
##
## Pearson Test score for zn :
##
## Pearson's product-moment correlation
##
```

```

## data:  x and y
## t = -10.309, df = 464, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.5028019 -0.3547564
## sample estimates:
##      cor
## -0.4316818
##
##
## Pearson Test score for indus :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = 16.361, df = 464, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5438976 0.6594549
## sample estimates:
##      cor
## 0.6048507
##
##
## Pearson Test score for chas :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = 1.7297, df = 464, p-value = 0.08435
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.01087336 0.16964461
## sample estimates:
##      cor
## 0.08004187
##
##
## Pearson Test score for nox :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = 22.748, df = 464, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6801291 0.7663936
## sample estimates:
##      cor
## 0.7261062
##
##
## Pearson Test score for rm :
##

```

```

## Pearson's product-moment correlation
##
## data:  x and y
## t = -3.325, df = 464, p-value = 0.0009542
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.24006288 -0.06258443
## sample estimates:
##      cor
## -0.1525533
##
##
## Pearson Test score for age :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = 17.479, df = 464, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5720099 0.6819122
## sample estimates:
##      cor
## 0.6301062
##
##
## Pearson Test score for dis :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = -16.963, df = 464, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.6717579 -0.5592666
## sample estimates:
##      cor
## -0.6186731
##
##
## Pearson Test score for tax :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = 16.631, df = 464, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5508558 0.6650327
## sample estimates:
##      cor
## 0.6111133
##
##

```

```

## Pearson Test score for ptratio :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = 5.5819, df = 464, p-value = 4.053e-08
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1637438 0.3340729
## sample estimates:
##      cor
## 0.2508489
##
##
## Pearson Test score for lstat :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = 11.443, df = 464, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3951287 0.5370764
## sample estimates:
##      cor
## 0.469127
##
##
## Pearson Test score for medv :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = -6.0536, df = 464, p-value = 2.925e-09
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.3527185 -0.1842424
## sample estimates:
##      cor
## -0.2705507
##
##
## Pearson Test score for target :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = Inf, df = 464, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  1 1
## sample estimates:
## cor
## 1

```

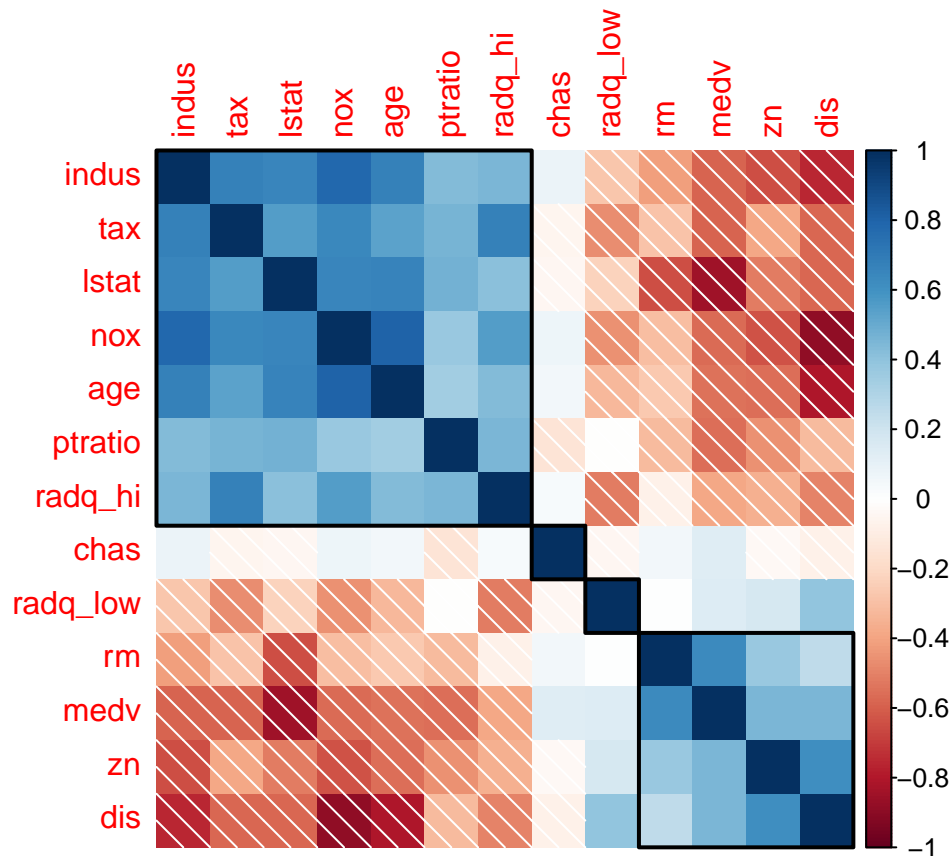


```
##
##
## Pearson Test score for radq_low :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = -8.5077, df = 464, p-value = 2.466e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4433864 -0.2860542
## sample estimates:
##      cor
## -0.3673453
##
##
## Pearson Test score for radq_hi :
##
## Pearson's product-moment correlation
##
## data:  x and y
## t = 17.599, df = 464, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5749042 0.6842126
## sample estimates:
##      cor
## 0.6326995
```

```
print(cor_results)
```

```
##      name      value
## 1      zn -0.43168176
## 2    indus  0.60485074
## 3     chas  0.08004187
## 4      nox  0.72610622
## 5       rm -0.15255334
## 6      age  0.63010625
## 7      dis -0.61867312
## 8      tax  0.61111331
## 9  ptratio  0.25084892
## 10    lstat  0.46912702
## 11    medv -0.27055071
## 12   target  1.00000000
## 13 radq_low -0.36734528
## 14 radq_hi  0.63269952
```

**Correlation Clusters** Next we can visualize the correlations in clusters.



Using “hclust”, our corplot shows four distinct groups, each with strong correlation between the parameters within each group. This suggests that we may want to select specific parameters from within these groups or conduct principal component analysis on each of these groups.

```
car::vif(model_full) |> sort()
```

### Variance Inflation Factor

```
##      chas radq_low      zn radq_hi      age      tax ptratio  lstat
## 1.208878 1.939100 1.979366 2.126242 2.146346 2.205786 2.314164 2.561351
##      indus      dis      rm      nox      medv
## 3.388295 4.138425 4.732821 5.329898 5.983892
```

A VIF test suggests that we should remove nox and medv.

### Presence of Outliers

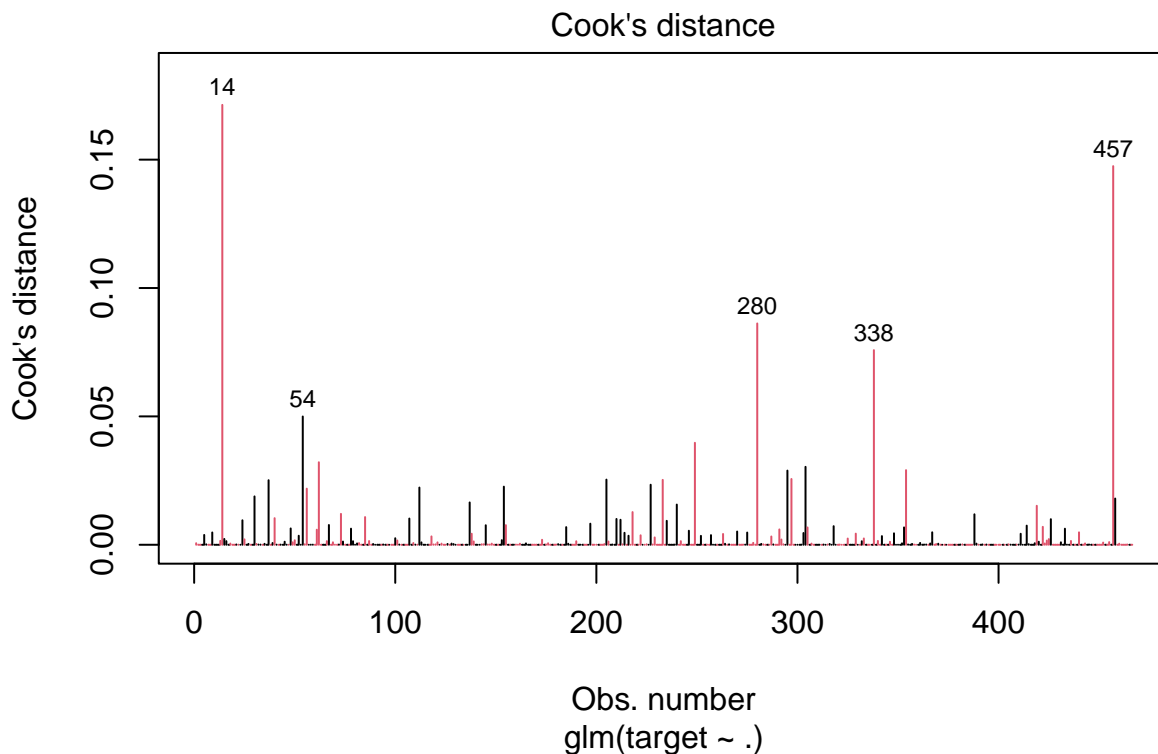
We can examine our diagnostic plots to find potential outliers and leverage. First we will examine the Cook’s Distance and Cook’s Distance vs Leverage plots. Cook’s Distance measures the influence of an observation on the fitted values of the model.

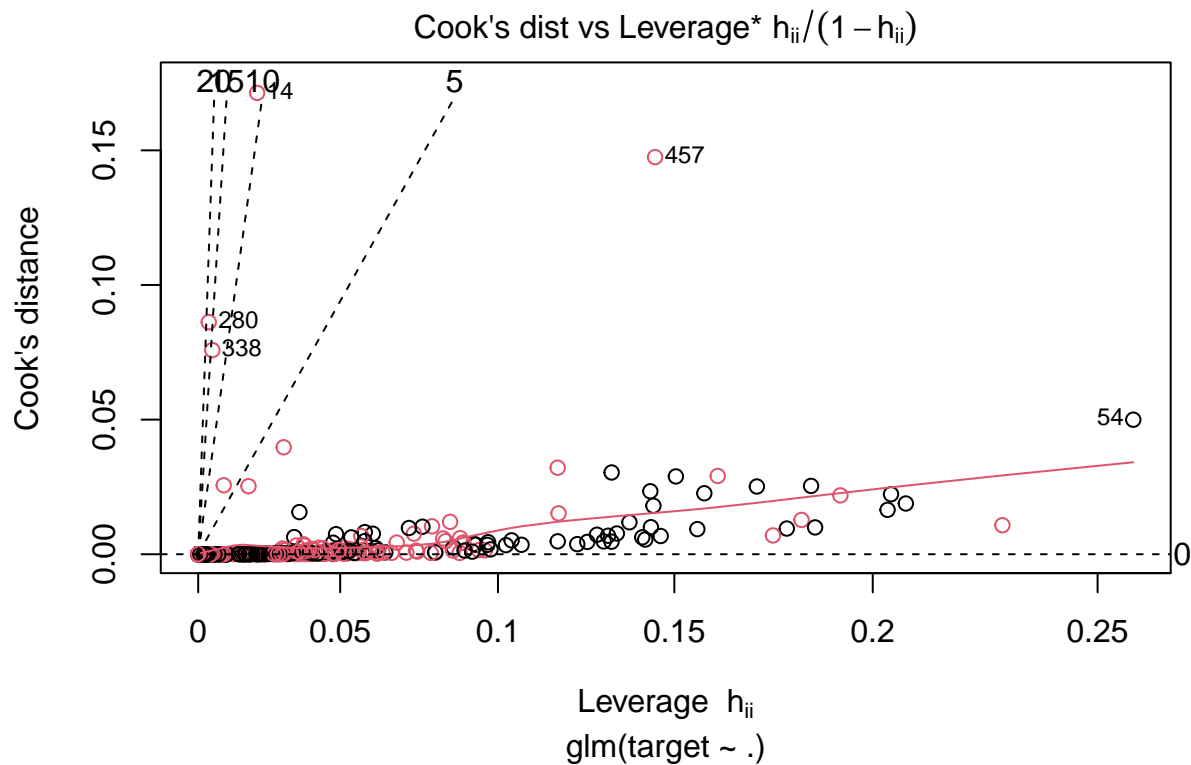
```
## Rows: 5
## Columns: 21
## $ index      <int> 14, 457, 280, 338, 54
## $ target     <chr> "Hi", "Hi", "Hi", "Hi", "Lo"
## $ zn         <dbl> 22, 0, 22, 20, 0
## $ indus      <dbl> 5.86, 10.59, 5.86, 6.96, 1.89
## $ chas       <fct> 0, 0, 0, 0, 0
```

```
## $ nox      <dbl> 0.431, 0.489, 0.431, 0.464, 0.518
## $ rm       <dbl> 8.259, 5.412, 6.108, 5.856, 6.540
## $ age      <dbl> 8.4, 9.8, 34.9, 42.1, 59.7
## $ dis      <dbl> 8.9067, 3.5875, 8.0555, 4.4290, 6.2669
## $ tax      <dbl> 330, 277, 330, 223, 422
## $ ptratio  <dbl> 19.1, 18.6, 19.1, 18.6, 15.9
## $ lstat    <dbl> 3.54, 29.55, 9.16, 13.00, 8.65
## $ medv     <dbl> 42.8, 23.7, 24.3, 21.1, 16.5
## $ radq_low <dbl> 0, 1, 0, 1, 1
## $ radq_hi  <dbl> 0, 0, 0, 0, 0
## $ .fitted  <dbl> -4.6773398, -2.3444828, -5.7239448, -5.3060581, 0.4048362
## $ .resid   <dbl> 3.061568, 2.207286, 3.384437, 3.259143, -1.353450
## $ .hat     <dbl> 0.021373959, 0.144810967, 0.003911828, 0.005210891, 0.25736~
## $ .sigma   <dbl> 0.6164625, 0.6234026, 0.6129971, 0.6144815, 0.6291214
## $ .cooksd  <dbl> 0.17134297, 0.14748407, 0.08620525, 0.07580775, 0.04996881
## $ .std.resid <dbl> 3.094821, 2.386863, 3.391076, 3.267668, -1.570564
```

The calculation above shows that points 14, 457, 280, 338, and 54 have the highest Cook's distance values (ordered from highest to lowest) and should be investigated as potential outliers.

```
par(mar = c(5, 4, 4, 2) + 0.1)
plot(model_full, which = c(4, 6), col=df_training_one_hot$target, id.n = 5)
```



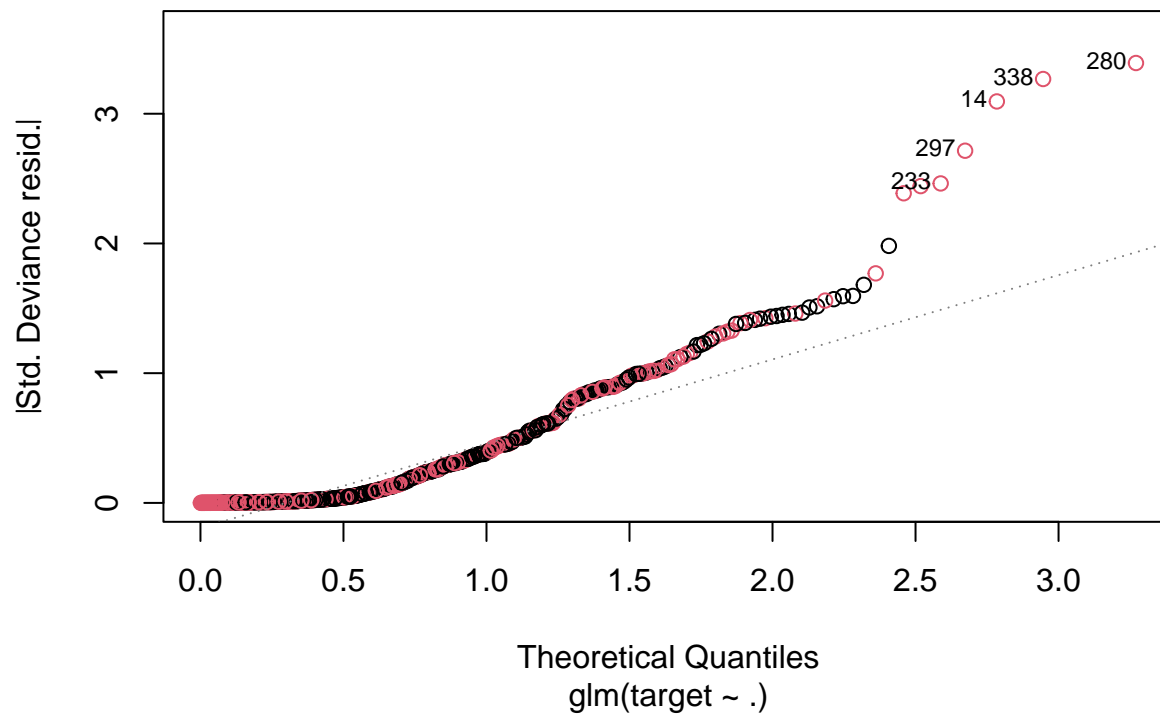
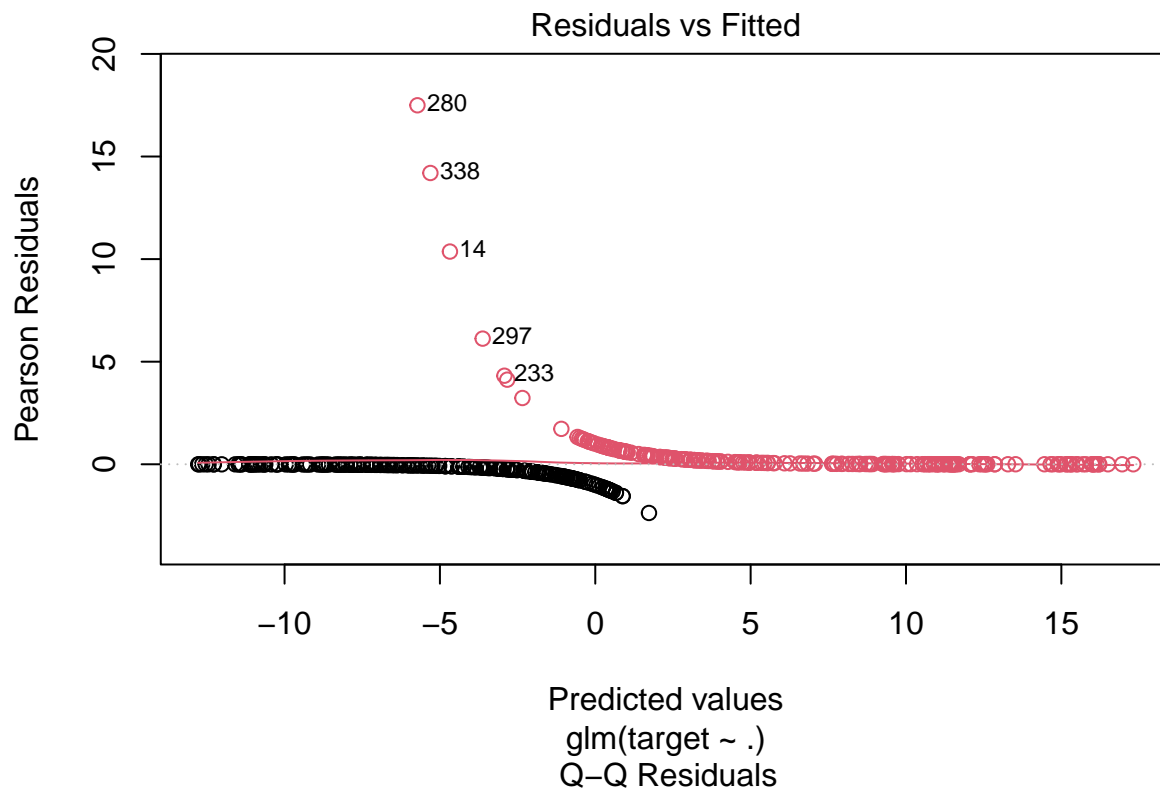


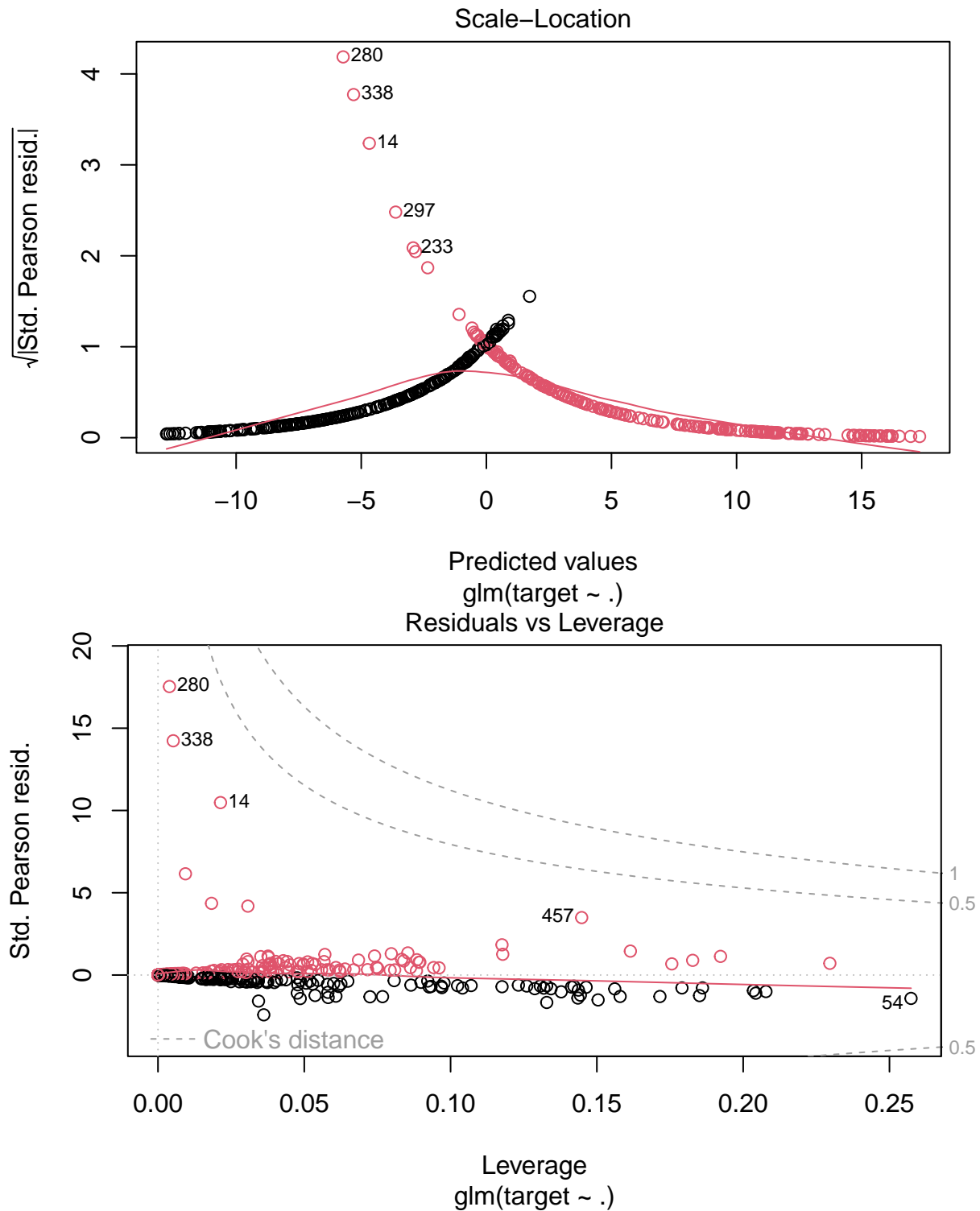
We see on the Cook's dist vs Leverage plot that points 280 and 338 may have very high leverage on our model, followed by point 14. Point 457 also stand out and should be investigated but appears to have less leverage.

```
# print influential points using cooks-distance
cooksd <- cooks.distance(model_full)
influential <- which(cooksd > (4 / length(cooksd)))
print(influential)
```

```
## 14 24 30 37 40 54 56 62 73 85 107 112 137 154 205 210 212 218 227 233
## 14 24 30 37 40 54 56 62 73 85 107 112 137 154 205 210 212 218 227 233
## 235 240 249 280 295 297 304 338 354 388 419 426 457 458
## 235 240 249 280 295 297 304 338 354 388 419 426 457 458
```

The formula above is used to identify influential points defined as points whose Cook's Distance value is greater than  $4 / \text{length of cooksd}$ . This contains all three points (280, 338, and 14) as being influential.



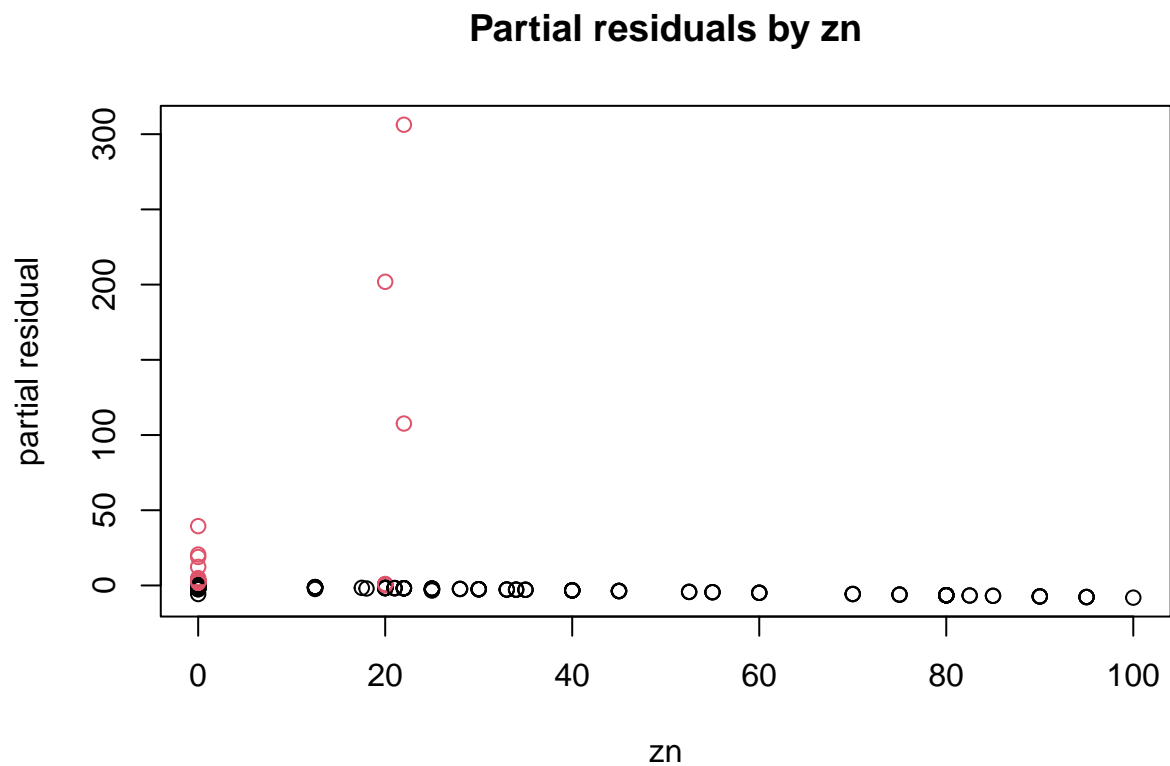


Our residual vs fitted, QQ, Scale-Location and Residual vs Leverage plots all confirm that points 280, 338, and 14 should be investigated and could be outliers with high influence. Points 457 appear to have less leverage and does not stand out in these plots.

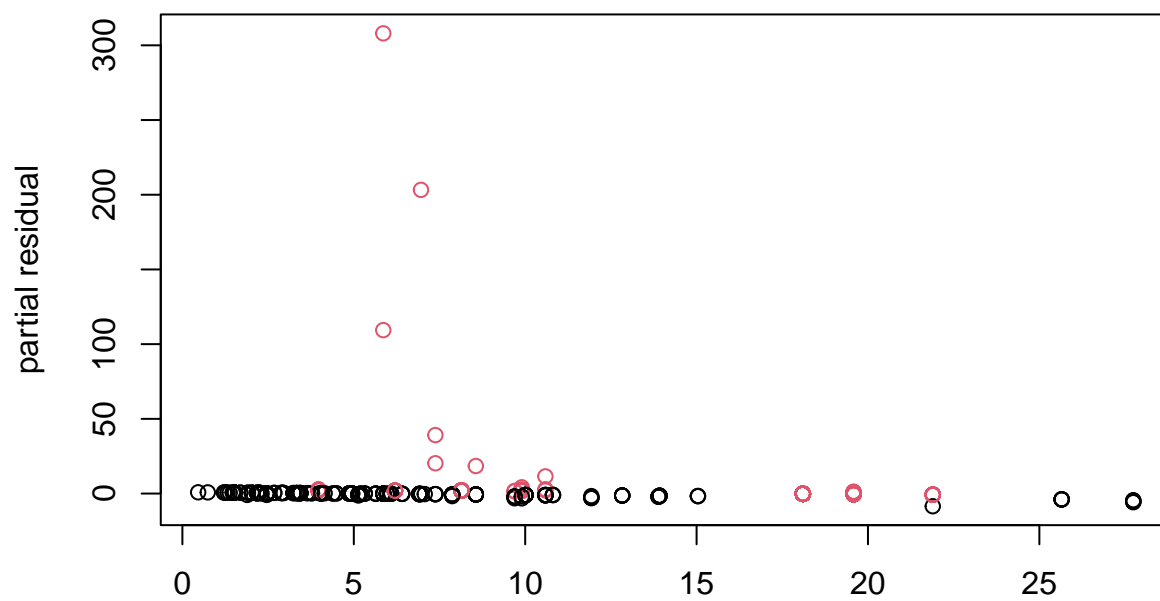
Below is the output for the three points identified as potential outliers in our diagnostic plots. A quick review of the data doesn't reveal anything that stands out as being out of the ordinary.

*Partial residual plots*

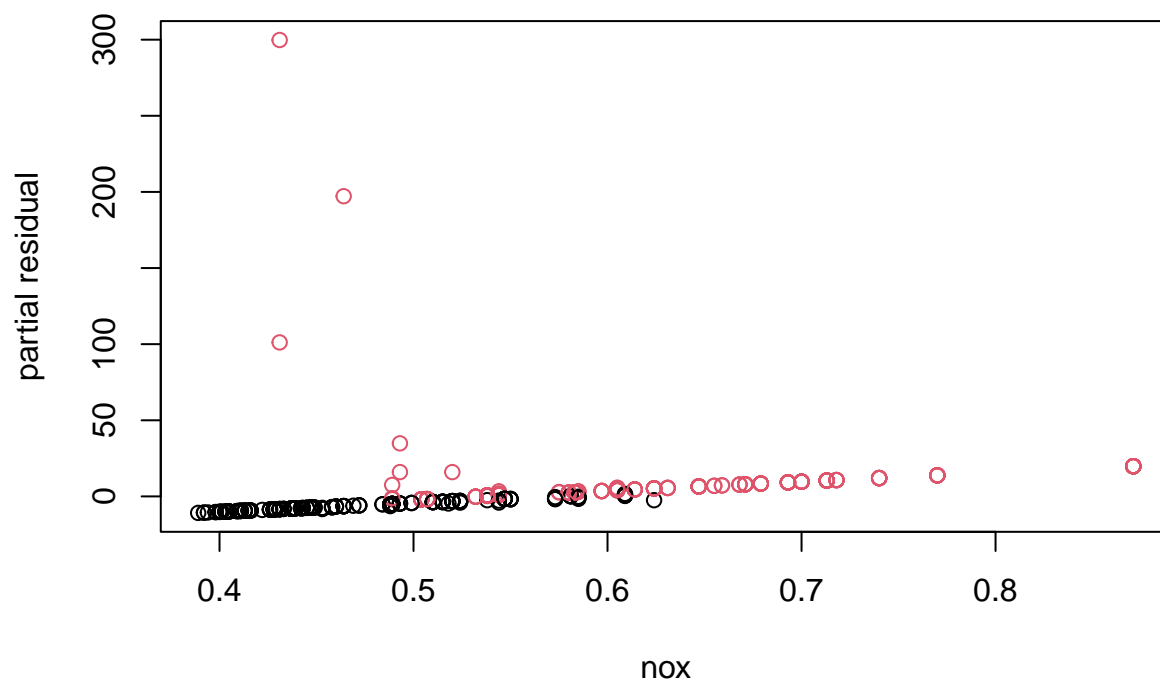
```
for (j in names(coef(model_full))[-1]) {  
  if (j != 'chas1' && j != 'chas2'){  
    plot(  
      x = df_training_one_hot[,j],  
      y = residuals(model_full, "partial")[,j],  
      col = df_training_one_hot$target,  
      main = paste0("Partial residuals by ", j),  
      xlab = j,  
      ylab = "partial residual"  
    )  
  }  
}
```



**Partial residuals by indus**

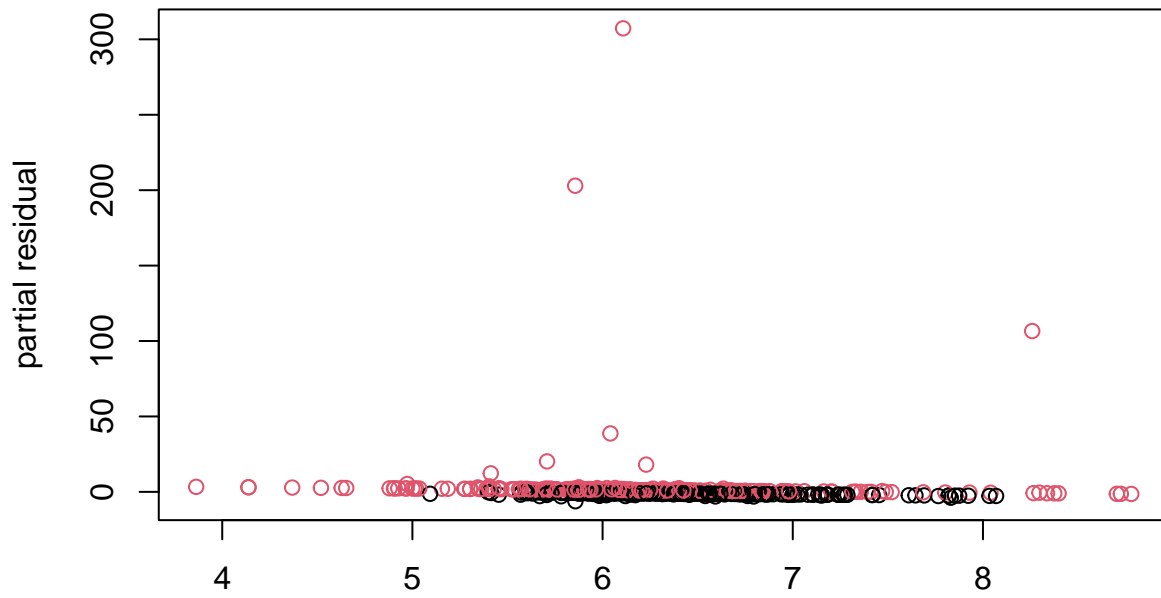


indus  
**Partial residuals by nox**

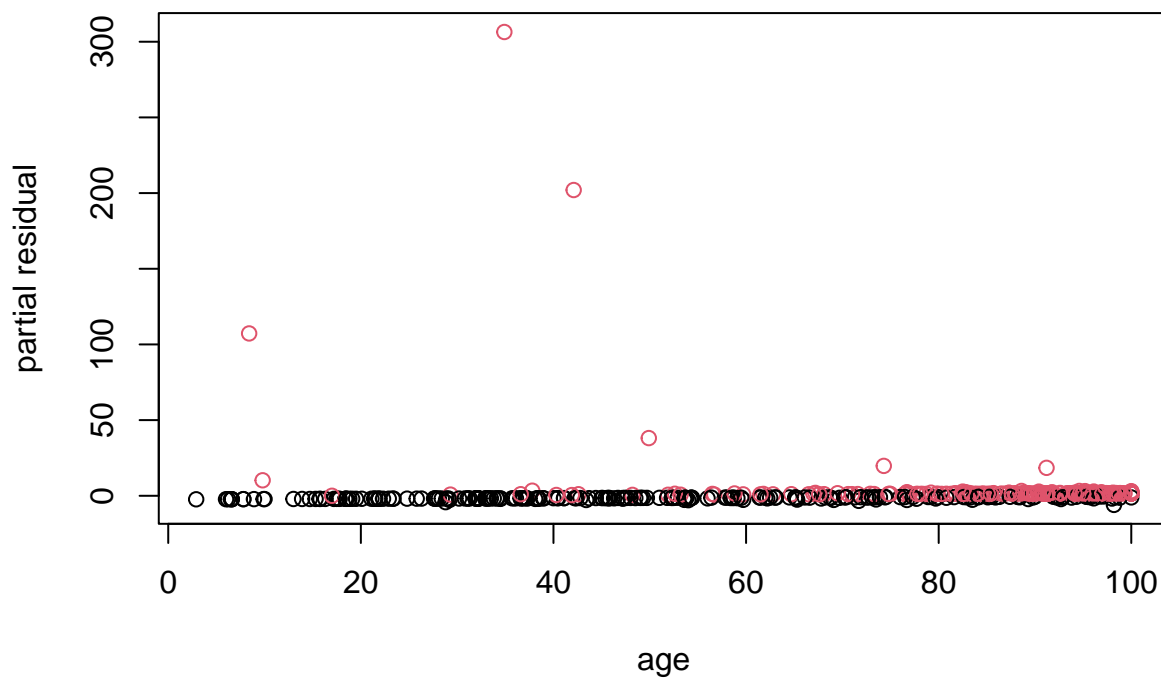




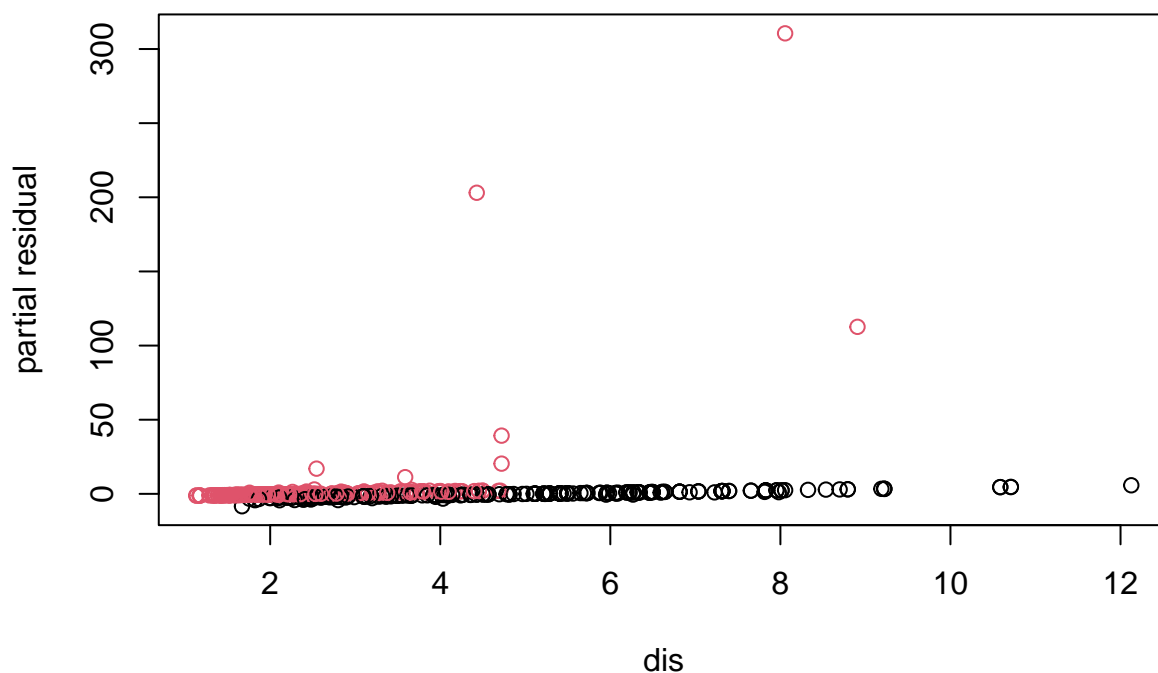
**Partial residuals by rm**



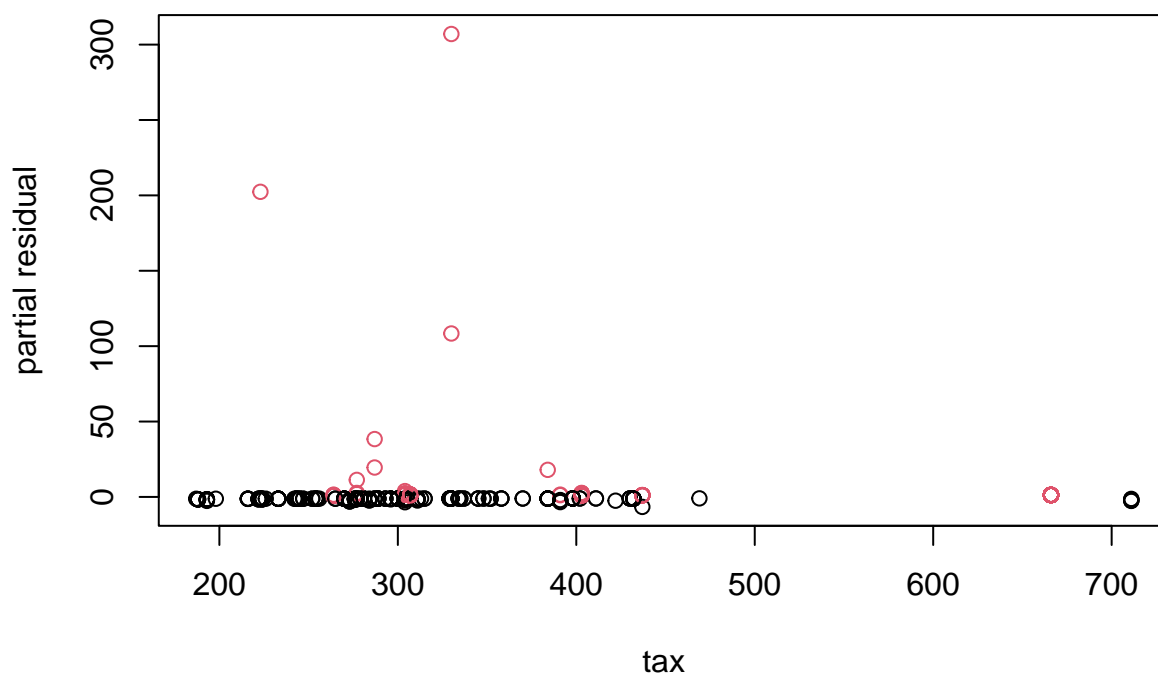
**Partial residuals by age**



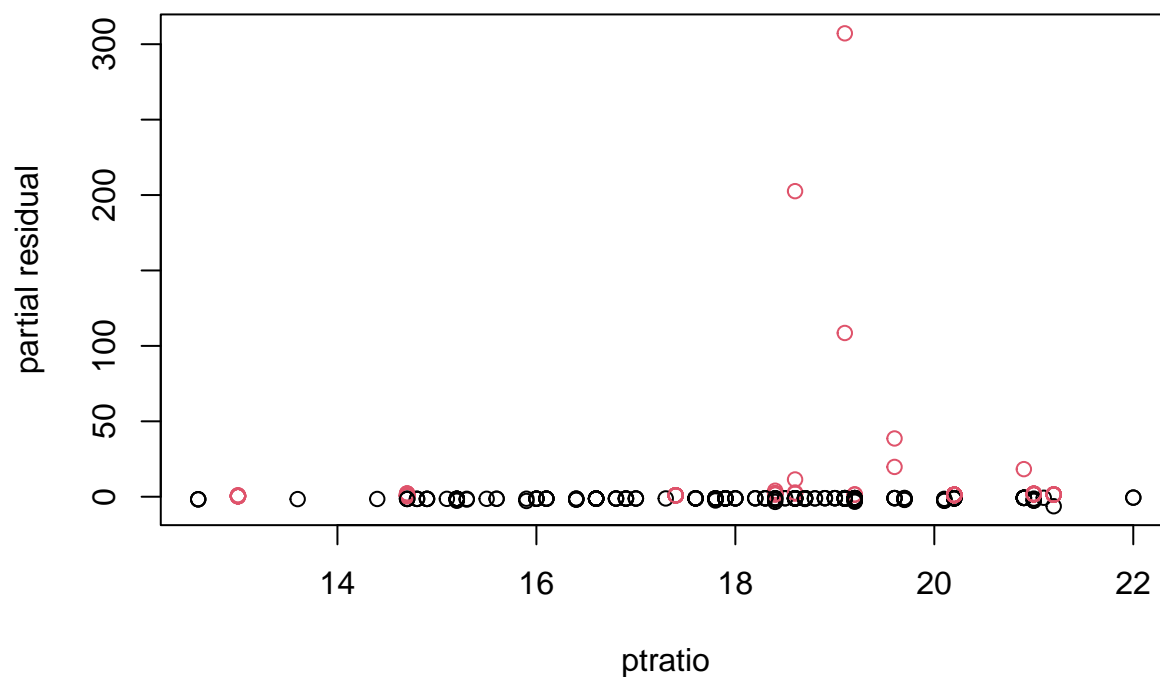
**Partial residuals by dis**



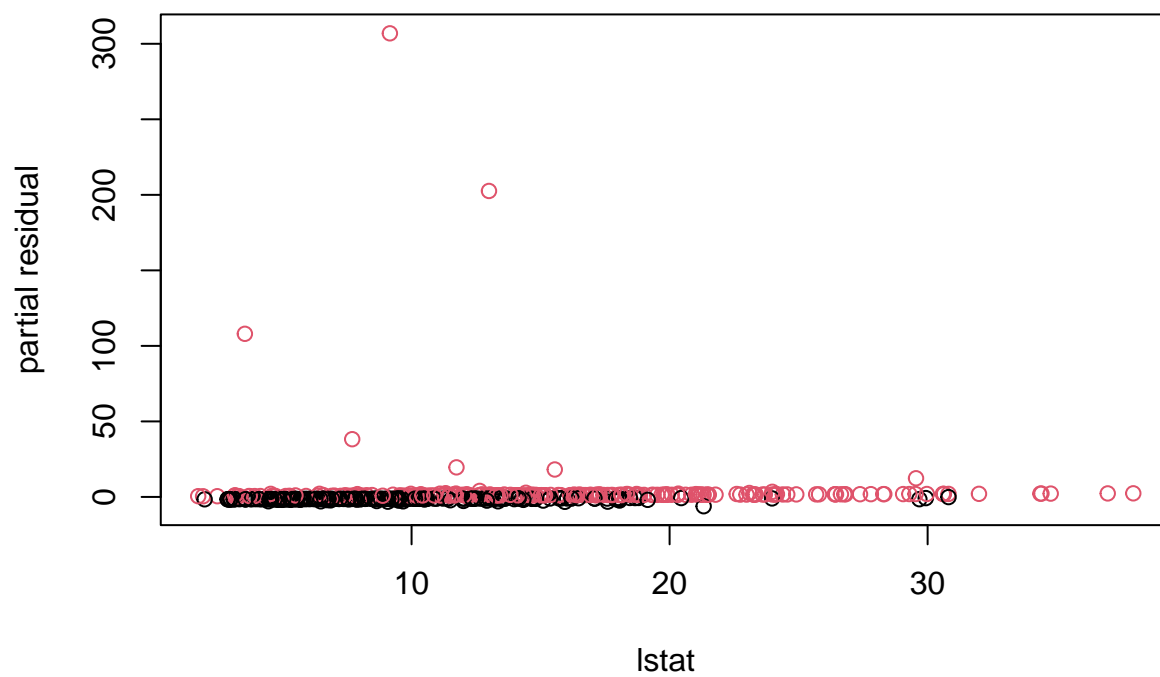
**Partial residuals by tax**



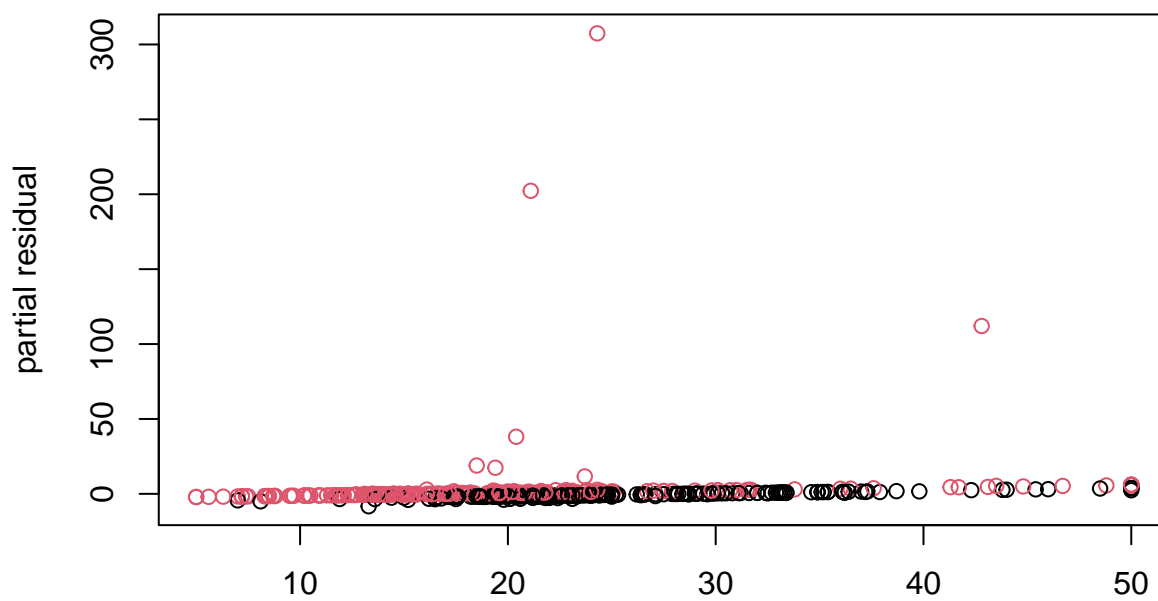
**Partial residuals by ptratio**



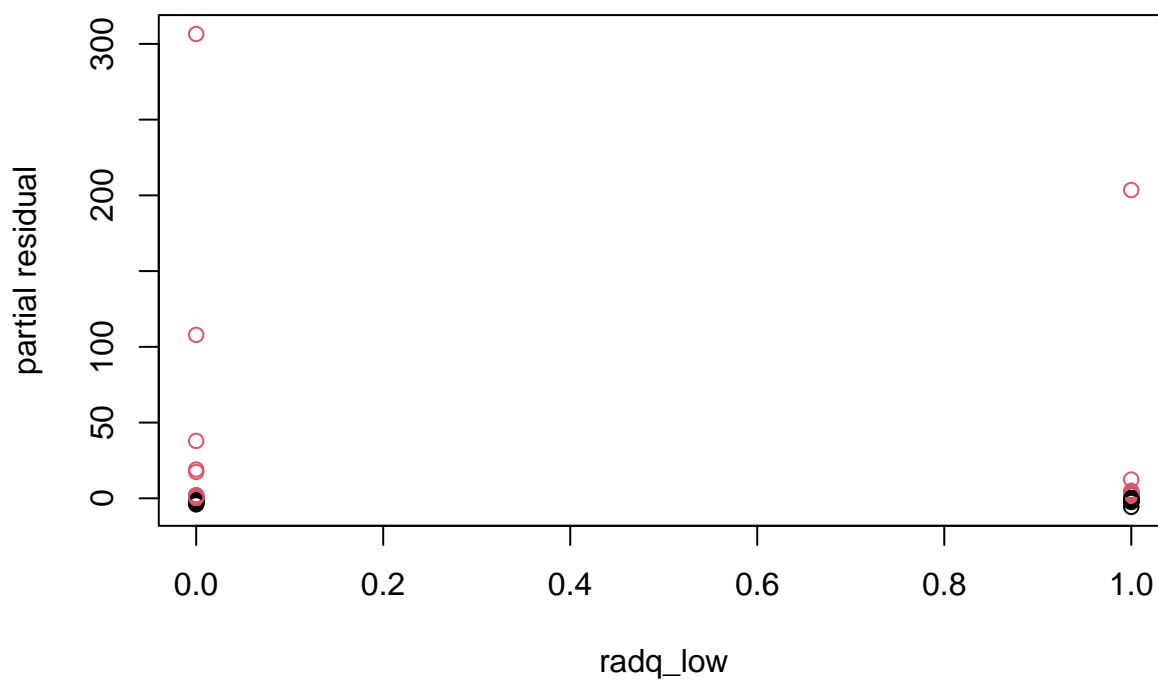
ptratio  
**Partial residuals by lstat**



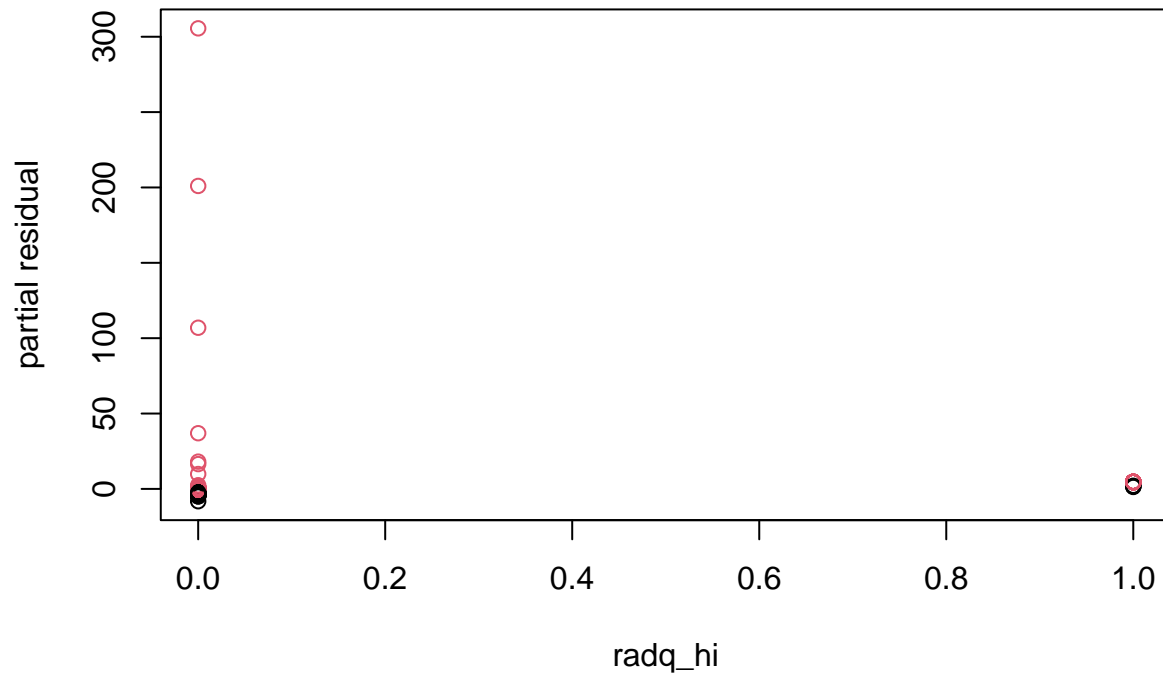
**Partial residuals by medv**



**Partial residuals by radq\_low**



## Partial residuals by radq\_hi

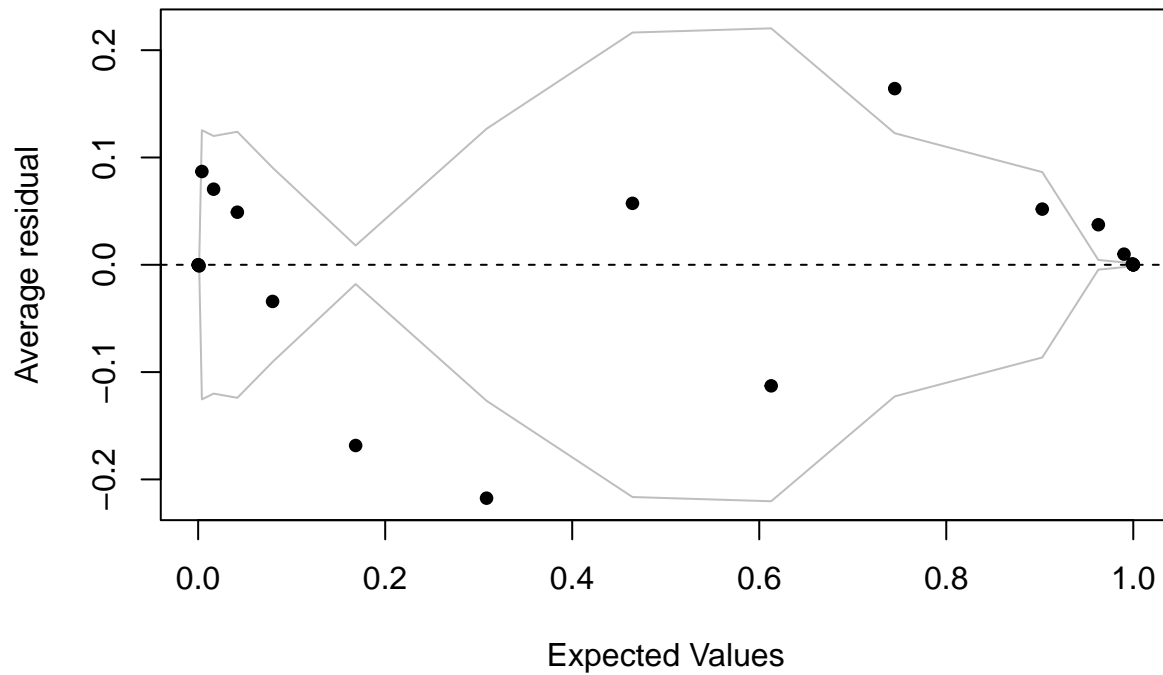


### *Binned Residuals Plot*

Below is a Binned Residuals Plot. The binned residuals plot divide the data into categories (bins) based on their fitted values, then plot the average residual versus the average fitted value for each bin.

```
binnedplot(  
  x = predict(model_full, newdata=df_training_one_hot, type="response"),  
  y = residuals(model_full, type="response")  
)
```

## Binned residual plot

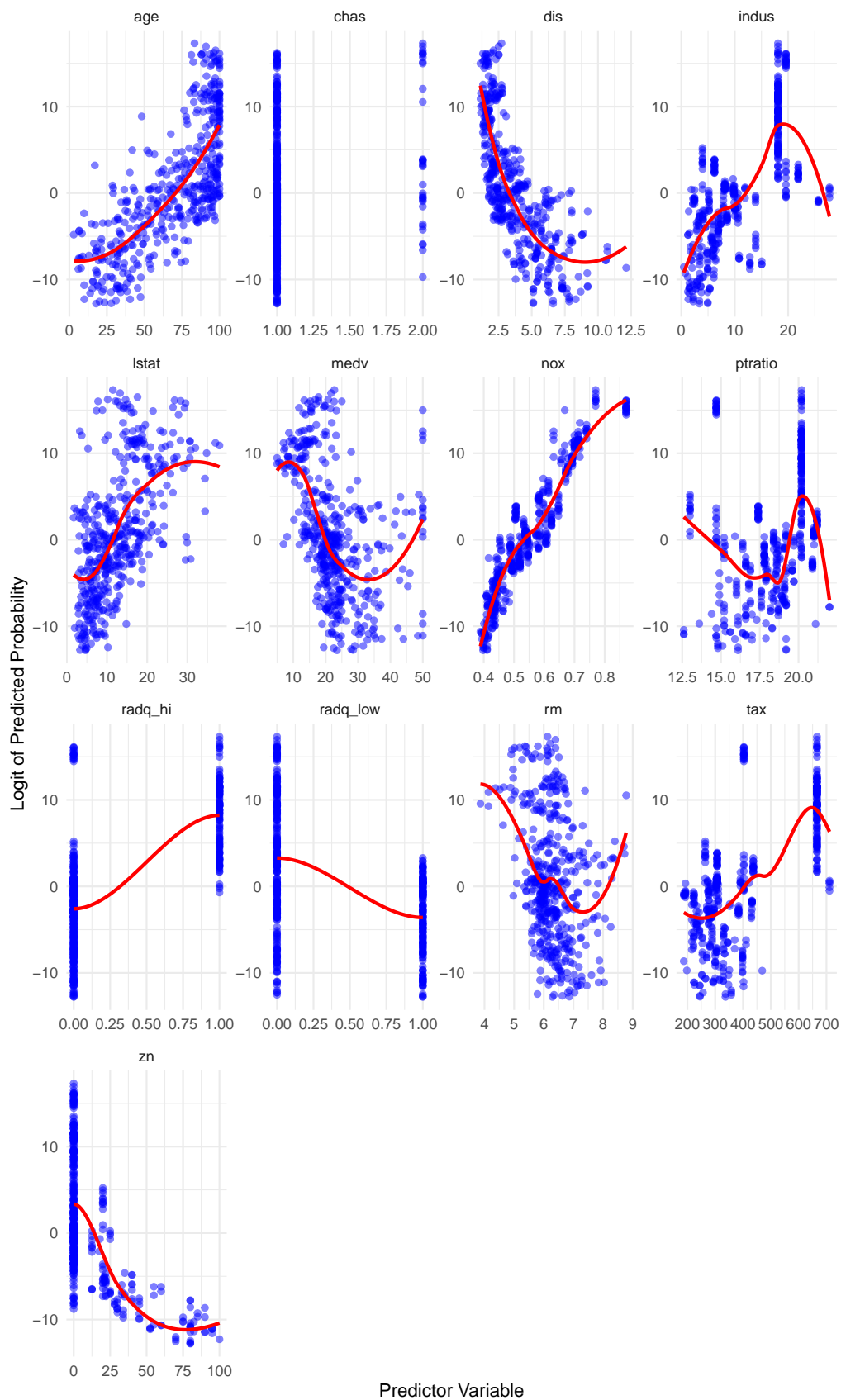


In a Binned Residuals Plot, the gray lines indicate plus and minus 2 standard-error bounds. We would expect about 95% of the binned residuals to fall within these lines. Several points fall outside of the 95% interval, but three points are more obviously outside.

### Linearity

To check this condition, I created a scatterplot with a loess line to check that there is a linear relationship between the logit of the dependent variable and the independent variables.

# Linearity of Logit Check for Binary Logistic Regression



---

## Using mathematical transformations

To reduce the influence of outliers and better align the data with the assumptions of logistic regression, log-transformations were applied to tax, zn, dis, and lstat. This transformation helps normalize the data, reduce variance, and enhance model interpretability. A small constant was added to zn before the transformation to account for zero values.

```
df_training_1h_log <- df_training_one_hot |>
  mutate(
    log_tax = log(tax),
    log_dis = log(dis),
    log_zn = log(zn + 1),
    log_lstat = log(lstat),
    log_medv = log(medv),
    log_indus = log(indus),
    log_ptratio = log(ptratio),
  ) |>
  subset(select = -c(tax, dis, zn, lstat, medv, indus, ptratio))

model_full_log <- glm(target ~ ., binomial(link = "logit"), data=df_training_1h_log)
summary(model_full_log)
```

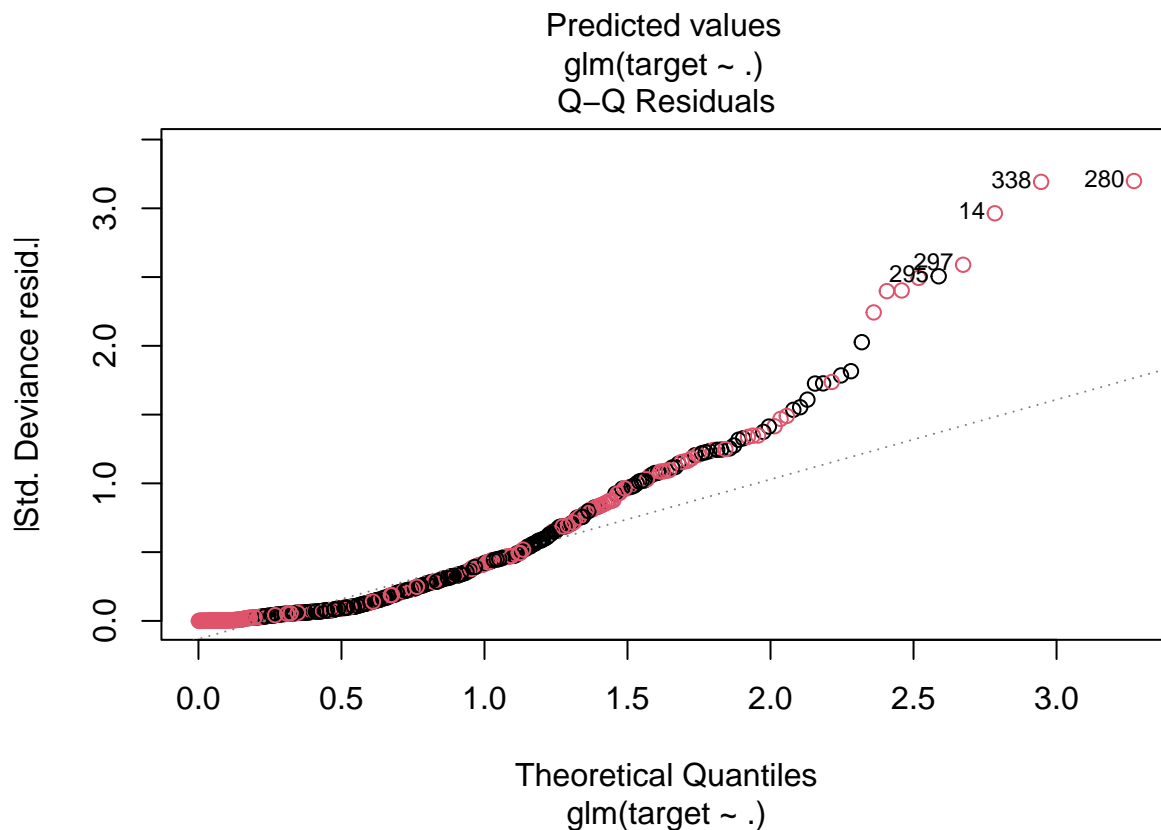
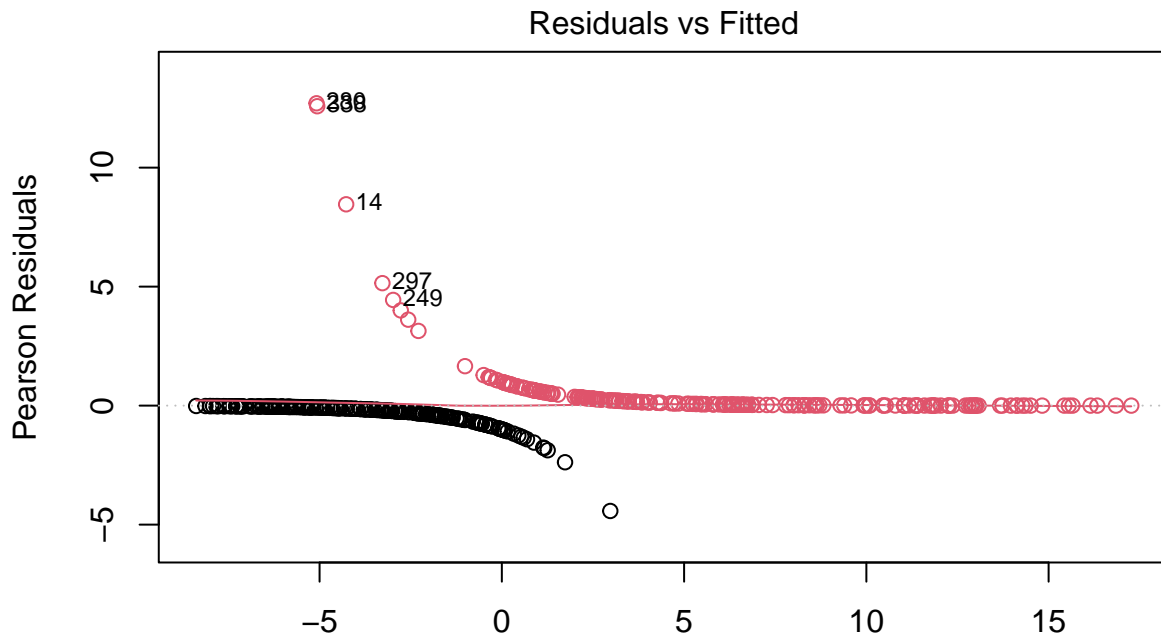
```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##      data = df_training_1h_log)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -61.55667   14.07134  -4.375 1.22e-05 ***
## chas1         0.76611    0.78810   0.972 0.33100
## nox          55.42383    8.35816   6.631 3.33e-11 ***
## rm          -0.44042    0.60357  -0.730 0.46557
## age           0.02181    0.01234   1.768 0.07708 .
## radq_low      1.54137    0.52446   2.939 0.00329 **
## radq_hi       4.56152    0.81652   5.587 2.32e-08 ***
## log_tax       1.42973    0.92183   1.551 0.12091
## log_dis       4.30622    0.96710   4.453 8.48e-06 ***
## log_zn       -0.46306    0.23731  -1.951 0.05102 .
## log_lstat     0.59773    0.64866   0.921 0.35680
## log_medv      4.37118    1.39379   3.136 0.00171 **
## log_indus    -0.45813    0.46068  -0.994 0.32000
## log_ptratio   1.43430    2.38065   0.602 0.54685
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 645.88  on 465  degrees of freedom
## Residual deviance: 186.45  on 452  degrees of freedom
## AIC: 214.45
##
```

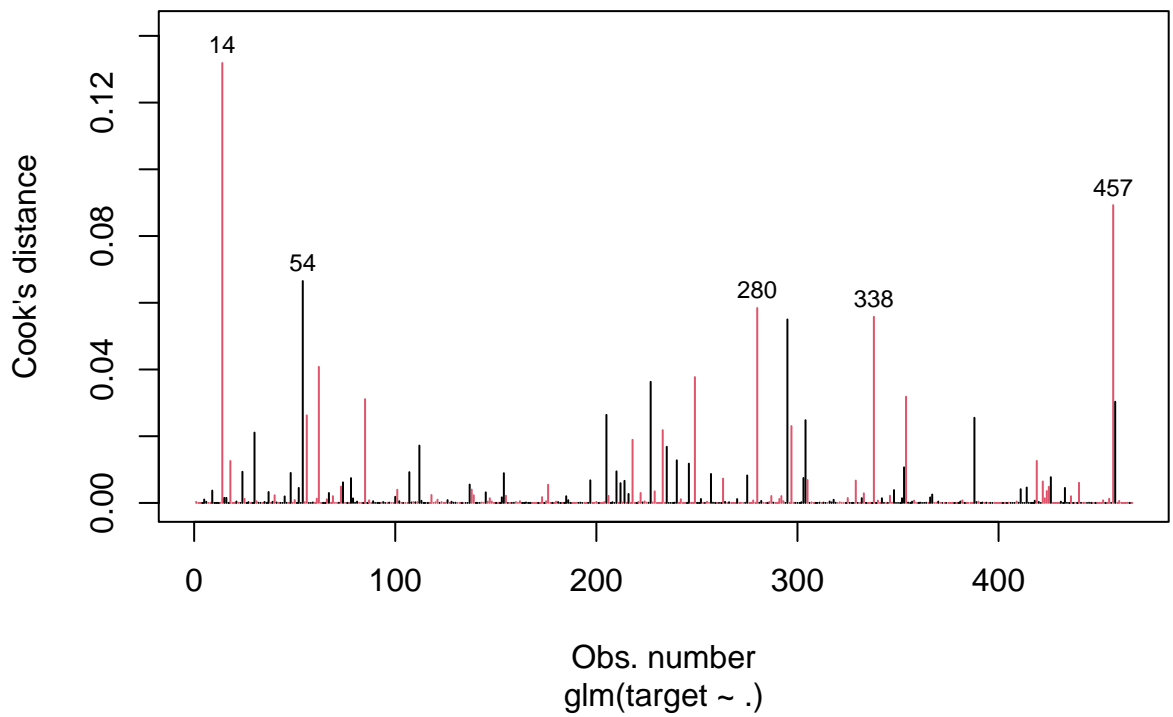
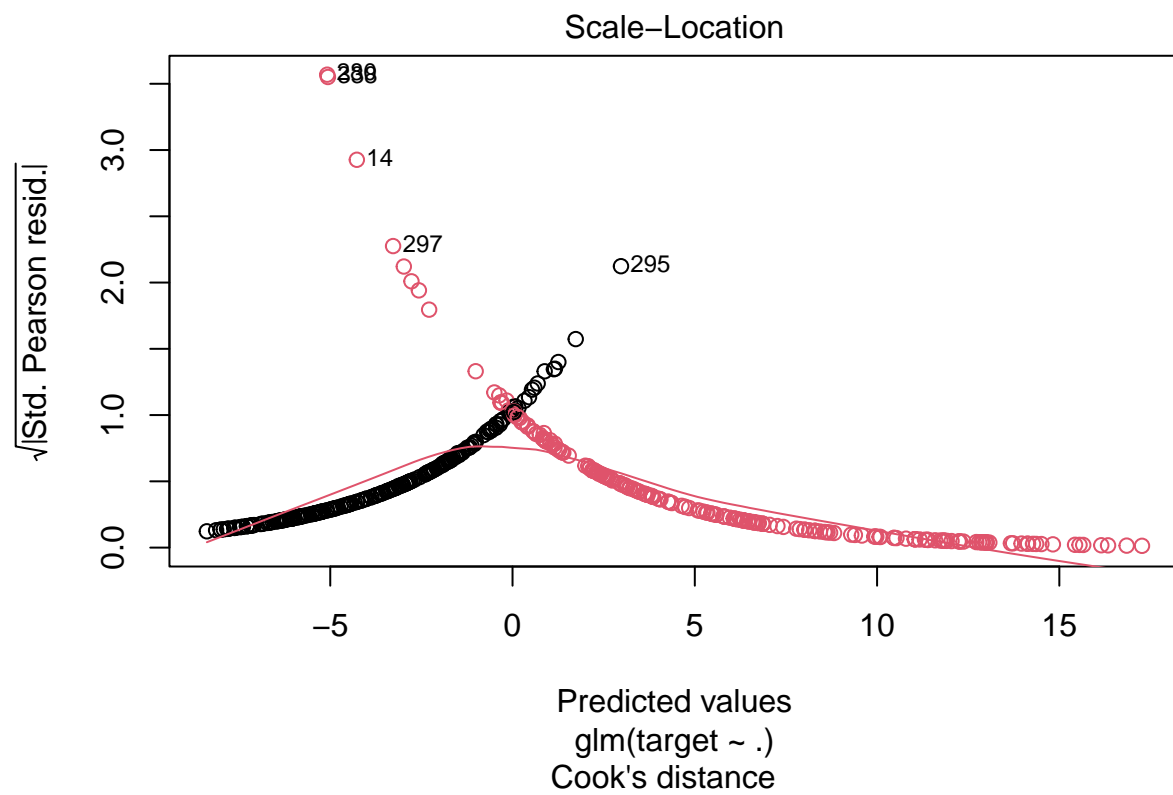


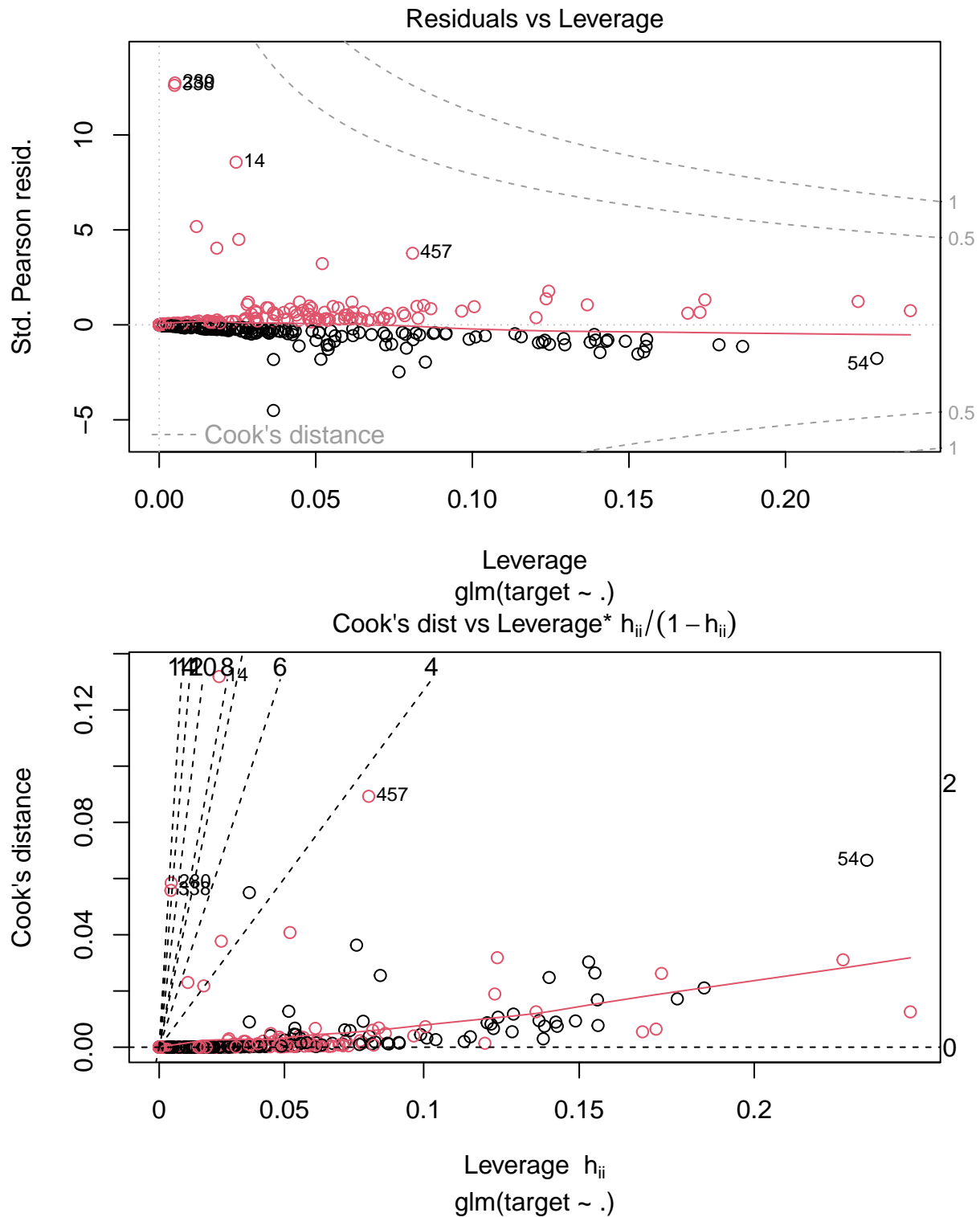
```
## Number of Fisher Scoring iterations: 8
```

**Outlier** Applying the log transformation didn't make too much of a difference with our questionable points (280, 338, and 14)

```
par(mar = c(5, 4, 4, 2) + 0.1)
plot(model_full_log, which = c(4, 6, 1, 2, 3, 5), col=df_training_1h_log$target, id.n = 5)
```

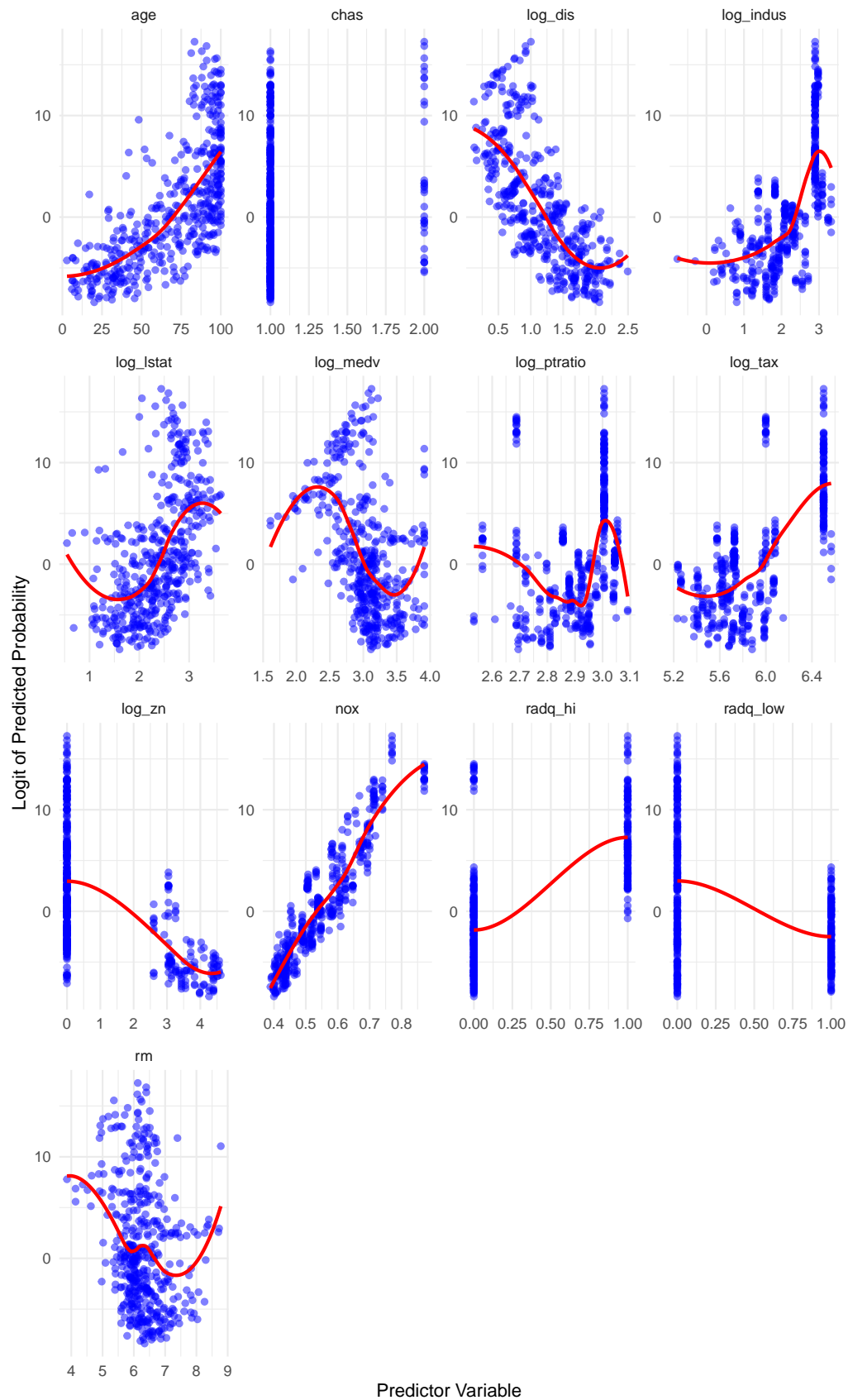






**Linearity** Applying the log transformation helped the linearity for some of the variables. It had less of an effect on medv, indus, and ptratio

# Linearity of Logit Check for Binary Logistic Regression



**Colinearity** A Variance Inflation Factor test on our model with logged predictors shows that `nox` and `medv` should be considered for removal. `rm` and `log_dis` may also need to be considered.

```
car::vif(model_full_log) |> sort()
```

```
##      chas      radq_hi      radq_low      log_tax      log_zn      age
##  1.231235  1.652783  1.964114  1.989751  2.016719  2.036268
##  log_indus log_ptratio log_lstat      rm      log_dis      nox
##  2.704867  2.745159  3.199281  3.993171  4.738826  5.340565
##  log_mdv
##  6.182031
```

## MODEL BUILDING

Using a binomial (`target`) for our dependent variable would violate the common assumptions for linear regression. Specifically:

- the observations will not be normally distributed as they are binary
- the variance of error may be heteroskedastic instead of homoskedastic
- R-squared may not be a good fit

To account for these violations, we will use a Generalized Linear Model (GLM) to conduct logistic regression.

### Backward Selection Model (BIC)

```
# Backward stepwise regression
backward_model <- stepAIC(model_full, direction = "backward", k=log(nrow(df_training_one_hot)))
```

```
## Start:  AIC=266.99
## target ~ zn + indus + chas + nox + rm + age + dis + tax + ptratio +
##      lstat + medv + radq_low + radq_hi
##
##      Df Deviance    AIC
## - tax      1  181.15 261.02
## - ptratio   1  181.76 261.64
## - lstat     1  182.04 261.92
## - rm        1  182.84 262.72
## - chas      1  183.04 262.92
## - age       1  183.46 263.33
## - zn        1  186.29 266.16
## <none>      180.97 266.99
## - medv      1  190.89 270.76
## - radq_low  1  190.93 270.81
## - dis       1  192.90 272.77
## - indus     1  194.95 274.82
## - radq_hi   1  233.24 313.11
## - nox       1  272.51 352.39
##
## Step:  AIC=261.02
## target ~ zn + indus + chas + nox + rm + age + dis + ptratio +
##      lstat + medv + radq_low + radq_hi
##
##      Df Deviance    AIC
## - ptratio   1  181.90 255.63
## - lstat     1  182.38 256.11
## - rm        1  182.89 256.62
```

```

## - chas      1   183.10 256.83
## - age       1   183.52 257.25
## - zn        1   186.43 260.16
## <none>      181.15 261.02
## - radq_low  1   190.96 264.69
## - medv      1   190.97 264.70
## - dis       1   192.90 266.63
## - indus     1   195.66 269.39
## - radq_hi   1   246.52 320.25
## - nox       1   273.18 346.91
##
## Step:  AIC=255.63
## target ~ zn + indus + chas + nox + rm + age + dis + lstat + medv +
##         radq_low + radq_hi
##
##           Df Deviance   AIC
## - rm      1   183.31 250.90
## - lstat    1   183.32 250.91
## - chas     1   183.57 251.16
## - age      1   183.76 251.34
## <none>     181.90 255.63
## - zn       1   188.59 256.18
## - medv     1   191.14 258.73
## - dis      1   193.32 260.90
## - indus    1   196.68 264.26
## - radq_low 1   197.70 265.29
## - radq_hi  1   258.34 325.92
## - nox      1   276.37 343.95
##
## Step:  AIC=250.9
## target ~ zn + indus + chas + nox + age + dis + lstat + medv +
##         radq_low + radq_hi
##
##           Df Deviance   AIC
## - age      1   184.18 245.62
## - chas     1   185.38 246.82
## - lstat    1   186.68 248.13
## <none>     183.31 250.90
## - zn       1   190.50 251.94
## - medv     1   193.27 254.71
## - dis      1   193.78 255.22
## - indus    1   197.50 258.94
## - radq_low 1   198.16 259.60
## - radq_hi  1   260.41 321.85
## - nox      1   276.66 338.10
##
## Step:  AIC=245.62
## target ~ zn + indus + chas + nox + dis + lstat + medv + radq_low +
##         radq_hi
##
##           Df Deviance   AIC
## - chas     1   186.52 241.82
## - lstat    1   188.79 244.09
## <none>     184.18 245.62

```

```

## - zn          1    191.45 246.75
## - dis          1    193.99 249.28
## - medv         1    194.41 249.70
## - indus        1    198.41 253.71
## - radq_low     1    200.10 255.40
## - radq_hi      1    262.55 317.84
## - nox          1    289.78 345.08
##
## Step: AIC=241.82
## target ~ zn + indus + nox + dis + lstat + medv + radq_low + radq_hi
##
##           Df Deviance    AIC
## - lstat     1    191.97 241.13
## <none>       186.52 241.82
## - zn        1    194.01 243.17
## - dis       1    195.90 245.05
## - medv      1    198.71 247.87
## - indus     1    199.42 248.57
## - radq_low  1    200.88 250.03
## - radq_hi   1    265.34 314.50
## - nox       1    289.83 338.99
##
## Step: AIC=241.13
## target ~ zn + indus + nox + dis + medv + radq_low + radq_hi
##
##           Df Deviance    AIC
## <none>       191.97 241.13
## - zn        1    198.69 241.70
## - medv      1    198.73 241.74
## - dis       1    199.72 242.73
## - indus     1    202.54 245.55
## - radq_low  1    206.64 249.65
## - radq_hi   1    270.27 313.28
## - nox       1    298.32 341.33

```

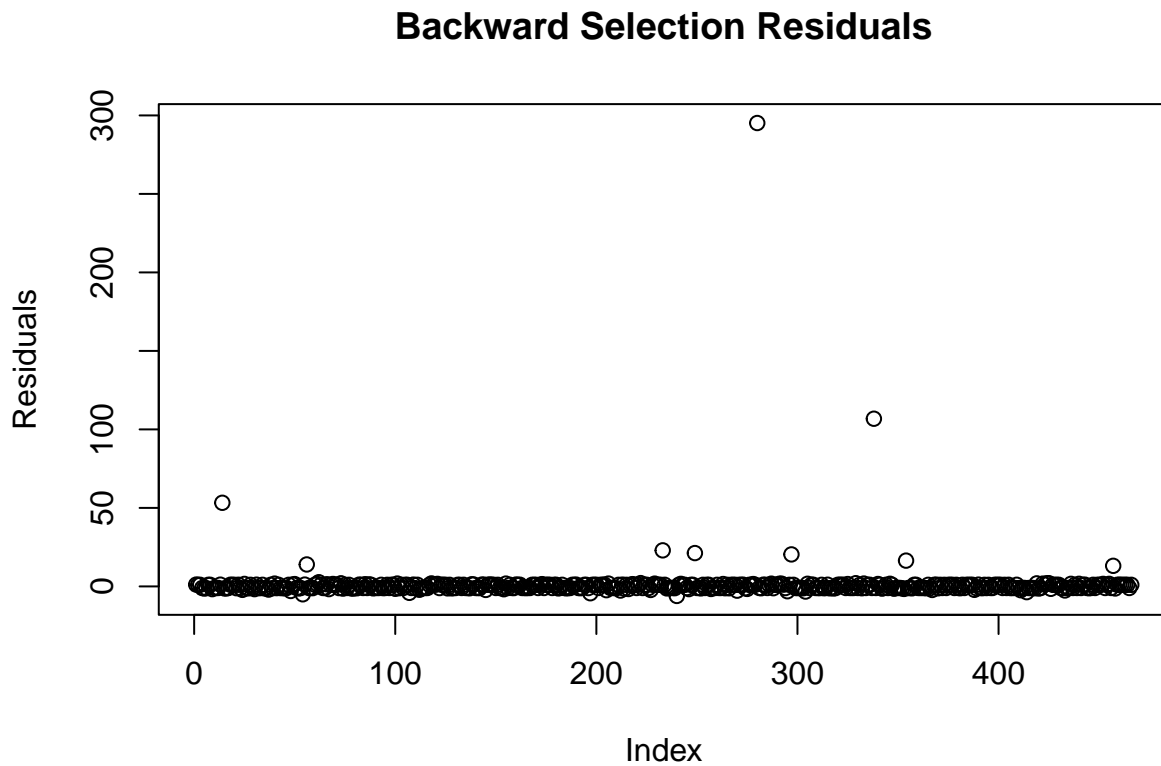
```
summary(backward_model)
```

```

##
## Call:
## glm(formula = target ~ zn + indus + nox + dis + medv + radq_low +
##       radq_hi, family = binomial(link = "logit"), data = df_training_one_hot)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -33.35748    5.01546  -6.651 2.91e-11 ***
## zn           -0.08021    0.03682  -2.178 0.029380 *
## indus        -0.13193    0.04263  -3.095 0.001968 **
## nox           55.14154    8.02184   6.874 6.25e-12 ***
## dis           0.60188    0.22176   2.714 0.006645 **
## medv          0.06570    0.02705   2.429 0.015141 *
## radq_low      1.58049    0.43618   3.623 0.000291 ***
## radq_hi       4.95421    0.77313   6.408 1.47e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##

```

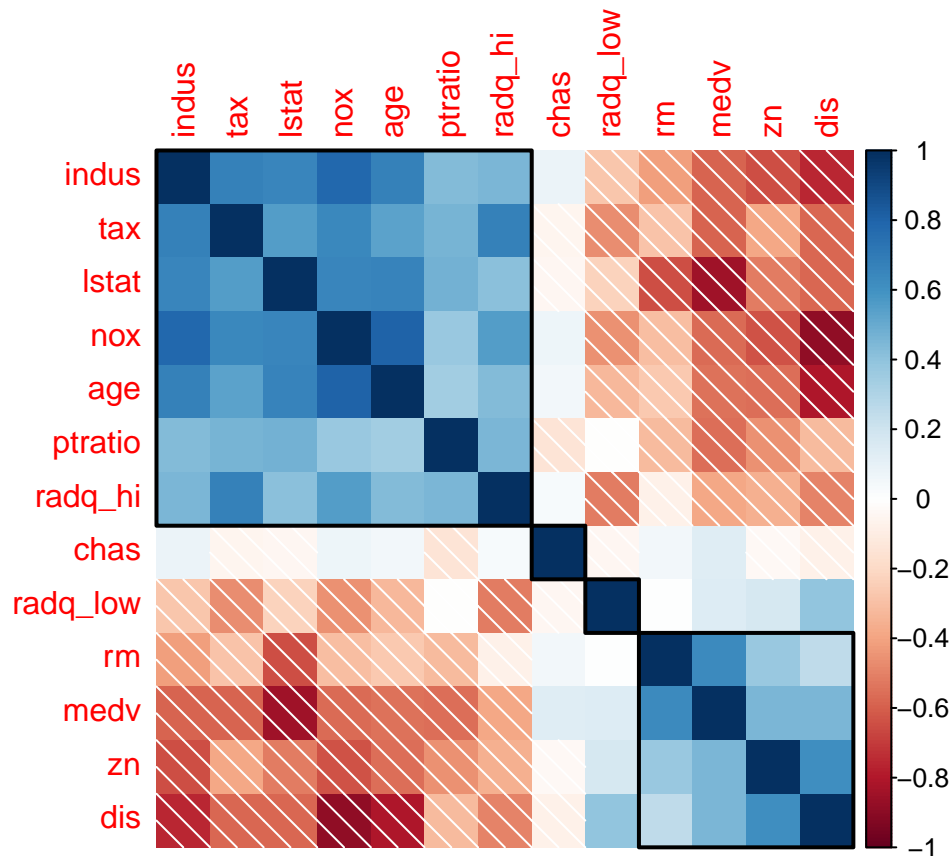
```
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 645.88 on 465 degrees of freedom
## Residual deviance: 191.97 on 458 degrees of freedom
## AIC: 207.97
##
## Number of Fisher Scoring iterations: 8
plot(backward_model$residuals, main = "Backward Selection Residuals", ylab = "Residuals")
```



### Models Cased on Predictor Correlation

These models were guided by the results of our correlation plot. The correlation plot shows strong correlation among predictors in two large clusters suggesting that selecting one variable from each cluster might be sufficient within our model.





```
##
## Call:
## glm(formula = target ~ indus + chas + radq_low + medv, family = binomial(link = "logit"),
##      data = df_training_one_hot)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.185085   0.577111  -3.786 0.000153 ***
## indus        0.215890   0.023736   9.095 < 2e-16 ***
## chas1        0.508463   0.484696   1.049 0.294162
## radq_low     -1.274556   0.261075  -4.882 1.05e-06 ***
## medv         0.008922   0.016893   0.528 0.597415
## ---
## 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: 426.57  on 461  degrees of freedom
## AIC: 436.57
##
## Number of Fisher Scoring iterations: 4
```

**Model using Principal Components** This section uses the correlation plot to perform Principal Component Analysis on the two large variable clusters shown in the plot. We will then substitute the variables in each of the two clusters with their respective PC scores in our model.

```

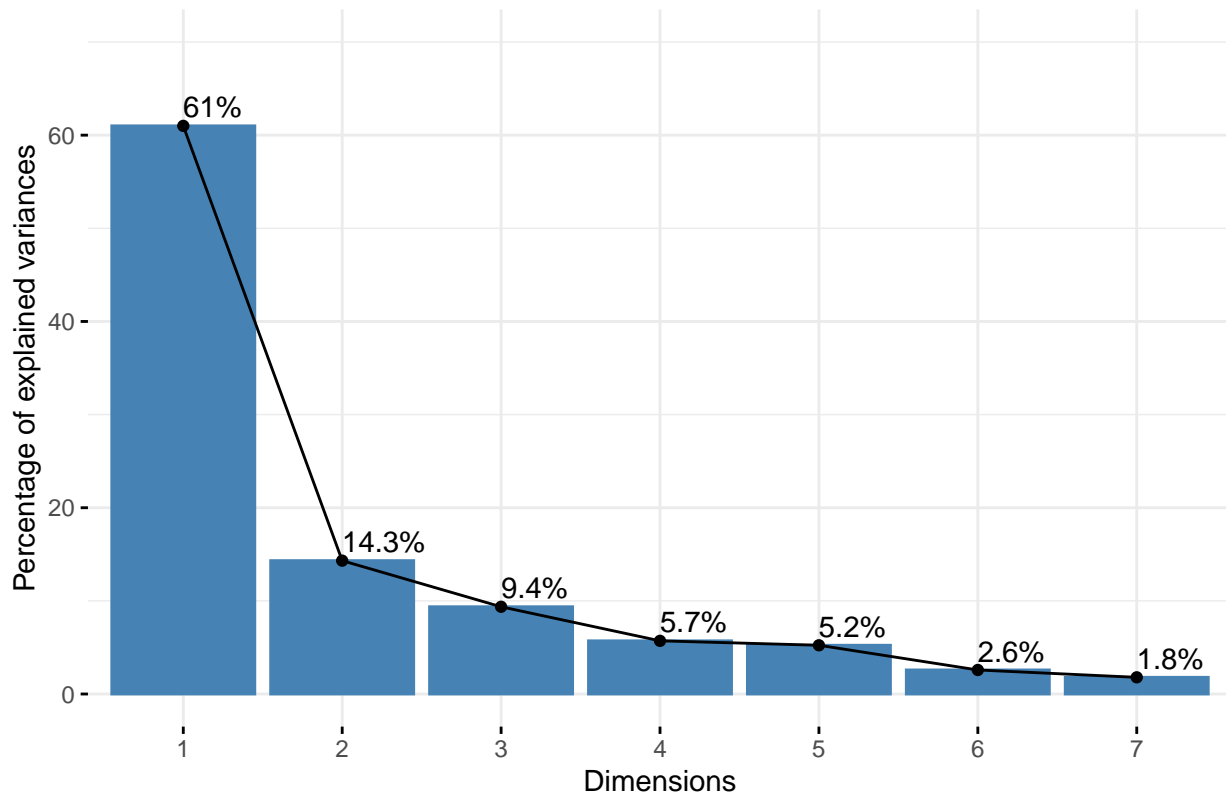
# Create PCA from first cluster in our correlation plot
df_pca_subset1 <- df_training_one_hot |>
  subset(select = c(indus, tax, lstat, nox, age, ptratio, radq_hi))

# calculate PCA
df_training_pca1 <- prcomp(df_pca_subset1, scale=TRUE)

# use eigen vectors to plot % of data explained by PCA1
fviz_eig(df_training_pca1, addlabels=TRUE, ylim=c(0, 70))

```

Scree plot

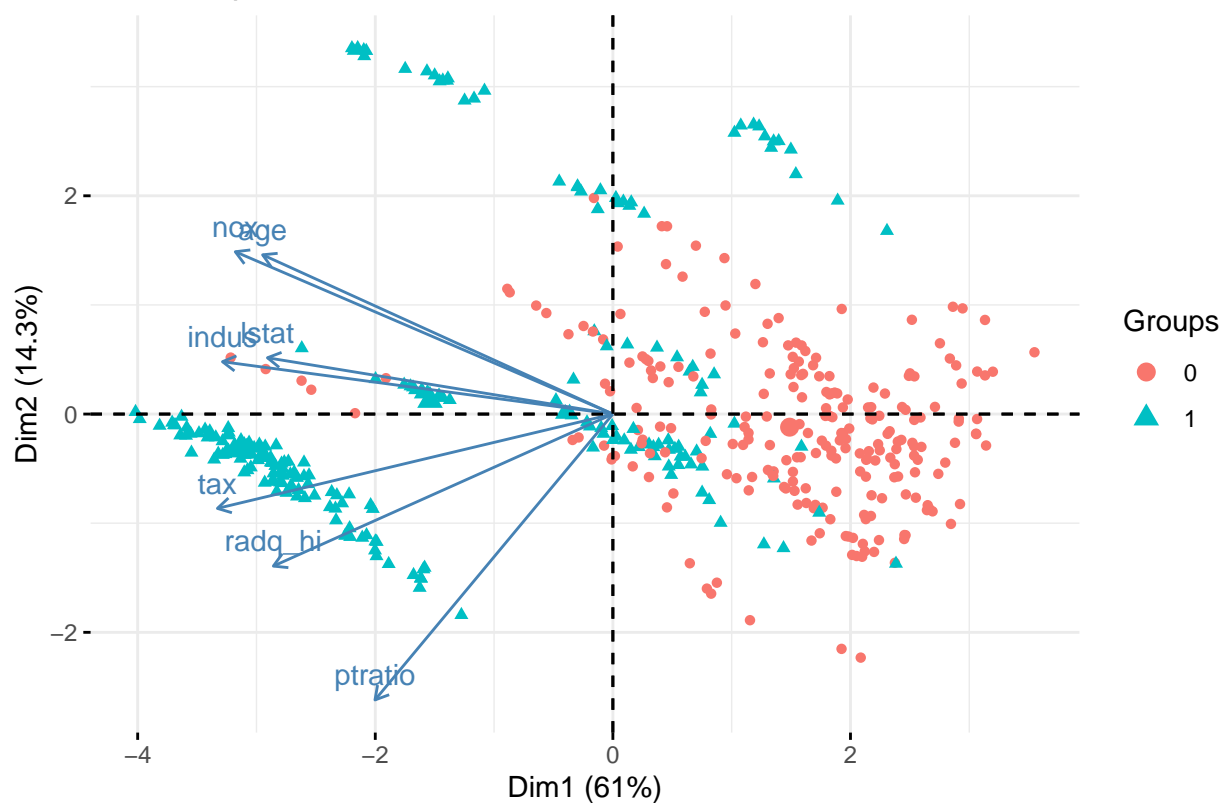


```

# plot PCA biplot
fviz_pca_biplot(df_training_pca1, label="var", habillage = df_training_one_hot$target)

```

## PCA – Biplot

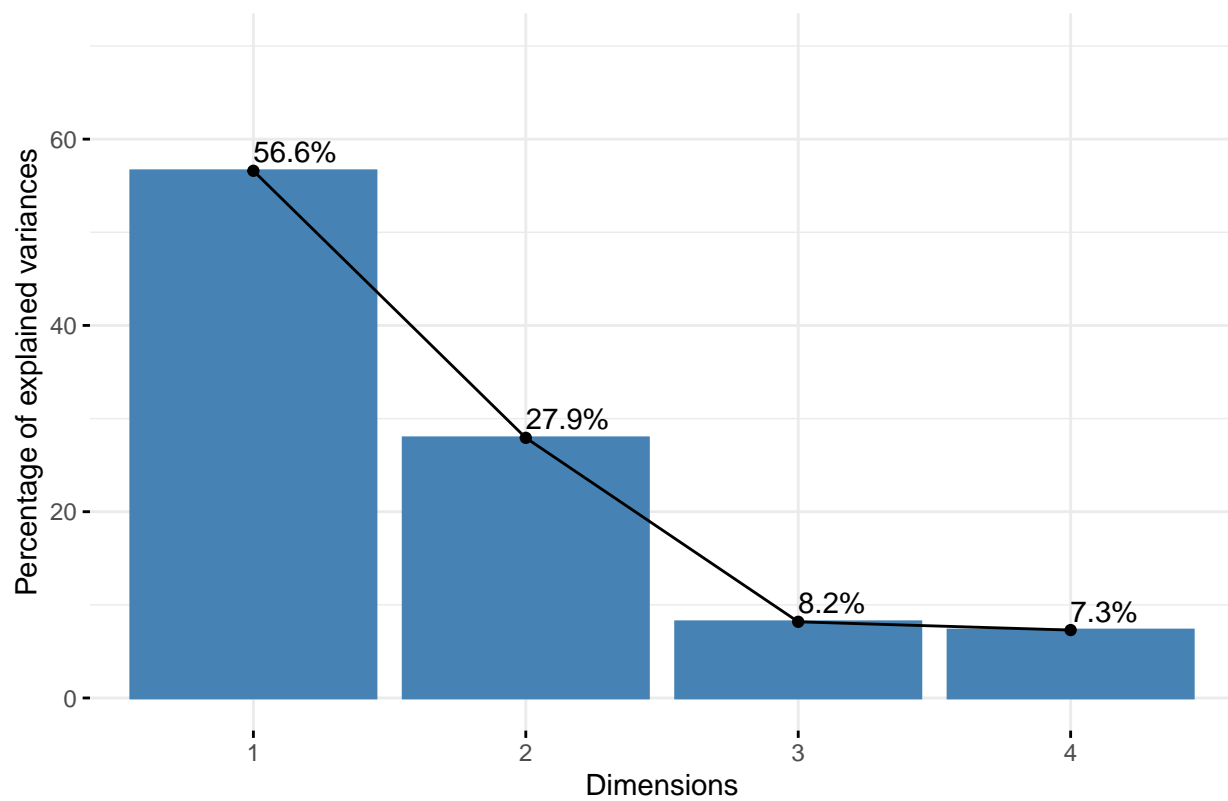


```
# Create PCA from second cluster in our correlation plot
df_pca_subset2 <- df_training_one_hot |>
  subset(select = c(rm, medv, zn, dis))

# calculate PCA
df_training_pca2 <- prcomp(df_pca_subset2, scale=TRUE)

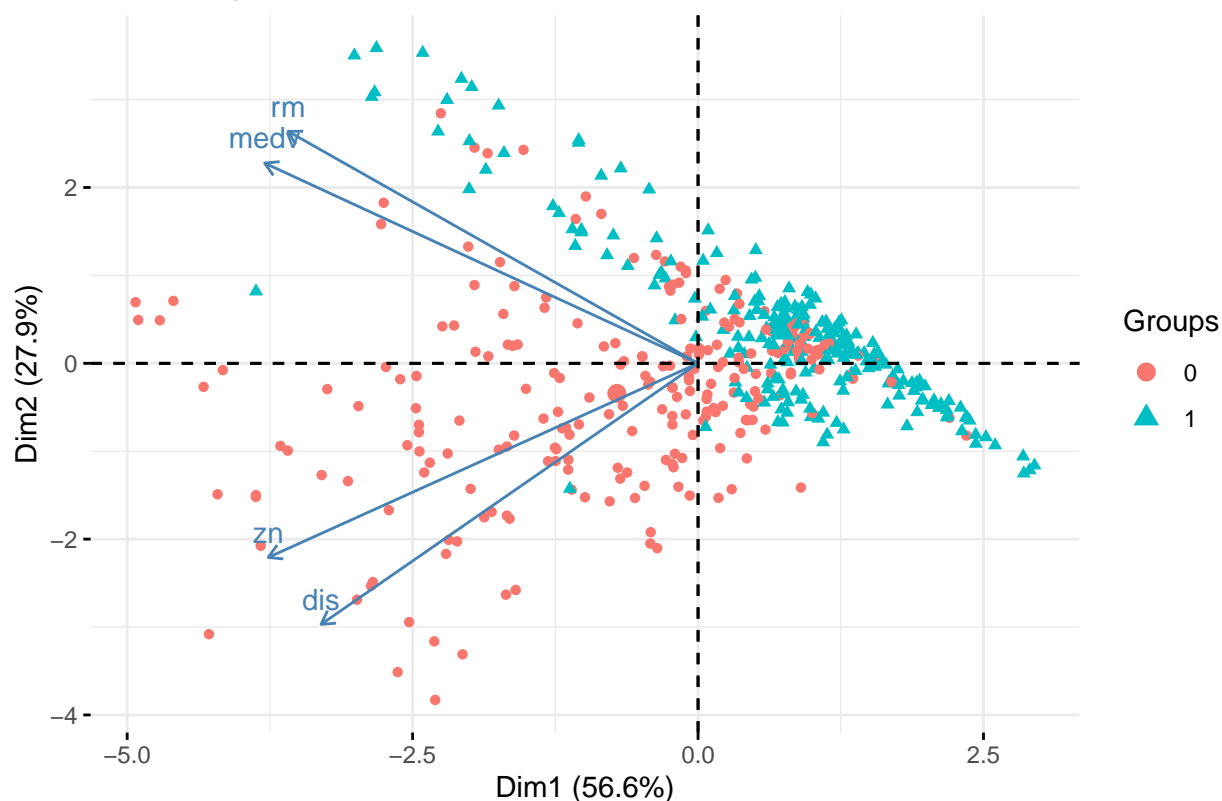
# use eigen vectors to plot % of data explained by PCA1
fviz_eig(df_training_pca2, addlabels=TRUE, ylim=c(0, 70))
```

Scree plot



```
# plot PCA biplot  
fviz_pca_biplot(df_training_pca2, label="var", habillage = df_training_one_hot$target)
```

## PCA – Biplot



```
# add pca's to our dataset
df_training_one_hot_pca <- df_training_one_hot |>
  subset(select = c(target, chas, radq_low)) |>
  mutate(
    group1_pc1 = df_training_pca1$x[, "PC1"],
    group1_pc2 = df_training_pca1$x[, "PC2"],
    group2_pc1 = df_training_pca2$x[, "PC1"],
    group2_pc2 = df_training_pca2$x[, "PC2"],
  )

#ggpairs(df_training_one_hot_pca |> subset(select = -c(target)))

model_pca <- glm(target ~ ., binomial(link = "logit"), data=df_training_one_hot_pca)
summary(model_pca)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##      data = df_training_one_hot_pca)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.14990    0.22856   0.656 0.511908
## chas1        0.35486    0.52302   0.678 0.497468
## radq_low     -0.04425    0.32215  -0.137 0.890753
## group1_pc1   -1.45138    0.18342  -7.913 2.52e-15 ***
## group1_pc2    0.17208    0.15573   1.105 0.269166
```

```
## group2_pc1 -0.30550    0.20533 -1.488 0.136790
## group2_pc2  0.67822    0.18654  3.636 0.000277 ***
## ---
## 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: 289.25  on 459  degrees of freedom
## AIC: 303.25
##
## Number of Fisher Scoring iterations: 6
```

Interestingly, only the primary principal component from group1 and the secondary principal component from group two have strong statistical significance. `radq_low` has a particularly high p-value and should be considered for removal.

```
model_pca2 <- update(model_pca, . ~ . - radq_low)
summary(model_pca2)
```

```
##
## Call:
## glm(formula = target ~ chas + group1_pc1 + group1_pc2 + group2_pc1 +
##      group2_pc2, family = binomial(link = "logit"), data = df_training_one_hot_pca)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1331     0.1933   0.689 0.491070
## chas1         0.3545     0.5219   0.679 0.497018
## group1_pc1   -1.4576     0.1779  -8.194 2.53e-16 ***
## group1_pc2    0.1751     0.1541   1.136 0.256022
## group2_pc1   -0.3094     0.2032  -1.523 0.127809
## group2_pc2    0.6808     0.1856   3.668 0.000244 ***
## ---
## 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: 289.27  on 460  degrees of freedom
## AIC: 301.27
##
## Number of Fisher Scoring iterations: 6
```

```
model_pca2 <- update(model_pca2, . ~ . - chas)
summary(model_pca2)
```

```
##
## Call:
## glm(formula = target ~ group1_pc1 + group1_pc2 + group2_pc1 +
##      group2_pc2, family = binomial(link = "logit"), data = df_training_one_hot_pca)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1640     0.1881   0.871 0.383506
## group1_pc1   -1.4598     0.1780  -8.201 2.38e-16 ***
```

```
## group1_pc2    0.1848    0.1532    1.206 0.227758
## group2_pc1   -0.3124    0.2027   -1.542 0.123190
## group2_pc2    0.6823    0.1845    3.697 0.000218 ***
## ---
## 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: 289.73  on 461  degrees of freedom
## AIC: 299.73
##
## Number of Fisher Scoring iterations: 6
model_pca2 <- update(model_pca2, . ~ . - group1_pc2)
summary(model_pca2)

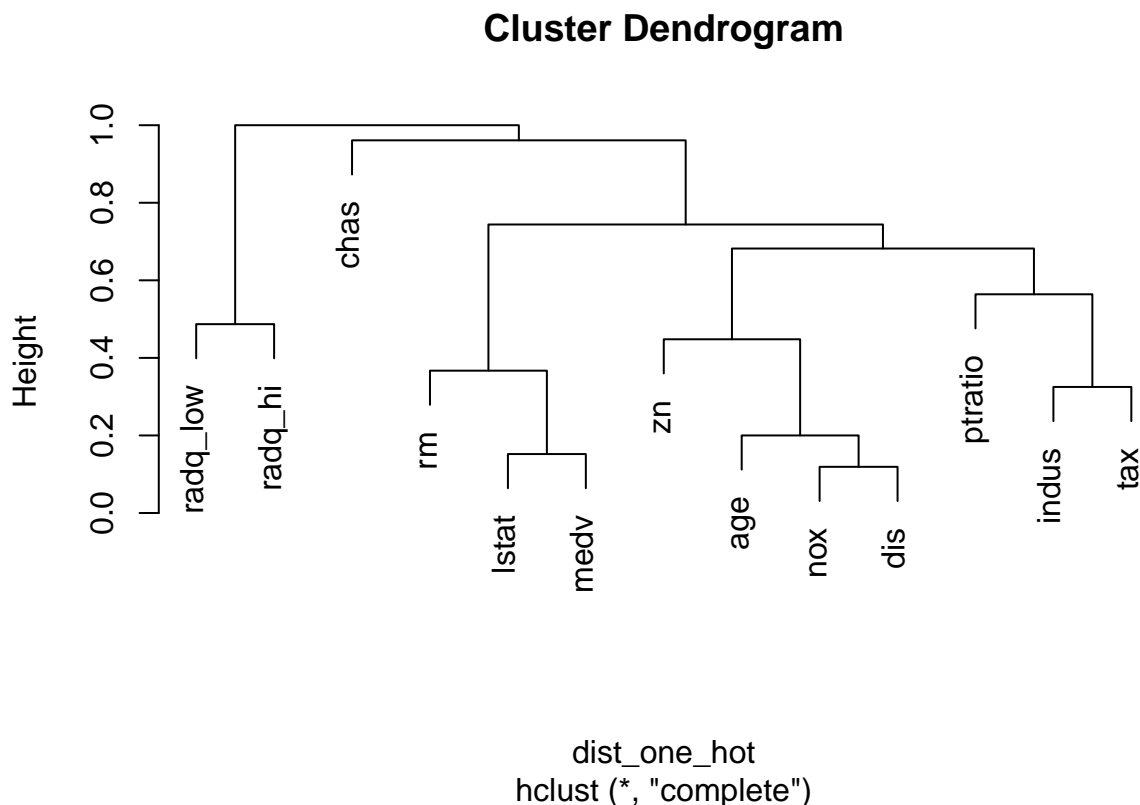
##
## Call:
## glm(formula = target ~ group1_pc1 + group2_pc1 + group2_pc2,
##      family = binomial(link = "logit"), data = df_training_one_hot_pca)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1950     0.1905   1.024   0.306
## group1_pc1   -1.4594     0.1818  -8.027 9.99e-16 ***
## group2_pc1    -0.2843     0.2034  -1.398   0.162
## group2_pc2     0.7457     0.1783   4.181 2.90e-05 ***
## ---
## 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: 291.22  on 462  degrees of freedom
## AIC: 299.22
##
## Number of Fisher Scoring iterations: 6
model_pca2 <- update(model_pca2, . ~ . - group2_pc1)
summary(model_pca2)

##
## Call:
## glm(formula = target ~ group1_pc1 + group2_pc2, family = binomial(link = "logit"),
##      data = df_training_one_hot_pca)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.05301    0.16131   0.329   0.742
## group1_pc1   -1.28804    0.12063 -10.678 < 2e-16 ***
## group2_pc2     0.87594    0.16084   5.446 5.15e-08 ***
## ---
## 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: 293.06 on 463 degrees of freedom
## AIC: 299.06
##
## Number of Fisher Scoring iterations: 6
```

**Model based on Variable Clustering** The dendrogram is a variable clustering technique that shows how the parameters progressively come together at different levels of similarity. It offers another way to visualize correlations between our parameters. In this model, we will use the dendrogram to prune parameters that are similar from the lower branches. In this model, we used the results from a T and Wilcox pairwise test to assist with the parameter selection.

```
dist_one_hot = as.dist(m = 1 - abs(df_training_cor))
par(mar = c(5, 4, 4, 2) + 0.1)
plot(hclust(dist_one_hot))
```



```
sapply(numeric_cols, function(param) {
  pairwise.t.test(
    x = df_training_one_hot[, param],
    g = df_training_one_hot$target,
    pool.sd = FALSE,
    paired = FALSE,
    alternative = "two.sided"
  )$p.value
}) |> sort()
```

```
##          nox          age          dis          indus          tax          lstat
```



```
## 1.486824e-70 3.953661e-52 1.762618e-48 7.522700e-48 2.028465e-45 4.663092e-26
##          zn          medv          ptratio          rm
## 1.545946e-21 3.868621e-09 4.851822e-08 1.036364e-03
```

```
sapply(numeric_cols, function(param) {
  pairwise.wilcox.test(
    x = df_training_one_hot[, param],
    g = df_training_one_hot$target,
    pool.sd = FALSE,
    paired = FALSE,
    alternative = "two.sided"
  )$p.value
}) |> sort()
```

```
##          nox          dis          age          indus          tax          lstat
## 1.505559e-59 7.713151e-46 4.570642e-44 1.169101e-40 8.311193e-38 4.704275e-25
##          zn          medv          ptratio          rm
## 1.999127e-24 4.781087e-18 1.305775e-14 1.331368e-04
```

```
model_dendo <- glm(target ~ radq_hi + chas + lstat + indus + age, binomial(link = "logit"), data=df_train)
summary(model_dendo)
```

```
##
## Call:
## glm(formula = target ~ radq_hi + chas + lstat + indus + age,
##      family = binomial(link = "logit"), data = df_training_one_hot)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.090764   0.560770  -9.078 < 2e-16 ***
## radq_hi      3.827787   0.576717   6.637 3.20e-11 ***
## chas1        0.266750   0.554497   0.481  0.6305
## lstat        0.003477   0.028225   0.123  0.9020
## indus        0.061046   0.025708   2.375  0.0176 *
## age          0.050076   0.008265   6.059 1.37e-09 ***
## ---
## 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: 297.52  on 460  degrees of freedom
## AIC: 309.52
##
## Number of Fisher Scoring iterations: 6
```

## Model Using Quasi-Logit

### Model comparison

```
library(pscl)

stats <- LRstats(model_full, model_full_log,
  backward_model, model_corr, model_pca, model_dendo)

stats$McFaddenR2 <- NA
```

```

stats$Accuracy <- NA
stats$Precision <- NA
stats$Recall <- NA
stats$Sensitivity <- NA
stats$Specificity <- NA
stats$F1_score <- NA
stats$AUC <- NA

enhanceEvaluationMetrics <- function(df, model_name) {
  model <- get(model_name)

  if (model_name == "model_full_log") {
    model_data <- df_training_1h_log
  } else if (model_name == "model_pca") {
    model_data <- df_training_one_hot_pca
  } else {
    model_data <- df_training_one_hot
  }

  df[model_name, "McFaddenR2"] <- pR2(model)["McFadden"]

  pred_probs <- predict(model, type = "response")

  pred_probs_factor <- as.factor(ifelse(pred_probs > 0.5, 1, 0))
  conf_matrix <- confusionMatrix(pred_probs_factor, model_data$target)
  df[model_name, "Accuracy"] <- conf_matrix$overall['Accuracy']
  df[model_name, "Precision"] <- conf_matrix$byClass['Precision']
  df[model_name, "Recall"] <- conf_matrix$byClass['Recall']
  df[model_name, "F1_score"] <- conf_matrix$byClass['F1']

  #roc_model <- roc(as.factor(model_data$target), pred_probs)
  #plot(roc_model, main = "ROC Curve using pROC", col = "red", lwd = 2)

  TPR <- conf_matrix$byClass["Sensitivity"]
  FPR <- conf_matrix$byClass["Specificity"]
  df[model_name, "Sensitivity"] <- TPR
  df[model_name, "Specificity"] <- FPR
  # roc_auc not working
  df[model_name, "AUC"] <- MLmetrics::AUC(y_true = model_data$target, y_pred = pred_probs)

  return(df)
}

models <- list(model_full, model_full_log, backward_model, model_corr, model_pca, model_dendo)

# Loop through the list of models and update the dataframe for each
for (model_name in rownames(stats)) {
  stats <- enhanceEvaluationMetrics(stats, model_name)
}

## fitting null model for pseudo-r2
## fitting null model for pseudo-r2

```

```
## fitting null model for pseudo-r2
## fitting null model for pseudo-r2
## fitting null model for pseudo-r2
## fitting null model for pseudo-r2
```

```
stats
```

```
## Likelihood summary table:
```

##	AIC	BIC	LR	Chisq	Df	Pr(>Chisq)	McFaddenR2	Accuracy
## model_full	208.97	266.99	180.97	452		1.00000	0.71981	0.92275
## model_full_log	214.45	272.47	186.45	452		1.00000	0.71132	0.92918
## backward_model	207.97	241.13	191.97	458		1.00000	0.70277	0.92704
## model_corr	436.57	457.29	426.57	461		0.87313	0.33955	0.78326
## model_pca	303.25	332.26	289.25	459		1.00000	0.55216	0.85193
## model_dendo	309.52	334.39	297.52	460		1.00000	0.53935	0.85837
##	Precision	Recall	Sensitivity	Specificity	F1_score	AUC		
## model_full	0.92405	0.92405	0.92405	0.92140	0.92405	0.97771		
## model_full_log	0.93220	0.92827	0.92827	0.93013	0.93023	0.97483		
## backward_model	0.93939	0.91561	0.91561	0.93886	0.92735	0.97291		
## model_corr	0.75564	0.84810	0.84810	0.71616	0.79920	0.84462		
## model_pca	0.83871	0.87764	0.87764	0.82533	0.85773	0.94054		
## model_dendo	0.84615	0.88186	0.88186	0.83406	0.86364	0.93288		

**Checking the Model's Conditions** We will examine the following key conditions for fitting a logistic model:

1. dependent variable is binary
2. large enough sample
3. observations are independent, not matched
4. independent (predictor) variables do not correlate too strongly with each other
5. linearity of independent variables and log odds
6. no outliers in data

### Confidence Interval

##	2.5 %	97.5 %
## (Intercept)	-51.220070886	-25.123420900
## zn	-0.167059221	-0.009805196
## indus	-0.277621666	-0.079569873
## chas1	-0.437329365	2.774890137
## nox	43.146882327	78.772474084
## rm	-2.343126185	0.413916124
## age	-0.004788160	0.047225213
## dis	0.345497559	1.314977464
## tax	-0.003432766	0.005692181
## ptratio	-0.145335266	0.382263413
## lstat	-0.049284710	0.156305233
## medv	0.064587356	0.297709207
## radq_low	0.575629081	2.654531102
## radq_hi	3.463840203	7.223064642

```
exp(coef(model_full))
```

## Odds Ratio

```
## (Intercept)          zn          indus          chas1          nox          rm
## 5.965983e-17 9.228761e-01 8.416752e-01 3.251869e+00 6.638993e+25 3.842654e-01
##          age          dis          tax          ptratio          lstat          medv
## 1.020511e+00 2.254949e+00 1.000962e+00 1.126409e+00 1.055980e+00 1.192431e+00
##          radq_low          radq_hi
## 4.772780e+00 1.670461e+02
```

```
model_summary <- summary(model_full)
#param_pvalue <- model_summary$coefficients["target", "Pr(>|z|)"]
anova(model_full, test = "Chisq")
```

## ANOVA Test

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: target
##
## Terms added sequentially (first to last)
##
##
##          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                465      645.88
## zn          1  127.411            464      518.46 < 2.2e-16 ***
## indus       1   86.433            463      432.03 < 2.2e-16 ***
## chas        1    1.274            462      430.76 0.2589811
## nox         1  150.804            461      279.95 < 2.2e-16 ***
## rm          1    6.755            460      273.20 0.0093493 **
## age         1    0.217            459      272.98 0.6415150
## dis         1    7.981            458      265.00 0.0047265 **
## tax         1   14.205            457      250.80 0.0001639 ***
## ptratio     1    3.659            456      247.14 0.0557589 .
## lstat       1    0.640            455      246.50 0.4236364
## medv        1   12.753            454      233.74 0.0003555 ***
## radq_low    1    0.504            453      233.24 0.4775972
## radq_hi     1   52.270            452      180.97 4.838e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```