Using Losistic Regression to Predict Boston Neighborhood Crime Levels

Daniel Brooks (daniel.brooks@spsmail.cuny.edu), Daniel Fanelli (daniel.fanelli@spsmail.cuny.edu), Christopher Fenton (christopher.fenton@spsmail.cuny.edu), James Hamski (james.hamski@spsmail.cuny.edu), Youqing Xiang (youqing.xiang@spsmail.cuny.edu)

7/3/2016

The purpose of this analysis is to build a logistic regression model that will predict whether a particular neighborhood in Boston is above or below the median crime level for the city.

Our dataset includes information on 466 Boston neighborhoods. Each neighborhood has 13 potential predictor variables, and 1 response variable. The response variable is “target”, which is “1” if the neighbhorhood is above the city’s median crime level, and 0 if not.

1 Data Exploration

The first thing we checked for was if there was missing data in any of the variables. There was no missing data.

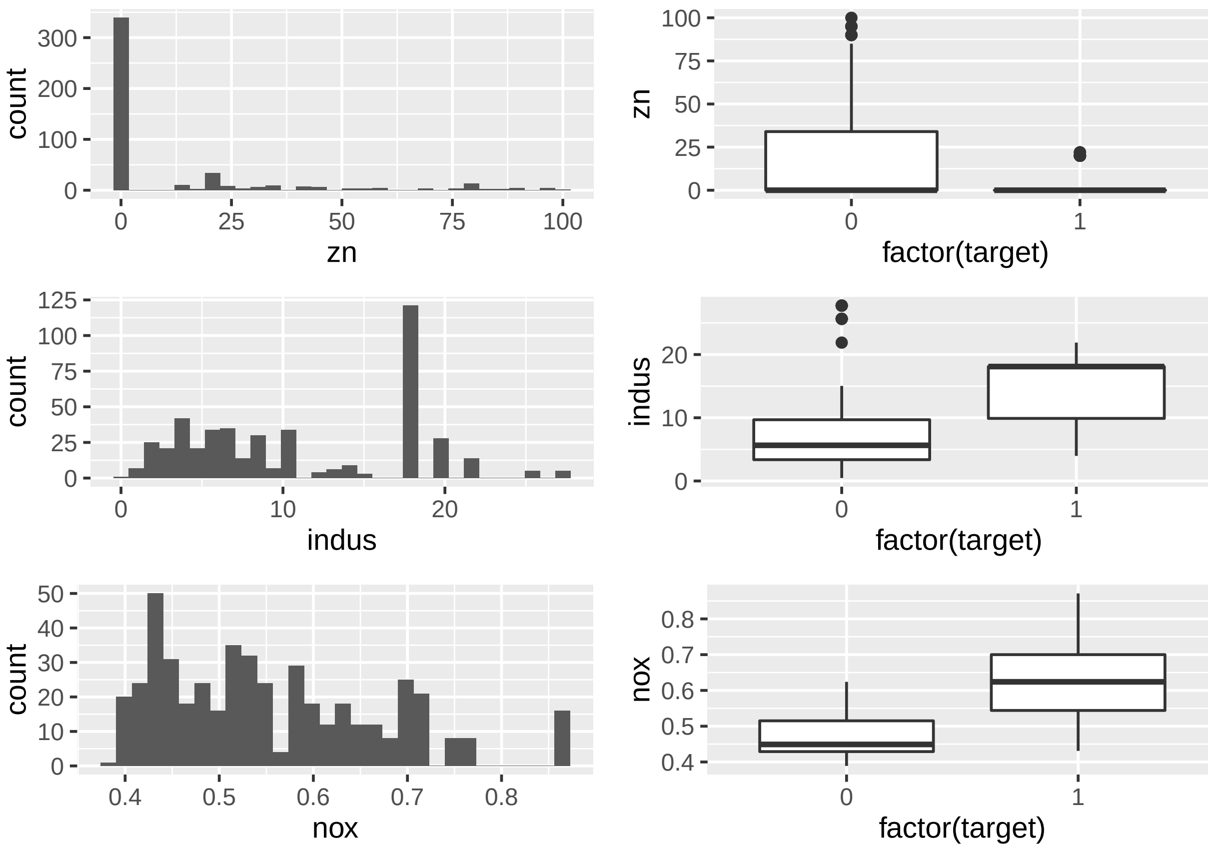
Of the 13 predictor variables, 12 were numeric and 1 was caterogical. The categorical data would need to be converted in order to be approrpriately used in a generalized linear model.

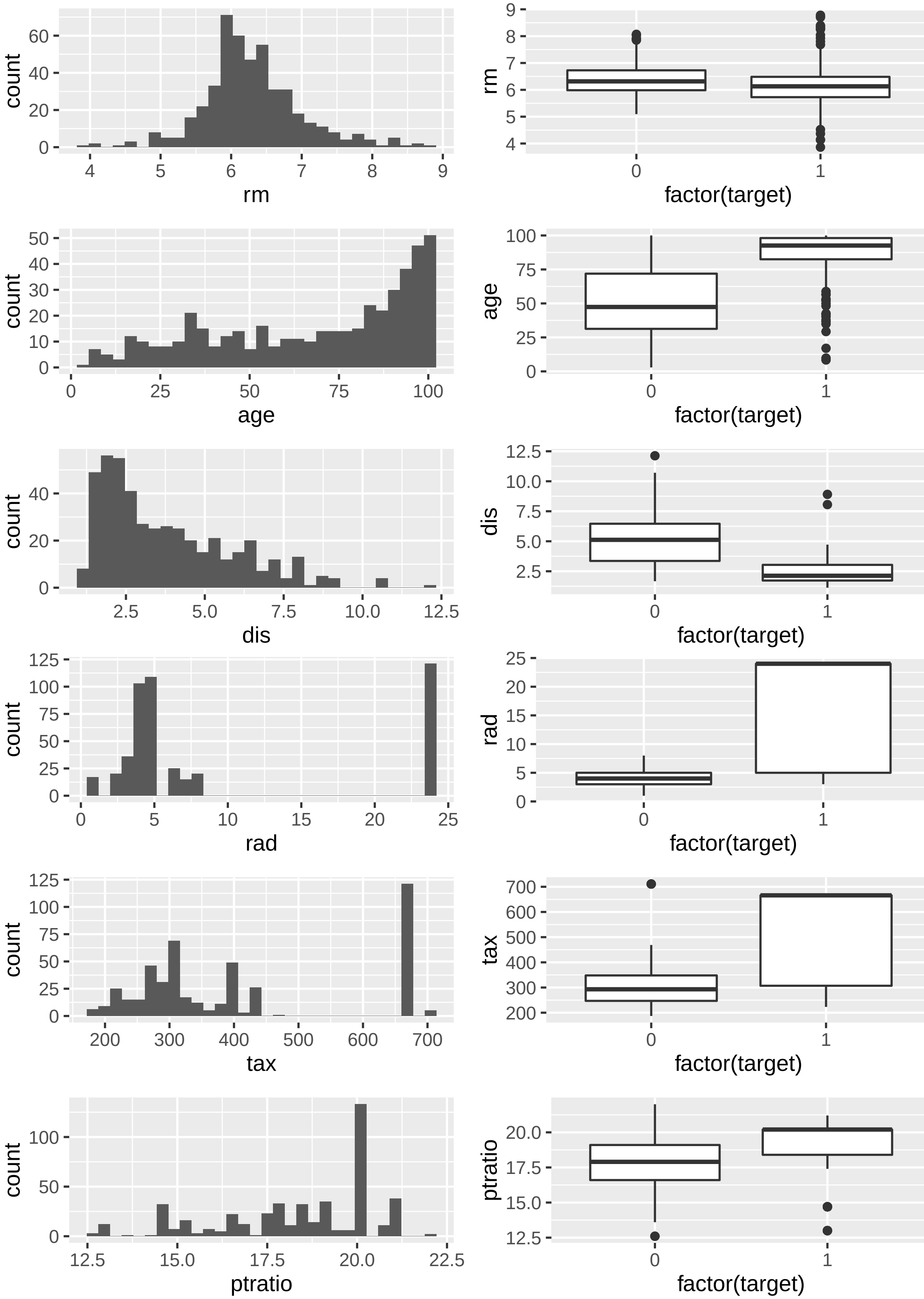
We also examined the distribution of the response variable, `target`. 229 neighborhoods were marked with a 1, thus above the median crime level, and 237 were not. The fact that the two numbers were not within 1 of each other implies that some neighbohood data was not included. However, the split was roughly even, allowing us to proceed.

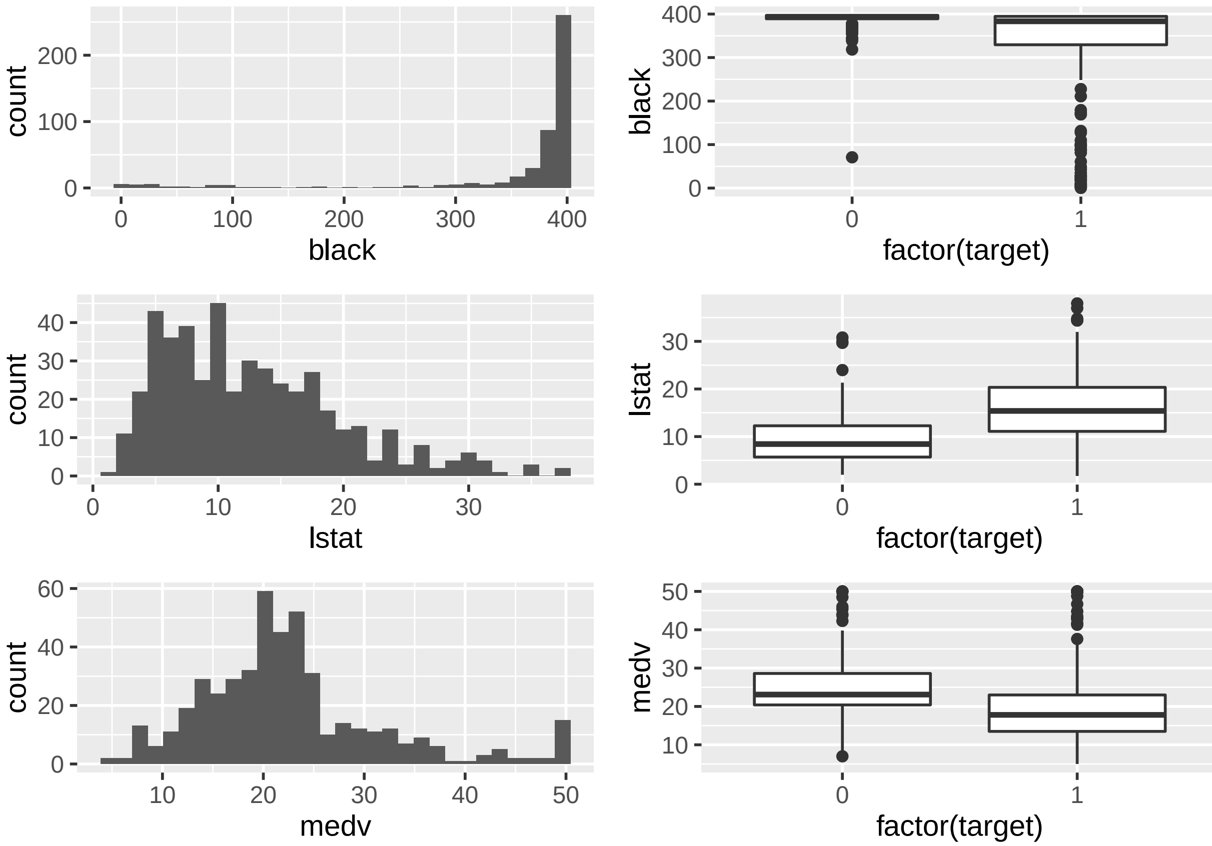
Figure 1.1

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

```
##  Max.   :711.0   Max.   :22.0   Max.   :396.90   Max.   :37.970
##       medv           target
##  Min.   : 5.00   Min.   :0.0000
##  1st Qu.:17.02   1st Qu.:0.0000
##  Median :21.20   Median :0.0000
##  Mean   :22.59   Mean   :0.4914
##  3rd Qu.:25.00   3rd Qu.:1.0000
##  Max.   :50.00   Max.   :1.0000
```




count
rm
factor(target)
age
dis
rad
tax
ptratio



Using histograms of variables and boxplots of variables grouped by predictor (Figure 1.1), we could see some outliers that may need to be dealt with. Also, correlations with the response variable (`target`) showed that variables such as `zn`, `indus`, `dis` and `rad` may have more potential as predictor variables while `chas` and `rm` may not be as useful.

Figure 1.2

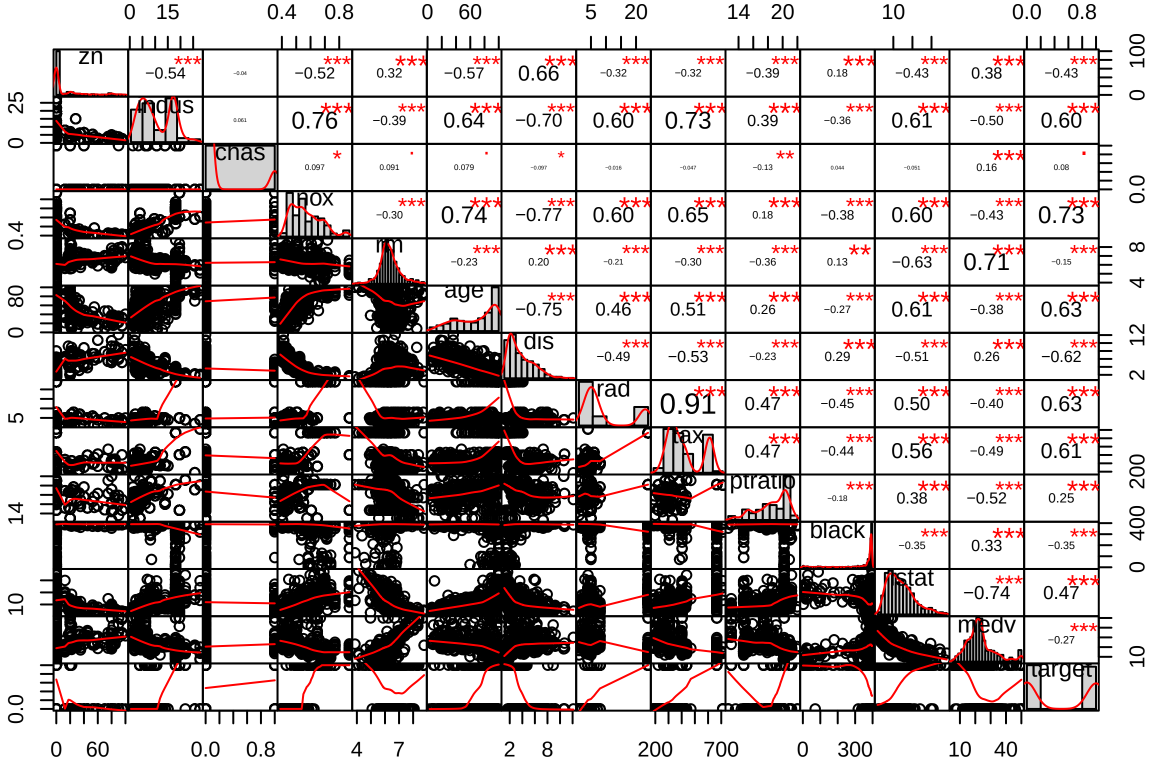


Figure 1.2 shows that every variable other than adjancency to the Charles river has correlation significance level of .001 with our response variable, `target`. This graph also shows that multicollinearity is something that will have to be dealt with in this model. Principal component analysis could be useful to mitigate this.

The fact that the predictor variables show a high degree of correlation with each other makes intuitive sense. Most variables relate to a notion of "desirability", which would hypothetically have significant impacts on other variables.

For instance, one might suppose that a high degree of industrial real estate in a neighborhood would have a negative effect on real estate values. One might than hypothesize that a neighborhood with lower median real estate values would be more highly susceptible to higher than usual crime.

At least in the provided dataset, there are strong correlations that bear out both of those hypotheses. `Indus`, which describes the proportion of non-retail businesses in a neighborhood, is negatively correlated by a factor of -.49617 with `Medv`, the median value of owner occupied homes; `Medv` is negative correlated (-.27) with higher than normal crime rates.

2 Data Preparation

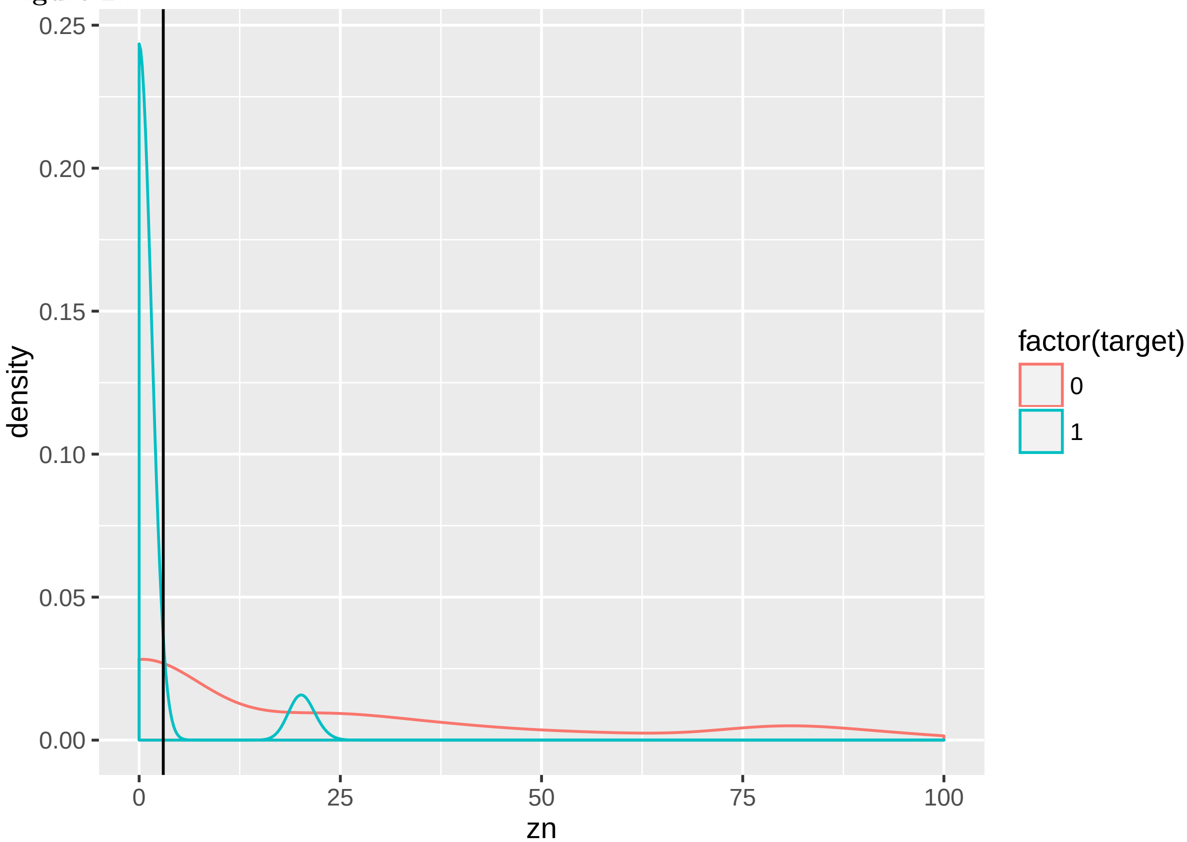
We used two separate approaches to preparing the data. One approach dealt with each variable separately, while the second approach normalized all variables.

2.1 Individual Variable Preparation Approach

This approach looked at each variable independently of the others.

2.1.1 zn

Figure 2.1



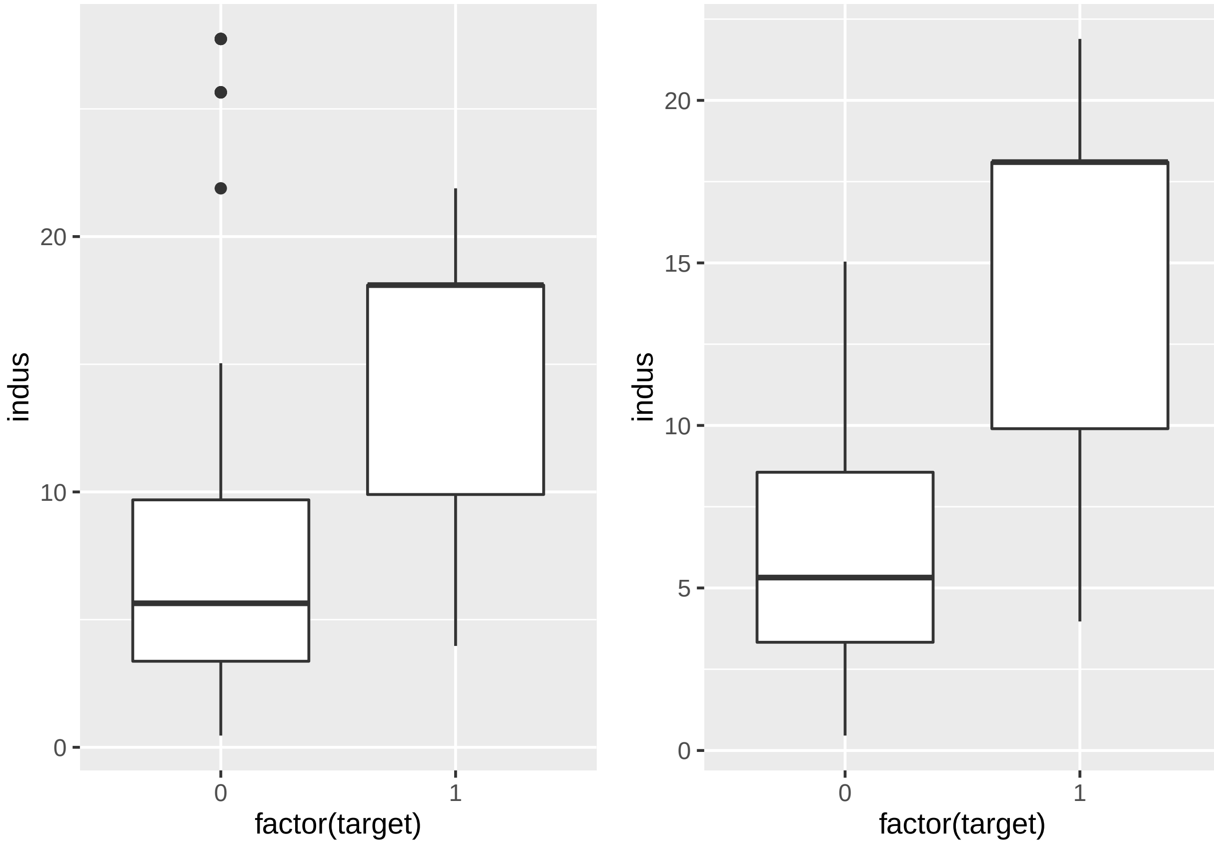
znN	Target	Freq
0	0	125
1	0	112
0	1	214
1	1	15

Using Figure 2.1, we decided to transform the numeric `z` variable to a derived categorical variable called `znN`. For this new `znN` variable, *1* means more than 3% of residential land zoned for large lots (over 25000 square feet), and *0* means less than or equal to 3% of residential land zoned for large lots (over 25000 square feet).

2.1.2 indus

Each observation in the dataset is for a different Boston area neighborhood. This means the data are somewhat arbitrarily binned by geography - the area that is considered a neighborhood is influenced by historical factors. For instance, we see high-value outliers in the `indus` variable because historic land-use and zoning laws mean that non-retail business use is concentrated in specific industrial areas. In order to remove these potential leverage points, we removed neighborhoods with indus values above 20. However, we do not consider this to be invalid data.

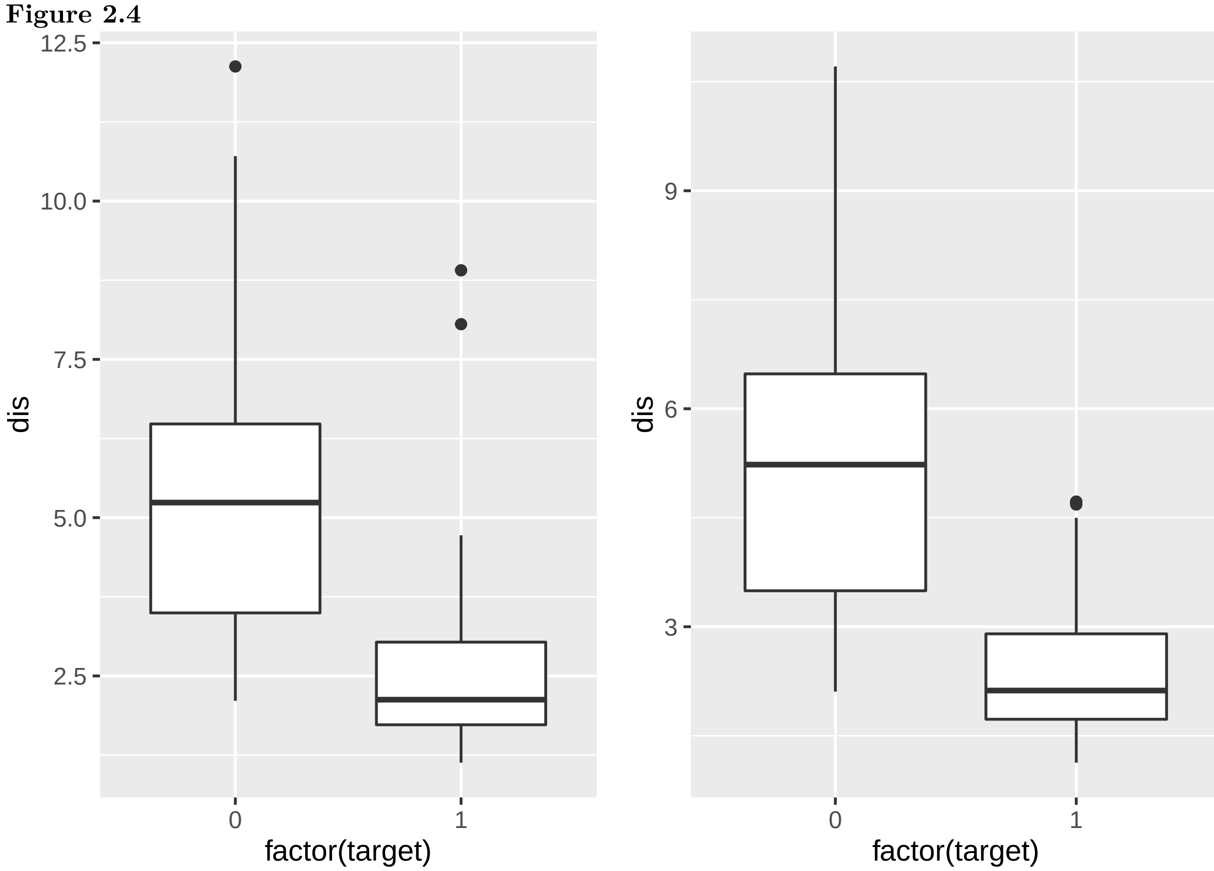
Figure 2.3



For `indus`, we removed observations where `indus` was greater than 20 and `target` was 0.

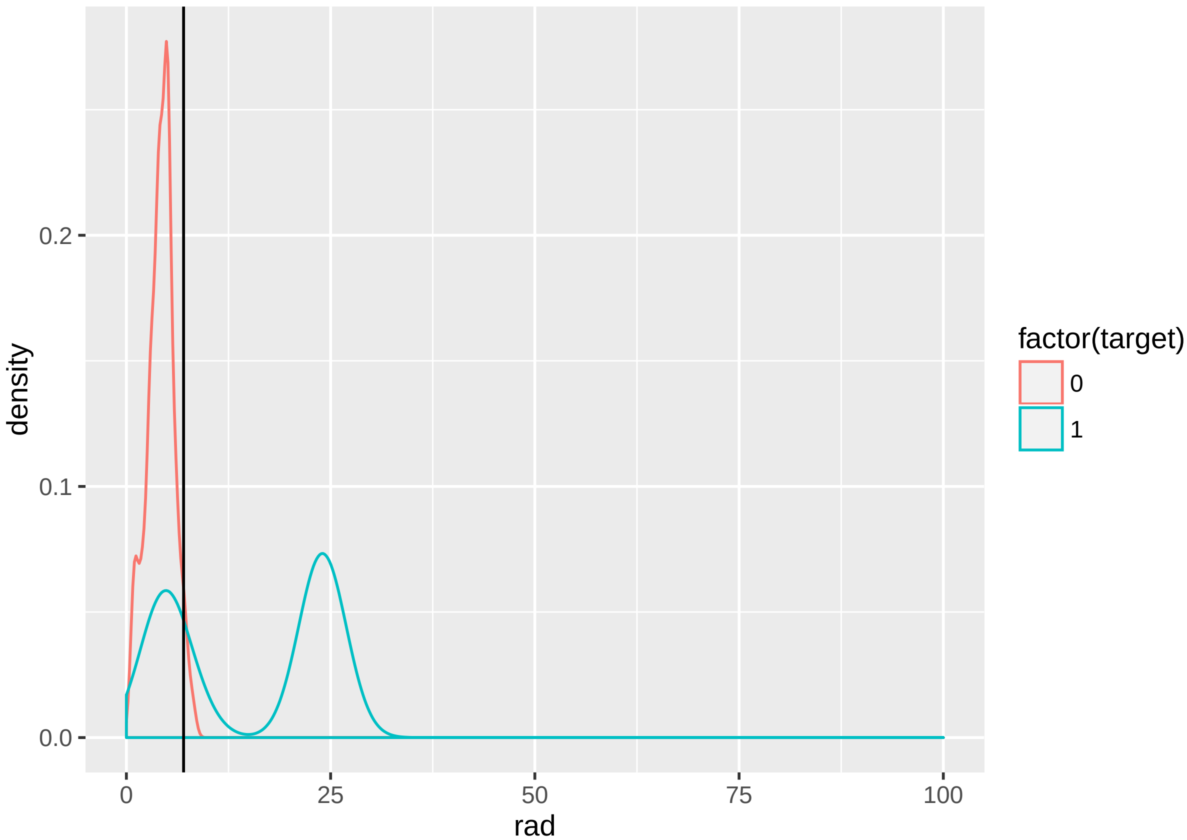
2.1.3 dis

Figure 2.4



We removed observations where `dis` was greater than 11 and `target` was 0, and observations where `dis` was greater than 7.5 and `target` was 1.

2.1.4 7. rad



radN	Target	Freq
0	0	221
1	0	4
0	1	90
1	1	137

Here we applied the same method as `zn`. We set up a new variable called `radN`, where *1* means index of accessibility to radial highways was greater than 7 and *0* means index of accessibility to radial highways was less than or equal to 7.

2.1.5 Summary

Under our first approach, we removed 14 rows and added 2 new variables: `znN` and `radN`.

2.2 Normalization Approach

Since the variables were on a variety of scales, we created a dataset of centered (predictor variable mean subtracted from each observation) and scaled (each observation divided by the predictor variable's standard deviation) data. The two datasets are distinguished as 'non-normalized' and 'normalized' below, with the latter having been centered and scaled using R's 'scale' function.

3 Build Models

Since the evaluation data set did not include the response variable, we were not able to use it to cross validate our models. Instead, we split the training data on a 70-30 training/testing split to evaluate our models.

3.1 Model 1: Non-normalized Baseline

This model used the non-normalized original variables. It utilized the probit for the link function. The model began with all the original values, and via backwards selection removed the `zn`, `chas`, `rm`, `dis`, and `black` variables.

```
##
## Call:
## glm(formula = target ~ indus + nox + age + rad + tax + ptratio +
##     lstat + medv, family = binomial(link = "probit"), data = training)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.343  -0.026   0.000   0.000   2.882
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -24.171357   4.325079  -5.589 2.29e-08 ***
## indus         0.134991   0.060490   2.232 0.025639 *
## nox          28.874385   5.510981   5.239 1.61e-07 ***
## age           0.015291   0.007279   2.101 0.035654 *
## rad           0.491912   0.136730   3.598 0.000321 ***
## tax          -0.017047   0.004355  -3.914 9.07e-05 ***
## ptratio       0.344176   0.103985   3.310 0.000933 ***
## lstat         0.099590   0.035125   2.835 0.004579 **
## medv          0.079078   0.029423   2.688 0.007196 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 438.75  on 316  degrees of freedom
## Residual deviance: 102.98  on 308  degrees of freedom
## AIC: 120.98
##
## Number of Fisher Scoring iterations: 11
```

From the output, the model is as follows:

the log odds of `target` = -24.17 + 0.135 `indus` + 28.874 `nox` + 0.015 `age` + 0.492 `rad` - 0.017 `tax` + 0.344 `ptratio` + 0.1 `lstat` + 0.079 `medv`

We can see that `indus`, `nox`, `age`, `rad`, `ptratio`, `lstat`, and `medv` have positive effects on `target`, but `tax` has a negative effect on `target`.

For this model, the most unexpected result is that `medv` (median value of owner-occupied homes in $1000s) has a positive effect on crime. And if we go back to check figure 1.2, we saw the weak negative correlationship (-0.27) between `medv` and `target`. Since this model includes `indus`, `nox`, `age`, `rad`, `ptratio`, `lstat`, `medv` and `tax`, **multicollinearity** could be the root cause.

3.2 Model 2: Non-normalized with derived variables

This model also used backward selection on the non-normalized variables and the probit for the link function, but instead used the derived variables from part 2, instead of their original counterparts.

```
##
## Call:
## glm(formula = target ~ nox + tax + lstat + radN, family = binomial(link = "probit"),
##     data = training)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.32446  -0.04097   0.00000   0.02465   2.62981
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -16.708448   2.552283  -6.546 5.89e-11 ***
## nox          32.430637   5.246318   6.182 6.35e-10 ***
## tax          -0.005576   0.002087  -2.672  0.00754 **
## lstat         0.080376   0.027954   2.875  0.00404 **
## radN1         2.932473   0.556149   5.273 1.34e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 438.75  on 316  degrees of freedom
## Residual deviance: 107.01  on 312  degrees of freedom
## AIC: 117.01
##
## Number of Fisher Scoring iterations: 9
```

From the output, the model is as follows:

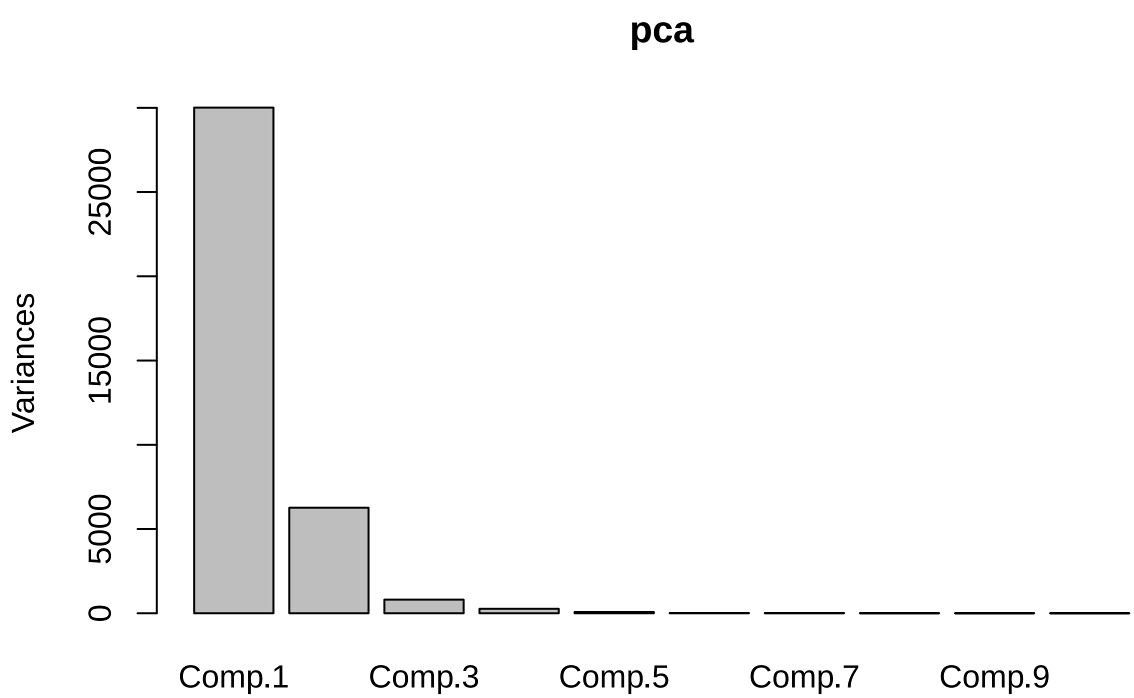
the log odds of `target` = -16.71 + 32.43 `nox` - 0.006 `tax` + 0.08 `lstat` + 2.93 radN1

We can see that `nox` and `lstat` have positive effects on the log odds of `target`, but `tax` has a negative effect on the log odds of `target`. In addition, when radN equals to 1, the log odds of `target` increases by 2.93.

This model only include 4 variables and we transformed `rad` to categorical variable `radN`, so **multicollinearity** became less of an issue to us. Although we saw a positve correlation between `tax` and `target` in figure 1.2, both Model 1 and Model 2 show that `tax` has a negative yet weak effect on the log odds of `target`, which fits our intuition.

3.3 Model 3: Principle Component Analysis

For this model, we used an orthogonal transformation to convert our variables (normalization applied to all variables) into a set of values of linearly uncorrelated variables, which is called principal components. And then we chose the first two principal components that account for around 95% proportion of variance in the data. Finally, we used those chosen principal components to build a logistic regression model.




```
##   target      Comp.1     Comp.2
## 1      1    7.982215  -10.79214
## 2      1   14.663799  -36.79382
## 3      1 -237.185595 -109.13296
## 4      0  115.531924   17.45648
## 5      0  214.333148   31.11811
## 6      0   35.789981  -29.36113
```

```
##
## Call:
## glm(formula = target ~ ., data = training_pca)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.59191  -0.27404  -0.10935   0.04818   0.83770
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.5116620  0.0217455  23.530   <2e-16 ***
## Comp.1      -0.0018248  0.0001246 -14.639   <2e-16 ***
## Comp.2      -0.0001726  0.0002618  -0.659     0.51
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1496637)
##
##     Null deviance: 79.073  on 316  degrees of freedom
## Residual deviance: 46.994  on 314  degrees of freedom
## AIC: 302.49
##
## Number of Fisher Scoring iterations: 2
```

From the output, the formula we got was the following:

the log odds of `target` = 0.512 - 0.00182 `Comp.1` - 0.00017 `Comp.2`

For this model, since we did an orthogonal transformation of variables, **multicollinearity** was no longer an issue. Both principal components have negative effects on the log odds of `target`. And we only chose two principal components, so we keep the model as is, even though the p value of `Comp.2` is not significant.

3.4 Model 4: Normalized Backward Selection

This model used the normalized data, the logit in the link function, and used backward selection using R's step() function with the backward option to arive at the below model. Insignificant variables (those with a P value > .05) were discarded.

```
##
## Call:
## glm(formula = target ~ nox + age + rad + tax + ptratio + black +
##     medv, family = binomial, data = training_norm)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.85486  -0.18466  -0.01769   0.00474   2.84168
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.7989     0.9758   3.893 9.90e-05 ***
## nox           3.7313     0.6959   5.362 8.23e-08 ***
## age           0.7877     0.3430   2.297 0.021636 *
## rad           6.2349     1.4908   4.182 2.89e-05 ***
## tax          -1.9056     0.5618  -3.392 0.000693 ***
## ptratio       1.0324     0.3163   3.264 0.001099 **
## black        -4.5850     1.5835  -2.896 0.003785 **
## medv          0.8730     0.3800   2.297 0.021603 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 452.80  on 326  degrees of freedom
## Residual deviance: 131.34  on 319  degrees of freedom
## AIC: 147.34
##
## Number of Fisher Scoring iterations: 9
```

The backward selection model is below:

the log odds of `target` = 3.80 + 3.73 `nox` + 0.79 `age` + 6.23 `rad` - 1.91 `tax` 1.03 `ptratio` - 4.59 `black` + .87 `medv`

As in model 1, this model predicts a positive effect of 'medv' (median value of owner occupied homes) on `target` or crime, which is counterintuitive. This is an indication that multicollinearity could be an issue.

The high number of predictors (7) in this model also has the effect of decreasing the interpretive ability of the model.

3.5 Model 5: Normalized Forward Selection

This model used the normalized data, the logit in the link function, and used forward selection using R's step() function with the forward option to arive at the below model.

```
##
## Call:
## glm(formula = target ~ nox + rad + tax + ptratio + black + medv +
##     age + dis + zn + lstat, family = binomial, data = data3b)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7582  -0.1591  -0.0017   0.0029   3.3169
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   3.4969     1.0333   3.384 0.000714 ***
## nox           4.4006     0.9301   4.731 2.23e-06 ***
## rad           6.4586     1.6456   3.925 8.68e-05 ***
## tax          -1.7948     0.5993  -2.995 0.002746 **
## ptratio       0.9818     0.3278   2.995 0.002746 **
## black        -4.2670     1.5925  -2.679 0.007375 **
## medv          1.4443     0.5357   2.696 0.007017 **
## age           0.8985     0.3753   2.394 0.016661 *
## dis           1.0075     0.5447   1.850 0.064354 .
## zn           -1.6529     0.8970  -1.843 0.065360 .
## lstat         0.3301     0.3718   0.888 0.374665
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 452.80  on 326  degrees of freedom
## Residual deviance: 124.91  on 316  degrees of freedom
## AIC: 146.91
##
## Number of Fisher Scoring iterations: 9
```

The normalized forward selection model is below:

the log odds of `target` = 3.50 + 4.40 `nox` + 6.46 `rad` - 1.79 `tax` + 0.98 `ptratio` - 4.27 `black` + 1.44 `medv` + 0.90 `age` + 1.01 `dis` - 1.65 `zn` + 0.33 `lstat`

Many of the concerns that appeared in Model 4 (normalized backward selection) reappear in Model 5. Medv again positvely predicts a higher crime level, indicating multicollinearity.

This model also includes a large number of predictors (11), even larger than model 4. The inclusion of some insginificant variables was used to compare it's performance against a similar model that did not include insignificant variables (Model 4).

4 Model Selection

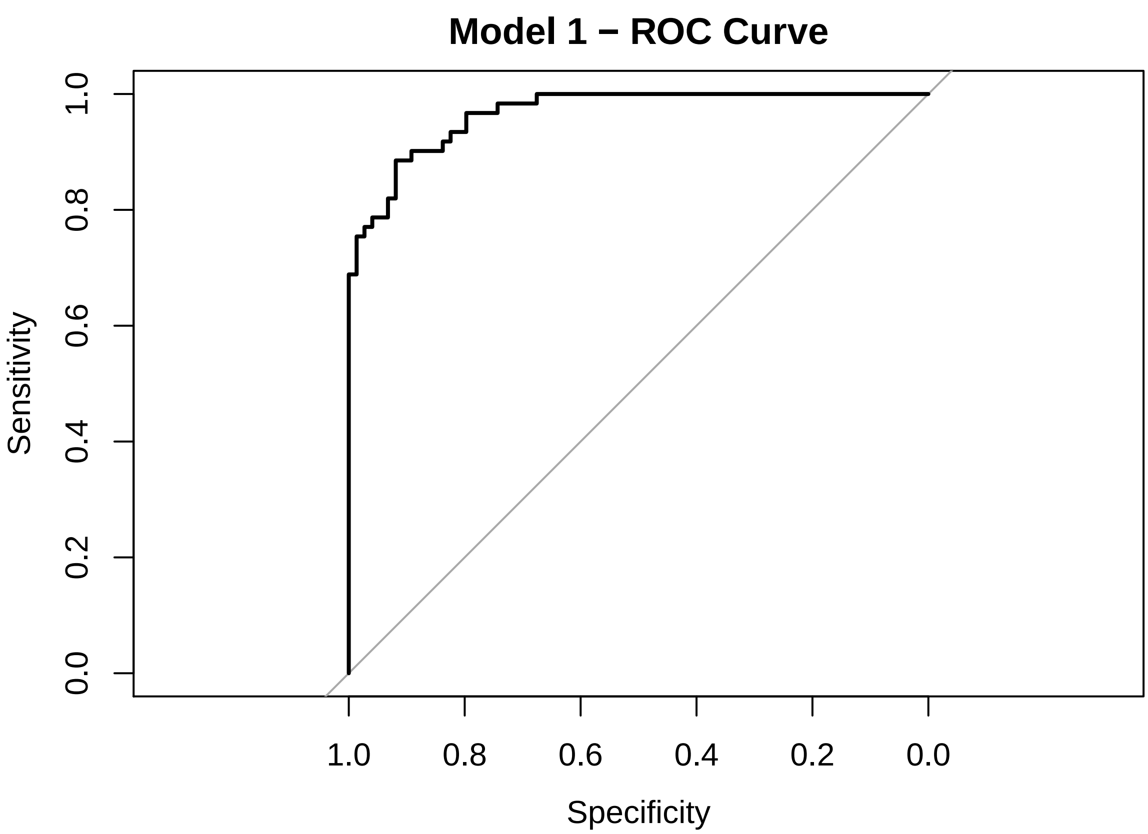
During our modeling process, **multicollinearity** was the biggest issue we faced. Model 3 completely took care of this issue; Model 2 minimized this issue to a great extent and Models 1, 4 and 5 partially solved this problem. We used cross validation technique to further study our models and try to find the best model.

4.1 Confusion Matrices

Table 3: Confusion Matrix

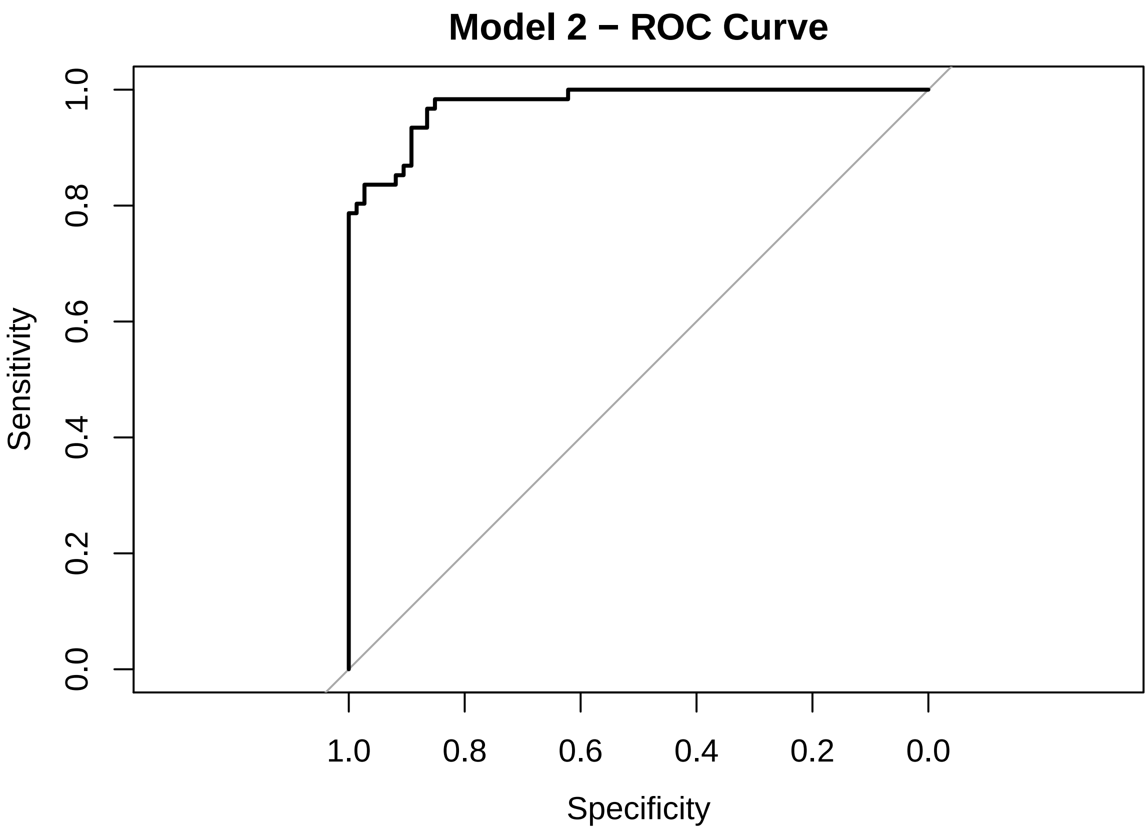
	Model1	Model2	Model3	Model4	Model5
Accuracy	0.8814815	0.9037037	0.8222222	0.8920863	0.8920863
Kappa	0.7621145	0.8075447	0.6315670	0.7840050	0.7840050
AccuracyLower	0.8146770	0.8409602	0.7471282	0.8282642	0.8282642
AccuracyUpper	0.9307163	0.9477237	0.8826473	0.9383301	0.9383301
AccuracyNull	0.5481481	0.5481481	0.5481481	0.5179856	0.5179856
AccuracyPValue	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
McnemarPValue	0.4532547	0.0960923	0.0005202	1.0000000	1.0000000
Sensitivity	0.9016393	0.9508197	0.6557377	0.8888889	0.8888889
Specificity	0.8648649	0.8648649	0.9594595	0.8955224	0.8955224
Pos Pred Value	0.8461538	0.8529412	0.9302326	0.9014085	0.9014085
Neg Pred Value	0.9142857	0.9552239	0.7717391	0.8823529	0.8823529
Prevalence	0.4518519	0.4518519	0.4518519	0.5179856	0.5179856
Detection Rate	0.4074074	0.4296296	0.2962963	0.4604317	0.4604317
Detection Prevalence	0.4814815	0.5037037	0.3185185	0.5107914	0.5107914
Balanced Accuracy	0.8832521	0.9078423	0.8075986	0.8922056	0.8922056

4.2 ROC Curve and Area Under the Curve

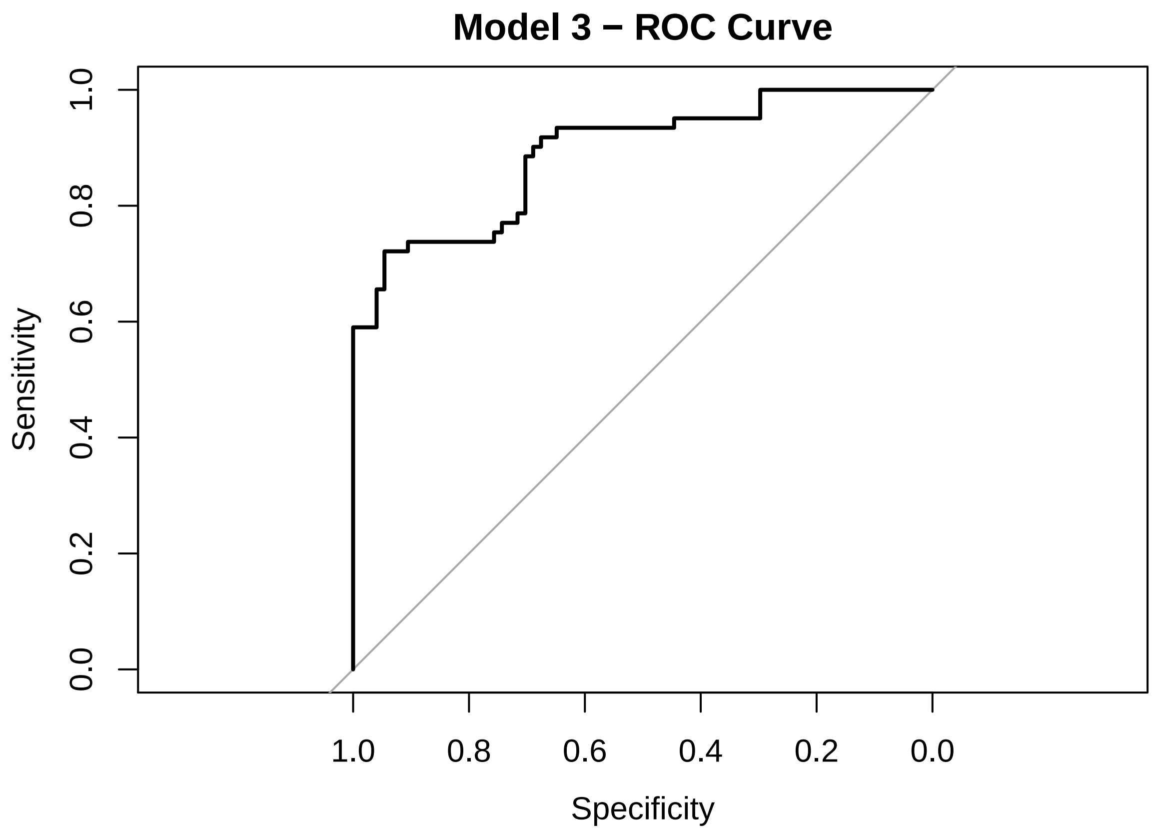



```
##
## Call:
```

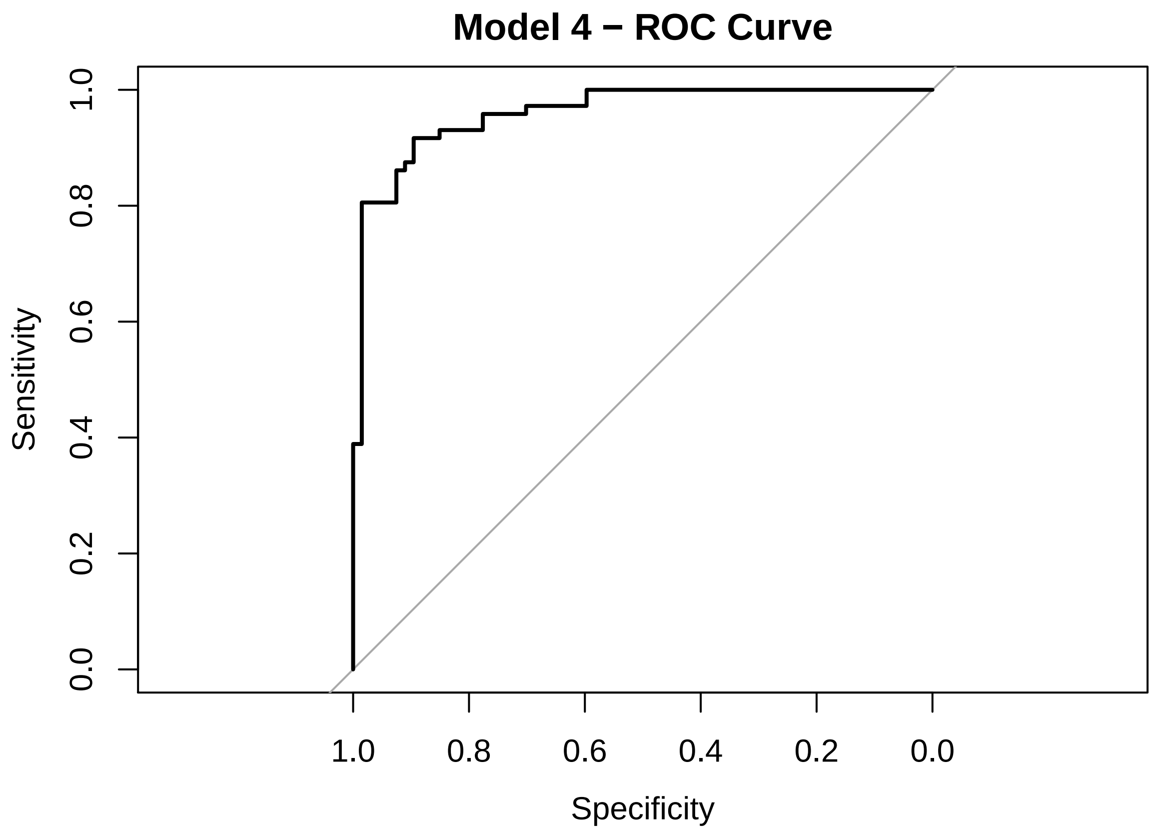
```
## roc.formula(formula = factor(target) ~ predict_1, data = testing)
##
## Data: predict_1 in 74 controls (factor(target) 0) < 61 cases (factor(target) 1).
## Area under the curve: 0.967
```



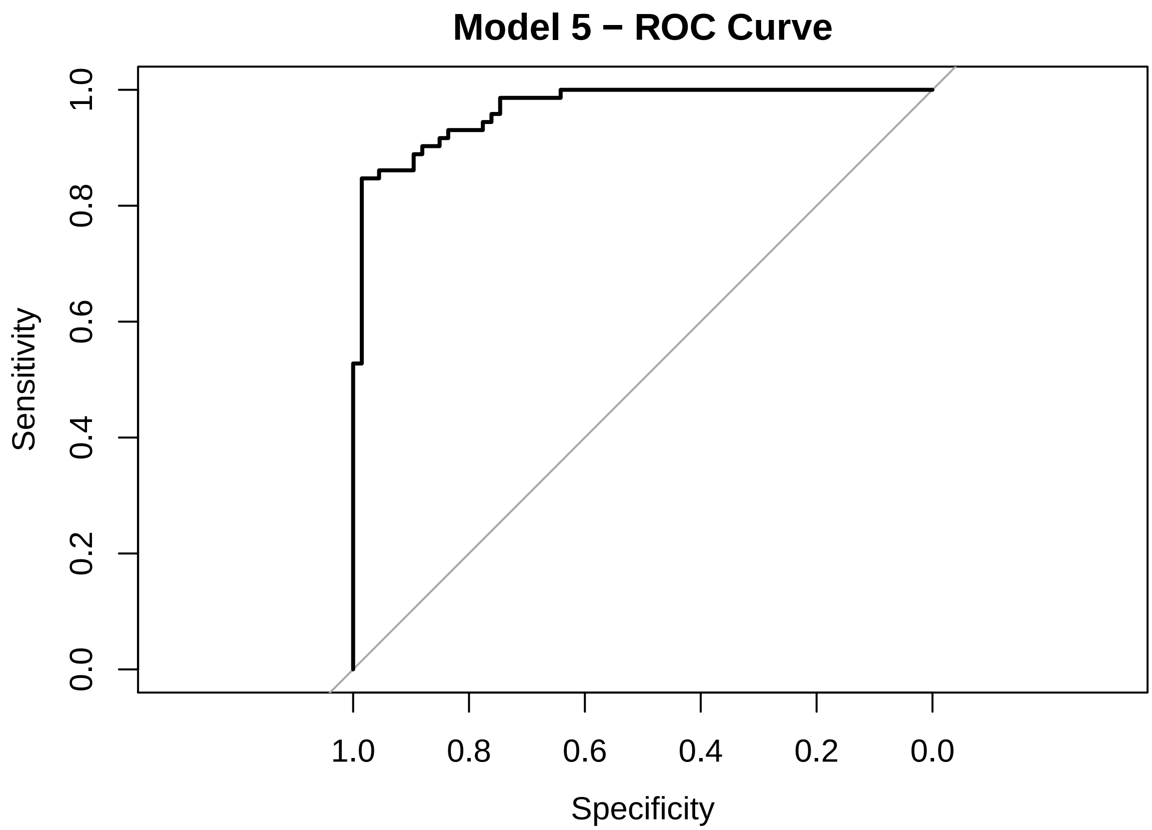

```
##
## Call:
## roc.formula(formula = factor(target) ~ predict_2, data = testing)
##
## Data: predict_2 in 74 controls (factor(target) 0) < 61 cases (factor(target) 1).
## Area under the curve: 0.9759
```




```
##
## Call:
## roc.formula(formula = factor(target) ~ predict_3, data = testing_pca)
##
## Data: predict_3 in 74 controls (factor(target) 0) < 61 cases (factor(target) 1).
## Area under the curve: 0.8903
```




```
##
## Call:
## roc.formula(formula = factor(target) ~ predict_4, data = testing_norm)
##
## Data: predict_4 in 67 controls (factor(target) 0) < 72 cases (factor(target) 1).
## Area under the curve: 0.9604
```




```
##
## Call:
## roc.formula(formula = factor(target) ~ predict_5, data = testing_norm)
##
## Data: predict_5 in 67 controls (factor(target) 0) < 72 cases (factor(target) 1).
## Area under the curve: 0.9672
```

Table 4: Area under the curve

Model	AUC
Model 1	0.9669916
Model 2	0.9758529
Model 3	0.8903412
Model 4	0.9604063
Model 5	0.9672471

4 of the 5 area under the curve measurements were above .96, indicating that all but model 3 have excellent predictive power when measured against the test set.

4.3 Log-likelihood/AIC/BIC

Table 5: Log-likelihood/AIC/BIC

	Log Likelihood	AIC	BIC
Model 1	-51.49	120.98	154.81
Model 2	-53.51	117.01	135.81
Model 3	-147.25	302.49	317.53
Model 4	-65.67	147.34	177.66
Model 5	-62.45	146.91	188.60

Overall, Model 2 stands out with high Accuracy, high Sensitivity, high Specificity, high AUC, high Log-likelihood number, low AIC and low BIC. Although Model 3 was the best model to deal with **multicollinearity** issues, we still wanted a model with good prediction.

Model 2 not only greatly minimized the **multicollinearity** issue, but also gave the better prediction. We can verify the low level of multicollinearity in Model 2 by looking at variance inflation factors (using the 'car' library's vif() function. VIF measures how much the variance of each variable is inflated due to multicollinearity.

```
##      nox      tax    lstat     radN
## 2.009370 1.959690 1.262095 1.970707
```

Since the scores are all below 4, we can conclude multicollinearity has been handled appropriately.

More importantly, Likelihood, AIC and BIC numbers also indicate Model 2 is a better model to fit.

5 Predictions on Evaluation Data

```
##
##  0  1
## 16 24
```

After we applied our chosen model (Model 2) to crime-evaluation-data set, we predicted that for 16 observations `target` equals 0, and 24 observations, which `target` equals 1. Please check the file *result.csv* for detailed information.

6 Code Appendix

```
library(PerformanceAnalytics)
library(ggplot2)
library(gridExtra)
library(knitr)
library(caret)
library(lattice)
library(caret)
library(dplyr)
library(knitr)
library(pROC)
library(caTools)
library(car)
crime <- read.csv('crime-training-data.csv', na.strings=c("","NA"))
data <- crime

summary(crime)
# Histogram of one variable
zn_hist <- ggplot(crime, aes(zn)) + geom_histogram()
indus_hist <- ggplot(crime, aes(indus)) + geom_histogram()
nox_hist <- ggplot(crime, aes(nox)) + geom_histogram()
rm_hist <- ggplot(crime, aes(rm)) + geom_histogram()
age_hist <- ggplot(crime, aes(age)) + geom_histogram()
dis_hist <- ggplot(crime, aes(dis)) + geom_histogram()
rad_hist <- ggplot(crime, aes(rad)) + geom_histogram()
tax_hist <- ggplot(crime, aes(tax)) + geom_histogram()
ptratio_hist <- ggplot(crime, aes(ptratio)) + geom_histogram()
black_hist <- ggplot(crime, aes(black)) + geom_histogram()
lstat_hist <- ggplot(crime, aes(lstat)) + geom_histogram()
medv_hist <- ggplot(crime, aes(medv)) + geom_histogram()

# Boxplot: one variable ~ target
zn_bp <- ggplot(crime, aes(factor(target), zn)) + geom_boxplot()
indus_bp <- ggplot(crime, aes(factor(target), indus)) + geom_boxplot()
nox_bp <- ggplot(crime, aes(factor(target), nox)) + geom_boxplot()
rm_bp<- ggplot(crime, aes(factor(target), rm)) + geom_boxplot()
age_bp <- ggplot(crime, aes(factor(target), age)) + geom_boxplot()
dis_bp <- ggplot(crime, aes(factor(target), dis)) + geom_boxplot()
rad_bp <- ggplot(crime, aes(factor(target), rad)) + geom_boxplot()
tax_bp <- ggplot(crime, aes(factor(target), tax)) + geom_boxplot()
ptratio_bp <- ggplot(crime, aes(factor(target), ptratio)) + geom_boxplot()
black_bp <- ggplot(crime, aes(factor(target), black)) + geom_boxplot()
lstat_bp <- ggplot(crime, aes(factor(target), lstat)) + geom_boxplot()
medv_bp <- ggplot(crime, aes(factor(target), medv)) + geom_boxplot()

# Histogram and boxplot showing together
grid.arrange(zn_hist,zn_bp,indus_hist,indus_bp,nox_hist,nox_bp,ncol=2,nrow=3)
grid.arrange(rm_hist,rm_bp,age_hist,age_bp,dis_hist,dis_bp,ncol=2,nrow=3)
grid.arrange(rad_hist,rad_bp,tax_hist,tax_bp,ptratio_hist,ptratio_bp,ncol=2,nrow=3)
grid.arrange(black_hist,black_bp,lstat_hist,lstat_bp,medv_hist,medv_bp,ncol=2,nrow=3)

table(crime$target)
```

```
chart.Correlation(crime)

ggplot(crime, aes(x=zn)) + geom_density(aes(colour=factor(target))) + xlim(0,100) +
  geom_vline(xintercept = 3)
crime$znN <- ifelse(crime$zn > 3, 1, 0)
crime$znN <- as.factor(crime$znN)
t <- as.data.frame(table(znN=crime$znN, Target=crime$target))
kable(t, align='c')

attach(crime)
p0 <- ggplot(crime, aes(factor(target), indus)) + geom_boxplot()
crime <- crime[-which(target==0 & indus > 20),]
p1 <- ggplot(crime, aes(factor(target), indus)) + geom_boxplot()
grid.arrange(p0, p1,ncol=2,nrow=1)
detach(crime)

attach(crime)
p0 <- ggplot(crime, aes(factor(target), dis)) + geom_boxplot()
crime <- crime[-which(target==0 & dis > 11),]
crime <- crime[-which(target==1 & dis > 7.5),]
p1 <- ggplot(crime, aes(factor(target), dis)) + geom_boxplot()
grid.arrange(p0, p1, ncol=2,nrow=1)
detach(crime)

ggplot(crime, aes(x=rad)) + geom_density(aes(colour=factor(target))) + xlim(0,100) + geom_vline(xinterc
crime$radN <- ifelse(crime$rad > 7, 1, 0)
crime$radN <- as.factor(crime$radN)
t <- as.data.frame(table(radN=crime$radN, Target=crime$target))
kable(t)

#Since the variables were on a variety of scales, we also tried models that normalized the data before
#$\frac{(x_{i} - x_{min})}{(x_{max} - x_{min}) \times 100}$

normalize <- function(x)
{
 return((x - min(x)) / (max(x) - min(x)) * 100)
}

target <- data$target
#data_n <- as.data.frame(lapply(data[1:13], normalize))
#normalized_data <- cbind(data_n, target)

data_n <- data %>%
  select(-target) %>%
  scale(center = TRUE, scale = TRUE)

normalized_data <- data_n %>%
  cbind(target) %>%
  as.data.frame()

set.seed(45)
inTrain <- createDataPartition(y=crime$target, p=0.7,list=FALSE)
training <- crime[inTrain,]
```

```r
testing <- crime[-inTrain,]

inTrain2 <- createDataPartition(y=normalized_data$target, p=0.7,list=FALSE)
training_norm <- normalized_data[inTrain2,]
testing_norm <- normalized_data[-inTrain2,]

m11 <- glm(target ~ . -znN-radN, data=training, family = binomial(link='probit'))
#summary(m11)
m12 <- update(m11, .~. - zn-chas-rm-dis-black)
m1 <- m12
summary(m1)

## Model2 - Using the three new created variances

m21 <- glm(target ~ .-age-zn-rad, data=training,
               family = binomial(link='probit'))
#summary(m21)
m22 <- update(m21, .~. - indus-chas-rm-dis-black-znN)
#summary(m22)
m23 <- update(m22, .~. -medv)
#summary(m23)
m24 <- update(m23, .~. -ptratio)
m2 <- m24
summary(m2)

crime_pca <- crime[,1:14]
crime_pca <- select(crime_pca,-chas)
#names(crime_pca)
target <- crime_pca$target
A <- as.matrix(select(crime_pca,-target))
pca <- princomp(A,center=T,scale.=T)
plot(pca)
#summary(pca)
pca <- as.data.frame(pca$scores[,1:2])
crime_pca <- cbind(target=target,pca)

head(crime_pca)
set.seed(45)
inTrain_pca <- createDataPartition(y=crime_pca$target, p=0.7,list=FALSE)
training_pca <- crime_pca[inTrain_pca,]
testing_pca <- crime_pca[-inTrain_pca,]
m3 <- glm(target ~ ., data=training_pca)
summary(m3)

data3a <- training_norm[,-c(5,12)]

model3a <- glm(target ~ nox + age + rad + tax + ptratio + black + medv, data = training_norm, family = 
predict3a <- round(predict(model3a, type = 'response'),4)

result3a <- predict3a

for (i in 1:NROW(predict3a))
{
```

```
 if(predict3a[i] > .50)
  {
   result3a[i] <- 1
  } else
  {
   result3a[i]  <- 0
  }
}

final3a <- as.data.frame(cbind(predict3a, result3a))
t <- table(actual = data3a$target, predicted = final3a$result3a)
summary(model3a)

data3a <- training_norm[,-c(5,12)]

model3a <- glm(target ~ nox + age + rad + tax + ptratio + black + medv, data = training_norm, family = 
predict3a <- round(predict(model3a, type = 'response'),4)

result3a <- predict3a

for (i in 1:NROW(predict3a))
{
 if(predict3a[i] > .50)
  {
   result3a[i] <- 1
  } else
  {
   result3a[i]  <- 0
  }
}

final3a <- as.data.frame(cbind(predict3a, result3a))
t <- table(actual = data3a$target, predicted = final3a$result3a)
summary(model3a)

data3b <- training_norm[,-c(2,3,5)]

model3b <- glm(target ~ nox + rad + tax + ptratio + black + medv + age + dis + zn + lstat, data = data3
predict3b <- round(predict(model3b, type = 'response'),4)

result3b <- predict3b

for (i in 1:NROW(predict3b))
{
 if(predict3b[i] > .50)
  {
   result3b[i] <- 1
  } else
  {
   result3b[i]  <- 0
  }
}
```

```r
final3b <- as.data.frame(cbind(predict3b, result3b))

summary(model3b)

# Model1
predict_1 <- predict(m1, newdata=testing, type='response')
glm.pred1 = ifelse(predict_1 > 0.5, 1, 0)
cM1 <- confusionMatrix(glm.pred1, testing$target, positive = "1")

# Model2
predict_2 <- predict(m2, newdata=testing, type='response')
glm.pred2 = ifelse(predict_2 > 0.5, 1, 0)
cM2 <- confusionMatrix(glm.pred2, testing$target, positive = "1")

# Model3
predict_3 <- predict(m3,newdata=testing_pca,type='response')
glm.pred3 = ifelse(predict_3 > 0.5, 1, 0)
cM3 <- confusionMatrix(glm.pred3, testing_pca$target, positive = "1")

# Model4
predict_4 <- predict(model3a, newdata=testing_norm, type ='response')
glm.pred4 = ifelse(predict_4 > 0.5, 1, 0)
cM4 <- confusionMatrix(glm.pred4, testing_norm$target, positive = "1")

# Model5
predict_5 <- predict(model3b, newdata=testing_norm, type ='response')
glm.pred5 = ifelse(predict_4 > 0.5, 1, 0)
cM5 <- confusionMatrix(glm.pred5, testing_norm$target, positive = "1")

df1b <- as.data.frame(cM1$byClass)
df1a <- as.data.frame(cM1$overall)
colnames(df1a) <- 'Model1'
colnames(df1b) <- 'Model1'
df1 <- rbind(df1a, df1b)

df2b <- as.data.frame(cM2$byClass)
df2a <- as.data.frame(cM2$overall)
colnames(df2a) <- 'Model2'
colnames(df2b) <- 'Model2'
df2 <- rbind(df2a, df2b)

df3b <- as.data.frame(cM3$byClass)
df3a <- as.data.frame(cM3$overall)
colnames(df3a) <- 'Model3'
colnames(df3b) <- 'Model3'
df3 <- rbind(df3a, df3b)

df4b <- as.data.frame(cM4$byClass)
df4a <- as.data.frame(cM4$overall)
colnames(df4a) <- 'Model4'
colnames(df4b) <- 'Model4'
df4 <- rbind(df4a, df4b)
```

```
df5b <- as.data.frame(cM5$byClass)
df5a <- as.data.frame(cM5$overall)
colnames(df5a) <- 'Model5'
colnames(df5b) <- 'Model5'
df5 <- rbind(df5a, df5b)

df <- cbind(df1,df2,df3,df4,df5)
kable(df,caption='Confusion Matrix')

rc1 <- roc(factor(target) ~ predict_1, data=testing)
rc2 <- roc(factor(target) ~ predict_2, data=testing)
rc3 <- roc(factor(target) ~ predict_3, data=testing_pca)
rc4 <- roc(factor(target) ~ predict_4, data=testing_norm)
rc5 <- roc(factor(target) ~ predict_5, data=testing_norm)

plot(rc1,main='Model 1 - ROC Curve')
plot(rc2,main='Model 2 - ROC Curve')
plot(rc3,main='Model 3 - ROC Curve')
plot(rc4,main='Model 4 - ROC Curve')
plot(rc5,main='Model 5 - ROC Curve')

model <- c('Model 1', 'Model 2', 'Model 3','Model 4','Model 5')
area <- c(auc(rc1),auc(rc2),auc(rc3),auc(rc4),auc(rc5))
df <- data.frame(Model=model,AUC=area)
kable(df,caption='Area under the curve')

LL.1 <- logLik(m1)
LL.2 <- logLik(m2)
LL.3 <- logLik(m3)
LL.4 <- logLik(model3a)
LL.5 <- logLik(model3b)
LL <- rbind(LL.1, LL.2, LL.3, LL.4, LL.5) %>% round(2)

#Akaike Information Criterion
AIC.1 <- AIC(m1)
AIC.2 <- AIC(m2)
AIC.3 <- AIC(m3)
AIC.4 <- AIC(model3a)
AIC.5 <- AIC(model3b)
AIC <- rbind(AIC.1, AIC.2, AIC.3, AIC.4, AIC.5) %>% round(2)

#Coefficient of Determination
# http://stats.stackexchange.com/questions/577/is-there-any-reason-to-prefer-the-aic-or-bic-over-the-ot
BIC.1 <- BIC(m1)
BIC.2 <- BIC(m2)
BIC.3 <- BIC(m3)
BIC.4 <- BIC(model3a)
BIC.5 <- BIC(model3b)
BIC <- rbind(BIC.1, BIC.2, BIC.3, BIC.4, BIC.5) %>% round(2)

eval.table <- cbind(LL, AIC, BIC)

rownames(eval.table) <- c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5")
```

```r
colnames(eval.table) <- c("Log Likelihood", "AIC", "BIC")

kable(eval.table,caption = 'Log-likelihood/AIC/BIC')

vif(m2)

eval <- read.csv('crime-evaluation-data.csv')
eval$radN <- ifelse(eval$rad > 7, 1, 0)
eval$radN <- as.factor(eval$radN)
predict_eval <- predict(m2, newdata=eval, type='response')
glm.pred_eval = ifelse(predict_eval > 0.5, 1, 0)
eval$target <- glm.pred_eval
table(eval$target)
write.csv(eval, 'result.csv')
```