DATA 621—Assignment no. 3

Critical Thinking Group 2

October 30, 2019

Contents

Executive Overview

In this homework assignment, you will explore, analyze and model a data set containing information on crime for various neighborhoods of a major city. Each record has a response variable indicating whether or not the crime rate is above the median crime rate (1) or not (0).

Your objective is to build a binary logistic regression model on the training data set to predict whether the neighborhood will be at risk for high crime levels.

Below is a short description of the variables of interest in the data set:

Variable	Description
zn	proportion of residential land zoned for large lots (over 25000 square feet) (predictor variable)
indus	proportion of non-retail business acres per suburb (predictor variable)
chas	a dummy var. for whether the suburb borders the Charles River (1) or not (0) (predictor variable)
nox	nitrogen oxides concentration (parts per 10 million) (predictor variable)
rm	average number of rooms per dwelling (predictor variable)
age	proportion of owner-occupied units built prior to 1940 (predictor variable)
dis	weighted mean of distances to five Boston employment centers (predictor variable)
rad	index of accessibility to radial highways (predictor variable)
tax	full-value property-tax rate per $10,000 (predictor variable)
ptratio	pupil-teacher ratio by town (predictor variable)
lstat	lower status of the population (percent) (predictor variable)

Variable	Description
medv	median value of owner-occupied homes in $1000s (predictor variable)
target	whether the crime rate is above the median crime rate (1) or not (0) (response variable)

Create train and test sets using the `caret` machine learning package:

Only use the train data frame until the very end of the process, when we use test to evaluate how effective the model is!

```
df <- read.csv('crime-training-data_modified.csv', stringsAsFactors=FALSE)
set.seed(1804)
#80% train, 20% test split
train_ix <- createDataPartition(df$target, p=0.8, list=FALSE)
train <- df[train_ix, ]
test <- df[-train_ix, ]
rm(df)
```

Data Exploration

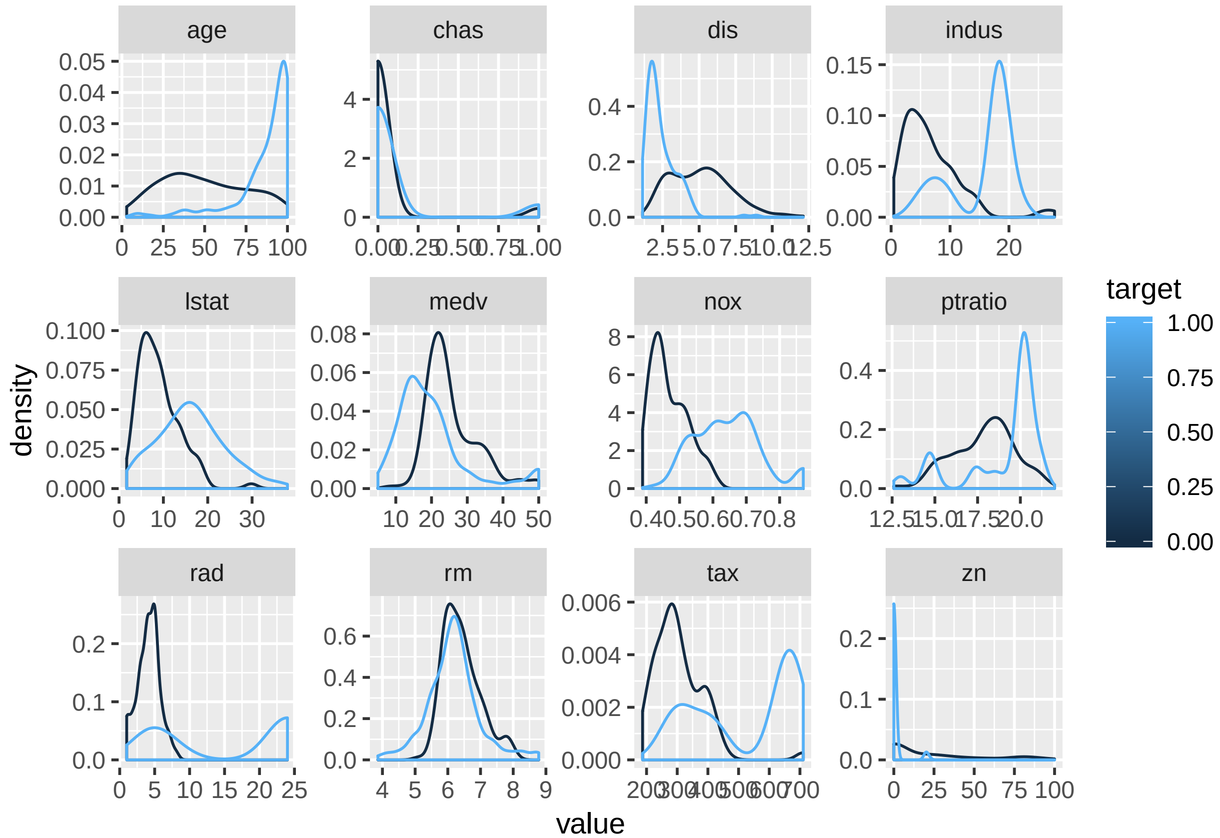
Below is descriptive statistic of the variables. There are no NA values.

```
(train.summary <- data.frame(unclass(summary(train)), row.names = NULL))
```

```
##          X......zn      X....indus        X.....chas        X.....nox
## 1 Min.   :  0.00   Min.   : 0.46   Min.   :0.00000   Min.   :0.3890
## 2 1st Qu.:  0.00   1st Qu.: 5.13   1st Qu.:0.00000   1st Qu.:0.4480
## 3 Median :  0.00   Median : 9.90   Median :0.00000   Median :0.5380
## 4 Mean   : 11.66   Mean   :11.14   Mean   :0.07775   Mean   :0.5571
## 5 3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000   3rd Qu.:0.6470
## 6 Max.   :100.00   Max.   :27.74   Max.   :1.00000   Max.   :0.8710
##         X......rm         X.....age        X.....dis         X.....rad
## 1 Min.   :3.863   Min.   :  2.90   Min.   : 1.130   Min.   : 1.000
## 2 1st Qu.:5.913   1st Qu.: 43.70   1st Qu.: 2.022   1st Qu.: 4.000
## 3 Median :6.226   Median : 78.70   Median : 3.092   Median : 5.000
## 4 Mean   :6.298   Mean   : 68.47   Mean   : 3.748   Mean   : 9.729
## 5 3rd Qu.:6.635   3rd Qu.: 94.60   3rd Qu.: 5.212   3rd Qu.:24.000
## 6 Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000
##         X.....tax      X...ptratio       X....lstat      X.....medv
## 1 Min.   :187.0   Min.   :12.60   Min.   : 1.73   Min.   : 5.00
## 2 1st Qu.:281.0   1st Qu.:17.00   1st Qu.: 6.75   1st Qu.:17.00
## 3 Median :345.0   Median :18.90   Median :11.32   Median :21.50
## 4 Mean   :412.6   Mean   :18.41   Mean   :12.63   Mean   :22.81
## 5 3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:17.09   3rd Qu.:26.20
## 6 Max.   :711.0   Max.   :22.00   Max.   :37.97   Max.   :50.00
##        X....target
## 1 Min.   :0.0000
## 2 1st Qu.:0.0000
## 3 Median :0.0000
## 4 Mean   :0.4987
## 5 3rd Qu.:1.0000
## 6 Max.   :1.0000
```

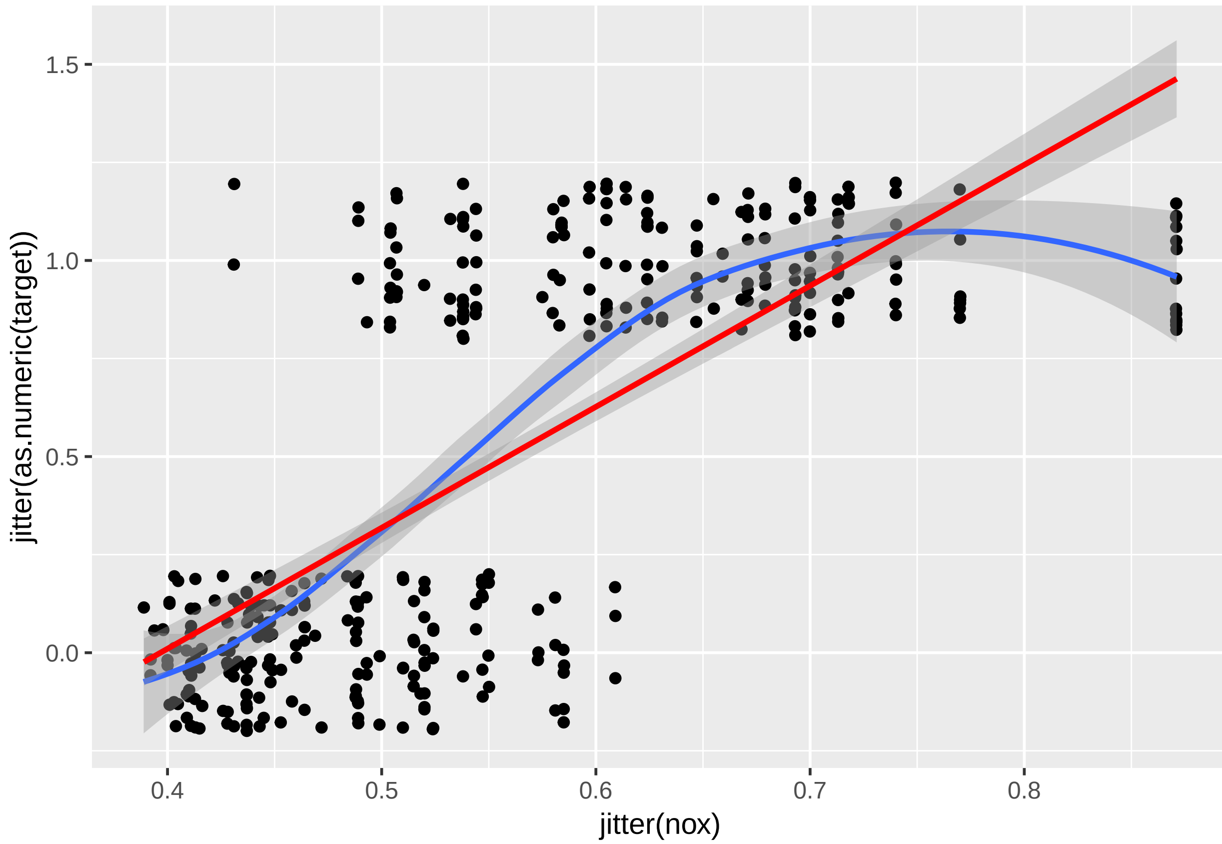
There don't seem to be any outliers or missing data, so we will proceed directly to examining the variables. First, histograms of each variable for each `target` class:

```r
train %>%
  gather(-target, key='variable', value='value') %>%
  ggplot(aes(x=value, group=target, color=target)) +
    facet_wrap(~ variable, scales='free') +
    geom_density()
```



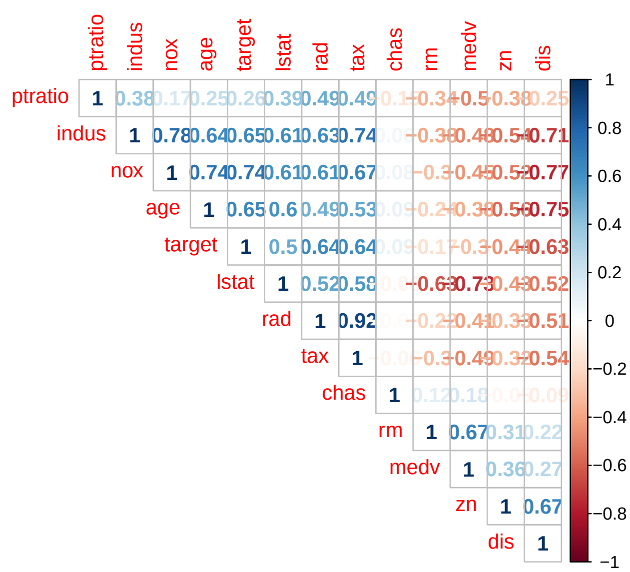
Most variables have distinct shapes for each `target` class. `chas` and `zn` are quite skewed, and do not appear terribly informative. `indus` and `tax` have two peaks for `target = 1`, indicating there are two separate processes at work there.

```r
ggplot(train, aes(x=jitter(nox), y=jitter(as.numeric(target)))) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method='lm', color='red')
```



It is to be expected that many of these variables will be correlated with each other:

```
corrplot(cor(train), type='upper', method='number', order='hclust')
```



Obviously, the concentration of industry is strongly and positively correlated with nitrogen oxide concentration $\rho = 0.78$). Parent-teacher ratio is negatively correlated with median property values ($\rho = -0.5$), and positively correlated with property taxes ($\rho = 0.49$). What these and other variables are really getting at is *economic class*. Each measures a different phenomenon, but can be conceived of as operationalizing one thing. This suggests PCA may be useful on this dataset.

Checking for interactions

Given the high correlation between the variables, it may be the case that there are numerous interactions that can improve our modeling. In this section, we attempt to determine if this is the case. We will group numeric variables by membership in quartile, and examine line plots.

```
calc_percentile <- function(x){
  trunc(rank(x)) / length(x)
}
```

Data Preparation

There doesn't seem to be any missing data or any obvious outliers.

Modeling

Function to calculate McFadden's pseudo-R^2 for logistic models:

```
calc_r2 <- function(model) {
  1 - model$deviance / model$null.deviance
}
```

M_0: Dummy model

Baseline model, which just predicts the class proportion, which is nearly balanced between the two classes. If we are having trouble improving on this model, we know we are doing something wrong.

This dummy model has an accuracy of about 0.50, sensitivity of 1, and specificity of 0. Since it has zero predictive power, we know that it has a pseudo-R^2 of 0.

```
m_0 <- glm(target ~ 1, train, family=binomial())
pred_0 <- factor(round(predict(m_0, train, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_0, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 187 186
##          1   0   0
##
##                Accuracy : 0.5013
##                  95% CI : (0.4494, 0.5532)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : 0.5207
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.5013
##          Neg Pred Value :    NaN
##              Prevalence : 0.5013
```

```
##          Detection Rate : 0.5013
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```r
m_0 #null deviance and residual deviance are the same --> R2 is zero
```

```
##
## Call:  glm(formula = target ~ 1, family = binomial(), data = train)
##
## Coefficients:
## (Intercept)
##   -0.005362
##
## Degrees of Freedom: 372 Total (i.e. Null);  372 Residual
## Null Deviance:       517.1
## Residual Deviance: 517.1     AIC: 519.1
```

M_1: Full model

The next simplest model uses all available data, without transformations or interactions or polynomials:

```r
m_1 <- glm(target ~ zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + lstat + medv, trai
pred_1 <- factor(round(predict(m_1, train, type='response')), levels=c('0', '1'))
calc_r2(m_1)
```

```
## [1] 0.7216759
```

```r
confusionMatrix(data=pred_1, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 173  19
##          1  14 167
##
##                Accuracy : 0.9115
##                  95% CI : (0.878, 0.9383)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.823
##
##  Mcnemar's Test P-Value : 0.4862
##
##             Sensitivity : 0.9251
##             Specificity : 0.8978
##          Pos Pred Value : 0.9010
##          Neg Pred Value : 0.9227
##              Prevalence : 0.5013
##          Detection Rate : 0.4638
##    Detection Prevalence : 0.5147
##       Balanced Accuracy : 0.9115
```

```
##
##        'Positive' Class : 0
##
```

M_2: Stepwise variable selection with interactions

https://www.theanalysisfactor.com/interpreting-interactions-in-regression/

We know that variable interaction is probably likely. We can automatically test all interactions using stepwise selection:

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred Explanation found: https://stats.stackexchange.com/questions/11109/how-to-deal-with-perfect-separation-in-logistic-regression "If you have a variable which perfectly separates zeroes and ones in target variable, R will yield the following"perfect or quasi perfect separation" "We still get the model but the coefficient estimates are inflated." We are suppressing the warning.

```r
#https://stat.ethz.ch/R-manual/R-patched/library/MASS/html/stepAIC.html
m_2 <- stepAIC(m_1, trace=0, scope=list(upper = ~ zn * indus * chas * nox * rm *
                                        age * dis * rad * tax * ptratio *
                                        lstat*medv, lower= ~1))
pred_2 <- factor(round(predict(m_2, train, type='response')), levels=c('0', '1'))
calc_r2(m_2)
```

```
## [1] 0.9022719
```

```r
confusionMatrix(data=pred_2, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 183   5
##          1   4 181
##
##                Accuracy : 0.9759
##                  95% CI : (0.9547, 0.9889)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9517
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9786
##             Specificity : 0.9731
##          Pos Pred Value : 0.9734
##          Neg Pred Value : 0.9784
##              Prevalence : 0.5013
##          Detection Rate : 0.4906
##    Detection Prevalence : 0.5040
##       Balanced Accuracy : 0.9759
##
##        'Positive' Class : 0
##
```

```r
summary(m_2)
```

```
##
## Call:
## glm(formula = target ~ zn + indus + chas + nox + rm + age + dis +
##     rad + tax + ptratio + lstat + medv + ptratio:lstat + chas:tax +
##     nox:age + rm:lstat + rm:age + age:medv + nox:ptratio + dis:tax +
##     indus:tax + tax:medv + indus:dis + age:lstat, family = binomial(),
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.70636  -0.00332   0.00000   0.00000   2.57482
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    2.503e+00  8.456e+01   0.030 0.976389
## zn            -4.539e-01  1.865e-01  -2.433 0.014958 *
## indus         -2.261e+00  8.363e-01  -2.703 0.006862 **
## chas          -6.926e+03  4.370e+05  -0.016 0.987355
## nox            3.607e+02  1.952e+02   1.848 0.064661 .
## rm            -2.264e+01  7.012e+00  -3.228 0.001247 **
## age           -2.219e+00  5.930e-01  -3.742 0.000182 ***
## dis           -1.474e+01  5.996e+00  -2.459 0.013933 *
## rad            2.495e+00  6.857e-01   3.639 0.000274 ***
## tax           -3.465e-01  1.186e-01  -2.922 0.003473 **
## ptratio        8.824e+00  4.858e+00   1.817 0.069279 .
## lstat         -1.399e+00  2.456e+00  -0.570 0.568786
## medv           9.272e-01  7.603e-01   1.220 0.222642
## ptratio:lstat  2.242e-01  1.271e-01   1.764 0.077731 .
## chas:tax       2.502e+01  1.578e+03   0.016 0.987343
## nox:age        1.362e+00  5.341e-01   2.549 0.010789 *
## rm:lstat      -5.908e-01  2.796e-01  -2.113 0.034585 *
## rm:age         3.657e-01  9.577e-02   3.819 0.000134 ***
## age:medv      -3.075e-02  9.117e-03  -3.372 0.000745 ***
## nox:ptratio   -1.843e+01  9.957e+00  -1.851 0.064239 .
## dis:tax        5.026e-02  1.892e-02   2.656 0.007913 **
## indus:tax      5.110e-03  1.884e-03   2.713 0.006672 **
## tax:medv       4.637e-03  1.871e-03   2.477 0.013231 *
## indus:dis      1.913e-01  1.284e-01   1.490 0.136209
## age:lstat      7.722e-03  3.927e-03   1.966 0.049257 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.085  on 372  degrees of freedom
## Residual deviance:  50.534  on 348  degrees of freedom
## AIC: 100.53
##
## Number of Fisher Scoring iterations: 25
```

However, this model is probably overfit. By common heuristic, we have enough data for:

```r
min(table(train$target)) / 15
```

```
## [1] 12.4
```

i.e., 12 variables.

M_3: Adjusting for multiple significance tests

To correct for this overfitting, we will use the `p.adjust` function to revise our p-values, and then use those that remain significant at $p = 0.05$ for the next model:

```r
m_2_p <- summary(m_2)$coefficients[,4]
sort(p.adjust(m_2_p))
```

```
##       rm:age          age          rad     age:medv           rm
##  0.003352611  0.004375544  0.006294925  0.016389911  0.026180977
##          tax        indus    indus:tax      dis:tax      nox:age
##  0.069463666  0.126774120  0.126774120  0.134529104  0.172628530
##           zn          dis     tax:medv     rm:lstat    age:lstat
##  0.198464177  0.198464177  0.198464177  0.415017379  0.541826029
##          nox      ptratio ptratio:lstat  nox:ptratio    indus:dis
##  0.642394932  0.642394932  0.642394932  0.642394932  0.817253780
##  (Intercept)         chas        lstat         medv     chas:tax
##  1.000000000  1.000000000  1.000000000  1.000000000  1.000000000
```

Using the top values (including any variable as well as interaction effect:

```r
m_3 <- glm(target ~ age*rm + rad + age*medv, train, family=binomial())
pred_3 <- factor(round(predict(m_3, train, type='response')), levels=c('0', '1'))
calc_r2(m_3)
```

```
## [1] 0.609916
```

```r
confusionMatrix(data=pred_3, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 160  22
##          1  27 164
##
##                Accuracy : 0.8686
##                  95% CI : (0.8301, 0.9012)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7373
##
##  Mcnemar's Test P-Value : 0.5677
##
##             Sensitivity : 0.8556
##             Specificity : 0.8817
##          Pos Pred Value : 0.8791
##          Neg Pred Value : 0.8586
##              Prevalence : 0.5013
##          Detection Rate : 0.4290
```

```
##    Detection Prevalence : 0.4879
##       Balanced Accuracy : 0.8687
##
##        'Positive' Class : 0
##
```

The pseudo-R^2 is naturally much less than the overfit M_2. Presumably, it will be better fit to the hold-out sample, however. We do see that sensitivity, specificity, and pos/neg predictive value are actually still pretty strong. As expected and required, all variables are extremely significant.

```r
summary(m_3)
```

```
##
## Call:
## glm(formula = target ~ age * rm + rad + age * medv, family = binomial(),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9933  -0.3056  -0.0112   0.0131   3.9635
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) 29.933949   9.484583   3.156 0.001599 **
## age         -0.391606   0.116138  -3.372 0.000747 ***
## rm          -9.166313   2.215904  -4.137 3.52e-05 ***
## rad          0.572715   0.131517   4.355 1.33e-05 ***
## medv         0.767617   0.175859   4.365 1.27e-05 ***
## age:rm       0.108631   0.026477   4.103 4.08e-05 ***
## age:medv    -0.008883   0.002072  -4.288 1.80e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.09  on 372  degrees of freedom
## Residual deviance: 201.71  on 366  degrees of freedom
## AIC: 215.71
##
## Number of Fisher Scoring iterations: 8
```

M_4: Previous model + a few more predictors

We noted above that we have data for up to 12 variables in this model, so I will include the first 12 significant variables of the p-value adjustment:

```r
m_4 <- glm(target ~ age*rm + rad + age*medv +
             indus*tax + dis*tax + nox*age + zn, train, family=binomial())
pred_4 <- factor(round(predict(m_4, train, type='response')), levels=c('0', '1'))
calc_r2(m_4)
```

```
## [1] 0.7624825
```

```r
confusionMatrix(data=pred_4, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   0   1
##          0 174  14
##          1  13 172
##
##                Accuracy : 0.9276
##                  95% CI : (0.8964, 0.9518)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8552
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9305
##             Specificity : 0.9247
##          Pos Pred Value : 0.9255
##          Neg Pred Value : 0.9297
##              Prevalence : 0.5013
##          Detection Rate : 0.4665
##    Detection Prevalence : 0.5040
##       Balanced Accuracy : 0.9276
##
##        'Positive' Class : 0
##
```

Despite adding all these variables, we see that the confusion matrix evaluations are not that much higher. Pseudo-R^2 did take a nice bump, though. Nonetheless, it is possible that this model does not fit the hold out sample as well as M_3.

```r
summary(m_4)
```

```
##
## Call:
## glm(formula = target ~ age * rm + rad + age * medv + indus *
##     tax + dis * tax + nox * age + zn, family = binomial(), data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.99998  -0.17796   0.00000   0.00012   2.94942
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) 36.3490667 16.0747814   2.261 0.023744 *
## age         -0.7613783  0.2221787  -3.427 0.000611 ***
## rm          -8.3921084  2.5514316  -3.289 0.001005 **
## rad          1.0682193  0.2427743   4.400 1.08e-05 ***
## medv         0.8831142  0.2527643   3.494 0.000476 ***
## indus       -0.3370128  0.1593284  -2.115 0.034412 *
## tax         -0.0572609  0.0206903  -2.768 0.005648 **
## dis         -2.7775922  1.4321662  -1.939 0.052448 .
## nox          0.9108840 17.1773095   0.053 0.957709
## zn          -0.2092179  0.0602181  -3.474 0.000512 ***
## age:rm       0.1058911  0.0318826   3.321 0.000896 ***
## age:medv    -0.0105802  0.0029532  -3.583 0.000340 ***
```

```
## indus:tax    0.0008713  0.0004702   1.853 0.063907 .
## tax:dis      0.0122552  0.0045798   2.676 0.007452 **
## age:nox      0.7561247  0.2713618   2.786 0.005330 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.09  on 372  degrees of freedom
## Residual deviance: 122.82  on 358  degrees of freedom
## AIC: 152.82
##
## Number of Fisher Scoring iterations: 10
```

M_5: PCA

```
pca <- prcomp(train[,1:12], retx=TRUE, center=TRUE, scale=TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3    PC4     PC5     PC6
## Standard deviation     2.4660 1.2794 1.04731 0.9172 0.88763 0.63278
## Proportion of Variance 0.5067 0.1364 0.09141 0.0701 0.06566 0.03337
## Cumulative Proportion  0.5067 0.6431 0.73455 0.8047 0.87031 0.90368
##                            PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     0.53845 0.52925 0.45673 0.42813 0.36816 0.24159
## Proportion of Variance 0.02416 0.02334 0.01738 0.01527 0.01129 0.00486
## Cumulative Proportion  0.92784 0.95118 0.96857 0.98384 0.99514 1.00000
```

The first five account for 87 percent of variation, so we will use those for modeling:

```
pca_df <- as.data.frame(cbind(train$target, pca$x[,1:5]))
colnames(pca_df) <- c('target', 'PC1', 'PC2' ,'PC3', 'PC4', 'PC5')
m_5 <- glm(target ~ ., pca_df, family=binomial())
pred_5 <- factor(round(predict(m_5, pca_df, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_5, reference=factor(train$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 162  36
##          1  25 150
##
##                Accuracy : 0.8365
##                  95% CI : (0.7949, 0.8725)
##     No Information Rate : 0.5013
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6729
##
##  Mcnemar's Test P-Value : 0.2004
##
##             Sensitivity : 0.8663
##             Specificity : 0.8065
```

```
##          Pos Pred Value : 0.8182
##          Neg Pred Value : 0.8571
##              Prevalence : 0.5013
##          Detection Rate : 0.4343
##    Detection Prevalence : 0.5308
##       Balanced Accuracy : 0.8364
##
##        'Positive' Class : 0
##
```

```r
calc_r2(m_5)
```

```
## [1] 0.5806149
```

This model has similar confusion matrix evaluation values as some models above, though it's pseudo-R^2 value is a bit low.

The results of this exercise with PCA seem to suggests there are three separate 'clusters' of phenomenon that affect crime level, at least at a statistically significant level. All three are negative related.

```r
summary(m_5)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(), data = pca_df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.59273  -0.43190  -0.07356   0.21743   2.63898
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.33271    0.26973   1.233   0.2174
## PC1         -1.18169    0.13143  -8.991  < 2e-16 ***
## PC2         -0.90733    0.15730  -5.768 8.01e-09 ***
## PC3         -0.62027    0.24223  -2.561   0.0104 *
## PC4         -0.02799    0.18400  -0.152   0.8791
## PC5         -0.15736    0.20946  -0.751   0.4525
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 517.09  on 372  degrees of freedom
## Residual deviance: 216.86  on 367  degrees of freedom
## AIC: 228.86
##
## Number of Fisher Scoring iterations: 6
```

Summary of Psuedo-R^2 and Accuracy of Different Models on Training Data

Model	pseudo-R^2	Accuracy	Description
m_0	0	0.5013	Dummy model

Model	pseudo-R^2	Accuracy	Description
m_1	0.7216759	0.9115	Full Model
m_2	0.9022719	0.9759	Stepwise variable selection with interactions
m_3	0.609916	0.8686	Adjusting m_2 for multiple significance tests
m_4	0.7624825	0.9276	m_3 plus few more predictors
m_5	0.5806149	0.8365	PCA

Evaluating the Models on the Test Set

```r
# Don't run until the very end
# confusionMatrix(data=predict(model, test), reference=test$target)
# Evaluate on F1 score
# For PCA prediction:
# pred_xx <- factor(round(predict(m_5, as.data.frame(predict(pca, newdata=test)), type='response')), le
```

Model `m_1` on the test set:

```r
pred_test_1 <- factor(round(predict(m_1, test, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_test_1, reference=factor(test$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 47  6
##          1  3 37
##
##                Accuracy : 0.9032
##                  95% CI : (0.8242, 0.9548)
##     No Information Rate : 0.5376
##     P-Value [Acc > NIR] : 2.426e-14
##
##                   Kappa : 0.8044
##
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity : 0.9400
##             Specificity : 0.8605
##          Pos Pred Value : 0.8868
##          Neg Pred Value : 0.9250
##              Prevalence : 0.5376
##          Detection Rate : 0.5054
##    Detection Prevalence : 0.5699
##       Balanced Accuracy : 0.9002
##
##        'Positive' Class : 0
##
```

```r
F1_Score(y_pred = pred_test_1, y_true = test$target, positive = "0")
```

```
## [1] 0.9126214
```

Model `m_2` on the test set:

```r
pred_test_2 <- factor(round(predict(m_2, test, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_test_2, reference=factor(test$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48  3
##          1  2 40
##
##                Accuracy : 0.9462
##                  95% CI : (0.879, 0.9823)
##     No Information Rate : 0.5376
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8917
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9600
##             Specificity : 0.9302
##          Pos Pred Value : 0.9412
##          Neg Pred Value : 0.9524
##              Prevalence : 0.5376
##          Detection Rate : 0.5161
##    Detection Prevalence : 0.5484
##       Balanced Accuracy : 0.9451
##
##        'Positive' Class : 0
##
```

```r
F1_Score(y_pred = pred_test_2, y_true = test$target, positive = "0")
```

```
## [1] 0.950495
```

Model `m_3` on the test set:

```r
pred_test_3 <- factor(round(predict(m_3, test, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_test_3, reference=factor(test$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 42  8
##          1  8 35
##
##                Accuracy : 0.828
##                  95% CI : (0.7357, 0.8983)
##     No Information Rate : 0.5376
##     P-Value [Acc > NIR] : 3.831e-09
##
##                   Kappa : 0.654
##
##  Mcnemar's Test P-Value : 1
##
```

```
##             Sensitivity : 0.8400
##             Specificity : 0.8140
##          Pos Pred Value : 0.8400
##          Neg Pred Value : 0.8140
##              Prevalence : 0.5376
##          Detection Rate : 0.4516
##    Detection Prevalence : 0.5376
##       Balanced Accuracy : 0.8270
##
##        'Positive' Class : 0
##
```

```
F1_Score(y_pred = pred_test_3, y_true = test$target, positive = "0")
```

```
## [1] 0.84
```

Model `m_4` on the test set:

```
pred_test_4 <- factor(round(predict(m_4, test, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_test_4, reference=factor(test$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48  8
##          1  2 35
##
##                Accuracy : 0.8925
##                  95% CI : (0.8111, 0.9472)
##     No Information Rate : 0.5376
##     P-Value [Acc > NIR] : 1.783e-13
##
##                   Kappa : 0.7816
##
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.9600
##             Specificity : 0.8140
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.9459
##              Prevalence : 0.5376
##          Detection Rate : 0.5161
##    Detection Prevalence : 0.6022
##       Balanced Accuracy : 0.8870
##
##        'Positive' Class : 0
##
```

```
F1_Score(y_pred = pred_test_4, y_true = test$target, positive = "0")
```

```
## [1] 0.9056604
```

Model `m_5` on the test set:

```
pca_test <- prcomp(test[,1:12], retx=TRUE, center=TRUE, scale=TRUE)
pca_test_df <- as.data.frame(cbind(test$target, pca_test$x[,1:5]))
```

```r
colnames(pca_test_df) <- c('target', 'PC1', 'PC2' ,'PC3', 'PC4', 'PC5')
pred_test_5 <- factor(round(predict(m_5, pca_test_df, type='response')), levels=c('0', '1'))
confusionMatrix(data=pred_test_5, reference=factor(test$target, levels=c('0', '1')))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  8 30
##          1 42 13
##
##                Accuracy : 0.2258
##                  95% CI : (0.1455, 0.3242)
##     No Information Rate : 0.5376
##     P-Value [Acc > NIR] : 1.0000
##
##                   Kappa : -0.5274
##
##  Mcnemar's Test P-Value : 0.1949
##
##             Sensitivity : 0.16000
##             Specificity : 0.30233
##          Pos Pred Value : 0.21053
##          Neg Pred Value : 0.23636
##              Prevalence : 0.53763
##          Detection Rate : 0.08602
##    Detection Prevalence : 0.40860
##       Balanced Accuracy : 0.23116
##
##        'Positive' Class : 0
##
```

```r
F1_Score(y_pred = pred_test_5, y_true = test$target, positive = "0")
```

```
## [1] 0.1818182
```

Summary of Accuracy and F1 Score on Test Set

Model	Accuracy	F1 Score	Description
m_1	0.9032	0.9126214	Full Model
m_2	0.9462	0.950495	Stepwise variable selection with interactions
m_3	0.828	0.84	Adjusting m_2 for multiple significance tests
m_4	0.8925	0.9056604	m_3 plus few more predictors
m_5	0.2258	0.1818182	PCA

Analysis of Final Model