DATA 621—Assignment no. 3

Critical Thinking Group 2

October 30, 2019

Contents

Executive Overview **1**

Data Exploration **2**

Data Preparation **6**

Modeling **6**
M_0: Dummy model 6
M_1: Full model 6
M_2: Stepwise variable selection with interactions 8
M_3: Interactions, part II 8
M_4: PCA 9

Evaluating the Models on the Test Set **11**
Inference on the Final Model 12

Executive Overview

This paper describes our attempt to model whether a geographical zone in Boston is at risk for high crime levels. Our data set contains 466 such zones, and each zone is quantified by 12 numeric variables. The `target` variable is marked `1` for high crime areas, and `0` otherwise.

We find that our variables are in four clusters: Proximity to the Charles River, the economic class of a zone's citizens, how business-industrial v. residential it is, and its status as a suburb.

Four substantial models are fit. An overfit model with 28 variables performs the best on a holdout sample, although a smaller one with only 6 variables performs nearly as well. Due to correlations between most of the variables, we also used PCA transformation in one model, which performed only adequately.

For the purposes of statistical inference, the conclusion of the modeling is that the proportion of a geographic zone's 'lower-status' population and concentration of nitrogen oxides are the most powerful variables to explain whether a zone has a high crime level. Both quantities are significantly and strongly related to crime levels.

First, we create `train` and `test` dataframes (75 and 25 percent of the data set, respectively) using the `caret` machine learning package. The `test` data set provides an unbiased estimate of how well our models perform, and was not be used for any purpose until the very end of the modeling process.

```
## [1] "Rows in training data set: 350"
```

```
## [1] "Rows in test data set: 116"
```

Data Exploration

The variables at hand are:

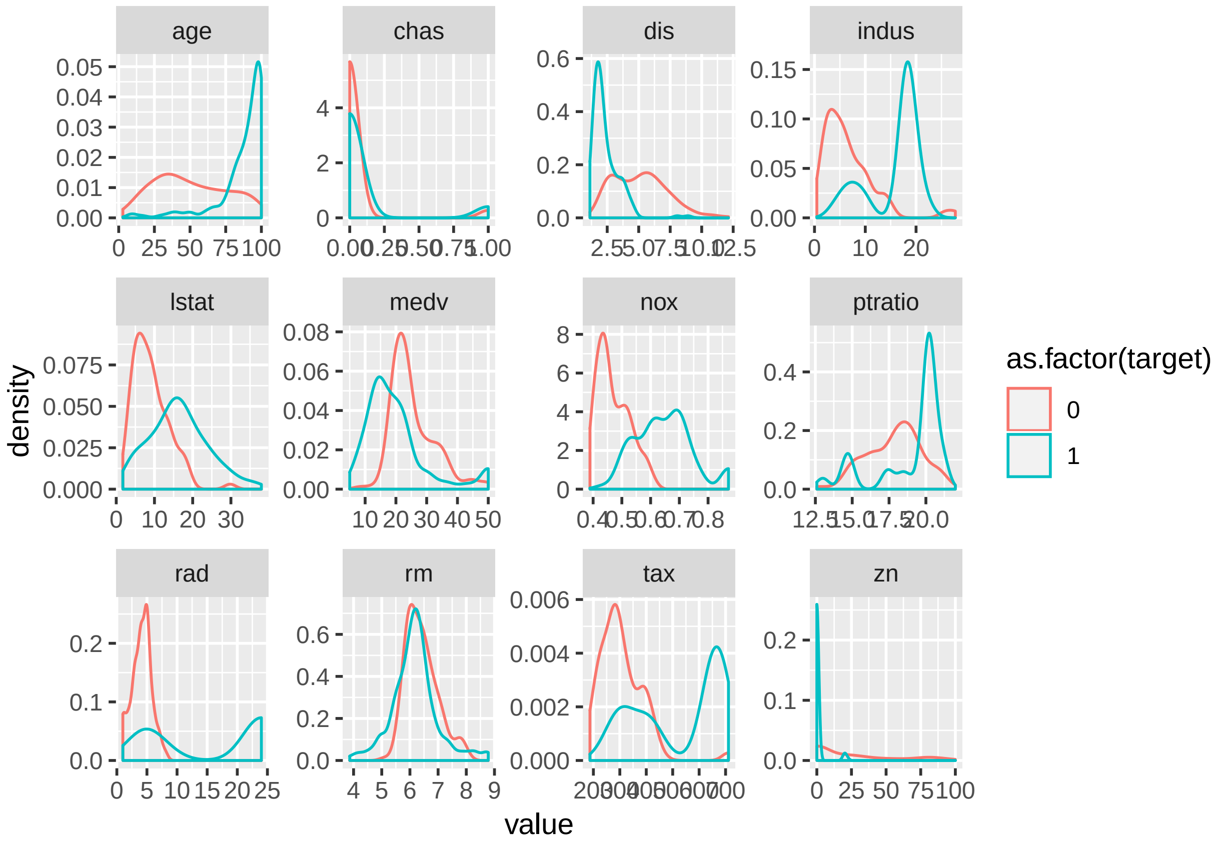
Variable	Description
zn	proportion of residential land zoned for large lots (over 25000 square feet) (predictor variable)
indus	proportion of non-retail business acres per suburb (predictor variable)
chas	a dummy var. for whether the suburb borders the Charles River (1) or not (0) (predictor variable)
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)
black	1000(Bk - 0.63)2 where Bk is the proportion of blacks by town (predictor variable)
lstat	lower status of the population (percent) (predictor variable)
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)

A quick summary of each:

```
##          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.:0.4490
## 3 Median :  0.00   Median : 9.90   Median :0.00000   Median :0.5380
## 4 Mean   : 12.09   Mean   :11.22   Mean   :0.07143   Mean   :0.5587
## 5 3rd Qu.: 16.25   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.: 44.55   1st Qu.: 1.999   1st Qu.: 4.000
## 3 Median :6.224   Median : 80.10   Median : 3.011   Median : 5.000
## 4 Mean   :6.295   Mean   : 69.19   Mean   : 3.737   Mean   : 9.846
## 5 3rd Qu.:6.630   3rd Qu.: 94.97   3rd Qu.: 5.214   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.: 6.80   1st Qu.:16.80
## 3 Median :350.0   Median :19.00   Median :11.49   Median :21.30
## 4 Mean   :415.5   Mean   :18.42   Mean   :12.70   Mean   :22.71
## 5 3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:17.10   3rd Qu.:26.35
## 6 Max.   :711.0   Max.   :22.00   Max.   :37.97   Max.   :50.00
##     X....target
## 1 Min.   :0.0
## 2 1st Qu.:0.0
## 3 Median :0.5
## 4 Mean   :0.5
## 5 3rd Qu.:1.0
## 6 Max.   :1.0
```

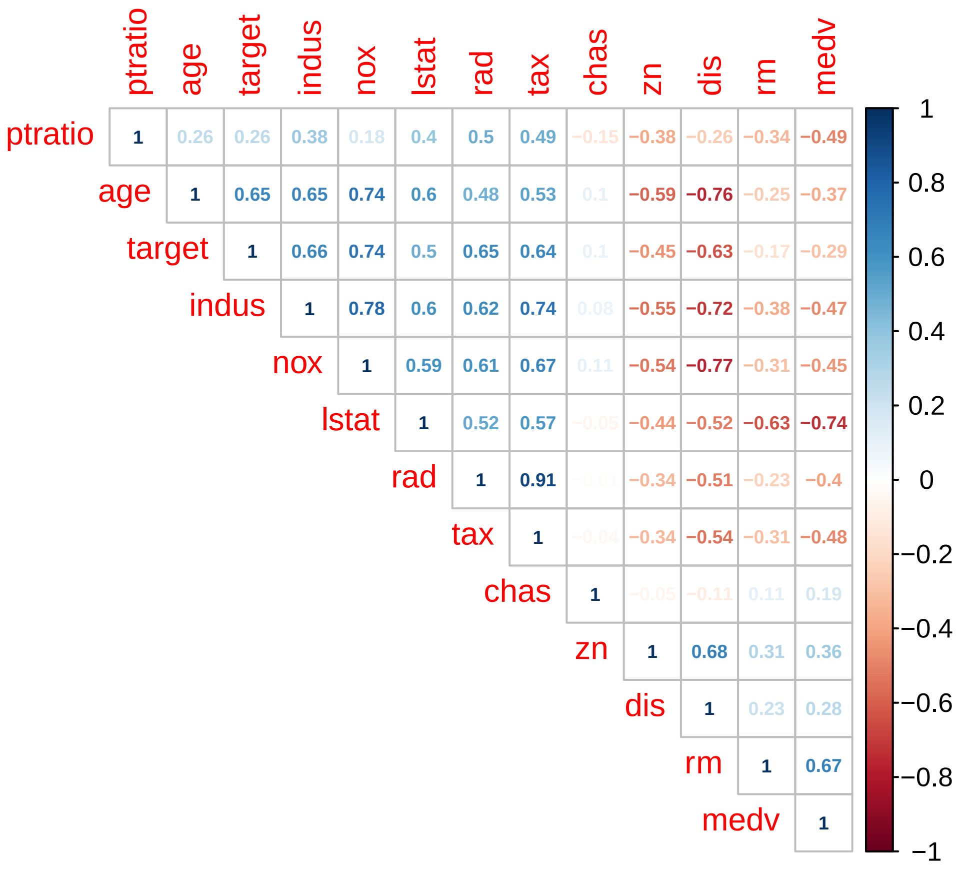
There don't seem to be any outliers or missing data, so we will proceed directly to examining the variables.

First, histograms of each variable by `target` class:



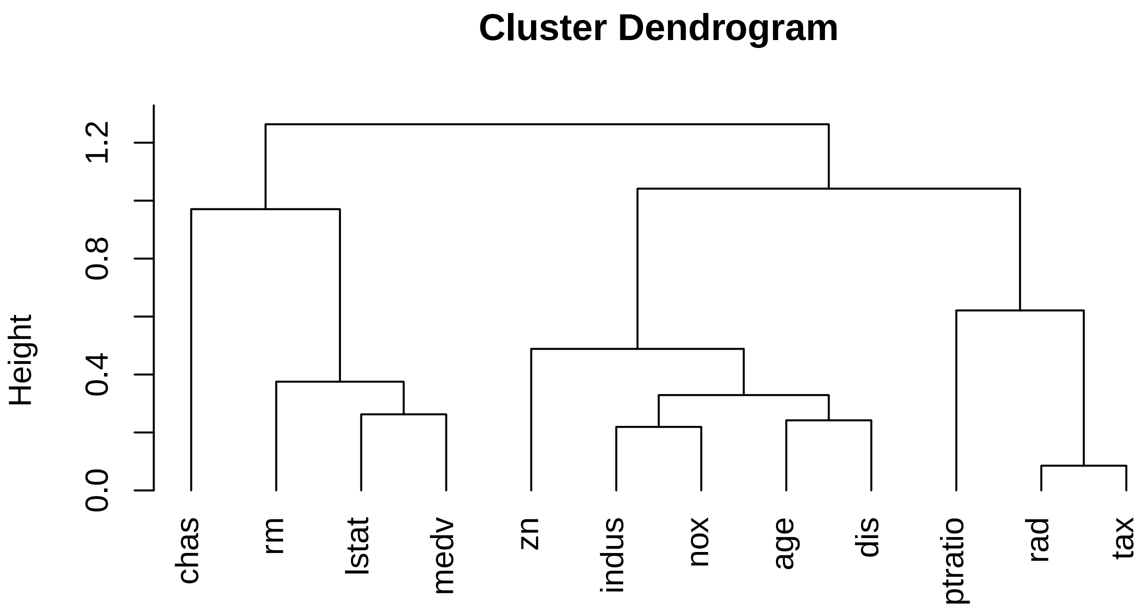
Most variables have distinct shapes for each `target` class, suggesting that they will have good discriminative ability. `chas` and `zn` are quite skewed, and do not appear terribly informative. `indus` and `tax` have two peaks for `target = 1`, which hints that there may be two seperate processes at work there.

As would be expected, many of these variables are correlated with each other:



Obviously, the concentration of industry is strongly and positively correlated with nitrogen oxide concentration ($r = 0.78$). Parent-teacher ratio is negatively correlated with median property values ($r = -0.5$), and positively correlated with property taxes ($r = 0.49$).

A further way to understand how these variables may be related is to use the `ClustOfVar` package. It provides a function to use hierarchical clustering of the independent variables:

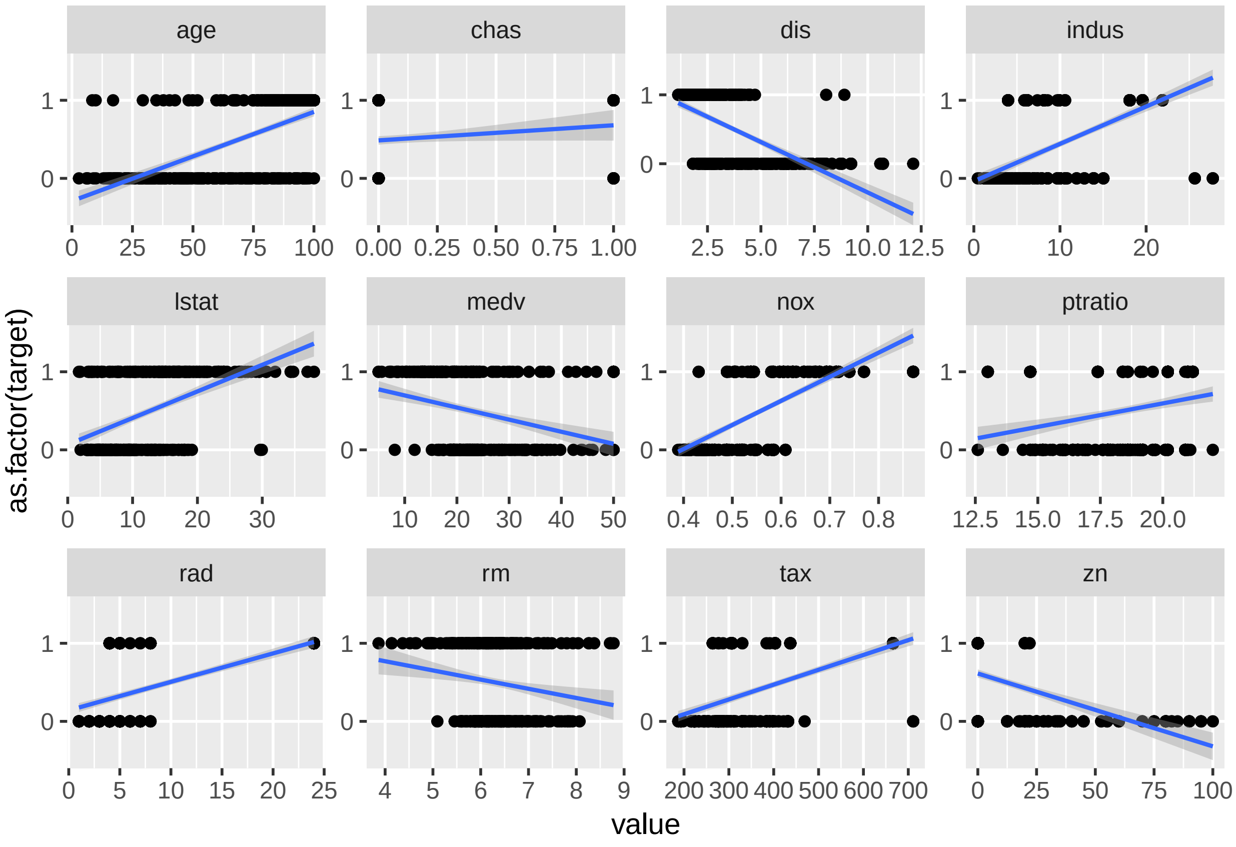


There are four major groups of variables:

1. *Charles River* — Whether the geographical zone borders the Charles River or not is its own cluster. This is consistent with the correlation chart presented above, showing that `chase` was only weakly correlated with other variables.
2. *Economic Class* — The average number of rooms per dwelling, the 'lower status of the population' percent, and the median value of homes in the zone, are the second cluster. These appear to directly relate to economic class. Wealthier areas have larger homes and lower proportions of 'lower-status' populations.
3. *Industrial-Business v. Residential* – This cluster is composed of the weighted mean of distance to employment center, the nitrogen oxides concentration, and the proportions of residential land zoned for large lots, of non-retail business acres, and of older homes. It seems to represent whether a zone is primarily industrial-business, or residential.
4. *Suburban* – Made up of the pupil-teacher ratio, an index of accessibility to the highway, and the property tax. We propose this cluster is composed of variables that distinguish whether a zone is suburban or otherwise.

This exercise will inform our modeling, allowing us to avoid including highly correlated variables in the model.[1]

Toward that end, we can directly observe the relationships between each independent variable and `target`, with the blue line representing a linear fit:



Most of these variables appear very strongly related to the dependent variable. Most of these relationships make sense: Geographical zones that are more industrial suffer from increases crime, higher parent ratio is correlated with higher crime levels, etc.

The property tax variables, `tax`, is counter-intuitive, however. The plot is essentially suggesting crime

[1]Although it is not clear from our reading that logistic regression has this same constraint as linear regression!

is associted with wealthier neighbhorhoods! Perhaps controlling for the other variables statistically will straighten this relationship out.

Data Preparation

No data prepartion was necessary, as there neither missing data points or outliers. Furthermore, the distributions of each variable is normal enough, and does warrent transformation (for the most part).

Modeling

In this section, we will fit a number of models on the training data. The 'winning' model will be selected according to its $F1$ score on the test data. Along the way, we will also examine important metrics about how each model fit the training dataset, including a confusion matrix. We will also use Nagelkerke's measure of R^2 for logistic regression, via the `DescTools::PseudoR2()` function.

M_0: Dummy model

This first model 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 far off and should re-examine our assumptions.

Ommitting the full confusion matrxi, this dummy model has an accuracy of about 0.50, a sensitivity of 1, and specificity of 0. Since it has zero predictive power, we know that it has a pseudo-R^2 of 0. Its $F1$ score is 0.67.

```
m_0 <- glm(target ~ 1, train, family=binomial())
```

```
##        F1
## 0.6666667
```

M_1: Full model

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

```
m_1 <- glm(target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
             ptratio + lstat + medv, train, family=binomial())
summary(m_1)
```

```
##
## Call:
## glm(formula = target ~ zn + indus + chas + nox + rm + age + dis +
##     rad + tax + ptratio + lstat + medv, family = binomial(),
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.73083  -0.13168  -0.00032   0.00204   2.65879
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -43.260627   8.188099  -5.283 1.27e-07 ***
## zn           -0.083548   0.043154  -1.936 0.052863 .
## indus        -0.018223   0.055825  -0.326 0.744094
## chas          0.860469   0.885425   0.972 0.331143
## nox          45.313255   9.269718   4.888 1.02e-06 ***
## rm            0.381239   0.909266   0.419 0.675010
## age           0.027249   0.015561   1.751 0.079930 .
## dis           0.730083   0.276563   2.640 0.008295 **
## rad           0.716975   0.196530   3.648 0.000264 ***
## tax          -0.006848   0.003460  -1.979 0.047804 *
## ptratio       0.356687   0.161345   2.211 0.027055 *
## lstat         0.086439   0.062657   1.380 0.167721
## medv          0.129097   0.085462   1.511 0.130897
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 485.20  on 349  degrees of freedom
## Residual deviance: 135.91  on 337  degrees of freedom
## AIC: 161.91
##
## Number of Fisher Scoring iterations: 9
```

A few variables come out highly significant. Our issue with `tax` has been fixed: there is now a negative relationship, whereas a positive was plotted. Evaluation:

```
##        F1
## 0.9126761
```

```
## Nagelkerke
##  0.8418415
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 162  18
##          1  13 157
##
##                Accuracy : 0.9114
##                  95% CI : (0.8766, 0.939)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8229
##
##  Mcnemar's Test P-Value : 0.4725
##
##             Sensitivity : 0.9257
##             Specificity : 0.8971
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.9235
##              Prevalence : 0.5000
##          Detection Rate : 0.4629
##    Detection Prevalence : 0.5143
##       Balanced Accuracy : 0.9114
##
```

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

The full model has very high $F1$ and R^2 scores. The confusion metrics also look very nice.

M_2: Stepwise variable selection with interactions

We could stop here, although we know that variable interaction is likely. We can automatically test all interactions using stepwise selection:

```
m_2 <- stepAIC(m_1, trace=0, scope=list(upper = ~ zn * indus * chas * nox * rm *
                                          age * dis * rad * tax * ptratio *
                                          lstat*medv, lower= ~1))
```

However, this model is almost certainly overfit. We've ommitted the model summary, as none of the parameters are signficant. Naturally, the $F1$ and R^2 values are impossibly high.

```
## F1
##  1
```

```
## Nagelkerke
##  0.9999427
```

By a common heuristic, we have enough data for:

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

```
## [1] 11.66667
```

i.e., 12 variables. The next model tries to par them down while keeping important interaction effects.

M_3: Interactions, part II

We need a way to include interactions without using so many variables. To this end, this model uses only one variable of each variable cluster from above, plus their interactions. The variables highest in absolute correlation with `target` was chosen from each group. The `chas` variable was dropped entirely.

```
m_3 <- glm(target ~ lstat*nox + lstat*rad + nox*rad, train, family=binomial())
summary(m_3)
```

```
##
## Call:
## glm(formula = target ~ lstat * nox + lstat * rad + nox * rad,
##     family = binomial(), data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.93168  -0.22771  -0.00930   0.02243   2.42266
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -25.50831    7.03554  -3.626 0.000288 ***
## lstat         0.73322    0.25254   2.903 0.003692 **
## nox          37.77281   13.81440   2.734 0.006251 **
## rad           0.60720    1.11339   0.545 0.585507
## lstat:nox    -1.09095    0.43391  -2.514 0.011930 *
## lstat:rad    -0.02389    0.01446  -1.652 0.098532 .
## nox:rad       0.58297    2.35787   0.247 0.804718
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 485.20  on 349  degrees of freedom
## Residual deviance: 159.93  on 343  degrees of freedom
## AIC: 173.93
##
## Number of Fisher Scoring iterations: 10
```

Strangely, `rad` was not found significant at any reasonable level. Nor was the `nox:rad` interaction. The other two interactions were both significant at at least $p < .05$. Evaluation:

```
##        F1
## 0.8764045
```

```
## Nagelkerke
##   0.806926
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 156  25
##          1  19 150
##
##                Accuracy : 0.8743
##                  95% CI : (0.8349, 0.9072)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7486
##
##  Mcnemar's Test P-Value : 0.451
##
##             Sensitivity : 0.8914
##             Specificity : 0.8571
##          Pos Pred Value : 0.8619
##          Neg Pred Value : 0.8876
##              Prevalence : 0.5000
##          Detection Rate : 0.4457
##    Detection Prevalence : 0.5171
##       Balanced Accuracy : 0.8743
##
##        'Positive' Class : 0
##
```

Both $F1$ and R^2 are very close to the full model M_1, and with only 6 variables!

M_4: PCA

Given all of the correlations between the variables, it makes sense to attempt PCA on this data set.

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

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     2.4701 1.2831 1.04445 0.91975 0.87133 0.62125
## Proportion of Variance 0.5084 0.1372 0.09091 0.07049 0.06327 0.03216
## Cumulative Proportion  0.5084 0.6456 0.73654 0.80703 0.87030 0.90246
##                            PC7     PC8     PC9    PC10    PC11   PC12
## Standard deviation     0.54482 0.53167 0.45817 0.43277 0.36572 0.2449
## Proportion of Variance 0.02474 0.02356 0.01749 0.01561 0.01115 0.0050
## Cumulative Proportion  0.92720 0.95075 0.96825 0.98386 0.99500 1.0000
```

Since we know from above there are four main groups of varaibles, and the first four components account for 80 percent of variation, we use those for modeling:

```r
pca_df <- as.data.frame(cbind(train$target, pca$x[,1:4]))
colnames(pca_df) <- c('target', 'PC1', 'PC2' ,'PC3', 'PC4')
m_4 <- glm(target ~ ., pca_df, family=binomial())
summary(m_4)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(), data = pca_df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.55238  -0.40759  -0.00738   0.24318   2.66704
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22744    0.26053   0.873  0.38265
## PC1         -1.18019    0.13072  -9.029  < 2e-16 ***
## PC2         -0.92710    0.16291  -5.691 1.26e-08 ***
## PC3         -0.62204    0.24103  -2.581  0.00986 **
## PC4         -0.09532    0.19549  -0.488  0.62584
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 485.20  on 349  degrees of freedom
## Residual deviance: 201.58  on 345  degrees of freedom
## AIC: 211.58
##
## Number of Fisher Scoring iterations: 6
```

The first three components are highly significant, and negatively related to crime level. Examine its metrics:

```
##        F1
## 0.8403361
```

```
## Nagelkerke
##  0.7403957
```

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

```
##           Reference
## Prediction   0   1
##          0 150  32
##          1  25 143
##
##                Accuracy : 0.8371
##                  95% CI : (0.7942, 0.8743)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6743
##
##  Mcnemar's Test P-Value : 0.4268
##
##             Sensitivity : 0.8571
##             Specificity : 0.8171
##          Pos Pred Value : 0.8242
##          Neg Pred Value : 0.8512
##              Prevalence : 0.5000
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.5200
##       Balanced Accuracy : 0.8371
##
##        'Positive' Class : 0
##
```

There is similar performance to previous models, although the $F1$ is higher and the R^2 lower compared to M_3.

Evaluating the Models on the Test Set

Evaluate each model based on its $F1$ score on the test set, although we include other metrics as a convenianceS:

```
# pred_0_test <- factor(round(predict(m_0, test, type='response')), levels=c('0', '1'))
# m_0_test <- caret::confusionMatrix(data=pred_0_test, reference=factor(test$target, levels=c('0', '1')
# m_0_test$byClass['F1']
#
# pred_1_test <- factor(round(predict(m_1, test, type='response')), levels=c('0', '1'))
# m_1_test <- caret::confusionMatrix(data=pred_1_test, reference=factor(test$target, levels=c('0', '1')
# m_1_test$byClass['F1']
#
# pred_2_test <- factor(round(predict(m_2, test, type='response')), levels=c('0', '1'))
# m_2_test <- caret::confusionMatrix(data=pred_2_test, reference=factor(test$target, levels=c('0', '1')
# m_2_test$byClass['F1']
#
# pred_3_test <- factor(round(predict(m_3, test, type='response')), levels=c('0', '1'))
# m_3_test <- caret::confusionMatrix(data=pred_3_test, reference=factor(test$target, levels=c('0', '1')
# m_3_test$byClass['F1']
#
# pred_4_test <- factor(round(predict(m_4, as.data.frame(predict(pca, newdata=test)), type='response'))
#                         levels=c('0', '1'))
# m_4_test <- caret::confusionMatrix(data=pred_4_test, reference=factor(test$target, levels=c('0', '1')
# m_4_test$byClass['F1']
```

	Description	F1	Accuracy	Sensitivity	Specificity
M_0	dummy	0.6966	0.53448	1.0000	0.0000
M_1	full	0.9076	0.8965	**0.9516**	0.8333
M_2	all + interactions	**0.9344**	**0.9310**	0.9193	**0.9444**
M_3	parred down + interactions	0.8615	0.8448	0.9032	0.7777
M_4	PCA	0.8062	0.7844	0.8387	0.7222

All of our models performed significantly better than the dummy model. This shows that implementing this model for some purpose would make sense.

Surpisingly, the large model with all variables plus all their interactions proved to be *not* overfit, even though it was composed of almost entirely insignicant variables. This model also beat all the others on accuracy and specificity. The full model also performed very well.

The PCA model M_4 was the weakest of our serious models. It also suffers from the issue of not being as interpretable.

Inference on the Final Model

Since M_2 performed the best on the test data set, it makes sense to use it if out purposes are purely predictive. However, it is remarkable that a model with only 6 variables could come so close to one with 28. For purposes of inference, it makes sense to use M_3.

Re-running on the full model:

```r
df <- rbind(train, test)
m_3_star <- glm(target ~ lstat*nox + lstat*rad + nox*rad, df, family=binomial())
summary(m_3_star)
```

```
##
## Call:
## glm(formula = target ~ lstat * nox + lstat * rad + nox * rad,
##     family = binomial(), data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0218  -0.2839  -0.0398   0.0214   2.5044
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -23.59296    5.91825  -3.986 6.71e-05 ***
## lstat         0.76309    0.19688   3.876 0.000106 ***
## nox          35.34895   11.56633   3.056 0.002242 **
## rad           0.02625    0.95644   0.027 0.978101
## lstat:nox    -1.18581    0.34392  -3.448 0.000565 ***
## lstat:rad    -0.02540    0.01130  -2.249 0.024539 *
## nox:rad       1.61203    2.00028   0.806 0.420298
## ---
## 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: 225.85  on 459  degrees of freedom
## AIC: 239.85
##
## Number of Fisher Scoring iterations: 10
```

Transforming the coefficients:

```r
format(exp(coef(m_3_star)), scientific=FALSE)
```

```
##                           (Intercept)                                 lstat
## "                0.00000000005671612" "                2.14488716162942739"
##                                   nox                                   rad
## "2248305377840071.00000000000000000" "                1.02660189102795552"
##                             lstat:nox                             lstat:rad
## "                0.30549979032242131" "                0.97492180512146454"
##                               nox:rad
## "                5.01299060478035940"
```

The intercept is highly significant, and for once makes sense with this data set. If a geographic zones contains 0 `lower status` population, no nitrous oxide, and is directly accessible to the highway, we can expect it to have `target=0` on average.

This data does not provide sufficient evidence to reject the null hypotheses that the coefficents for `rad` is 0, nor that the interaction between `nox` and `rad` is 0.

We have found that there is a positive relationship between `lstat`, the porportion of 'lower status' population, and crime levels. When `nox` and `rad` are zero, each additional percentage point of `lstat` increases the odds of high crime levels by 114 percent. The interaction effects show that the effect of `lstat` decreases as `nox` and `rad` increase.

`nox` has a very strong effect on whether a geographic zone suffers from high crime levels. It seems to suggest that each additional part per million of nitrogen oxides concentration increases the odds of high crime level by an almost absurd amount. It is worth noting that other models, including M_1, returned similarly large estimated coefficients.