HW3

Critical Thinking Group One

2021-04-16

Contents

Authorship

Critical Thinking Group 1

- Angel Claudio
- Bonnie Cooper
- Manolis Manoli
- Magnus Skonberg
- Christian Thieme
- Leo Yi

Background

In the following exercises we will be working with **The Boston Housing Dataset**. The dataset was gathered by the US Census Bureau regarding housing in the Boston Massachussetts area and can be obtained from the StatLib archive.

This dataset has been used extensively in academic literature and for Kaggle competitions because it can be used as a benchmark for different algorithms. The dataset is relatively small (506 observations) and contains the following variables:

- `zn` proportion of residential land zoned for lots over 25,000 sq.ft.
- `indus` proportion of non-retail business acres per town
- `chas` Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- `nox` nitric oxides concentration (parts per 10 million)
- `rm` average number of rooms per dwelling
- `age` proportion of owner-occupied units built prior to 1940
- `dis` weighted distances to five Boston employment centres
- `rad` index of accessibility to radial highways
- `tax` full-value property-tax rate per $10,000
- `ptratio` pupil-teacher ratio by town
- `lstat` % lower status of the population
- `medv` median value of owner-occupied homes in $1000's

- `target` **response variable** indicating whether or not the crime rate is above the median crime rate (1) or not (0)

The purpose of the assignment is to build logistic regression models on the boston housing training data to predict whether or not neighborhoods are at risk for high crime, and validate the model with the greatest predictive accuracy.

Our Approach

The goal of our modeling approach is to find the optimal binary logistic regression model that utilizes this feature set to predict whether or not a neighborhood has a high crime rate.

Along the way, we will explore numerous logistic regression models:

1. **baseline model**: no data preparation –> "baseline" predictive accuracy
2. **outlier optimized model**: observe the affect that dealing with outliers (alone) has on our predictive accuracy
3. **outlier and feature engineered model**: observe the affect that dealing with outliers *and* engineering features has on our predictive accuracy
4. **AIC optimized model**: apply stepAIC() function to outlier and feature engineered model to *automatically* identify (and then remove) impertinent features.

We start by importing the data, performing a *thorough* exploratory data analysis, and preparing our data with a series of transformational steps (ie. dealing with outliers, feature engineering and removal). We then move on to building each model, exploring corresponding predictive accuracies, verifying each model's predictive capability with our test data, and then highlighting the strongest model from those listed above.

……………………………………………………………………..

Data Exploration & Baseline Model

> "Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise." - John Tukey

The goal of exploratory data analysis (or EDA for short) is to enhance the precision of the questions we're asking. To generate questions, we search for answers via visualization, transformation, and modeling. Then, we use what we learn to refine our questions or generate new questions. It's an iterative process where the end goal is to *really* grasp and understand the data at hand.

For our approach, we first get to know the structure and value ranges, we then look at the distributions of our features, visualize the relationship between our numeric variables and the `target` variable, visualize the relationship our numeric variables have with one another, and then build our "baseline" logistic regression model with the aim of verifying multicollinearity via **vif()** function. After this point, we should have enough insight to prepare our data and then build our model.

To start, we utilize the built-in `glimpse()` method to gain insight into the dimensions, variable characteristics, and value range for our training dataset:

```
## 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    <dbl> 0, 1, 0, 0, 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     <dbl> 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  <dbl> 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0,~
```

From above, we see that our training dataset has **13** variables (of type double) and **466** observations, all positive numeric values with varying ranges (shown across each row), and two features (`chas` and `target`) that should be factors.

We update these features accordingly, combine datasets (to reduce duplication of work in data cleaning and feature engineering), and utilize **skimr()** and **inspectdf()** functions to take a more detailed look at the visualization of categorical vs. numeric predictor variables:

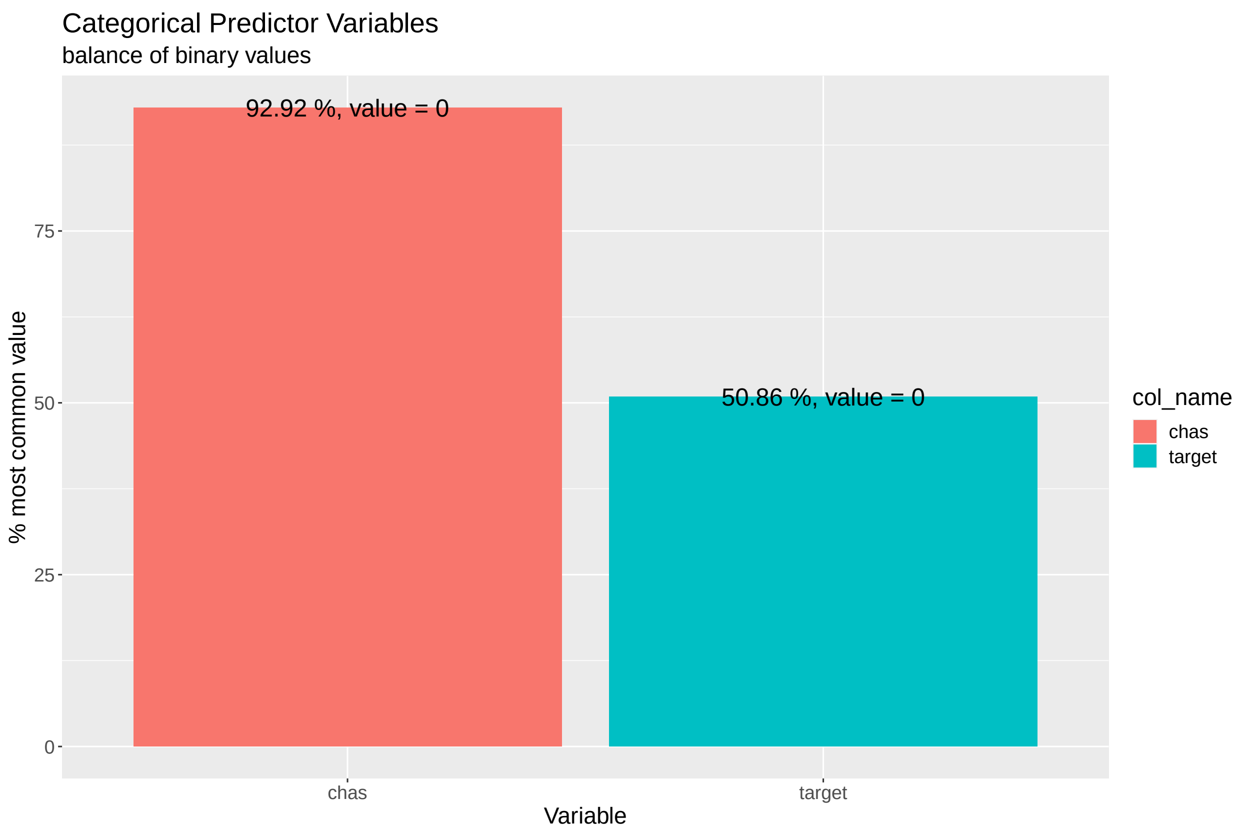

Figure 1: Most common level of categorical features. The balance of values for binary feature variables: `chaz` (red) : ~93% value=0 whereas `target` is evenly balanced with ~51% value = 0

From our earlier `glimpse` output and the visualization features (`chas` and `target`) we observe that:

- there are no nulls in our dataset,
- `target` is split almost fifty-fifty (a balanced set), and
- more than 90 percent of `chas` values are 0, which means most of neighborhoods are not near Charles River.

With regard to numeric predictor variables:

We observe that:

- `age` is highly left skewed, meaning a lot of homes in our dataset were built prior to 1940.

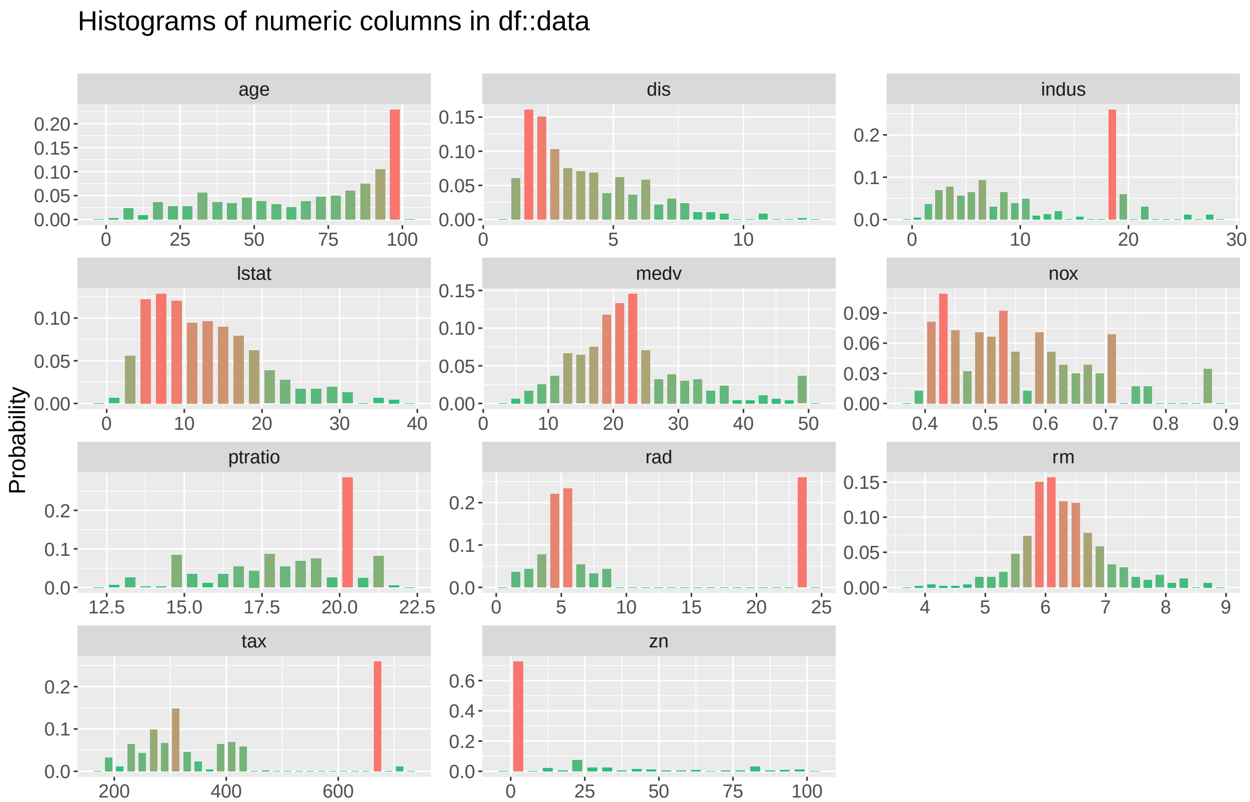

Figure 2: Distributions of numeric feature variables

- `dis` is highly right skewed, meaning many of the homes are in close proximity to Boston employment centers. We might venture to say that being close to one, indicates that there is more need for one.
- `indus` appears to overall have a fairly low proportion of non-retail business acres, although we see a spike ~18%.
- `lstat` is right skewed, reaching its peak between 5-20%.
- `medv` is slightly right skewed, reaching its peak between $17-25K.
- `nox` - is a multi-modal distribution, concentrated between 0.4-0.6, with its last significant spike ~0.7.
- `ptratio` - is a relatively uniform distribution ranging from 12.5-22.5 with a significant spike ~21.
- `rad` - is a bimodal distribution with peaks ~5 and ~22.5. It appears that individuals are either very close OR very far from radial highways.
- `rm` - is a relatively normal distribution with a peak ~6 and the greatest concentration of rooms between 5.5-7.
- `tax`- is a bimodal distribution with one peak ~300 and a larger peak ~650. There looks to be a split where either the tax value is on the lower end, or very high (600K+).
- `zn` - is highly right skewed and may be adjusted as a categorical variable (0,1). It looks like the majority of land is not zoned for large lots.

With an initial understanding of our data, we visualize the relationship between our numeric variables and `target` variable via boxplots:

There are outliers in 9/11 features. Being that generalized regression is sensitive to outliers, we'll need to pay attention to the effect these outliers (and our dealing with them) have on the predictive accuracy of our model. We saw what appeared to be some outliers in our earlier histograms and so we'll focus our analysis on looking at the median values of each feature:

- `age` appears to be highly correlated with target. The higher the proportion of homes that are built prior to 1940, the higher risk for crime. This makes sense as generally older homes and neighborhoods tend to be less expensive
- `dis` appears to be highly correlated. The closer to employment centers, the higher the risk of crime.
- `indus` appears to be correlated. Neighborhoods where there is a higher proportion of non-retail business acres are associated with higher crime
- `lstat` - appears to be somewhat correlated. The higher the proportion of 'lower status' the more crime.

- `medv` - appears to be somewhat correlated. The higher the median value of a home, the lower the rate of crime.
- `nox` - appears to be highly correlated. The more pollution, the higher the risk of crime.
- `ptratio` - appears to be correlated. Where there are more students per teacher, there's a higher risk of crime.
- `rad` - has an on relationship / correlation. The values that have a higher index make up a big chunk of the population and make our boxplot for 1-higher crime look a little strange. This may need to be broken into a separate feature.
- `rm` - does not appear to be correlated, and we'll verify via correlation matrix later.
- `tax`- has a similar correlation relationship to `rad` and may need to be broken into a separate feature as well.
- `zn` - appears to be weakly correlated. A large proportion of the values are 0. This feature does not appear to provide much signal and may be deemed a "throwaway".

Having reviewed the relationship each of our numeric features has with the `target` variable, we'll turn our attention to exploring the relationship these variables have with one another via **pairs** plot and **correlation matrix**:

From the pairs plot (Figure 4) we observe that:

- `age`, `indus`, `ptratio`, `rad`, `tax`, and `zn` all have odd distributions,
- many features appear to be highly correlated with one another, with some distributions even taking on an exponential curve, and
- it appears there are more features that are related than not.

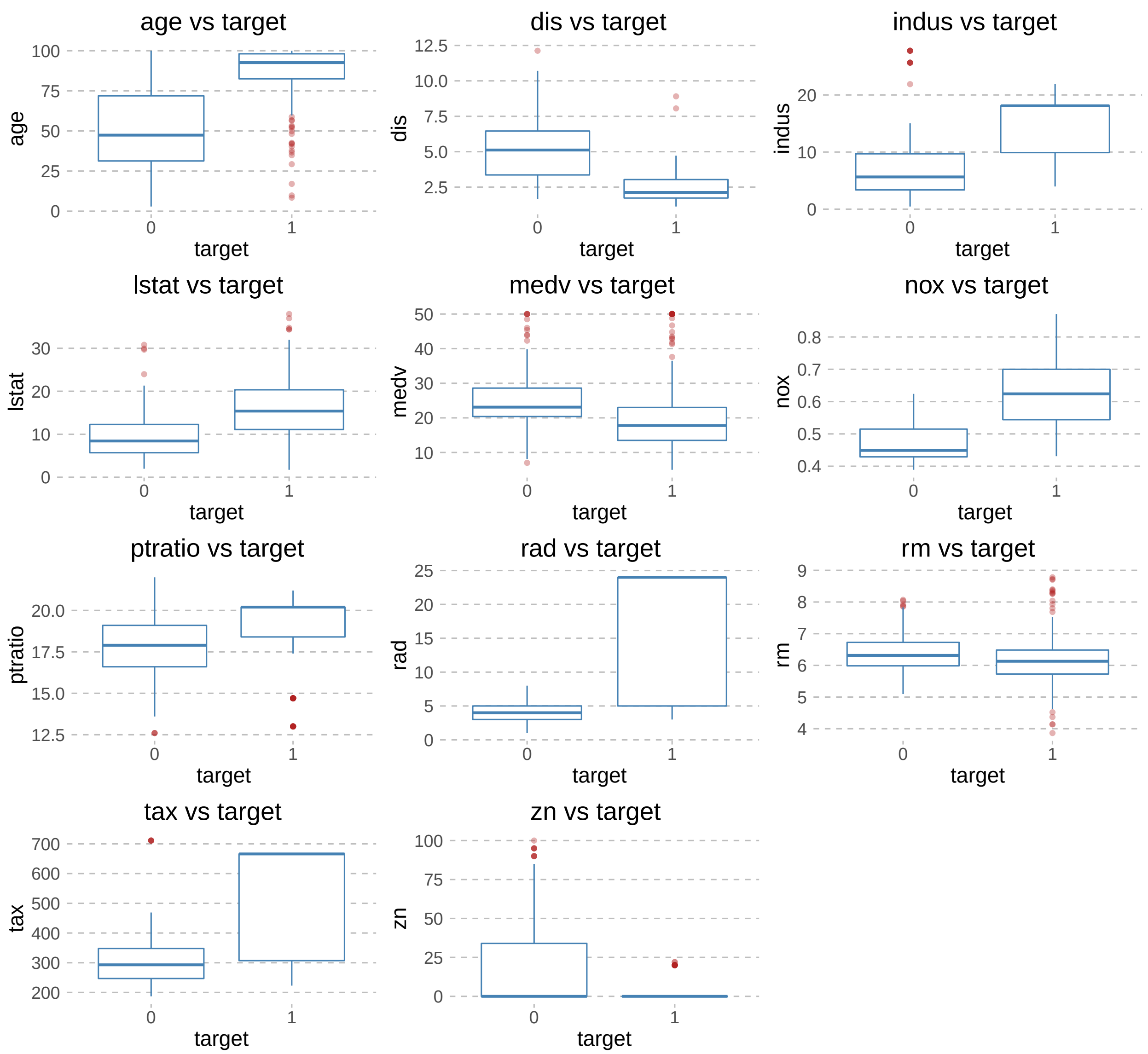

Figure 3: Relationship of numeric feature variables to the 'target' variable

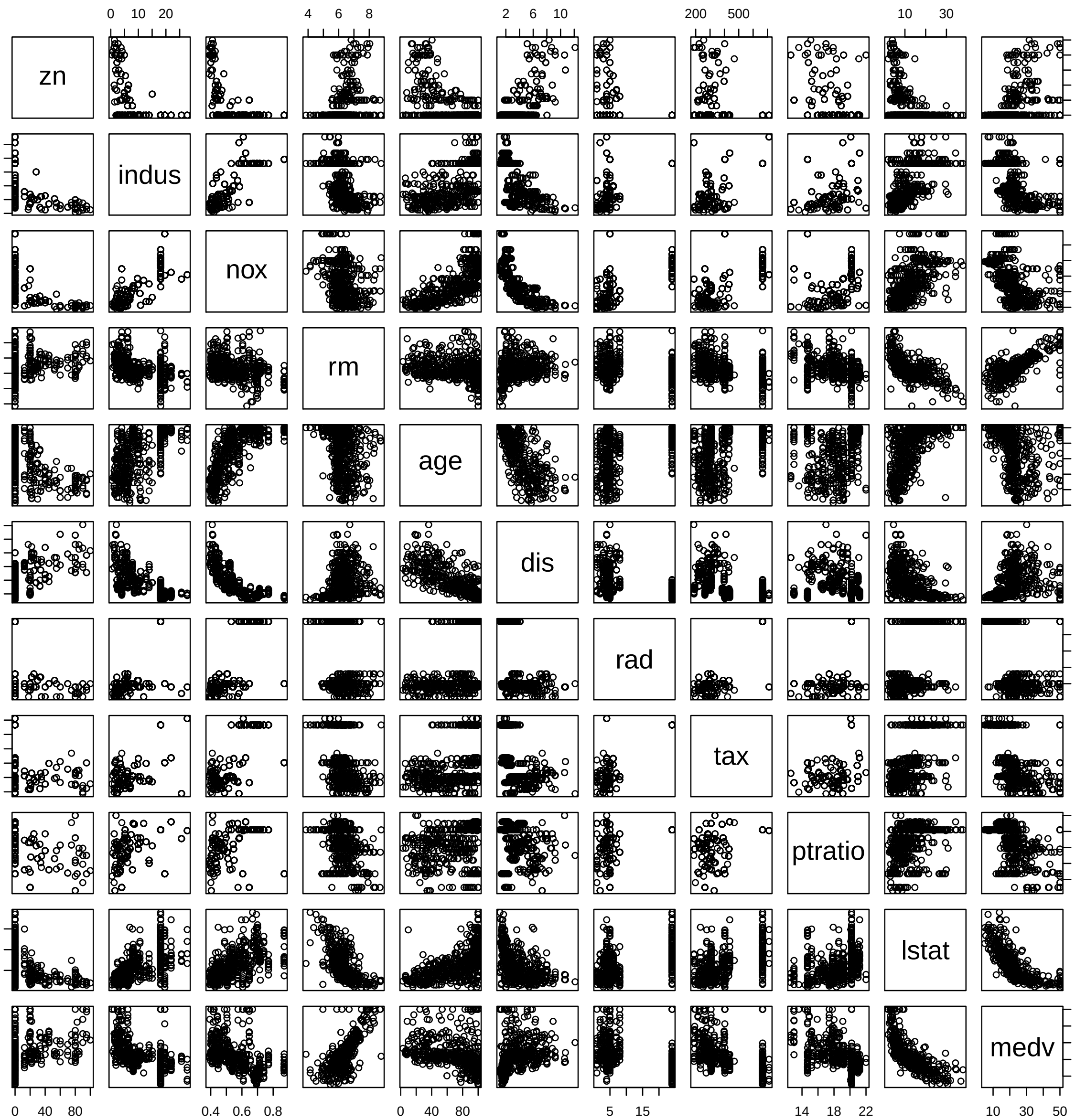

Figure 4: Paired plots of Feature Variables. clear relationships between pairs of feature variables are observed

To confirm or deny this fact, we explore the corresponding correlation matrix:

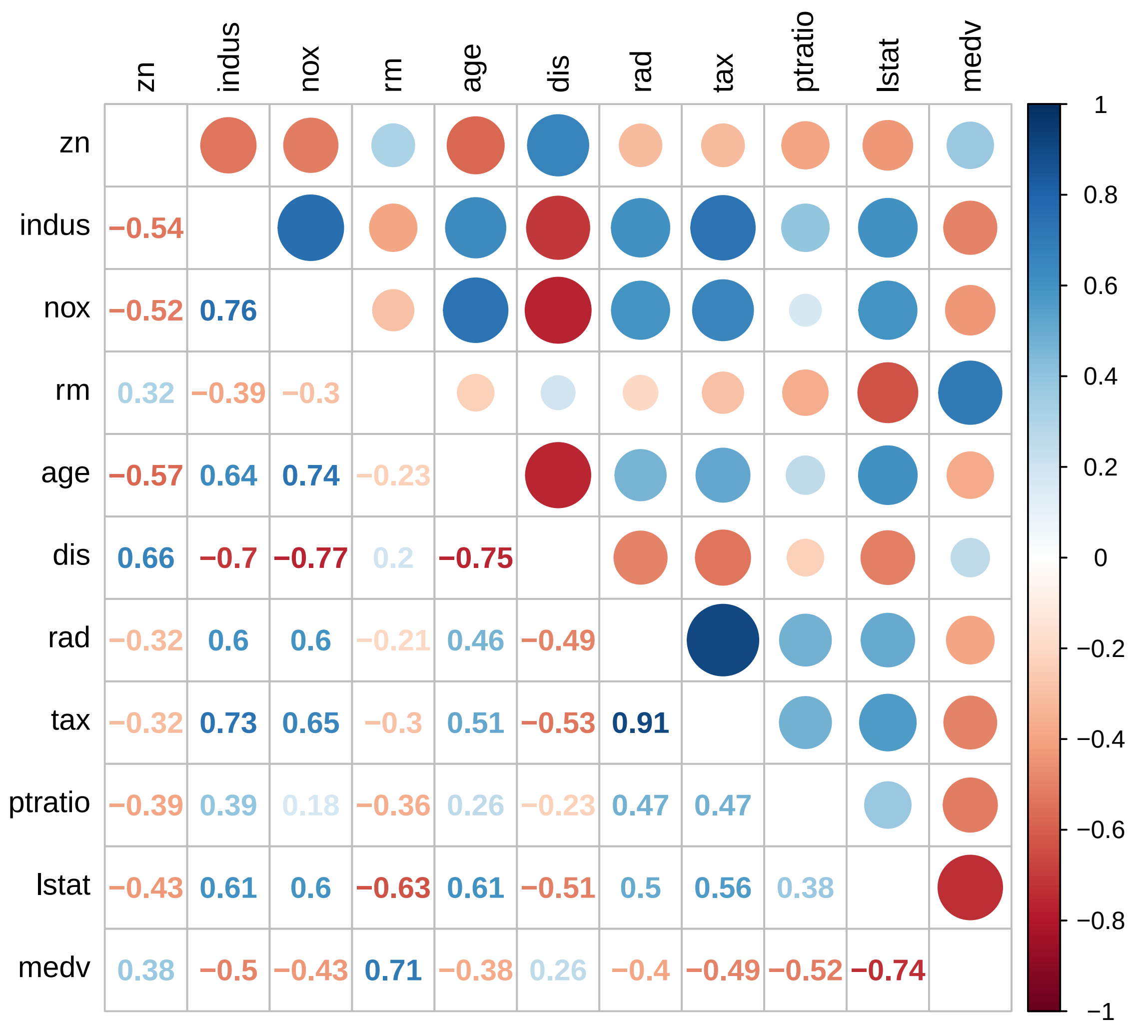

Figure 5: Correlation Matrix of feature variables. Multicolinearity between variables are observed

Our correlation matrix confirms that multicollinearity is a concern.

Model 1: Baseline

We dig deeper into this issue by running a "baseline" logistic regression model - one without data preparation, feature removal or engineering - and then applying variance inflation factor (VIF) analysis:

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
```

```
##     data = train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8464  -0.1445  -0.0017   0.0029   3.4665
##
## Coefficients:
##               Estimate Std. Error z value       Pr(>|z|)
## (Intercept) -40.822934   6.632913  -6.155 0.000000000753 ***
## zn           -0.065946   0.034656  -1.903        0.05706 .
## indus        -0.064614   0.047622  -1.357        0.17485
## chas1         0.910765   0.755546   1.205        0.22803
## nox          49.122297   7.931706   6.193 0.000000000590 ***
## rm           -0.587488   0.722847  -0.813        0.41637
## age           0.034189   0.013814   2.475        0.01333 *
## dis           0.738660   0.230275   3.208        0.00134 **
## rad           0.666366   0.163152   4.084 0.000044204596 ***
## tax          -0.006171   0.002955  -2.089        0.03674 *
## ptratio       0.402566   0.126627   3.179        0.00148 **
## lstat         0.045869   0.054049   0.849        0.39608
## medv          0.180824   0.068294   2.648        0.00810 **
## ---
## 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: 192.05  on 453  degrees of freedom
## AIC: 218.05
##
## Number of Fisher Scoring iterations: 9
```

From the above output statistics we observe that our baseline model AIC value is 218.05 and we may indeed be carrying impertinent features (ie. `indus`, `chas`, `rm`, and `lstat`).

Next we verify the statistical significance of our "baseline" model:

```
## [1] 453.8289
```

```
## [1] 1.452336e-89
```

Based on the relatively high null deviance (the 1st output) and extraordinarily low p-value (the 2nd output), our model is a better fit than the null-model.

We proceed to the corresponding confusion matrix:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Risk Normal
##     Risk    207     17
##     Normal   22    220
##
##                Accuracy : 0.9163
##                  95% CI : (0.8874, 0.9398)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                   Kappa : 0.8325
##
##  Mcnemar's Test P-Value : 0.5218
##
##             Sensitivity : 0.9039
##             Specificity : 0.9283
##          Pos Pred Value : 0.9241
##          Neg Pred Value : 0.9091
##              Prevalence : 0.4914
##          Detection Rate : 0.4442
##    Detection Prevalence : 0.4807
##       Balanced Accuracy : 0.9161
##
##        'Positive' Class : Risk
##
```

We observe a relatively high "baseline". Our model, with no data cleansing or feature engineering, predicts with an accuracy of ~92%.

To verify multicollinearity, we make use of the **vif()** function from the car library:

```
##       zn    indus     chas      nox       rm      age      dis      rad
## 1.823146 2.682271 1.241479 4.160497 5.813851 2.569961 3.887981 1.942967
##      tax  ptratio    lstat     medv
## 2.144040 2.275557 2.642656 8.122037
```

Surprisingly, not one of our VIF values surpass the >= 10 threshold for high correlation.

With multicollinearity in check, we proceed to data preparation with all features.

……………………………………………………………………..

Data Preparation & Model Building

With insights gained via EDA, we can now identify and handle outliers, engineer features that may improve our model, and drop impertinent features from consideration based on accuracy and AIC value.

Handle Outliers

Multicollinearity and outliers can have a strong negative influence on general regression models. Being that multicollinearity was addressed at the end of EDA, we're going to deal with the outliers in our set here (before engineering and removing features).

Consulting diagnostic plots confirmed the presence of outliers (i.e. observations 338, 62, 457, 14). With the understanding, that these points could heavily skew our model, we determined it best to identify and remove outliers.

To do so, we calculate the cooks distance for all observations, filter these observations for the most extreme values (), and then remove the identified observations from consideration:

```
##  [1]   9  14  37  56  62  78 100 137 138 145 210 212 216 218 235 240 280 297 304
## [20] 338 353 354 433 440 457 458
```

Using $4 * mean(cooksDistance)$, 26 observations (noted above) were removed from consideration. Resulting in a revised training dataset with 466 (original observations) - 26 (outliers) = 440 observations.

Model 2: Outlier Optimized

Let's observe the impact of outlier removal on model performance:

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##     data = train3)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9084  -0.0011   0.0000   0.0000   2.2383
##
## Coefficients:
##                Estimate   Std. Error z value     Pr(>|z|)
## (Intercept) -109.6344700   19.8228344  -5.531 0.0000000319 ***
## zn            -0.3778651    0.1396592  -2.706     0.006818 **
## indus         -0.2728470    0.1154607  -2.363     0.018122 *
## chas1          3.2850914    4.0660192   0.808     0.419126
## nox          143.7442389   26.0662736   5.515 0.0000000350 ***
## rm            -1.8059257    1.4784220  -1.222     0.221888
## age            0.1289781    0.0358352   3.599     0.000319 ***
## dis            2.4256151    0.6018874   4.030 0.0000557734 ***
## rad            1.5119714    0.3779760   4.000 0.0000632947 ***
## tax           -0.0142585    0.0062133  -2.295     0.021743 *
## ptratio        0.8243433    0.2985155   2.761     0.005754 **
## lstat          0.0008247    0.1123041   0.007     0.994141
## medv           0.4530878    0.1577972   2.871     0.004087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 609.933  on 439  degrees of freedom
## Residual deviance:  71.153  on 427  degrees of freedom
## AIC: 97.153
##
## Number of Fisher Scoring iterations: 11
```

While the high p-values of certain features (i.e. `chas1`, `rm`, and `lstat`) is indicative that we may be carrying impertinent features, **our AIC score dropped from 218.05 to 97.15**.

The Akaike Information Criterion (AIC score) deals with the model's goodness of fit and simplicity (i.e. feature pertinence). It estimates the relative amount of information lost by a given model, accounts for over vs. under fitting, and is typically used as a means of model selection.

The lower the score, the better the perceived prediction error rate. As noted earlier, we're going to use the AIC score in conjunction with model accuracy to determine model optimization (and later model selection).

As for the impact of outlier removal on model accuracy:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Risk Normal
##     Risk    210      5
##     Normal    8    217
##
##                Accuracy : 0.9705
##                  95% CI : (0.95, 0.9842)
```

```
##      No Information Rate : 0.5045
##      P-Value [Acc > NIR] : <2e-16
##
##                    Kappa : 0.9409
##
##   Mcnemar's Test P-Value : 0.5791
##
##              Sensitivity : 0.9633
##              Specificity : 0.9775
##           Pos Pred Value : 0.9767
##           Neg Pred Value : 0.9644
##               Prevalence : 0.4955
##           Detection Rate : 0.4773
##     Detection Prevalence : 0.4886
##        Balanced Accuracy : 0.9704
##
##         'Positive' Class : Risk
##
```

From the above output, we see that **our accuracy improved from less than 92% to greater than 97%**.

Removing outliers was a major step in optimizing our model's accuracy and AIC score.

Engineer Features

To deal with the odd distribution shapes observed earlier (i.e. `tax` and `rad`), we work to identify feature adaptations that may improve our model's accuracy.

To engineer features we remove outliers from the training set, combine our training and testing sets so that we don't have to make features twice, and then create new features based on patterns identified during EDA:

We can confirm our observation number via simple arithmetic: 466 (training) - 26 (outliers) + 40 (testing) = 480 observations. And thus, `final_df` is a 480 observation x 22 feature (1 `target`) dataframe.

We engineered eight features in total (5 flag features and 3 combination features):

- `age_greater_than_85` receives a value of 1 if >= 85 years, 0 otherwise
- `distance_band` receives a value of 1 if < 4 miles from employment centers, 0 otherwise
- `indus_flag` receives a value of 1 if > 15 proportion of non-retail business acres per town, 0 otherwise
- `lstat_and_rad` receives a value of 1 if > 20 % of population are lower status *and* accessibility to radial highways is > 4, 0 otherwise
- `lstat_flag` receives a value of 1 if > 12 % of population are lower status, 0 otherwise
- `medv_and_tax` receives a value of 1 if median value of owner-occupied homes in $1000's is < 17 *and* full-value property-tax rate per $10,000 is > 350, 0 otherwise
- `high_nox` receives a value of 1 if nitric oxides concentration (parts per 10 million) is > .6, 0 otherwise
- `ptratio_and_lstat` receives a value of 1 if the parent-teacher ratio is > 20 *and* > 15% of population are lower status, 0 otherwise

While it may seem a bit excessive to have engineered this many features, we found that their combination improved our accuracy. Additionally, we know we can drop impertinent features later during feature removal (if deemed necessary).

Model 3: Feature Engineered

As a next step, we filter for training data, drop the `dataset` feature, fit our model and visit the corresponding summary statistics and confusion matrix:

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##     data = train4)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.77951  -0.00008   0.00000   0.00000   2.48396
##
## Coefficients:
##                        Estimate Std. Error z value  Pr(>|z|)
## (Intercept)          -140.10957   33.88769  -4.135 0.0000356 ***
## zn                     -0.52652    0.25284  -2.082  0.037306 *
## indus                  -0.15226    0.42480  -0.358  0.720018
## chas1                   4.19629    5.96507   0.703  0.481758
## nox                   185.03842   49.90604   3.708  0.000209 ***
## rm                     -2.91561    1.65781  -1.759  0.078627 .
## age                     0.18746    0.05935   3.158  0.001586 **
## dis                     3.46235    1.17068   2.958  0.003101 **
## rad                     2.01558    0.61124   3.298  0.000975 ***
## tax                    -0.02622    0.01375  -1.906  0.056605 .
## ptratio                 1.02105    0.45513   2.243  0.024870 *
## lstat                  -0.01904    0.21933  -0.087  0.930814
## medv                    0.64043    0.21747   2.945  0.003231 **
## age_greater_than_851   -1.97673    1.39022  -1.422  0.155059
## distance_band1          1.10020    1.79472   0.613  0.539863
## indus_flag1           -11.74926   93.38354  -0.126  0.899877
## lstat_and_rad1         -0.34362    8.64468  -0.040  0.968293
## lstat_flag1            -0.40122    1.59115  -0.252  0.800922
## medv_and_tax1          -7.24980    3.01840  -2.402  0.016311 *
## high_nox1              13.20227   93.30537   0.141  0.887479
## ptratio_and_lstat1      4.62637    2.32561   1.989  0.046666 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 609.933  on 439  degrees of freedom
## Residual deviance:  49.605  on 419  degrees of freedom
## AIC: 91.605
##
## Number of Fisher Scoring iterations: 12
```

While the high p-values of certain features (i.e. `indus`, `chas1` and `lstat`) are indicative that we may be carrying impertinent features, **our AIC score dropped from 97.15 to 91.605**.

As for the impact of feature engineering on model accuracy:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Risk Normal
##     Risk    214      3
##     Normal    4    219
##
```

```
##                Accuracy : 0.9841
##                  95% CI : (0.9675, 0.9936)
##     No Information Rate : 0.5045
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9682
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9817
##             Specificity : 0.9865
##          Pos Pred Value : 0.9862
##          Neg Pred Value : 0.9821
##              Prevalence : 0.4955
##          Detection Rate : 0.4864
##    Detection Prevalence : 0.4932
##       Balanced Accuracy : 0.9841
##
##        'Positive' Class : Risk
##
```

From the above output, we see that **our accuracy improved from 97.05% to 98.41%**.

Engineering features, although not as impactful as outlier removal, was another positive step in optimizing our model's accuracy and AIC score.

Remove Features

Before finalizing our model, we've got to put its features under the microscope. We've got to determine whether or not each feature adds real value to the model.

In reviewing summary statistics up to this point there have been numerous features with high p-values. These high p-values can be indicative of impertinent features and thus our model may be optimized by their dropping.

As a next step, we aim to maintain or optimize our model's accuracy and AIC score as we identify and drop impertinent features.

For removing insignificant features, we utilize the **stepAIC()** function to identify an AIC-optimized model (without impertinent features):

```
## Start:  AIC=91.61
## target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + lstat + medv + age_greater_than_85 + distance_band +
##     indus_flag + lstat_and_rad + lstat_flag + medv_and_tax +
##     high_nox + ptratio_and_lstat
##
##                       Df Deviance    AIC
## - lstat_and_rad        1   49.607  89.607
## - lstat                1   49.613  89.613
## - lstat_flag           1   49.669  89.669
## - indus                1   49.734  89.734
## - distance_band        1   49.998  89.998
## - indus_flag           1   50.175  90.175
## - chas                 1   50.465  90.465
## - high_nox             1   50.829  90.829
## <none>                     49.605  91.605
```

```
## - age_greater_than_85  1   51.875  91.875
## - rm                  1   53.197  93.197
## - tax                 1   54.213  94.213
## - ptratio_and_lstat    1   54.879  94.879
## - ptratio             1   55.050  95.050
## - medv_and_tax         1   58.638  98.638
## - zn                  1   59.408  99.408
## - dis                 1   60.615 100.615
## - medv                1   61.427 101.427
## - age                 1   67.135 107.135
## - rad                 1   84.375 124.375
## - nox                 1  106.371 146.371
##
## Step:  AIC=89.61
## target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + lstat + medv + age_greater_than_85 + distance_band +
##     indus_flag + lstat_flag + medv_and_tax + high_nox + ptratio_and_lstat
##
##                       Df Deviance     AIC
## - lstat                1   49.615  87.615
## - lstat_flag           1   49.670  87.670
## - indus                1   49.735  87.735
## - distance_band        1   49.998  87.998
## - indus_flag           1   50.178  88.178
## - chas                 1   50.575  88.575
## - high_nox             1   50.832  88.832
## <none>                     49.607  89.607
## - age_greater_than_85  1   51.895  89.895
## - rm                  1   53.225  91.225
## - tax                 1   54.220  92.220
## - ptratio_and_lstat    1   54.942  92.942
## - ptratio             1   55.056  93.056
## - medv_and_tax         1   58.643  96.643
## - zn                  1   59.546  97.546
## - dis                 1   60.617  98.617
## - medv                1   61.474  99.474
## - age                 1   67.142 105.142
## - rad                 1   84.387 122.387
## - nox                 1  108.938 146.938
##
## Step:  AIC=87.61
## target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + medv + age_greater_than_85 + distance_band + indus_flag +
##     lstat_flag + medv_and_tax + high_nox + ptratio_and_lstat
##
##                       Df Deviance     AIC
## - indus                1   49.748  85.748
## - lstat_flag           1   49.753  85.753
## - distance_band        1   49.999  85.999
## - indus_flag           1   50.180  86.180
## - chas                 1   50.575  86.575
## - high_nox             1   50.832  86.832
## <none>                     49.615  87.615
## - age_greater_than_85  1   51.942  87.942
```

```
## - rm                   1   53.261  89.261
## - tax                  1   54.334  90.334
## - ptratio              1   55.056  91.056
## - ptratio_and_lstat    1   55.644  91.644
## - medv_and_tax         1   58.644  94.644
## - zn                   1   59.654  95.654
## - dis                  1   60.667  96.667
## - medv                 1   61.793  97.793
## - age                  1   67.169 103.169
## - rad                  1   85.370 121.370
## - nox                  1  109.012 145.012
##
## Step:  AIC=85.75
## target ~ zn + chas + nox + rm + age + dis + rad + tax + ptratio +
##     medv + age_greater_than_85 + distance_band + indus_flag +
##     lstat_flag + medv_and_tax + high_nox + ptratio_and_lstat
##
##                       Df Deviance     AIC
## - lstat_flag           1   49.904  83.904
## - distance_band        1   50.046  84.046
## - chas                 1   50.588  84.588
## <none>                     49.748  85.748
## - age_greater_than_85  1   52.163  86.163
## - high_nox             1   53.401  87.401
## - rm                   1   53.545  87.545
## - ptratio              1   55.730  89.730
## - ptratio_and_lstat    1   55.755  89.755
## - indus_flag           1   57.782  91.782
## - medv_and_tax         1   58.684  92.684
## - zn                   1   60.178  94.178
## - dis                  1   60.935  94.935
## - medv                 1   62.298  96.298
## - tax                  1   64.800  98.800
## - age                  1   67.563 101.563
## - nox                  1  119.197 153.197
## - rad                  1  129.099 163.099
##
## Step:  AIC=83.9
## target ~ zn + chas + nox + rm + age + dis + rad + tax + ptratio +
##     medv + age_greater_than_85 + distance_band + indus_flag +
##     medv_and_tax + high_nox + ptratio_and_lstat
##
##                       Df Deviance     AIC
## - distance_band        1   50.480  82.480
## - chas                 1   50.691  82.691
## <none>                     49.904  83.904
## - age_greater_than_85  1   52.685  84.685
## - rm                   1   53.592  85.592
## - high_nox             1   55.922  87.922
## - ptratio_and_lstat    1   56.197  88.197
## - ptratio              1   56.752  88.752
## - medv_and_tax         1   59.089  91.089
## - zn                   1   60.299  92.299
## - dis                  1   60.941  92.941
```

```
## - indus_flag           1   61.839  93.839
## - medv                 1   62.400  94.400
## - age                  1   67.644  99.644
## - tax                  1   69.082 101.082
## - nox                  1  119.709 151.709
## - rad                  1  134.775 166.775
##
## Step:  AIC=82.48
## target ~ zn + chas + nox + rm + age + dis + rad + tax + ptratio +
##     medv + age_greater_than_85 + indus_flag + medv_and_tax +
##     high_nox + ptratio_and_lstat
##
##                       Df Deviance     AIC
## - chas                 1   51.184  81.184
## <none>                     50.480  82.480
## - rm                   1   53.821  83.821
## - age_greater_than_85  1   53.896  83.896
## - high_nox             1   56.447  86.447
## - ptratio              1   56.776  86.776
## - ptratio_and_lstat    1   56.849  86.849
## - medv_and_tax         1   59.689  89.689
## - zn                   1   60.372  90.372
## - indus_flag           1   62.401  92.401
## - medv                 1   62.426  92.426
## - dis                  1   63.914  93.914
## - age                  1   68.869  98.869
## - tax                  1   69.696  99.696
## - nox                  1  120.323 150.323
## - rad                  1  134.935 164.935
##
## Step:  AIC=81.18
## target ~ zn + nox + rm + age + dis + rad + tax + ptratio + medv +
##     age_greater_than_85 + indus_flag + medv_and_tax + high_nox +
##     ptratio_and_lstat
##
##                       Df Deviance     AIC
## <none>                     51.184  81.184
## - rm                   1   54.429  82.429
## - age_greater_than_85  1   54.803  82.803
## - ptratio_and_lstat    1   57.103  85.103
## - ptratio              1   57.910  85.910
## - high_nox             1   58.072  86.072
## - medv_and_tax         1   60.594  88.594
## - zn                   1   61.684  89.684
## - medv                 1   62.731  90.731
## - indus_flag           1   62.964  90.964
## - dis                  1   64.034  92.034
## - age                  1   70.422  98.422
## - tax                  1   73.661 101.661
## - nox                  1  120.433 148.433
## - rad                  1  148.067 176.067
```

After seven iterations, **our AIC score dropped from 91.605 to 81.18** and our model was narrowed from 21 features to 14.

Model 4: AIC Optimized

As a next step, we prepare our data, fit our *AIC optimized* model and visit the corresponding summary statistics and confusion matrix:

```
##
## Call:
## glm(formula = target ~ zn + nox + rm + age + dis + rad + tax +
##     ptratio + medv + age_greater_than_85 + indus_flag + medv_and_tax +
##     high_nox + ptratio_and_lstat, family = binomial(link = "logit"),
##     data = train5)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -3.01009  -0.00008   0.00000   0.00000   2.18557
##
## Coefficients:
##                        Estimate  Std. Error z value  Pr(>|z|)
## (Intercept)          -134.112469   30.854663  -4.347 0.0000138 ***
## zn                     -0.496845    0.225368  -2.205  0.027483 *
## nox                   172.769790   39.508650   4.373 0.0000123 ***
## rm                     -2.571473    1.511765  -1.701  0.088948 .
## age                     0.200006    0.059067   3.386  0.000709 ***
## dis                     2.821253    0.960275   2.938  0.003304 **
## rad                     2.310756    0.540241   4.277 0.0000189 ***
## tax                    -0.033941    0.009833  -3.452  0.000557 ***
## ptratio                 1.050797    0.429904   2.444  0.014515 *
## medv                    0.620841    0.215941   2.875  0.004040 **
## age_greater_than_851   -2.300642    1.304731  -1.763  0.077849 .
## indus_flag1           -15.397754  110.321021  -0.140  0.888998
## medv_and_tax1          -7.667206    3.040030  -2.522  0.011666 *
## high_nox1              16.638722  110.359041   0.151  0.880158
## ptratio_and_lstat1      3.920215    1.847238   2.122  0.033821 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 609.933  on 439  degrees of freedom
## Residual deviance:  51.184  on 425  degrees of freedom
## AIC: 81.184
##
## Number of Fisher Scoring iterations: 12
```

We'd already touched on the fact that **our AIC score dropped from 91.605 to 81.18**, but it's also interesting to note that where our earlier model carried 12 features with high p-values, our AIC optimized model only carries 4 such features.

As for the impact of feature removal on model accuracy:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Risk Normal
##     Risk    213      4
##     Normal    5    218
```

```
##
##                Accuracy : 0.9795
##                  95% CI : (0.9615, 0.9906)
##     No Information Rate : 0.5045
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9591
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9771
##             Specificity : 0.9820
##          Pos Pred Value : 0.9816
##          Neg Pred Value : 0.9776
##              Prevalence : 0.4955
##          Detection Rate : 0.4841
##    Detection Prevalence : 0.4932
##       Balanced Accuracy : 0.9795
##
##        'Positive' Class : Risk
##
```

From the above output, we see that **our accuracy decreased from 98.41% to 97.95%**.

This is a red flag, but it's also worth noting that we've been working primarily with seen data and have yet to assess model performance on unseen data.

With this in mind we proceed to model selection with the two models that held the most predictive promise:

- **Model 3**: the outlier and feature engineered model and
- **Model 4**: the AIC optimized model

…………………………………………………………………..

Model Selection

We'll compare how each of these models perform on unseen data and determine whether **Model 3** or **Model 4** hold a greater potential to predict whether or not neighborhoods are at risk for high crime.

Outlier and Feature Engineered Model

We prepare our data, perform a train-test split (on the training dataset), fit our *outlier and feature engineered* model and visit the corresponding summary statistics and confusion matrix:

For now, we'll just note that our *outlier and feature engineered* model had an accuracy of 91.95%. While quite good, this is a far cry from the 98.41% accuracy the model had produced on training data earlier on. After visiting the AIC model's confusion matrix, we'll interpret classification statistics for each model side-by-side.

AIC Optimized Model

We prepare our data, perform a train-test split (on the training dataset), fit our *AIC optimized* model and visit the corresponding summary statistics and confusion matrix:

For now, we'll just note that our *AIC optimized* model had an accuracy of 97.7%. While a little worse than the 97.95% accuracy the model had produced on training data earlier, this is a stronger performance than than the *outlier and feature engineered* model.

Side-by-Side Comparison

We put our models side-by-side to interpret their common classification metrics and determine which has the greatest predictive accuracy:

	OF_Model	AIC_Model
Accuracy	0.9195402	0.9770115
Sensitivity	0.9318182	0.9444444
Specificity	0.9069767	1.0000000
Pos Pred Value	0.9111111	1.0000000
Neg Pred Value	0.9285714	0.9622642
Precision	0.9111111	1.0000000
Recall	0.9318182	0.9444444
F1	0.9213483	0.9714286
Prevalence	0.5057471	0.4137931
Detection Rate	0.4712644	0.3908046
Detection Prevalence	0.5172414	0.3908046
Balanced Accuracy	0.9193975	0.9722222

We consider the following classification metrics:

- **Accuracy** : $\frac{TP+TN}{TP+FP+TN+FN}$
- **Sensitivity (Recall)** : true positive rate. $\frac{TP}{TP+FN}$
- **Specificity**: true negative rate. $\frac{TN}{TN+FP}$
- **Pos Pred Value (Precision)** : probability that predicted positive is truly positive. $\frac{TP}{TP+FP}$
- **Neg Pred Value**: probability that predicted negative is truly negative. $\frac{TN}{(TN+FN)}$
- **F1**: harmonic mean of model's precision and recall. $\frac{2*(Precision*Recall)}{Precision+Recall}$
- **Prevalence**: truly positive observations as proportion of total number of observations. $\frac{TP+FN}{TP+FP+FN+TN}$
- **Detection Rate**: true positives detected as proportion of entire total population. $\frac{TP}{TP+FP+FN+TN}$
- **Detection Prevalence**: predicted positive events over total number of predictions. $\frac{TP+FP}{TP+FP+FN+TN}$
- **Balanced Accuracy**: measure of model's that is especially useful when classes are imbalanced. $\frac{Sensitivity+Specificity}{2}$

From the above table and classification metric definitions, we find that the **AIC Model** is more accurate, sensitive, specific, and precise. While the **OF Model** scores a higher `Detection Rate` and `Detection Prevalence`, the **AIC Model** scores higher across the board. It's consistently more accurate and capable of a higher rate of both positive and negative prediction.

Thus, the **AIC Model** performs better for predicting neighborhoods at risk for crime (`target` = 1) *and* not at risk for crime (`target` = 0) and is our choice model.

……………………………………………………………………..

Conclusion

Model Interpretation

With the **AIC Model** selected, we'll revisit the summary statistics and interpret the coefficients:

```
##
## Call:
## glm(formula = target ~ zn + nox + rm + age + dis + rad + tax +
##     ptratio + medv + age_greater_than_85 + indus_flag + medv_and_tax +
##     high_nox + ptratio_and_lstat, family = binomial(link = "logit"),
##     data = final_train2)
##
```

```
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.398   0.000   0.000   0.000   1.597
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -155.87849   42.36097  -3.680 0.000233 ***
## zn                     -0.59437    0.32066  -1.854 0.063803 .
## nox                   214.43749   58.56629   3.661 0.000251 ***
## rm                     -3.46333    1.76709  -1.960 0.050007 .
## age                     0.20120    0.06880   2.925 0.003449 **
## dis                     2.97319    1.10887   2.681 0.007334 **
## rad                     2.53413    0.72247   3.508 0.000452 ***
## tax                    -0.03238    0.01168  -2.773 0.005555 **
## ptratio                 1.06010    0.50908   2.082 0.037308 *
## medv                    0.74127    0.26236   2.825 0.004722 **
## age_greater_than_851   -1.73751    1.44255  -1.204 0.228410
## indus_flag1           -17.50999  214.91338  -0.081 0.935064
## medv_and_tax1          -9.81343    3.89538  -2.519 0.011761 *
## high_nox1              15.30479  214.91853   0.071 0.943229
## ptratio_and_lstat1      4.04164    2.06274   1.959 0.050071 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 489.019  on 352  degrees of freedom
## Residual deviance:  42.104  on 338  degrees of freedom
## AIC: 72.104
##
## Number of Fisher Scoring iterations: 13
```

For **continuous variables**, the interpretation is as follows:

- For every unit increase in `zn`, the log odds of the neighborhood being at risk for crime decrease by 0.594.
- For every unit increase in `nox`, the log odds of the neighborhood being at risk for crime increase by 214.437.
- For every unit increase in `rm`, the log odds of the neighborhood being at risk for crime decrease by -3.463.
- For every unit increase in `age`, the log odds of the neighborhood being at risk for crime increase by 0.201.
- For every unit increase in `dis`, the log odds of the neighborhood being at risk for crime increase by 2.973.
- For every unit increase in `rad`, the log odds of the neighborhood being at risk for crime increase by 2.534.
- For every unit increase in `tax`, the log odds of the neighborhood being at risk for crime decrease by -0.032.
- For every unit increase in `ptratio`, the log odds of the neighborhood being at risk for crime increase by 1.060.
- For every unit increase in `medv`, the log odds of the neighborhood being at risk for crime increase by 0.741.

For **engineered, categorical variables**, the interpretation is as follows:

- If `age_greater_than_85` is 1 (the unit is greater than 85 years old), the log odds of the neighborhood

being at risk for crime decrease by -1.737.
- If `indus_flag` is 1 (proportion of non business acres per town is greater than 15), the log odds of the neighborhood being at risk for crime decrease by -17.509.
- If `medv_and_tax` is 1 (median value of owner-occupied homes in \$1000's is < 17 and full-value property-tax rate per \$10,000 is > 350), the log odds of the neighborhood being at risk for crime decrease by -9.813.
- If `high_nox` is 1 (nitric oxides concentration (parts per 10 million) is > .6), the log odds of the neighborhood being at risk for crime increase by 15.305.
- If `ptratio_and_lstat` is 1 (the parent-teacher ratio is > 20 and > 15% of population are lower status), the log odds of the neighborhood being at risk for crime increase by 4.042.

From the above coefficient interpretations, we get an idea of the factors at play in predicting high crime neighborhoods (ie. pollution as a major predictor), but at this point we're yet to cast a prediction.

Prediction

As a next natural step, it's the moment we've all been waiting for …

We prepare our data, cast predictions using our AIC optimized model and then output corresponding index numbers with the predicted `target` value:

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  0  1  1  1  0  0  0  0  0  0  0  0  1  1  1  0  0  1  0  0  0  0  0  0  0  1
## 27 28 29 30 31 32 33 34 35 36 37 38 39 40
##  0  1  1  1  1  1  1  1  1  1  1  1  1  0
```

In Closing

In closing, here we present our findings for binary logistic regression modeling of the Boston Housing Dataset. Our objective was to predict whether a given neighborhood will be at risk for high crime levels. To approach this goal, we start with a full model that uses all feature variables in their native state as a benchmark. A model built with the raw data performed well with a predictive accuracy of ~92%. However, we were able to build upon this by iteratively improving our model by removing outliers (Model 2), engineering new features to capture the structure of existing feature variables (Model 3), and using AIC scores to optimize feature selection (Model 4). We select an AIC optimized model which improves our predictive accuracy to ~98%.

Appendix: `R` Statistical Code

```
library(knitr)
library(skimr)
library(visdat)
library(inspectdf)
library(corrplot)
library(scales)
library(tidyverse)
library(tidyr)
library(car)
library(dplyr)
library(kableExtra)
library(tufte)
library(MASS) #stepAIC
library(rsample)
library(BBmisc)
library(tidymodels)
```

```
options(scipen = 9)
set.seed(123)
```

DEPENDANCIES

```

#import train and test data sets
train <- readr::read_csv('https://raw.githubusercontent.com/dataconsumer101/data621/main/hw3/crime-trai
test <- readr::read_csv('https://raw.githubusercontent.com/dataconsumer101/data621/main/hw3/crime-evalu

#add target column to test set
test$target <- NA
```

IMPORTING DATA

DATA EXPLORATION & BASELINE MODEL

```
#visualize a summary of the train dataset
glimpse(train)
```

reformatting categorical features

```
#convert features to factor
train$chas <- as.factor(train$chas)
test$chas <- as.factor(test$chas)

train$target <- as.factor(train$target)
test$target <- as.factor(test$target)

#add dataset feature for future use while engineering the data
train$dataset <- 'train'
test$dataset <- 'test'
```

Visualize a summary of the categorical variables

```
#setup visualization dataset and combine datasets
data <- train %>% dplyr::select(-dataset)
final_df <- rbind(train, test)

#plot font size
plotfontsize <- 16

#revisit thorough summary statistics
#skimr::skim(train) #output to HTML not PDF
fig1 <- inspectdf::inspect_imb(data)
fig1 <- ggplot( data = fig1, aes( x = as.factor( col_name ),
                                  y = pcnt,
                                  fill = col_name ) ) +
    geom_bar( stat = "identity") +
    geom_text( aes( label = paste( round( pcnt,2 ), '%, value =', value ) ), size = 6 ) +
    theme( text = element_text(size=plotfontsize)) +
    ggtitle( 'Categorical Predictor Variables', subtitle = 'balance of binary values' ) +
    ylab( "% most common value") +
    xlab( "Variable")
fig1
```

Visualize distributions of the numeric feature variable

```
fig2 <- inspectdf::inspect_num(data) %>%
  show_plot()
fig2 + theme(text = element_text(size=plotfontsize))
fig2
```

Visualize relationships of numeric variables to the target variable

```
train_int_names <- train %>% select_if(is.numeric)
int_names <- names(train_int_names)

for (i in int_names) {
  assign(paste0("var_",i), ggplot(train, aes_string(x = train$target, y = i)) +
          geom_boxplot(color = 'steelblue',
                       outlier.color = 'firebrick',
                       outlier.alpha = 0.35) +
          #scale_y_continuous(labels = comma) +
          labs(title = paste0(i,' vs target'), y = i, x= 'target') +
          theme_minimal() +
          theme(
            plot.title = element_text(hjust = 0.45),
            panel.grid.major.y =  element_line(color = "grey", linetype = "dashed"),
            panel.grid.major.x = element_blank(),
            panel.grid.minor.y = element_blank(),
            panel.grid.minor.x = element_blank(),
            axis.ticks.x = element_line(color = "grey"),
            text = element_text(size=plotfontsize)
          ))
}
gridExtra::grid.arrange(var_age, var_dis, var_indus,var_lstat,
                        var_medv,var_nox,var_ptratio,var_rad,
                        var_rm, var_tax, var_zn, nrow=4)
```

Visualize paired plots of the numeric feature variables

```
pairs(train %>% select_if(is.numeric))
```

Visualize correlation matrix

```
numeric_values <- train %>% select_if(is.numeric)
train_cor <- cor(numeric_values)
corrplot.mixed(train_cor, tl.col = 'black', tl.pos = 'lt')
```

Model 1: Baseline model

```
#prepare input data
train2 <- train %>% dplyr::select(-`dataset`)
#fit "baseline" model
log_model <- glm(target ~ ., family = binomial(link = "logit"), data = train2)
#review summary statistics
summary(log_model)

#calculate deviance and p-value vs. null
with(log_model, null.deviance - deviance)
with(log_model, pchisq(null.deviance - deviance,
                       df.null - df.residual,
```

```
                         lower.tail = FALSE))

#prepare confusion matrix
log_preds <- predict(log_model, type = 'response')
train2$preds <- ifelse(log_preds > 0.5, 1,0)
train2$target <- factor(train2$target, levels = c(1,0), labels = c('Risk', 'Normal'))
train2$preds <- factor(train2$preds, levels = c(1,0), labels = c('Risk', 'Normal'))
caret::confusionMatrix(data = train2$preds, reference = train2$target)
```

use vif() to evaluate the multicolinearity of the feature variables

```
#verify vif
car::vif(log_model)
```

DATA PREPARATION & MODEL BUILDING

Using cooks distance to evaluate outliers

```
#use cooks distance to identify
cooks_distance <- cooks.distance(log_model)
influential <- as.numeric(names(cooks_distance)
                          [(cooks_distance > 4 * mean(cooks_distance, na.rm = TRUE))])
influential
#train[influential,] #verify outliers
#remove outliers
train3 <- train[-influential, ]
#train3
```

Remove outlier and refit binomial model

```
#prepare input data
train3 <- train3 %>% dplyr::select(-dataset)
#fit model
log_model2 <- glm(target ~ ., family = binomial(link = "logit"), data = train3)
#output summary statistics
summary(log_model2)
```

Model 2: Outlier Optimized Evaluate the impact of removing outliers on model accuracy

```
#prepare confusion matrix
log_preds2 <- predict(log_model2, type = 'response')
#assign predicted probabilities
train3$preds <- ifelse(log_preds2 > 0.5, 1,0)
#assign predictions based on probabilities with 0.5 threshold
train3$target <- factor(train3$target, levels = c(1,0),
                        labels = c('Risk', 'Normal')) #relabel target column
train3$preds <- factor(train3$preds, levels = c(1,0),
                       labels = c('Risk', 'Normal')) #relabel predictions column
caret::confusionMatrix(data = train3$preds, reference = train3$target)
#output confusion matrix
```

Prepare data for feature engineering (comnied train & test)

```
#removing outliers from training set
train <- train[-influential, ]
#combining datasets so we don't have to make features twice
final_df <- rbind(train, test)
```

```
#Creating new features:
final_df$rm <- round(final_df$rm,0) #adaptation of original feature --> round 0.5 up
final_df$age_greater_than_85 <- as.factor(ifelse(final_df$age >= 85, 1, 0))
final_df$distance_band <- as.factor(ifelse(final_df$dis < 4, 1, 0))
final_df$indus_flag <- as.factor(ifelse(final_df$indus > 15, 1, 0))
final_df$lstat_and_rad <- as.factor(ifelse(final_df$lstat > 20 & final_df$rad > 4, 1, 0))
final_df$lstat_flag <- as.factor(ifelse(final_df$lstat > 12, 1, 0))
final_df$medv_and_tax <- as.factor(ifelse(final_df$medv < 17 & final_df$tax > 350, 1, 0))
final_df$high_nox <- as.factor(ifelse(final_df$nox > .6, 1, 0))
final_df$ptratio_and_lstat <- as.factor(ifelse(final_df$ptratio > 20 & final_df$lstat > 15,1,0))
```

Model 3: Feature Engineered

```
#prepare input data: filter for training data and drop dataset feature from consideration
train4 <- final_df %>% filter(dataset == 'train') %>% dplyr::select(-dataset)
#fit model
log_model3 <- glm(target ~ ., family = binomial(link = "logit"), data = train4)
#output summary statistics
summary(log_model3)
```

Evaluate feature engineering for model accuracy

```
#prepare confusion matrix
log_preds3 <- predict(log_model3, type = 'response')
train4$preds <- ifelse(log_preds3 > 0.5, 1,0)
train4$target <- factor(train4$target, levels = c(1,0), labels = c('Risk', 'Normal'))
train4$preds <- factor(train4$preds, levels = c(1,0), labels = c('Risk', 'Normal'))
caret::confusionMatrix(data = train4$preds, reference = train4$target)
```

Engineer feature inclusion by AIC optimization

```
#apply stepAIC to optimize model
aic_opt_model <- stepAIC(log_model3)
```

Model 4: AIC Optimized prepare data and fit the AIC optimized model

```
train5 <- final_df %>% filter(dataset == 'train') %>% dplyr::select(-dataset)
#fit AIC optimized model
final_model <- glm(target ~ zn + nox + rm + age + dis + rad + tax +
    ptratio + medv + age_greater_than_85 + indus_flag + medv_and_tax +
    high_nox + ptratio_and_lstat,
           family = binomial(link = "logit"), data = train5)
#output summary statistics
summary(final_model)
```

Evaluate model accuracy

```
#prepare confusion matrix
final_preds <- predict(final_model, type = 'response')
train5$preds <- ifelse(final_preds > 0.5, 1,0)
train5$target <- factor(train5$target, levels = c(1,0), labels = c('Risk', 'Normal'))
train5$preds <- factor(train5$preds, levels = c(1,0), labels = c('Risk', 'Normal'))
caret::confusionMatrix(data = train5$preds, reference = train5$target)
```

MODEL SELECTION

Outlier and Feature Engineered Models Validation

```r
#set seed
set.seed(123) #for reproducibility
#prepare input data
train6 <- final_df %>% filter(dataset == 'train') %>% dplyr::select(-dataset)
#perform train-test split
train_split <- initial_split(train6, prop = 0.80)
final_train <- training(train_split)
final_test <- testing(train_split)
results <- final_test$target
final_test$target <- NA
#fit model
final_log_model <- glm(target ~ ., family = binomial(link = "logit"), data = final_train)
#prepare confusion matrix
final_log_preds <- predict(final_log_model, type = 'response', newdata = final_test)
final_test$preds <- ifelse(final_log_preds > 0.5, 1,0)
final_test$target <- factor(results, levels = c(1,0), labels = c('Risk', 'Normal'))
final_test$preds <- factor(final_test$preds, levels = c(1,0), labels = c('Risk', 'Normal'))
#output to kable table
OF_Model <- caret::confusionMatrix(data = final_test$preds,
                                   reference = final_test$target)$byClass
AccuracyOF <- caret::confusionMatrix(final_test$preds, final_test$target)$overall['Accuracy']
OF_Model <- data.frame(OF_Model)
OF_Model <- rbind("Accuracy" = AccuracyOF, OF_Model)
```

AIC Optimized Model Validation

```r
#set seed
set.seed(124) #for reproducibility
#prepare input data
train7 <- final_df %>% filter(dataset == 'train') %>% dplyr::select(-dataset)
#perform train-test split
train_split2 <- initial_split(train7, prop = 0.80)
final_train2 <- training(train_split2)
final_test2 <- testing(train_split2)
results2 <- final_test2$target
final_test2$target <- NA
#fit model
final_log_model2 <- glm(target ~ zn + nox + rm + age + dis + rad + tax +
    ptratio + medv + age_greater_than_85 + indus_flag + medv_and_tax +
    high_nox + ptratio_and_lstat, family = binomial(link = "logit"), data = final_train2)
#prepare confusion matrix
final_log_preds2 <- predict(final_log_model2, type = 'response', newdata = final_test2)
final_test2$preds <- ifelse(final_log_preds2 > 0.5, 1,0)
final_test2$target <- factor(results2, levels = c(1,0), labels = c('Risk', 'Normal'))
final_test2$preds <- factor(final_test2$preds, levels = c(1,0), labels = c('Risk', 'Normal'))
#output to kable table
AIC_Model <- caret::confusionMatrix(data = final_test2$preds,
                                    reference = final_test2$target)$byClass
AccuracyAIC <- caret::confusionMatrix(final_test2$preds, final_test2$target)$overall['Accuracy']
AIC_Model <- data.frame(AIC_Model)
AIC_Model <- rbind("Accuracy" = AccuracyAIC, AIC_Model)
```

Generate table fro Side-by-side model comparison

```r
#accuracy and classification statistics as a kable table
tabularview <- data.frame(OF_Model, AIC_Model)
tabularview %>%  kableExtra::kbl() %>%
    kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"),
                  latex_options="hold_position")
```

CONCLUSIONS

Visualize summary for the selected model, Model 4: AIC Model

```r
summary(final_log_model2)
```

Use model to predict the target variable for the test dataset

```r
#prepare dataset
final_test <- final_df %>% filter(dataset == 'test') %>% dplyr::select(-dataset,-target)
#predict using selected model
homework_predictions <- predict(final_log_model2, type = 'response', newdata = final_test)
final_test$predictions <- ifelse(homework_predictions > 0.5, 1,0)
#output predictions
#final_test
final_test$predictions
```