Business Analytics and Data Mining

DATA621 Homework 03

William Outcault, Kevin Potter, Mengqin Cai, Philip Tanofsky, Robert Welk, Zhi Ying Chen

31 October 2020

The assignment attempts to predict whether a given neighborhood near a big city is at risk for a high crime rate. The project first provides an initial data exploration of the 12 predictor variables and the one categorical target variable. The data exploration step provides insight into which variables provide a higher level of correlation with the target variable. Due to the binary, categorical nature of the target variable, a binary logistic regression model is required to make proper predictions. Several binary logistic regression models are constructed and evaluated to determine the best performing model on the training data. Once selected, the best-performing binary logistic regression model is used to predict the classification and probabilities on the evaluation data set.

Data exploration

The data fields represent crime information for various neighborhoods in a major city. Each record has a response variable indicating whether or not the crime rate is above the median crime rate (1) or below the median crime rate (0).

The following exploratory data methods present a picture of the data to capture the distribution of the data and potential correlation with the target variable. The techniques used explore a summary of the variables,the distribution of each predictor variable against the target variable, density plot of each predictor variable against the target variable, along with a correlation plot across all the features.

```
str(train)
```

```
## tibble [466 x 13] (S3: tbl_df/tbl/data.frame)
##  $ zn     : num [1:466] 0 0 0 30 0 0 0 0 0 80 ...
##  $ indus  : num [1:466] 19.58 19.58 18.1 4.93 2.46 ...
##  $ chas   : int [1:466] 0 1 0 0 0 0 0 0 0 0 ...
##  $ nox    : num [1:466] 0.605 0.871 0.74 0.428 0.488 0.52 0.693 0.693 0.515 0.392 ...
##  $ rm     : num [1:466] 7.93 5.4 6.49 6.39 7.16 ...
##  $ age    : num [1:466] 96.2 100 100 7.8 92.2 71.3 100 100 38.1 19.1 ...
##  $ dis    : num [1:466] 2.05 1.32 1.98 7.04 2.7 ...
##  $ rad    : int [1:466] 5 5 24 6 3 5 24 24 5 1 ...
##  $ tax    : int [1:466] 403 403 666 300 193 384 666 666 224 315 ...
##  $ ptratio: num [1:466] 14.7 14.7 20.2 16.6 17.8 20.9 20.2 20.2 20.2 16.4 ...
##  $ lstat  : num [1:466] 3.7 26.82 18.85 5.19 4.82 ...
##  $ medv   : num [1:466] 50 13.4 15.4 23.7 37.9 26.5 5 7 22.2 20.9 ...
##  $ target : int [1:466] 1 1 1 0 0 0 1 1 0 0 ...
```

The structure of the training data indicates 466 records with 13 variables, 12 predictor variables and one target variable (target).

```
summary(train)
```

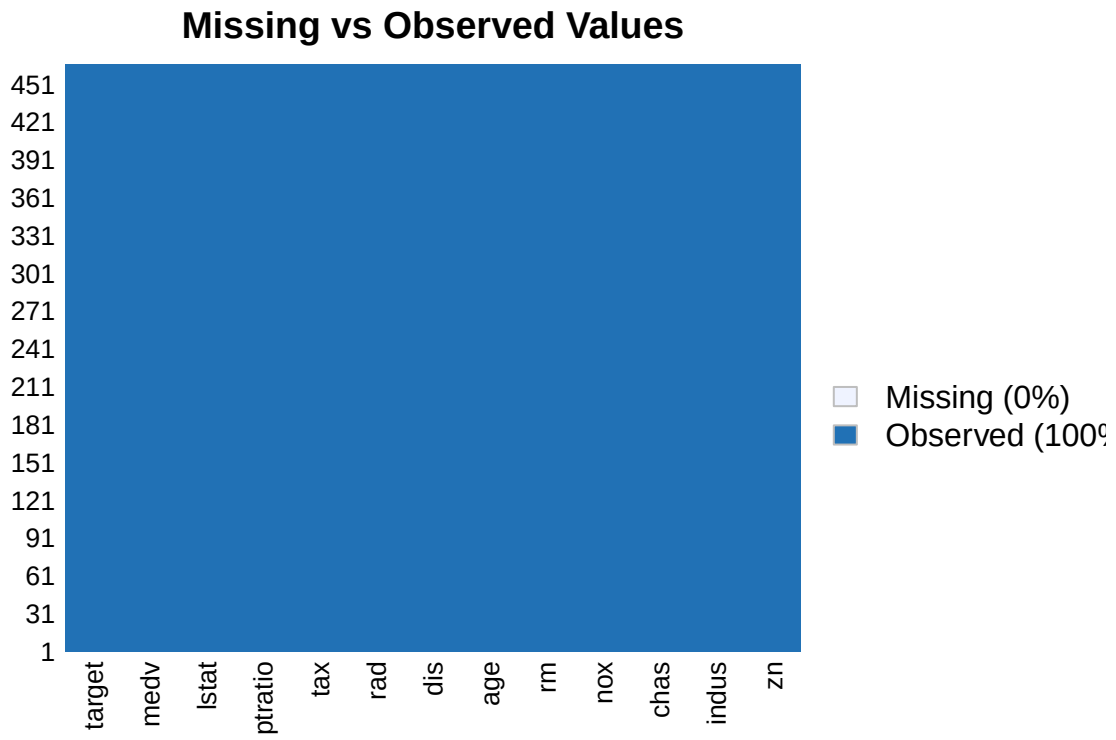
```
##        zn             indus             chas              nox        
##  Min.   :  0.00   Min.   : 0.460   Min.   :0.00000   Min.   :0.3890  
##  1st Qu.:  0.00   1st Qu.: 5.145   1st Qu.:0.00000   1st Qu.:0.4480  
##  Median :  0.00   Median : 9.690   Median :0.00000   Median :0.5380  
##  Mean   : 11.58   Mean   :11.105   Mean   :0.07082   Mean   :0.5543  
##  3rd Qu.: 16.25   3rd Qu.:18.100   3rd Qu.:0.00000   3rd Qu.:0.6240  
##  Max.   :100.00   Max.   :27.740   Max.   :1.00000   Max.   :0.8710  
##        rm              age              dis              rad        
##  Min.   :3.863   Min.   :  2.90   Min.   : 1.130   Min.   : 1.00  
##  1st Qu.:5.887   1st Qu.: 43.88   1st Qu.: 2.101   1st Qu.: 4.00  
##  Median :6.210   Median : 77.15   Median : 3.191   Median : 5.00  
##  Mean   :6.291   Mean   : 68.37   Mean   : 3.796   Mean   : 9.53  
##  3rd Qu.:6.630   3rd Qu.: 94.10   3rd Qu.: 5.215   3rd Qu.:24.00  
##  Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.00  
##       tax           ptratio          lstat             medv      
##  Min.   :187.0   Min.   :12.6   Min.   : 1.730   Min.   : 5.00  
##  1st Qu.:281.0   1st Qu.:16.9   1st Qu.: 7.043   1st Qu.:17.02  
##  Median :334.5   Median :18.9   Median :11.350   Median :21.20  
##  Mean   :409.5   Mean   :18.4   Mean   :12.631   Mean   :22.59  
##  3rd Qu.:666.0   3rd Qu.:20.2   3rd Qu.:16.930   3rd Qu.:25.00  
##  Max.   :711.0   Max.   :22.0   Max.   :37.970   Max.   :50.00  
##      target      
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.4914  
##  3rd Qu.:1.0000  
##  Max.   :1.0000
```

The summary table of the training data shows there are no missing values in the dataset along with the output below from the `missmap` function from the `Amelia` library.

Notes on the data:

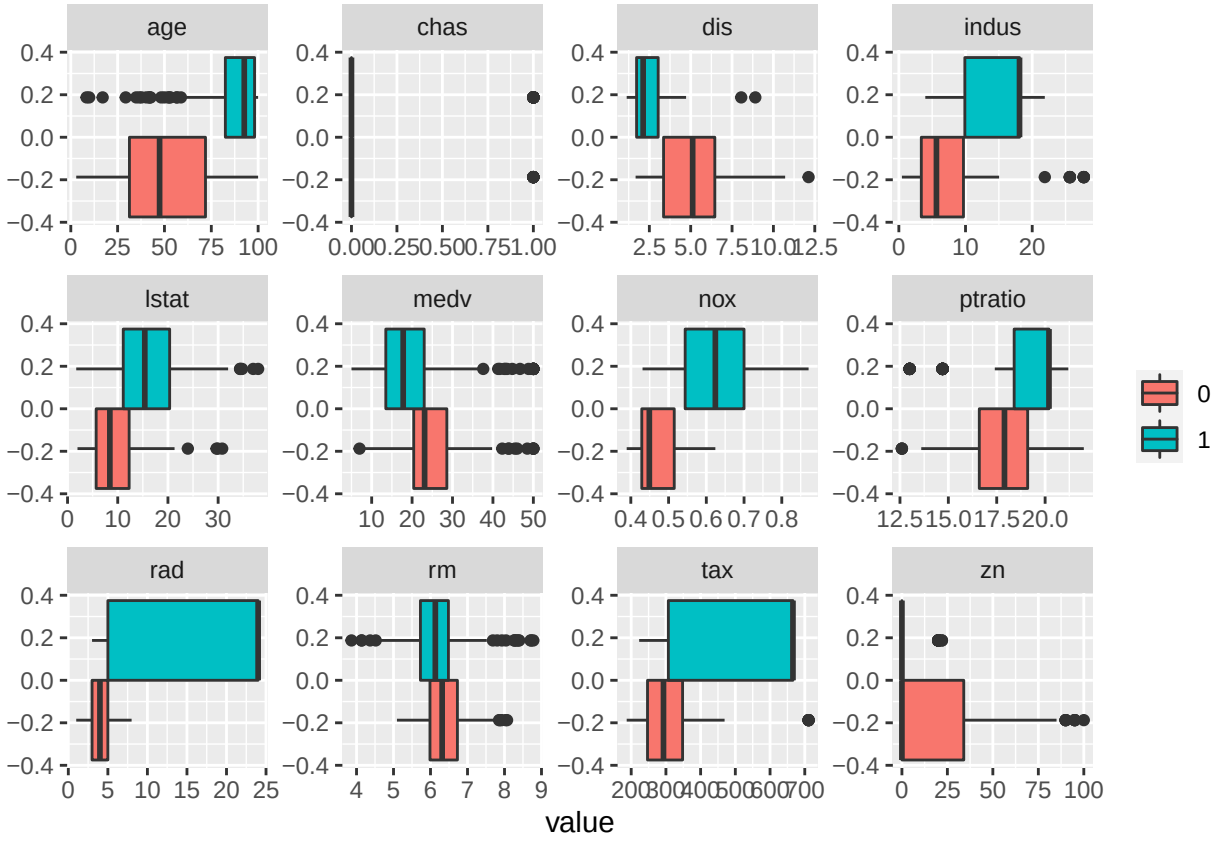
- 9 predictor variables defined as continuous numbers
- 3 predictor variables defined as integers
- chas is a dummy variable, containing the just the values 0 and 1
- rad is defined as an index, using integer values to define a category
- zn is a proportion and defined as an integer between 0 and 100
- Based on summary statistics, predictor variables appear valid given the range, median and mean.

```
missmap(train, main = "Missing vs Observed Values")
```



Boxplots

The dodged boxplot of each variable against the target variable highlights differences between target boxes which could mean the variable is useful for prediction. A dodged boxplot without overlapping boxes likely indicates a correlation in the value of the predictor variable to the target classes.



Boxplots with distinct difference:

- age
- dis
- indus
- nox
- rad

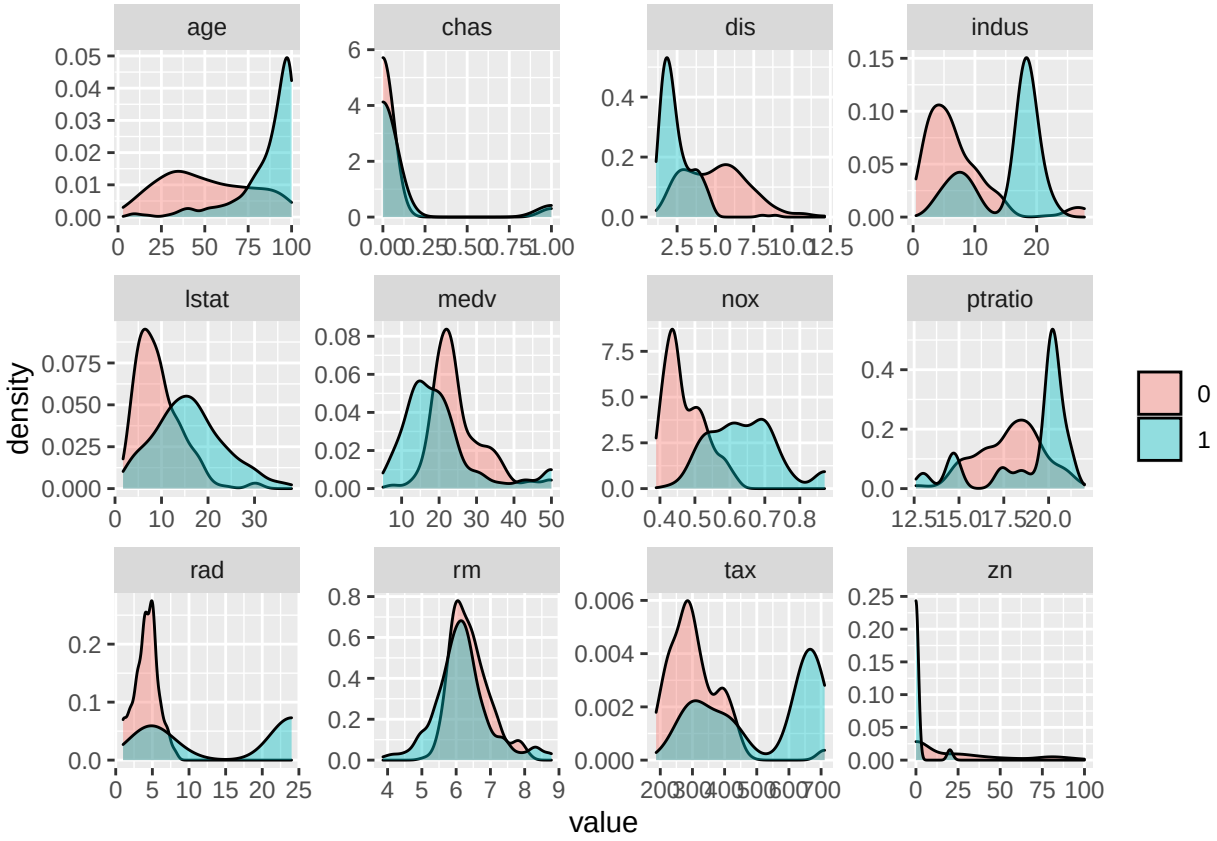
Boxplots without distinct difference:

- chas
- lstat
- medv
- ptratio
- rm
- tax
- zn

The above plots indicate age, dis, indus, nox and rad variables are likely to be more valuable to the model.

Density plots

Similar to the boxplots, the density plots are another tool to identify which predictor variables likely have a strong correlation with the target variable.



Density plots with clear distinction:

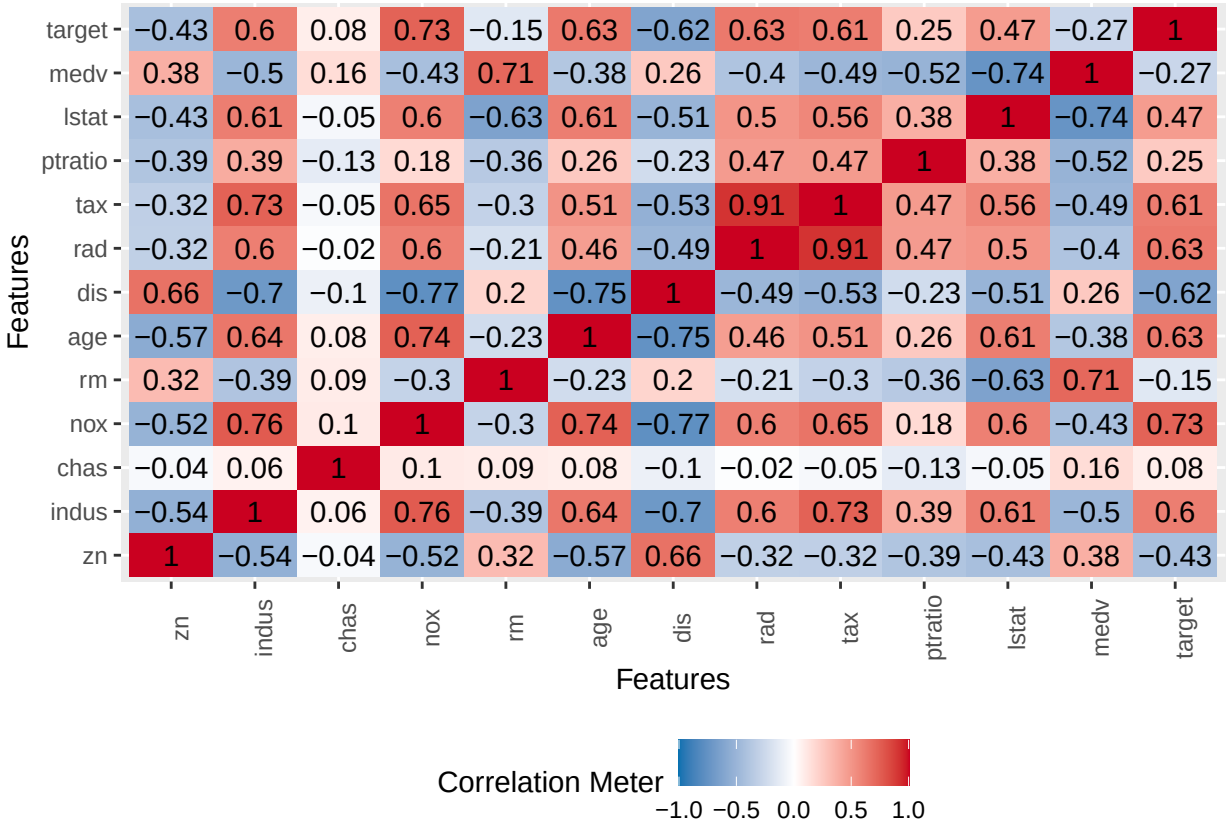
- age
- indus
- nox
- ptratio
- rad
- tax

Density plots without clear distinction:

- chas
- dis
- lstat
- medv
- rm
- zn

The above density plots indicate age, indus, nox, ptratio, rad and tax variables are likely to be more valuable to the model.

Correlation plot



Variables highly correlated with the target (0.6 or greater):

- indus
- nox
- age
- rad
- tax

Predictor variable combinations with high correlation (0.7 or greater):

- indus and nox
- indus and tax
- nox and age
- rm and medv

Predictor variable combinations with low correlation (-.7 or worse):

- indus and dis
- nox and dis
- age and dis
- lstat and medv

Based on the high correlation values with the target variable, the predictor variables of indus, nox, age, rad and tax are expected to be valuable to the model.

Data Preparation

Without any missing values in the predictor variables, no obvious data transformations are present. After performing additional research, the team concludes the logistic regression model contains no assumption of linearity. The model is based on data from the real world, and thus the transformation or cleaning of predictor data does expect to be a better representation of the real world data.

The team did attempt log transformations on several of the variables but witnesses no improvement to the model. The team has chosen to not prepare or transform the provided data in any meaningful manner.

For evaluating the training model, the training data set is split into train and test sets. The training set will contain 70 percent of the initial 466 records. The test set will be used to evaluate model for accuracy, sensitivity, specificity, AUC, and F1 score.

```
set.seed(1010)
trainIndex <- createDataPartition(train$target, p = .70,
                                  list = FALSE,
                                  times = 1)
train <- train[trainIndex,]
test <- train[-trainIndex,]
```

Build Models

The team created many binary logistic regression models, using both the `logit` and `probit` link functions, along with initial forays into data transformation. For each of the three models presented, the same 70 percent of the training data is used to evaluate the model and measure the predictions on the remaining 30 percent of the training data. The following sections explore the team's three primary methods for creating a model.

Raw Model

The raw model simply uses all the predictor variables in order to create a baseline for evaluation. The raw model uses the `glm` function to create the generalized linear model based on the `binomial` family and the link function `logit`. The same model is also created with the `probit` link function for initial comparison.

```
glm.full <- glm(target ~., data = train , family = "binomial" (link="logit"))
glm.full.probit <- glm(target ~., data = train , family = "binomial" (link="probit"))
summary(glm.full)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.95847  -0.09514  -0.00002   0.00079   3.01125
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -53.641514   9.582716  -5.598 2.17e-08 ***
## zn           -0.166833   0.066695  -2.501 0.012369 *
## indus        -0.088731   0.064186  -1.382 0.166852
## chas          0.252605   0.929337   0.272 0.785766
## nox          61.099868  11.475150   5.325 1.01e-07 ***
## rm           -0.337859   0.934782  -0.361 0.717777
## age           0.050229   0.017801   2.822 0.004778 **
## dis           1.372640   0.362765   3.784 0.000154 ***
## rad           0.762165   0.220322   3.459 0.000542 ***
## tax          -0.006152   0.003491  -1.762 0.078046 .
## ptratio       0.279135   0.165403   1.688 0.091487 .
## lstat         0.172248   0.069843   2.466 0.013655 *
## medv          0.285109   0.102456   2.783 0.005390 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 453.32  on 326  degrees of freedom
## Residual deviance: 119.55  on 314  degrees of freedom
## AIC: 145.55
##
## Number of Fisher Scoring iterations: 9
```

The output represents the raw model based on the `logit` link function. The resulting AIC for the raw model is 145.5463275. The AIC for the raw model using the `probit` link function is 146.655523.

The AIC (Akaike's Information Criteria) statistic is used to compare different models to determine the best fit for the data. The AIC is based on the count of independent variables as input into the model in addition to the how well the model reproduces the data. The purpose of the AIC best-fit model is to explain the greatest amount of variation with the fewest number of independent predictor variables.

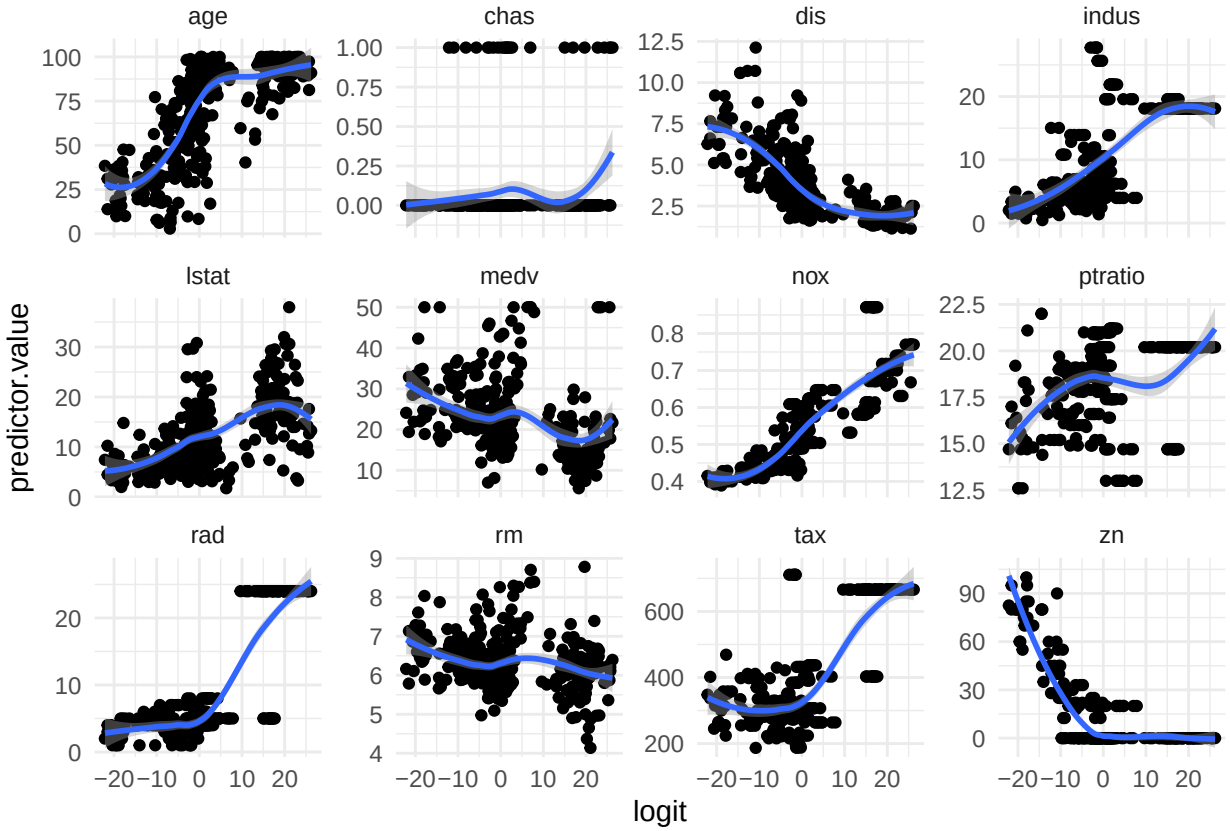
Simply put, the binary logistic regression model with the lower AIC is considered better. The raw model based on the `logit` link function performs slightly better.

Confusion Matrix

The following confusion matrix indicates the correct prediction of 84 records and the incorrect prediction of 10 records for the 30 percent of the training data set not used to build the model.

```
##           Reference
## Prediction  0  1
##          0 45  6
##          1  4 39
```

Before attempting to improve the raw model, the assumption of linearity is analyzed with the predictor variables from the first model.



The plots show relative linearity for variables:

- age
- indus
- nox
- rad
- tax

These results follow in line with the predictor variables with high correlations to the target variable found in the data exploration step.

These findings provide a guide for assessing the following models based on the reduction of predictor variables.

Stepwise Model

This model uses the raw model created above with the addition of the `stepAIC` function from the `MASS` package. `stepAIC` is a common package used to help with feature selection. This version of the model uses this package with no additional constraints to train and evaluate model performance.

```
glm.stepwise <- glm(target ~., data=train, family = "binomial"(link="logit")) %>%
  stepAIC(trace=FALSE)
summary(glm.stepwise)
```

```
##
## Call:
## glm(formula = target ~ zn + nox + age + dis + rad + tax + ptratio +
##     lstat + medv, family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.91478  -0.11523  -0.00002   0.00086   3.07021
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -50.526394   9.080721  -5.564 2.63e-08 ***
## zn           -0.169447   0.060627  -2.795 0.005191 **
## nox          52.962874   9.204715   5.754 8.72e-09 ***
## age           0.047734   0.015197   3.141 0.001684 **
## dis           1.306938   0.342028   3.821 0.000133 ***
## rad           0.865326   0.204170   4.238 2.25e-05 ***
## tax          -0.008003   0.003055  -2.619 0.008808 **
## ptratio       0.253776   0.150462   1.687 0.091672 .
## lstat         0.164516   0.062774   2.621 0.008774 **
## medv          0.256901   0.072166   3.560 0.000371 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 453.32  on 326  degrees of freedom
## Residual deviance: 121.77  on 317  degrees of freedom
## AIC: 141.77
##
## Number of Fisher Scoring iterations: 9
```

The stepwise model produces an AIC of 141.765417.

Confusion Matrix

The following confusion matrix indicates the correct prediction of 85 records and the incorrect prediction of 9 records for the 30 percent of the training data set not used to build the model.

```
##           Reference
## Prediction  0  1
##          0 45  5
##          1  4 40
```

Manual Backwards Step Model

This model uses the a manual backwards stepwise approach to remove the least valuable predictor variables one at a time until the model reaches a peak AIC value. The model completes after the removal of predictor variables, chas, rm, indus, and lstat. The model, performed independently of the above `stepAIC` model, excludes the same variables as the above model along with the `lstat` variable.

```
glm.back<-glm(target~.-chas, data=train,family=binomial)
glm.back<-update(glm.back,.~.-rm, data=train,family=binomial)
glm.back<-update(glm.back,.~.-indus, data=train,family=binomial)
glm.back<-update(glm.back,.~.-lstat, data=train,family=binomial)
summary(glm.back)
```

```
##
## Call:
## glm(formula = target ~ zn + nox + age + dis + rad + tax + ptratio +
##     medv, family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9250  -0.1672  -0.0001   0.0015   3.3500
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -43.385213   8.152307  -5.322 1.03e-07 ***
## zn           -0.144581   0.053551  -2.700 0.006937 **
## nox          48.808894   8.734689   5.588 2.30e-08 ***
## age           0.051558   0.015047   3.426 0.000612 ***
## dis           1.133801   0.325547   3.483 0.000496 ***
## rad           0.806492   0.188230   4.285 1.83e-05 ***
## tax          -0.007218   0.003002  -2.405 0.016191 *
## ptratio       0.227115   0.149937   1.515 0.129839
## medv          0.163989   0.055223   2.970 0.002982 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 453.32  on 326  degrees of freedom
## Residual deviance: 128.67  on 318  degrees of freedom
## AIC: 146.67
##
## Number of Fisher Scoring iterations: 9
```

The manual backwards stepwise model produces an AIC of 146.6671316.

Confusion Matrix

The following confusion matrix indicates the correct prediction of 83 records and the incorrect prediction of 11 records for the 30 percent of the training data set not used to build the model.

```
##            Reference
## Prediction  0  1
##          0 45  7
##          1  4 38
```

Select Model

All the models will be compared in order to select the model with the best fit in order to produce the most accurate results. The metrics we will be focused on are accuracy, AIC, and AUC (area under the curve). The models compared were the original raw model which included all variables. Next was a stepwise model which minimizes AIC in order to determine the variables which are necessary to include. The last model was a manual backwards stepwise model with only one variable different than the stepwise model.

	Full Model	AIC Stepwise	Manual Backwards
Accuracy	0.8936170	0.9042553	0.8829787
Classification Error Rate	0.1063830	0.0957447	0.1170213
Sensitivity	0.8666667	0.8888889	0.8444444
Specificity	0.9183673	0.9183673	0.9183673
Precision	0.9069767	0.9090909	0.9047619
Recall	0.8666667	0.8888889	0.8444444
F1	0.8863636	0.8988764	0.8735632
R Squared	0.7362843	0.7313891	0.7161641
AIC	145.5463275	141.7654170	146.6671316
Deviance	119.5463275	121.7654170	128.6671316
AUC	0.9623583	0.9600907	0.9541950

- Best AIC: AIC Stepwise
- Best Accuracy: AIC Stepwise
- Best AUC: Full Model

The selected model is the AIC stepwise model with the highest accuracy and the lowest AIC. As the results show, the three models perform well in comparison.

Deviance Residuals

To further assess the models, the deviance residuals are compared. Based on the distribution, the better model will produce deviance residuals centered at zero and more symmetrical.

Raw Model

```
summary(glm.full$residuals)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -6.806  -1.005  -1.000   0.328   1.000  93.113
```

Stepwise Model

```
summary(glm.stepwise$residuals)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -6.254  -1.007  -1.000   0.381   1.000 111.398
```

Manual Backwards Stepwise Model

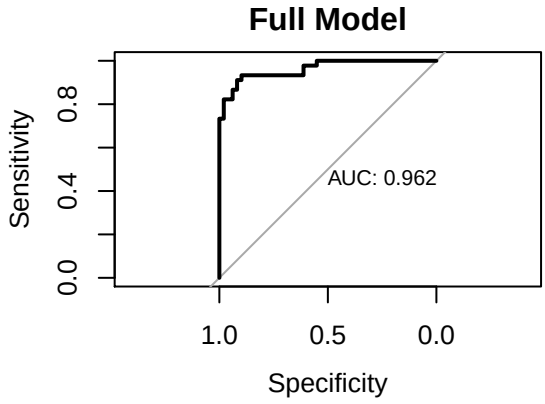
```
summary(glm.back$residuals)
```

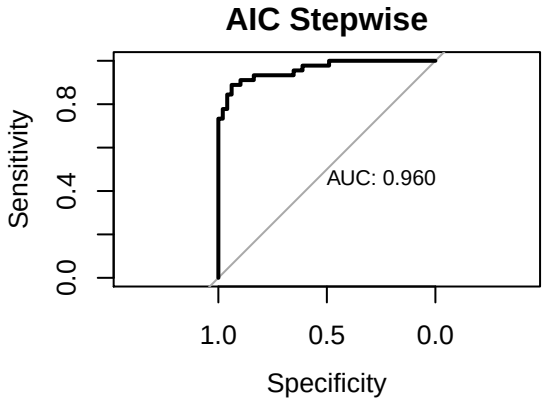
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -6.377  -1.014  -1.000   1.040   1.000 273.445
```

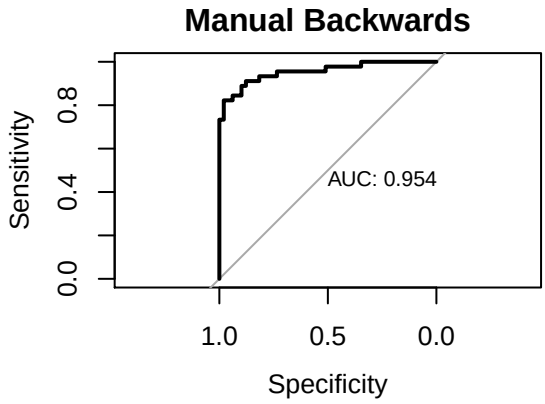
The raw model and stepwise model both produce results centered around zero.

ROC Plot

The ROC plots for each of the models indicates very similar performance with the best AUC value attained by the AIC Stepwise model as noted above.

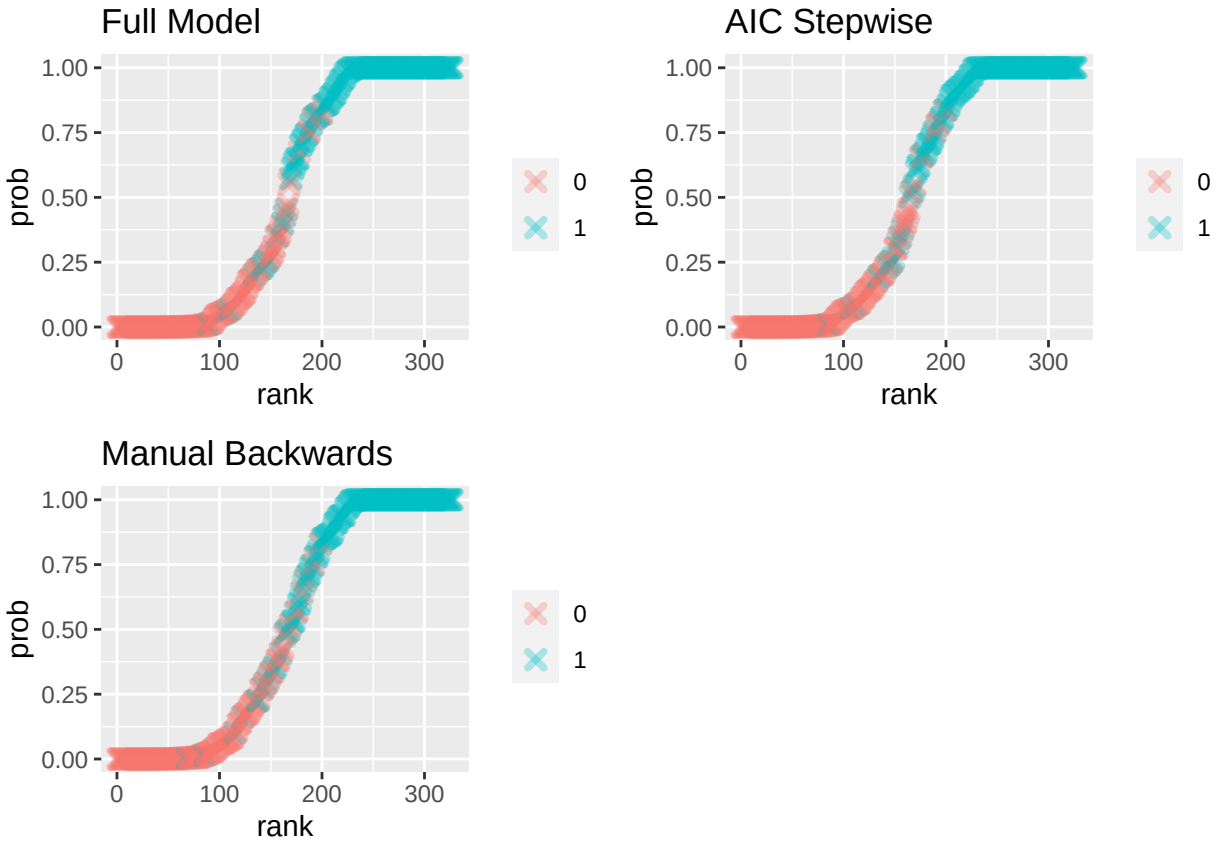






Probability Plot by Target Class

The plots of the probability for each targets from the initial training data set show very similar performance between the three chosen models. Each model identifies the target classification value predictably above and below the 0.5 threshold with only a few discrepancies.



Further Model Comparisons

As the raw model and stepwise model both performed well, further comparisons are made to tease out the difference in the models.

Anova

The `anova` function shows the deviance table for the generalized linear model fits for the raw model and the stepwise model. The table indicates a degrees of freedom of three less in the stepwise model along with a smaller deviance.

```
anova(glm.full, glm.stepwise, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + lstat + medv
## Model 2: target ~ zn + nox + age + dis + rad + tax + ptratio + lstat +
##     medv
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       314     119.55
## 2       317     121.77 -3  -2.2191   0.5282
```

Likelihood Ratio Test

The `lrtest` function evaluates the likelihood ratio test for the generalized linear model fits for the raw model and the stepwise model. The results show the stepwise model would not pass the p-value significance test of 0.05, which shows the two models are similar in evaluation.

```
lrtest(glm.full, glm.stepwise)
```

```
## Likelihood ratio test
##
## Model 1: target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + lstat + medv
## Model 2: target ~ zn + nox + age + dis + rad + tax + ptratio + lstat +
##     medv
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1  13 -59.773
## 2  10 -60.883 -3 2.2191     0.5282
```

Variable Importance

The `varImp` function calculates the variable importance for each predictor variable in the stepwise model.

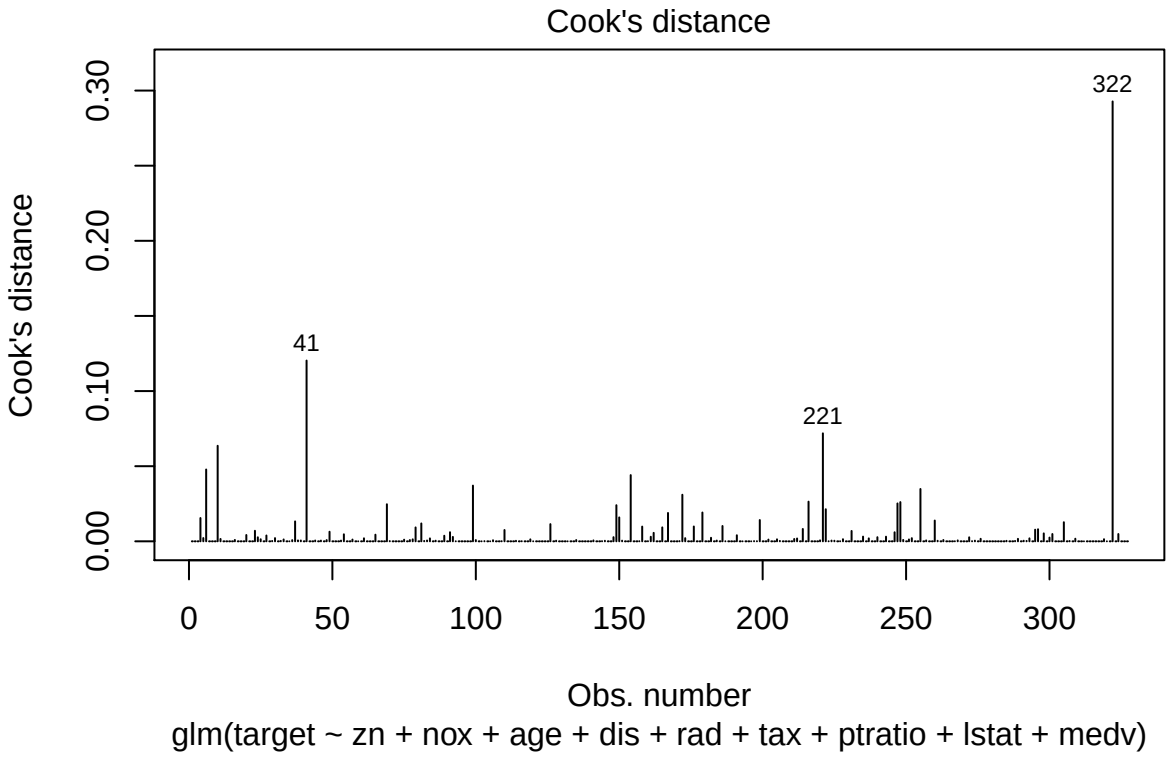
```
varImp(mod_fit)
```

```
## glm variable importance
##
##         Overall
## nox      100.00
## rad       62.74
## dis       52.48
## medv      46.06
## age       35.76
## zn        27.25
## lstat     22.97
## tax       22.93
## ptratio    0.00
```

Apparently, `ptratio` provides no importance to the stepwise model.

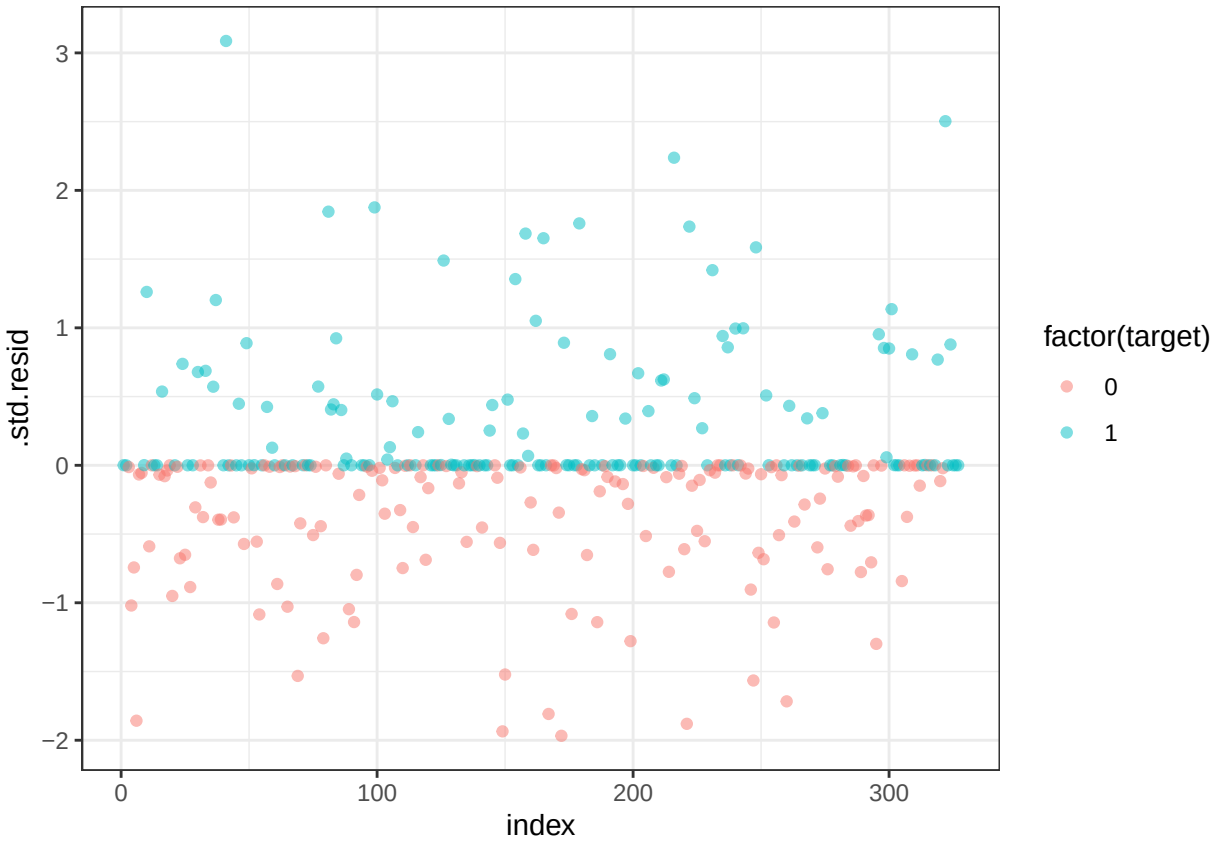
Checking Outliers

Not all outliers are influential observations. To check whether the data contains potential influential observations, the standardized residual error can be inspected. Data points with an absolute standardized residuals above 3 represent possible outliers and may deserve closer attention.



The following R code computes the standardized residuals `std.resid` and the Cook's distance using the R function `augment` from the `broom` package.

```
model.data <- augment(glm.stepwise) %>%
  mutate(index = 1:n())
ggplot(model.data, aes(index, .std.resid)) +
  geom_point(aes(color = factor(target)), alpha = .5) +
  theme_bw()
```



Checking for Collinearity

As a rule of thumb, a VIF value that exceeds greatly exceeds 5 indicates a problematic amount of collinearity. In our example of the stepwise model, there is no collinearity: all variables have a value of VIF below or just above 5.

```
##       zn      nox      age      dis      rad      tax  ptratio    lstat
## 2.615023 3.956144 2.099795 5.673126 1.605725 1.823703 1.850848 2.309572
##     medv
## 4.668272
```

Write Predictions

```
final.preds.probs <- predict(glm.stepwise,type="response", newdata=evaluation)
final.preds <- ifelse(final.preds.probs > 0.5, '1', '0')

final.preds <- cbind(target=final.preds, target_probability=final.preds.probs, evaluation)
write.csv(final.preds, "crime-evaluation-data-final.csv")
```

Output available in file `crime-evaluation-data-final.csv`.

Appendix

R statistical programming code

```
train <- read.csv(
  "https://raw.githubusercontent.com/Zchen116/data-621/master/crime-training-data_modified.csv") %>%
  as_tibble()

evaluation <- read.csv(
  "https://raw.githubusercontent.com/Zchen116/data-621/master/crime-evaluation-data_modified.csv") %>%
  as_tibble()
```

Data Exploration

```
train %>%
  gather("attribute", "value", 1:12) %>%
  ggplot(aes(x=value, fill=factor(target))) +
    geom_boxplot(position = 'dodge') +
    facet_wrap(~attribute, scales="free") +
    theme(legend.title=element_blank())
```

```
# density plots
train %>%
  gather("attribute", "value", 1:12) %>%
  ggplot(aes(x=value, fill=factor(target)))+
    geom_density(position = 'dodge', alpha=0.4)+
    facet_wrap(~attribute, scales="free")  +
    theme(legend.title=element_blank())
```

```
# corr matrix
plot_correlation(na.omit(train), maxcat = 5L)
```

Build Models

```
# Calculate McFaddens Psuedo R^2
ll.null <- glm.full$null.deviance/-2
ll.proposed <- glm.full$deviance/-2
R2_full <-(ll.null - ll.proposed)/ll.null
```

```
# plot the model
predictions.full <- data.frame(prob=glm.full$fitted.values, target=train$target) %>% arrange(prob)
predictions.full$rank <- 1:nrow(predictions.full)

## marginal effects for coefficients - now the coefficients can be interpreted
logit_scalar <- mean(dnorm(predict(glm.full, type="link")))
marginals.full <- logit_scalar * coef(glm.full)

## use the test data set to make predicts and calculate metrics from the confusion matrix
glm_full.probs <- predict(glm.full,type="response", newdata=test)
```

```
glm_predict.full <- ifelse(glm_full.probs > 0.5, '1','0')
attach(test)
```

```
# now can use the caret function
cm.full <- confusionMatrix(factor(glm_predict.full), factor(test$target), positive='1')
cm.full$table

# ROC and AUC
par(pty="s")
roc.full <- roc(test$target, glm_full.probs)
```

```
probabilities <- predict(glm.full, type='response')
predictors <- colnames(train)
vars <- names(glm.full$coefficients)[-1]

# Bind the logit and tidying the data for plot
train %>%
  dplyr::select(vars) %>%
  mutate(logit = log(probabilities/(1-probabilities))) %>%
  gather(key = "predictors", value = "predictor.value", -logit) %>%
  ggplot(aes(x=logit,y=predictor.value)) +
    geom_point() +
    geom_smooth(method='loess') +
    theme_minimal() +
    facet_wrap(~predictors, scales="free_y")
```

```
# Calculate McFaddens Psuedo R^2
ll.null <- glm.stepwise$null.deviance/-2
ll.proposed <- glm.stepwise$deviance/-2
R2_stepwise <-(ll.null - ll.proposed)/ll.null
```

```
predictions.stepwise <- data.frame(prob=glm.stepwise$fitted.values, target=train$target) %>% arrange(pr
predictions.stepwise$rank <- 1:nrow(predictions.stepwise)

## marginal effects for coefficients - now the coefficients can be interpreted
logit_scalar <- mean(dnorm(predict(glm.stepwise, type="link")))
marginals.stepwise <- logit_scalar * coef(glm.stepwise)

## use the test data set to make predicts and calculate metrics from the confusion matrix
glm_stepwise.probs <- predict(glm.stepwise,type="response", newdata=test)
glm_predict.stepwise <- ifelse(glm_stepwise.probs > 0.5, '1','0')
attach(test)
#table(glm_predict.stepwise, test$target)
```

```
# now can use the caret function
cm.stepwise <- confusionMatrix(factor(glm_predict.stepwise), factor(test$target), positive='1')
cm.stepwise$table

# ROC and AUC
par(pty="s")
roc.stepwise <- roc(test$target, glm_stepwise.probs)
```

```r
# Get McFaddens's R-squared
ll.null <- glm.back$null.deviance/-2
ll.proposed <- glm.back$deviance/-2
R2_back <-(ll.null - ll.proposed)/ll.null

# plot the model
predictions.back <- data.frame(prob=glm.back$fitted.values, target=train$target) %>% arrange(prob)
predictions.back$rank <- 1:nrow(predictions.back)


## marginal effects for coefficients - now the coefficients can be interpreted.. ie age, 0.1% more like
logit_scalar <- mean(dnorm(predict(glm.back, type="link")))
marginals.back <- logit_scalar * coef(glm.back)

## use the test data set to make predicts and calculate metrics from the confusion matrix
glm_back.probs <- predict(glm.back,type="response", newdata=test)
glm_predict.back <- ifelse(glm_back.probs > 0.5, '1','0')
attach(test)
```

```r
# now can use the caret function
cm.back <- confusionMatrix(factor(glm_predict.back), factor(test$target), positive='1')
cm.back$table

# ROC and AUC
par(pty="s")
roc.back <- roc(test$target, glm_back.probs)
```

Select Model

```r
temp <- data.frame(cm.full$overall,
                   cm.stepwise$overall,
                   cm.back$overall) %>%
  t() %>%
  data.frame() %>%
  dplyr::select(Accuracy) %>%
  mutate(`Classification Error Rate` = 1-Accuracy)
```

```r
eval <- data.frame(cm.full$byClass,
                   cm.stepwise$byClass,
                   cm.back$byClass)
eval <- data.frame(t(eval)) %>%
  cbind(temp) %>%
  mutate(eval = c("Full Model", "AIC Stepwise", "Manual Backwards"))
```

```r
eval <- dplyr::select(eval, Accuracy, `Classification Error Rate`, Sensitivity, Specificity, Precision,

# r-squared is better for the stepwise model, which is expected
R2.combined <- c(R2_full, R2_stepwise, R2_back)

# AIC is lower in the stepwise model suggesting it is closer to the "true" model
AIC.combined <- c(glm.full$aic, glm.stepwise$aic, glm.back$aic)
```

```r
# Residual Deviance are lower in the stepwise model
DEV.combined <- c(glm.full$deviance, glm.stepwise$deviance, glm.back$deviance)

# Area under the curve is slightly better for the stepwise model
AUC.combined <- c(roc.full$auc, roc.stepwise$auc, roc.back$auc)

eval <- cbind(eval, `R Squared`=R2.combined, AIC=AIC.combined, Deviance=DEV.combined, AUC=AUC.combined)

rownames(eval) = c("Full Model", "AIC Stepwise", "Manual Backwards")

t_eval <- t(eval)
colnames(t_eval) <- rownames(eval)
rownames(t_eval) <- colnames(eval)

knitr::kable(t_eval)
```

```r
par(mfrow=c(2,2))
plot(roc.full, print.auc=TRUE, main="Full Model")
plot(roc.stepwise, print.auc=TRUE, main="AIC Stepwise")
plot(roc.back, print.auc=TRUE, main="Manual Backwards")

par(mfrow=c(1,1))
```

```r
# model plots
p1 <- predictions.full %>% ggplot(aes(x=rank, y=prob)) +
  geom_point(aes(color=factor(target)), alpha=.3, shape=4, stroke=2) +
  ggtitle("Full Model") + theme(legend.title=element_blank())

p2 <- predictions.stepwise %>% ggplot(aes(x=rank, y=prob)) +
  geom_point(aes(color=factor(target)), alpha=.3, shape=4, stroke=2) +
  ggtitle("AIC Stepwise") + theme(legend.title=element_blank())

p3 <- predictions.back %>% ggplot(aes(x=rank, y=prob)) +
  geom_point(aes(color=factor(target)), alpha=.3, shape=4, stroke=2) +
  ggtitle("Manual Backwards") + theme(legend.title=element_blank())

grid.arrange(p1, p2, p3, ncol=2, nrow=2)
```

```r
mod_fit <- train(target ~ zn + nox + age + dis + rad + tax + ptratio +
      lstat + medv,  data=train, method="glm", family="binomial")
```

```r
# check that there are no outliers
plot(glm.stepwise, which = 4, id.n = 3)
```

```r
#compute variance inflation factors
car::vif(glm.stepwise)
```