HW3

Team 1

October 25, 2020

1. Data Exploration

The "Neighbourhood crime data" training data set contains 466 rows and 13 columns. The variables are thought to have a positive or negative effect on the crime rate being above median crime rate. Running a summary() function on the data set, we are able to get the mean, median, first and third quartile, and the minimum and maximum values for each variable. We included a correlation plot and pairs plot to visualize the relationship among the variables. From the correlation plot we see that dis is negatively correlated with the target (above-average median crime rate). This would suggest that lots of crime happens around these Boston employment centers. Unsurprisingly, the variables that describe a more affluent neighborhood seem to be less corrlated with crime and the variables associated with more poor neighborhoods seem more correlated with crime. We explored the structure of the variables for both the training and evaluation data sets and finally observed how Target variable is affected by other factors.

2. Data Preparation

Interestingly this data was clean and there were no NA values identified in the data set. But going through the dataset description we could identify that some values are numerical but represents certain classes and therefore should be converted into factors for the modeling process. These included chas and rad.

3. Build Models

We focused on building a logistic regression model since the target variable was binary variable with values limited to 0 or 1. We used the glm package available within R to perform this. We build 3 LR models using all independent variables or subsets of them and use stepwise regression. Once we trained the models, we stored the AIC, null, and residual deviance values in a table for easy representation of basic parameters of these models. When looking at the coefficients of our best model and their p-values, we keep the model because it has the best classification metrics that we care about. Most interesting is that it seems to find nox as the most significant with lowest p-value, which maybe suggests some closeness to industry.

4. Select Models

Out of the three models, the model that included all the parameters had lowest AIC and residual deviance value. That suggested that this model was the best model, which we verified by checking the ROC-AUC curve and the confusion matrix. We also generated the AIC for other models as well using stepAIC from the MASS package. Our evaluation metrics are as follows: Accuracy: .97 Classification Error Rate: .03

Precision: .9821 Sensitivity: .9563 Specificity: .9831 F1 score: .9690 AUC: .989 True Positive: 219 False Positive: 4 True Negative: 233 False Negative: 10

Our predictions are attached in a csv called eval_preds.csv. See the bottom of the appendix for more details about the metrics of our best model.

Appendix

Library

```
# load required packages
library(ggplot2)
library(dplyr)
#library(tidyr)
library(corrplot)
library(MASS)
library(caret)
library(RCurl)
library(tidyverse)
library(pROC)
library(kableExtra)
library(RCurl)
```

```
# Loading the data

git_dir <- 'https://raw.githubusercontent.com/Sizzlo/Data621/main'
train_df = read.csv(paste(git_dir, "/crime-training-data_modified.csv", sep=""))
test_df = read.csv(paste(git_dir, "/crime-evaluation-data_modified.csv", sep = ""))
head(train_df)
```

```
##   zn indus chas   nox    rm   age    dis rad tax ptratio lstat medv target
## 1  0 19.58    0 0.605 7.929  96.2 2.0459   5 403    14.7  3.70 50.0      1
## 2  0 19.58    1 0.871 5.403 100.0 1.3216   5 403    14.7 26.82 13.4      1
## 3  0 18.10    0 0.740 6.485 100.0 1.9784  24 666    20.2 18.85 15.4      1
## 4 30  4.93    0 0.428 6.393   7.8 7.0355   6 300    16.6  5.19 23.7      0
## 5  0  2.46    0 0.488 7.155  92.2 2.7006   3 193    17.8  4.82 37.9      0
## 6  0  8.56    0 0.520 6.781  71.3 2.8561   5 384    20.9  7.67 26.5      0
```

Data Exploration & Preparation

Summary of data

See a summary of each column in the train_df set

```
# view a summary of all columns
summary(train_df)
```

```
##        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  
```

Structure of the data

```r
str(train_df)
```

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

```r
str(test_df)
```

```
## 'data.frame':    40 obs. of  12 variables:
```

```
##  $ zn     : int  0 0 0 0 0 25 25 0 0 0 ...
##  $ indus  : num  7.07 8.14 8.14 8.14 5.96 5.13 5.13 4.49 4.49 2.89 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.469 0.538 0.538 0.538 0.499 0.453 0.453 0.449 0.449 0.445 ...
##  $ rm     : num  7.18 6.1 6.5 5.95 5.85 ...
##  $ age    : num  61.1 84.5 94.4 82 41.5 66.2 93.4 56.1 56.8 69.6 ...
##  $ dis    : num  4.97 4.46 4.45 3.99 3.93 ...
##  $ rad    : int  2 4 4 4 5 8 8 3 3 2 ...
##  $ tax    : int  242 307 307 307 279 284 284 247 247 276 ...
##  $ ptratio: num  17.8 21 21 21 19.2 19.7 19.7 18.5 18.5 18 ...
##  $ lstat  : num  4.03 10.26 12.8 27.71 8.77 ...
##  $ medv   : num  34.7 18.2 18.4 13.2 21 18.7 16 26.6 22.2 21.4 ...
```

NA check

```r
has_NA = names(which(sapply(train_df, anyNA)))
has_NA
```

```
## character(0)
```

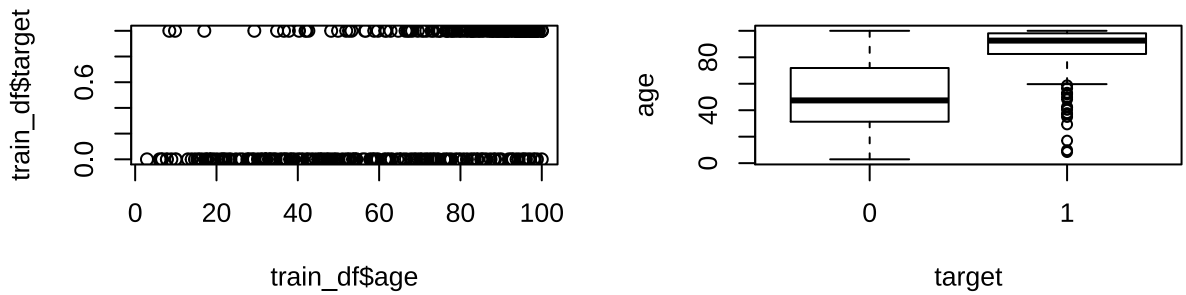
There are no NAs observed

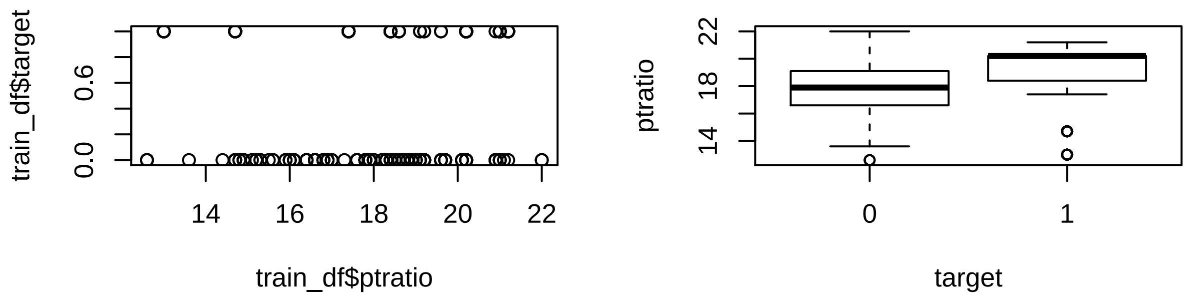
The summary() function for the training and testing data sets indicates that there are no missing values in the data. The response variable "target" is binary with 1 indicates crime rate is above median cirme rate and 0 indicates crime rate is not above median crime rate.

Let's observe how the target variable is effected by other factors: 1. The plot of "target" against "age" shows target equalling one (above median crime rate) increases as the proportion of owner-occupied units built prior to 1940 increaases; the boxplot further shows that a larger mean of proportions of owner-occupied units built prior to 1940 is assoicated with higher crime rate. 2. Plots of crime rate against pupil-teacher ratio indicate higher crime rate "1" is associated with higher pupil-teacher ratio.

Plotting

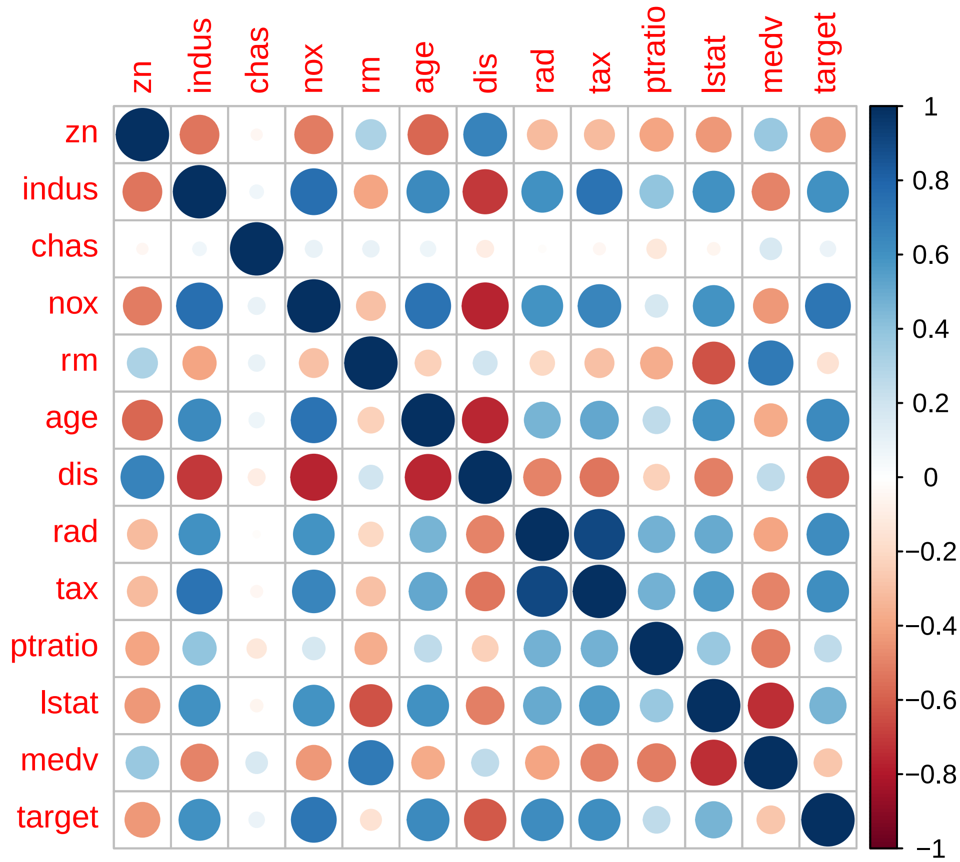
```r
par(mfrow=c(2,2))
# plot response variable "target" against predictor variable "age"
plot(train_df$age,train_df$target)
boxplot(age ~ target, train_df )

# plot response variable "target" against predictor variable "ptratio"
plot(train_df$ptratio,train_df$target)
boxplot(ptratio ~ target, train_df)
```



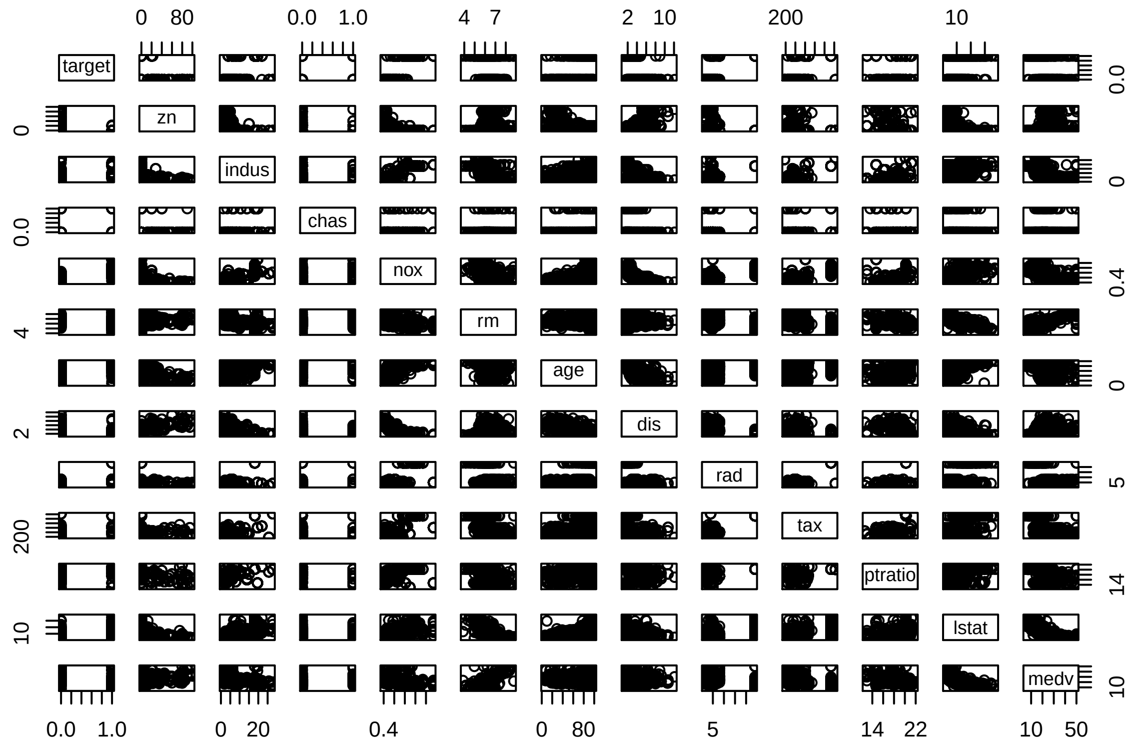


Corr analysis

```r
# Correlations
cor_train <- cor(train_df,  use = "na.or.complete")
corrplot(cor_train)
```




```r
pairs(~ target + zn + indus
      + chas + nox + rm + age + dis + rad + tax + ptratio + lstat + medv, data = train_df)
```



Converting to factors

```r
train_df$chas = as.factor(train_df$chas)
train_df$rad = as.factor(train_df$rad)

test_df$chas = as.factor(test_df$chas)
test_df$rad = as.factor(test_df$rad)
```

```r
model_metrics_df <- data.frame(Model=NA, AIC=NA, Null.Deviance=NA, Resid.Deviance=NA)


gather_metrics_func <- function(type, model_metrics_df, modelSummary) {
aic <- round(modelSummary$aic,4)
nullDeviance <- round(modelSummary$null.deviance, 4)
residDeviance <- round(modelSummary$df.residual, 4)

model_metrics_df <- rbind(model_metrics_df,c(type, aic, nullDeviance, residDeviance))
model_metrics_df <- na.omit(model_metrics_df)
return(model_metrics_df)
}
```

Modeling

Binary Logistic Regression

We are running Binary Logistic regression model with three 3 different set of parameters

Modeling with Target ~ Age

```
# preliminary exploration glm models
model1 <- glm(formula = target ~ age, family = binomial(), data = train_df)
summary(model1)
```

```
##
## Call:
## glm(formula = target ~ age, family = binomial(), data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9906  -0.6040  -0.1609   0.6659   2.9096
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.773112   0.465483  -10.25   <2e-16 ***
## age          0.066060   0.005922   11.15   <2e-16 ***
## ---
## 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: 424.75  on 464  degrees of freedom
## AIC: 428.75
##
## Number of Fisher Scoring iterations: 5
```

```
model_metrics_df <- gather_metrics_func('target ~ age', model_metrics_df, model1)
```

Modelling with Target ~ ptratio

```
# preliminary exploration glm models
model2 <- glm(formula = target ~ ptratio , family = binomial(), data = train_df)
summary(model2)
```

```
##
## Call:
## glm(formula = target ~ ptratio, family = binomial(), data = train_df)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max  
## -1.5439  -1.1075  -0.7538   1.0160   1.7812  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -4.51685    0.86035  -5.250 1.52e-07 ***
## ptratio      0.24303    0.04617   5.264 1.41e-07 ***
## ---
## 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: 615.64  on 464  degrees of freedom
## AIC: 619.64
## 
## Number of Fisher Scoring iterations: 4
```

```
model_metrics_df <- gather_metrics_func('target ~ ptratio', model_metrics_df, model2)
```

Modelling with Target ~ .(every other variable)

```
all_preds = glm(target ~ ., family = binomial, data = train_df)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(all_preds)
```

```
## 
## Call:
## glm(formula = target ~ ., family = binomial, data = train_df)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -2.5265  -0.0409   0.0000   0.0001   4.3848  
## 
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -5.526e+01  5.099e+03  -0.011   0.9914    
## zn          -1.609e-01  6.574e-02  -2.447   0.0144 *  
## indus       -1.562e-01  1.166e-01  -1.340   0.1802    
## chas1       -2.603e-01  9.626e-01  -0.270   0.7869    
## nox          6.863e+01  1.362e+01   5.038 4.71e-07 ***
## rm          -1.225e+00  1.010e+00  -1.213   0.2250    
## age          1.871e-02  1.569e-02   1.193   0.2330    
## dis          5.351e-01  2.671e-01   2.003   0.0452 *  
## rad2        -4.532e-01  7.114e+03   0.000   0.9999    
## rad3         1.783e+01  5.099e+03   0.003   0.9972    
## rad4         2.221e+01  5.099e+03   0.004   0.9965    
## rad5         1.950e+01  5.099e+03   0.004   0.9969    
```

```
## rad6         1.738e+01  5.099e+03   0.003   0.9973
## rad7         2.700e+01  5.099e+03   0.005   0.9958
## rad8         2.564e+01  5.099e+03   0.005   0.9960
## rad24        4.404e+01  5.457e+03   0.008   0.9936
## tax         -9.491e-03  5.442e-03  -1.744   0.0811 .
## ptratio      4.824e-02  2.040e-01   0.236   0.8131
## lstat        6.778e-02  6.441e-02   1.052   0.2927
## medv         2.195e-01  9.964e-02   2.203   0.0276 *
## ---
## 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: 116.98  on 446  degrees of freedom
## AIC: 156.98
##
## Number of Fisher Scoring iterations: 20
```

```
model_metrics_df <- gather_metrics_func('target ~ .', model_metrics_df, all_preds)
```

Comparing different models performance

```
model_metrics_df %>% kbl() %>% kable_styling()
```

	Model	AIC	Null.Deviance	Resid.Deviance
2	target ~ age	428.7471	645.8758	464
21	target ~ ptratio	619.6385	645.8758	464
3	target ~ .	156.9822	645.8758	446

Looking at the table, we can identify on a high level that 3rd model that includes all the parameters is better suited. Therefore, let's come up with a confusion matrix for 3rd model that includes all the parameters.

```
train_df$preds = ifelse(all_preds$fitted.values > 0.5, 1, 0)
# look at confusion matrix
cm = confusionMatrix(as_factor(train_df$preds), as_factor(train_df$target), positive = "1")
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 233  10
##          1   4 219
##
##                Accuracy : 0.97
##                  95% CI : (0.9501, 0.9835)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
##
```

```
##                   Kappa : 0.9399
##
##  Mcnemar's Test P-Value : 0.1814
##
##             Sensitivity : 0.9563
##             Specificity : 0.9831
##          Pos Pred Value : 0.9821
##          Neg Pred Value : 0.9588
##              Prevalence : 0.4914
##          Detection Rate : 0.4700
##    Detection Prevalence : 0.4785
##       Balanced Accuracy : 0.9697
##
##        'Positive' Class : 1
##
```

Using StepAIC

Using the MASS package provided 'stepAIC' lets try to further refine the available models within it

```r
step_all_preds = stepAIC(all_preds)
```

```
## Start:  AIC=156.98
## target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     ptratio + lstat + medv

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##           Df Deviance    AIC
## - ptratio  1   117.04 155.04
## - chas     1   117.06 155.06
## - lstat    1   118.07 156.07
```

```
## - age      1   118.44 156.44
## - rm       1   118.50 156.50
## - indus    1   118.82 156.82
## <none>         116.98 156.98
## - tax      1   120.42 158.42
## - dis      1   121.06 159.06
## - medv     1   122.98 160.98
## - zn       1   125.74 163.74
## - nox      1   185.39 223.39
## - rad      8   233.74 257.74

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##
## Step:  AIC=155.04
## target ~ zn + indus + chas + nox + rm + age + dis + rad + tax +
##     lstat + medv

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##         Df Deviance    AIC
## - chas   1   117.11 153.11
## - lstat  1   118.14 154.15
## - age    1   118.46 154.46
## - rm     1   118.53 154.53
## <none>       117.04 155.04
## - indus  1   119.35 155.35
## - tax    1   120.42 156.42
## - dis    1   121.17 157.17
## - medv   1   124.02 160.02
## - zn     1   127.07 163.07
## - nox    1   187.89 223.89
## - rad    8   242.93 264.93

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
##
## Step:  AIC=153.11
## target ~ zn + indus + nox + rm + age + dis + rad + tax + lstat +
##     medv

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##         Df Deviance    AIC
## - lstat  1   118.17 152.17
## - age    1   118.46 152.46
## - rm     1   118.54 152.54
## <none>       117.11 153.11
## - indus  1   120.17 154.17
## - tax    1   120.66 154.66
## - dis    1   121.41 155.41
## - medv   1   124.07 158.07
## - zn     1   127.10 161.10
## - nox    1   190.43 224.43
## - rad    8   247.55 267.55

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##
## Step:  AIC=152.17
## target ~ zn + indus + nox + rm + age + dis + rad + tax + medv

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

##         Df Deviance    AIC
## <none>       118.17 152.17
## - age    1   120.74 152.74
## - indus  1   120.93 152.93
## - rm     1   121.05 153.05
## - tax    1   121.73 153.73
## - dis    1   122.35 154.35
## - medv   1   125.18 157.18
## - zn     1   127.58 159.58
## - nox    1   191.60 223.60
## - rad    8   249.92 267.92
```

```r
summary(step_all_preds)
```

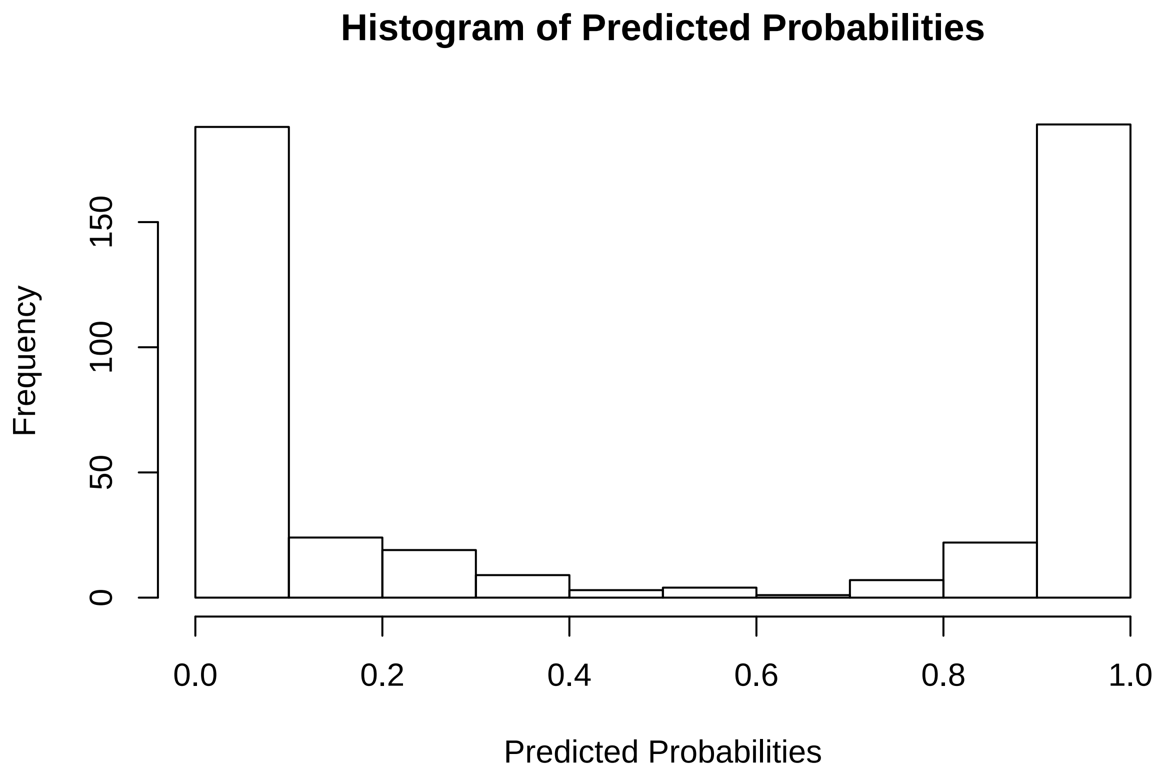
```
##
## Call:
## glm(formula = target ~ zn + indus + nox + rm + age + dis + rad +
##     tax + medv, family = binomial, data = train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3520  -0.0443   0.0000   0.0001   4.3170
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.053e+01  3.170e+03  -0.016   0.9873
## zn          -1.480e-01  5.772e-02  -2.564   0.0104 *
## indus       -1.613e-01  9.835e-02  -1.640   0.1009
## nox          6.718e+01  1.255e+01   5.353 8.64e-08 ***
## rm          -1.462e+00  8.701e-01  -1.681   0.0928 .
## age          2.172e-02  1.364e-02   1.592   0.1113
## dis          5.469e-01  2.689e-01   2.034   0.0420 *
## rad2        -1.873e-02  4.418e+03   0.000   1.0000
## rad3         1.695e+01  3.170e+03   0.005   0.9957
## rad4         2.139e+01  3.170e+03   0.007   0.9946
## rad5         1.839e+01  3.170e+03   0.006   0.9954
## rad6         1.661e+01  3.170e+03   0.005   0.9958
## rad7         2.563e+01  3.170e+03   0.008   0.9935
## rad8         2.434e+01  3.170e+03   0.008   0.9939
## rad24        4.192e+01  3.387e+03   0.012   0.9901
## tax         -8.591e-03  4.788e-03  -1.794   0.0728 .
## medv         2.047e-01  8.269e-02   2.475   0.0133 *
## ---
## 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: 118.17  on 449  degrees of freedom
## AIC: 152.17
##
## Number of Fisher Scoring iterations: 19
```

```r
train_df$preds = ifelse(step_all_preds$fitted.values > 0.5, 1, 0)
train_df$pred_proba = step_all_preds$fitted.values
# look at confusion matrix
cm <- confusionMatrix(as_factor(train_df$preds), as_factor(train_df$target), positive = "1")
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 233  10
##          1   4 219
##
##                Accuracy : 0.97
##                  95% CI : (0.9501, 0.9835)
##     No Information Rate : 0.5086
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9399
##
##  Mcnemar's Test P-Value : 0.1814
##
##             Sensitivity : 0.9563
##             Specificity : 0.9831
##          Pos Pred Value : 0.9821
##          Neg Pred Value : 0.9588
##              Prevalence : 0.4914
##          Detection Rate : 0.4700
##    Detection Prevalence : 0.4785
##       Balanced Accuracy : 0.9697
##
##        'Positive' Class : 1
##
```

```r
hist(step_all_preds$fitted.values, main= "Histogram of Predicted Probabilities", xlab="Predicted Probab
```



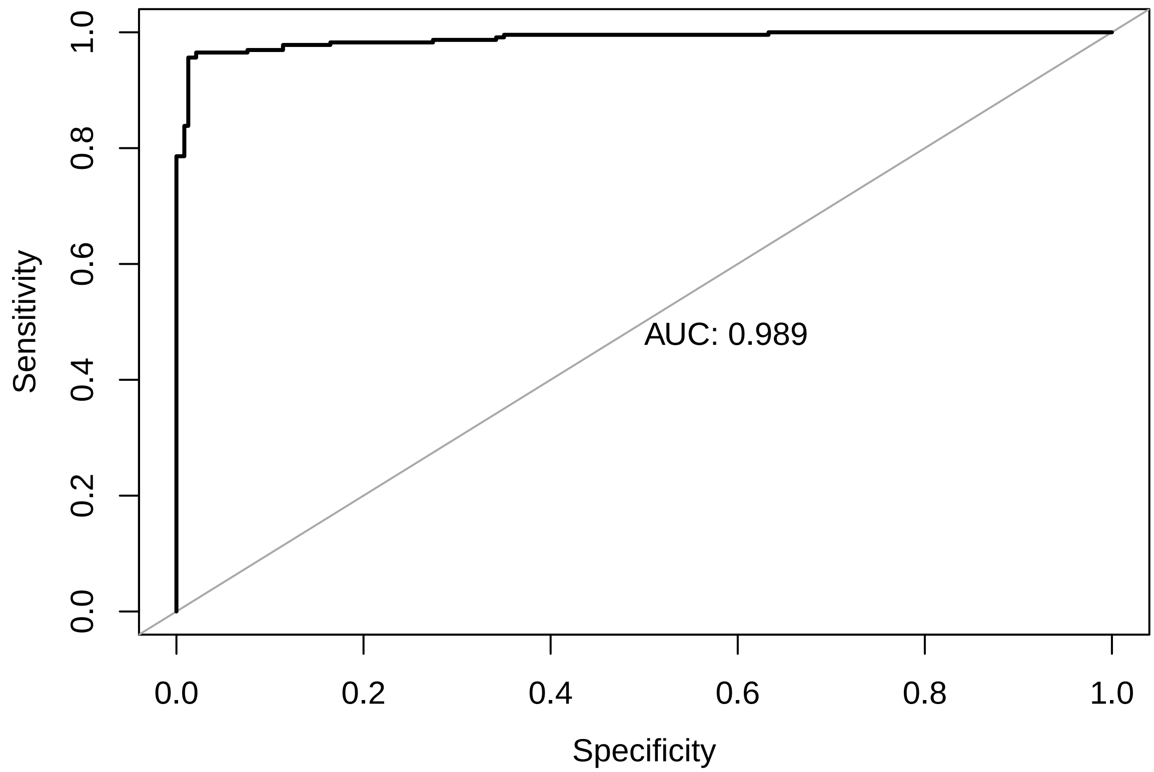
Plotting ROC

```
proc = roc(train_df$target, train_df$pred_proba)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(proc, asp=NA, legacy.axes=TRUE, print.auc=TRUE, xlab="Specificity")
```




```
recall = .9563
precision = .9821
f1 = 2*precision*recall/(precision+recall)
f1
```

```
## [1] 0.9690283
```

Conclusion

Using the above defined steps where using stepAIC and confusionMatrix we can derive at the model that has below specifications Sensitivity : 0.9563 Specificity : 0.9831 Accuracy: .97 Precision: 0.9821 AUC: .989

Predictions on evaluation set

```
model = step_all_preds
test_preds = round(predict(model, newdata=test_df, type='response')) #*162
test_df$PRED_TARGET = test_preds
write.csv(test_df, 'eval_with_preds.csv')
```