Crime Level Risk

Keeno Glanville

2023-09-14

Contents

DATA EXPLORATION

```
trainraw <- read_csv('https://raw.githubusercontent.com/kglan/MSDS/main/DATA621/HW3/crime-training-data_
```

```
## Rows: 466 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (13): zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, lstat, medv...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
testraw <- read_csv('https://raw.githubusercontent.com/kglan/MSDS/main/DATA621/HW3/crime-evaluation-data
```

```
## Rows: 40 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (12): zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, lstat, medv
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dim(trainraw)
```

```
## [1] 466  13
```

```r
head(trainraw)
```

```
## # A tibble: 6 x 13
##      zn indus  chas   nox    rm   age   dis   rad   tax ptratio lstat  medv
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
## 1     0 19.6      0 0.605  7.93  96.2  2.05     5   403    14.7  3.7   50
## 2     0 19.6      1 0.871  5.40 100    1.32     5   403    14.7 26.8   13.4
## 3     0 18.1      0 0.74   6.48 100    1.98    24   666    20.2 18.8   15.4
## 4    30  4.93     0 0.428  6.39   7.8  7.04     6   300    16.6  5.19  23.7
## 5     0  2.46     0 0.488  7.16  92.2  2.70     3   193    17.8  4.82  37.9
## 6     0  8.56     0 0.52   6.78  71.3  2.86     5   384    20.9  7.67  26.5
## # ... with 1 more variable: target <dbl>
```

```r
summary(trainraw)
```

```
##        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
```

```r
sapply(trainraw, function(x) sum(is.na(x)))
```

```
##      zn   indus    chas     nox      rm     age     dis     rad     tax ptratio
##       0       0       0       0       0       0       0       0       0       0
##   lstat    medv  target
##       0       0       0
```

```
sapply(testraw, function(x) sum(is.na(x)))
```

```
##      zn   indus    chas     nox      rm     age     dis     rad     tax ptratio
##       0       0       0       0       0       0       0       0       0       0
##   lstat    medv
##       0       0
```

```
sapply(trainraw, class)
```

```
##        zn     indus      chas       nox        rm       age       dis       rad
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##       tax   ptratio     lstat      medv    target
## "numeric" "numeric" "numeric" "numeric" "numeric"
```

```
sapply(testraw, class)
```

```
##        zn     indus      chas       nox        rm       age       dis       rad
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##       tax   ptratio     lstat      medv
## "numeric" "numeric" "numeric" "numeric"
```

DATA PREPARATION

```
train <- trainraw%>%
  rename(Residential_zone_Large = zn)%>%
  rename(Industrial_zone = indus )%>%
  rename(Charles_River_border = chas)%>%
  rename(NitrousOxide_conc = nox)%>%
  rename(Rooms_avg = rm)%>%
  rename(OwnerOccupiedUnits= age)%>%
  rename(Highway_Index = rad)%>%
  rename(dis_to_emplyoymentcenter=dis)%>%
  rename(Dangerous = target)%>%
  mutate(Charles_River_border= factor(Charles_River_border))%>%
  mutate(Highway_Index= factor(Highway_Index))%>%
  mutate(Dangerous= factor(Dangerous))

test <- testraw%>%
  rename(Residential_zone_Large = zn)%>%
  rename(Industrial_zone = indus )%>%
  rename(Charles_River_border = chas)%>%
  rename(NitrousOxide_conc = nox)%>%
  rename(Rooms_avg = rm)%>%
  rename(OwnerOccupiedUnits= age)%>%
  rename(Highway_Index = rad )%>%
  rename(dis_to_emplyoymentcenter=dis)%>%
  mutate(Charles_River_border= factor(Charles_River_border))%>%
  mutate(Highway_Index= factor(Highway_Index))
```

```
colnames(train)
```

```
##  [1] "Residential_zone_Large"   "Industrial_zone"
##  [3] "Charles_River_border"     "NitrousOxide_conc"
##  [5] "Rooms_avg"                "OwnerOccupiedUnits"
##  [7] "dis_to_emplyoymentcenter" "Highway_Index"
##  [9] "tax"                      "ptratio"
## [11] "lstat"                    "medv"
## [13] "Dangerous"
```

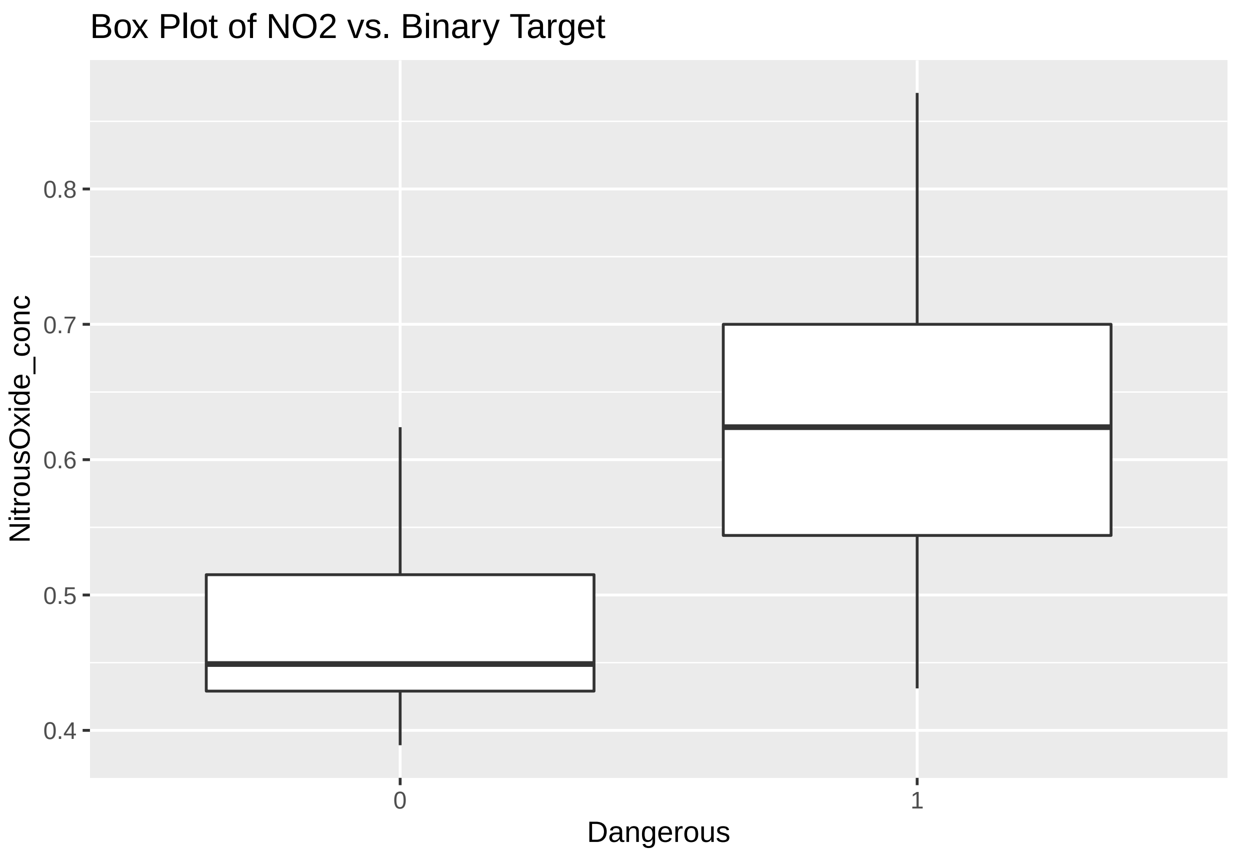
```
# Subset the dataset to include only numeric predictor variables and the target variable
numeric_predictors <- train%>%
  select(-c("Highway_Index", "Charles_River_border"))
colnames(numeric_predictors)
```

```
##  [1] "Residential_zone_Large"   "Industrial_zone"
##  [3] "NitrousOxide_conc"        "Rooms_avg"
##  [5] "OwnerOccupiedUnits"       "dis_to_emplyoymentcenter"
##  [7] "tax"                      "ptratio"
##  [9] "lstat"                    "medv"
## [11] "Dangerous"
```

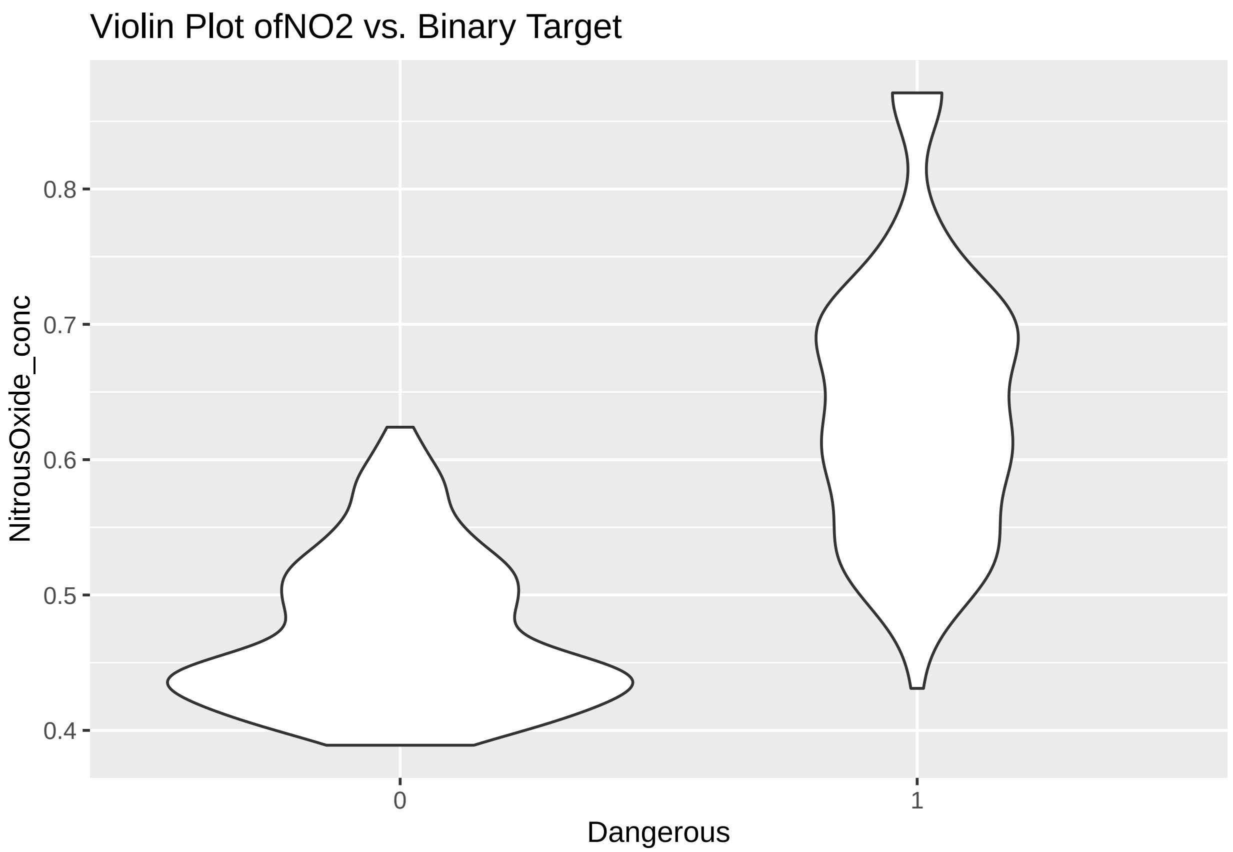
```
numeric_vars_list <- c("Residential_zone_Large", "Industrial_zone", "NitrousOxide_conc",
                       "Rooms_avg", "OwnerOccupiedUnits", "dis_to_emplyoymentcenter",
                       "tax", "ptratio", "lstat", "medv")

point_biserial_correlation <- cor(as.numeric(numeric_predictors$Dangerous), numeric_predictors[numeric_v

# View the sorted correlations
print(point_biserial_correlation)
```

```
##      Residential_zone_Large Industrial_zone NitrousOxide_conc  Rooms_avg
## [1,]             -0.4316818       0.6048507         0.7261062 -0.1525533
##      OwnerOccupiedUnits dis_to_emplyoymentcenter       tax   ptratio    lstat
## [1,]          0.6301062               -0.6186731 0.6111133 0.2508489 0.469127
##            medv
## [1,] -0.2705507
```

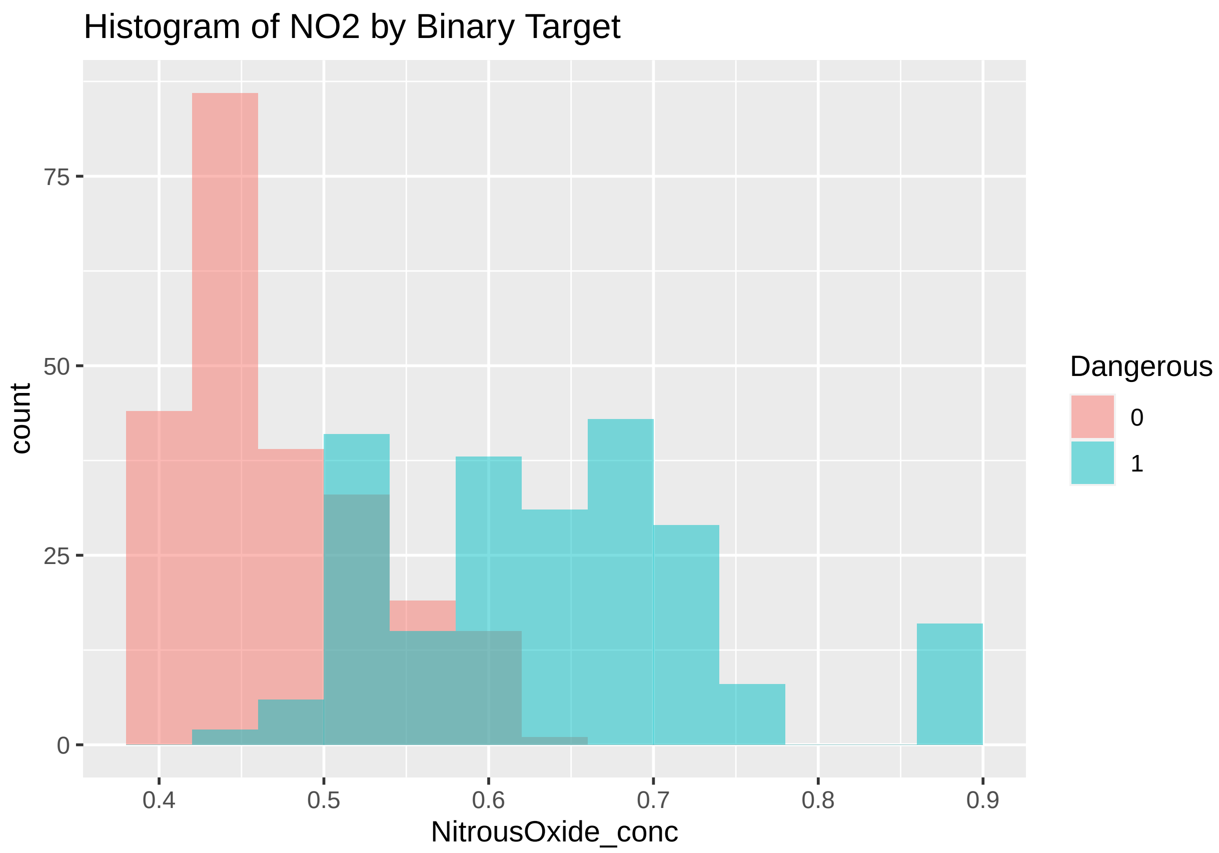
```
  # Box plot
ggplot(numeric_predictors, aes(x = Dangerous, y = NitrousOxide_conc)) +
    geom_boxplot() +
    labs(title = paste("Box Plot of NO2 vs. Binary Target"))
```



```r
ggplot(numeric_predictors, aes(x = Dangerous, y = NitrousOxide_conc)) +
    geom_violin() +
    labs(title = paste("Violin Plot ofNO2 vs. Binary Target"))
```



```
ggplot(numeric_predictors, aes(x = NitrousOxide_conc, fill = Dangerous)) +
    geom_histogram(binwidth = 0.04, position = "identity", alpha = 0.5) +
    labs(title = paste("Histogram of NO2 by Binary Target"))
```



Build Models

```
set.seed(124)  # For reproducibility
sample_indices <- sample(1:nrow(train), size = 140)  # Choose an appropriate size
validation_data <- train[sample_indices, ]
```

```
main_train_data <- train[-sample_indices, ]  # Exclude the validation subset
log_model <- train(Dangerous ~ ., data = main_train_data, method = "glm", family = binomial(link = "log
```

```
## 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

## 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

## 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
```

```r
validation_predictions <- predict(log_model, newdata = validation_data, type = "raw")
```

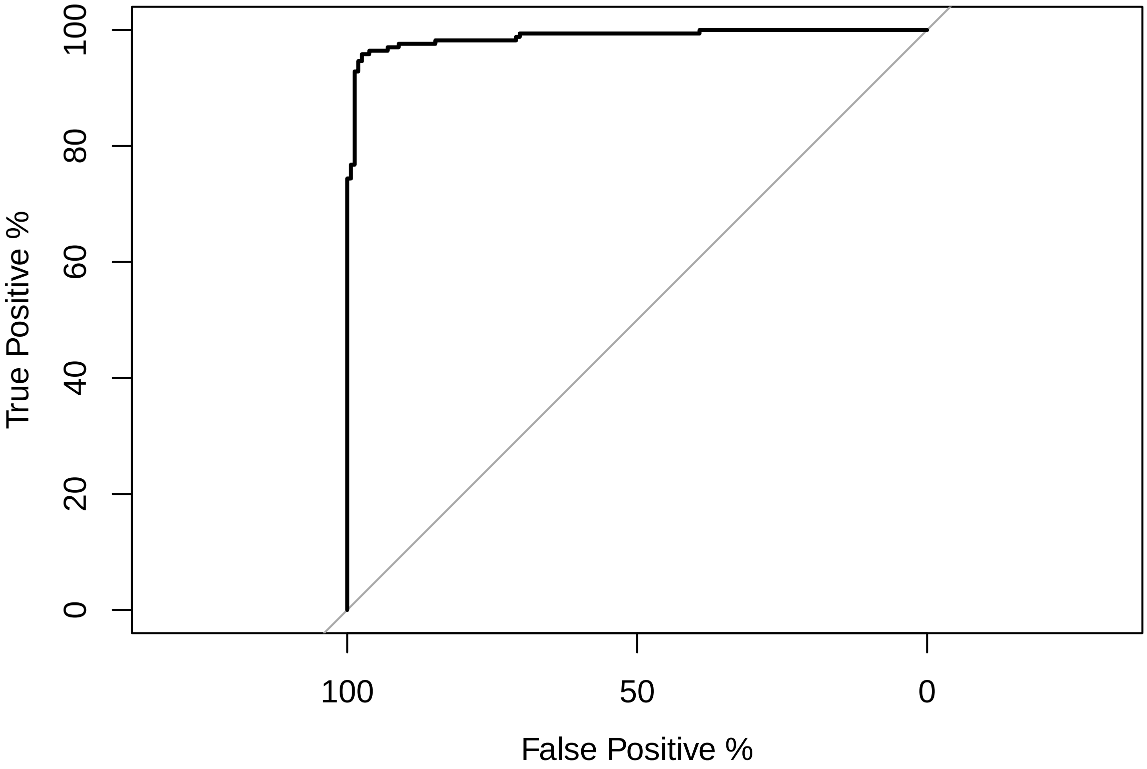
```r
conf_matrix <- confusionMatrix(validation_predictions, validation_data$Dangerous)
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  0  1
##          0 75  1
##          1  4 60
##
##                Accuracy : 0.9643
##                  95% CI : (0.9186, 0.9883)
##      No Information Rate : 0.5643
##      P-Value [Acc > NIR] : <2e-16
##
##                    Kappa : 0.9278
##
##   Mcnemar's Test P-Value : 0.3711
##
##              Sensitivity : 0.9494
##              Specificity : 0.9836
##           Pos Pred Value : 0.9868
##           Neg Pred Value : 0.9375
##               Prevalence : 0.5643
##           Detection Rate : 0.5357
##     Detection Prevalence : 0.5429
##        Balanced Accuracy : 0.9665
##
##         'Positive' Class : 0
##
```

```
roc(main_train_data$Dangerous, as.vector(fitted.values(log_model)), percent=T,   boot.n=1000, ci.alpha=
```

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

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



```
##
## Call:
## roc.default(response = main_train_data$Dangerous, predictor = as.vector(fitted.values(log_model)),
##
## Data: as.vector(fitted.values(log_model)) in 158 controls (main_train_data$Dangerous 0) < 168 cases
## Area under the curve: 98.8%
```

```
log_model2 <- train(Dangerous ~ Industrial_zone + NitrousOxide_conc + OwnerOccupiedUnits + tax , data =
```

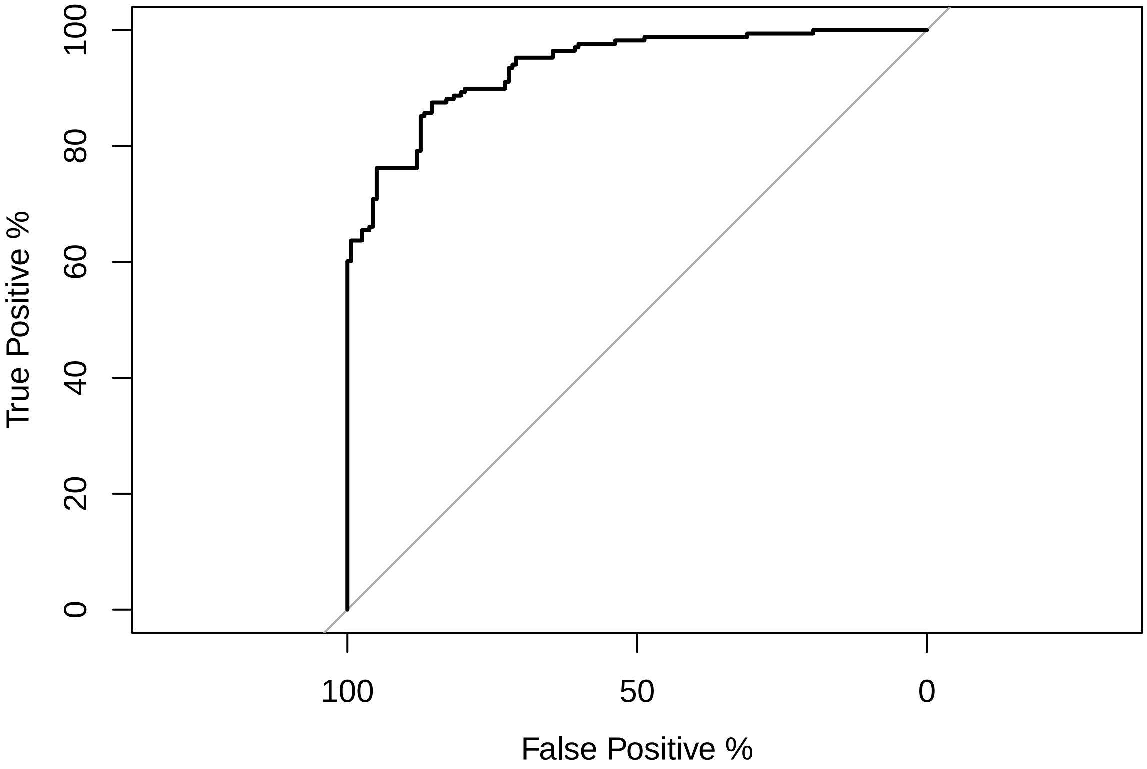
```
validation_predictions2 <- predict(log_model2, newdata = validation_data, type = "raw")
conf_matrix2 <- confusionMatrix(validation_predictions2, validation_data$Dangerous)
conf_matrix2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 73 13
##          1  6 48
##
##                Accuracy : 0.8643
##                  95% CI : (0.7962, 0.9163)
##     No Information Rate : 0.5643
##     P-Value [Acc > NIR] : 2.034e-14
##
##                   Kappa : 0.7204
##
##  Mcnemar’s Test P-Value : 0.1687
##
##             Sensitivity : 0.9241
##             Specificity : 0.7869
##          Pos Pred Value : 0.8488
##          Neg Pred Value : 0.8889
##              Prevalence : 0.5643
##          Detection Rate : 0.5214
##    Detection Prevalence : 0.6143
##       Balanced Accuracy : 0.8555
##
##        ’Positive’ Class : 0
##
```

```
roc(main_train_data$Dangerous, as.vector(fitted.values(log_model2)), percent=T,   boot.n=1000, ci.alpha
```

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

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



```
##
## Call:
## roc.default(response = main_train_data$Dangerous, predictor = as.vector(fitted.values(log_model2)),
##
## Data: as.vector(fitted.values(log_model2)) in 158 controls (main_train_data$Dangerous 0) < 168 cases
## Area under the curve: 93.66%
```

```
# Extract the numeric variables from your dataset
numeric_data <- main_train_data[, numeric_vars_list]

# Scale the numeric variables
scaled_data <- scale(numeric_data)

# Replace the original numeric variables with the scaled values
main_train_data[, numeric_vars_list] <- scaled_data






#Validation data
numeric_datav <- validation_data[, numeric_vars_list]

# Scale the numeric variables
scaled_datav <- scale(numeric_datav)
```

```r
# Replace the original numeric variables with the scaled values
validation_data[, numeric_vars_list] <- scaled_datav
```

```r
# Define the names of the variables to one-hot encode
categorical_vars_list <- c("Charles_River_border", "Highway_Index")

# Create one-hot encoded variables
one_hot_encoded <- model.matrix(~ . - 1, data = main_train_data[, categorical_vars_list])

# Add the one-hot encoded variables to the original dataset
main_train_data <- cbind(main_train_data, one_hot_encoded)

# Remove the original categorical variables
main_train_data <- main_train_data[, !names(main_train_data) %in% categorical_vars_list]
```

```r
#Validation data
# Create one-hot encoded variables
one_hot_encodedv <- model.matrix(~ . - 1, data = validation_data[, categorical_vars_list])

# Add the one-hot encoded variables to the original dataset
validation_data <- cbind(validation_data, one_hot_encodedv)

# Remove the original categorical variables
validation_data <- validation_data[, !names(validation_data) %in% categorical_vars_list]
```

```r
# Test rescaled model
log_model3 <- train(Dangerous ~ . , data = main_train_data, method = "glm", family = binomial(link = "lo
```

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning: glm.fit: algorithm did not converge

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

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

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

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

```
validation_predictions3 <- predict(log_model3, newdata = validation_data, type = "raw")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

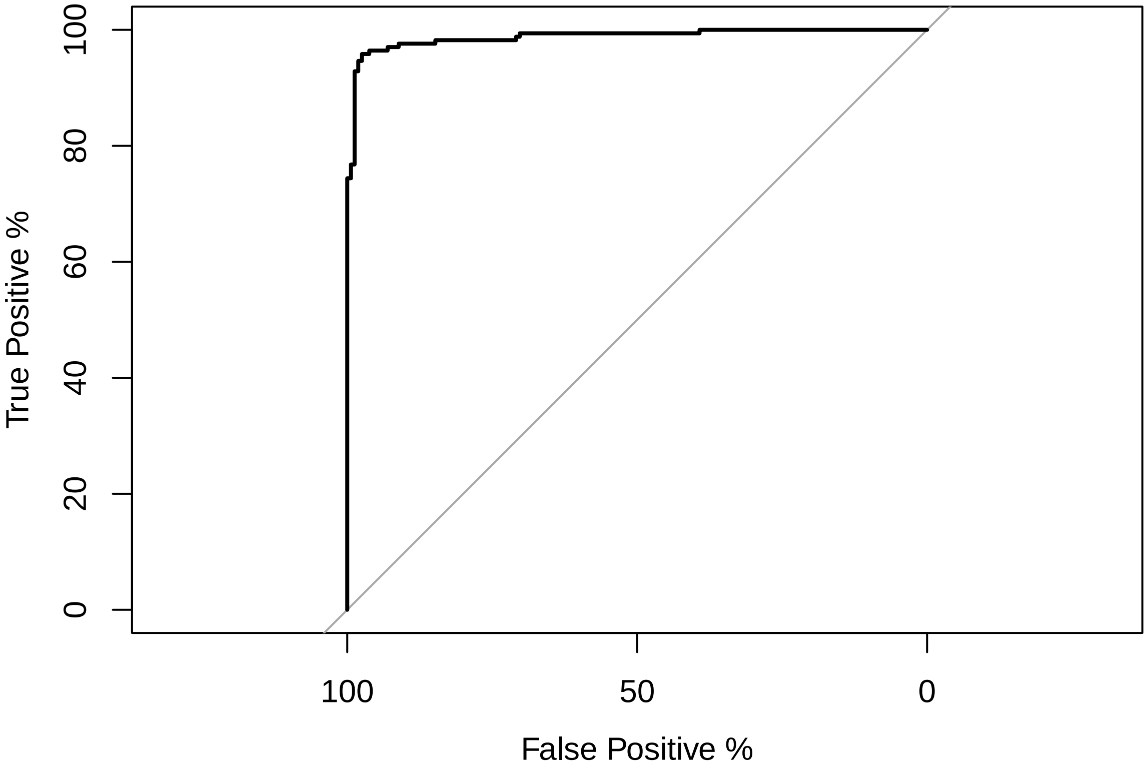
```
conf_matrix3 <- confusionMatrix(validation_predictions3, validation_data$Dangerous)
conf_matrix3
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction  0  1
##          0 71  1
##          1  8 60
##
##                Accuracy : 0.9357
##                  95% CI : (0.8815, 0.9702)
##     No Information Rate : 0.5643
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.871
##
##  Mcnemar’s Test P-Value : 0.0455
##
##             Sensitivity : 0.8987
##             Specificity : 0.9836
##          Pos Pred Value : 0.9861
##          Neg Pred Value : 0.8824
##              Prevalence : 0.5643
##          Detection Rate : 0.5071
##    Detection Prevalence : 0.5143
##       Balanced Accuracy : 0.9412
##
##        ’Positive’ Class : 0
##
```

```
roc(main_train_data$Dangerous, as.vector(fitted.values(log_model3)), percent=T,   boot.n=1000, ci.alpha
```

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

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



```
##
## Call:
## roc.default(response = main_train_data$Dangerous, predictor = as.vector(fitted.values(log_model3)),
##
## Data: as.vector(fitted.values(log_model3)) in 158 controls (main_train_data$Dangerous 0) < 168 cases
## Area under the curve: 98.8%
```

SELECT MODELS

Predicting on the given test data, I wanted to chose a model that was accurate but didnt seem to be overfitted. The way I accomplished this was by scaling the numerical variables and one-hot encoding the categorical variables. The test set provided didnt have values I could use for the prediction ROC and AUC curve so I instead subsetted the training data.

```
# Extract the numeric variables from your dataset
numeric_datat <- test[, numeric_vars_list]

# Scale the numeric variables
scaled_datat <- scale(numeric_datat)

# Replace the original numeric variables with the scaled values
test[, numeric_vars_list] <- scaled_datat
#Validation data
# Create one-hot encoded variables
```

```
one_hot_encodedt <- model.matrix(~ . - 1, data = test[, categorical_vars_list])

# Add the one-hot encoded variables to the original dataset
test <- cbind(test, one_hot_encodedt)

# Remove the original categorical variables
test <- test[, !names(test) %in% categorical_vars_list]
```

```
test_predictionst <- predict(log_model3, newdata = test, type = "raw")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
test_predictionst<- as.data.frame(test_predictionst)
```

```
test <- cbind(test, test_predictionst)
```

```
head(test)
```

```
##   Residential_zone_Large Industrial_zone NitrousOxide_conc  Rooms_avg
## 1             -0.3864190      -0.6244111        -0.8386288  1.4261128
## 2             -0.3864190      -0.4738235        -0.1969278 -0.1737603
## 3             -0.3864190      -0.4738235        -0.1969278  0.4124191
## 4             -0.3864190      -0.4738235        -0.1969278 -0.3882520
## 5             -0.3864190      -0.7806283        -0.5596284 -0.5351642
## 6              0.7020852      -0.8974393        -0.9874290 -0.6952984
##   OwnerOccupiedUnits dis_to_emplyoymentcenter        tax     ptratio
## 1         -0.3764993               0.55698900 -0.8543307 -0.80987496
## 2          0.5143080               0.31844789 -0.4877862  1.15345828
## 3          0.8911880               0.31504826 -0.4877862  1.15345828
## 4          0.4191362               0.09563010 -0.4877862  1.15345828
## 5         -1.1226456               0.06928292 -0.6456823  0.04908333
## 6         -0.1823490               1.62329420 -0.6174866  0.35585415
##         lstat        medv Charles_River_border0 Charles_River_border1
## 1 -1.15653394  1.46235907                     1                     0
## 2 -0.34465682 -0.41903856                     1                     0
## 3 -0.01365074 -0.39623374                     1                     0
## 4  1.92938102 -0.98915906                     1                     0
## 5 -0.53882968 -0.09977109                     1                     0
## 6  0.03196033 -0.36202651                     1                     0
##   Highway_Index2 Highway_Index3 Highway_Index4 Highway_Index5 Highway_Index6
## 1              1              0              0              0              0
## 2              0              0              1              0              0
## 3              0              0              1              0              0
## 4              0              0              1              0              0
## 5              0              0              0              1              0
## 6              0              0              0              0              0
##   Highway_Index7 Highway_Index8 Highway_Index24 test_predictionst
## 1              0              0               0                 0
## 2              0              0               0                 1
## 3              0              0               0                 1
## 4              0              0               0                 1
## 5              0              0               0                 0
## 6              0              1               0                 0
```