Classification

Vladimir Sokolov

9/26/2022

This notebook explores smoke detector data from Kaggle.

#Linear models for classification find a line that separates the observations such that they are placed into the correct classes. This line is called the decision boundary. Linear models are similar to linear regression and are high bias-low variance. The lines produced can underfit data and struggle to accurately represent non-linearly separable data sets.

Load the smoke_detection_iot.csv. Remove x, UTC, and CNT as they are not predictors. Change Fire.Alarm to a factor.

```
df <- read.csv("smoke_detection_iot.csv")
str(df)
```

```
## 'data.frame':    62630 obs. of  16 variables:
##  $ X              : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ UTC            : int  1654733331 1654733332 1654733333 1654733334 1654733335 1654733336 1654733337
##  $ Temperature.C.: num  20 20 20 20 20.1 ...
##  $ Humidity...    : num  57.4 56.7 56 55.3 54.7 ...
##  $ TVOC.ppb.      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ eCO2.ppm.      : int  400 400 400 400 400 400 400 400 400 400 ...
##  $ Raw.H2         : int  12306 12345 12374 12390 12403 12419 12432 12439 12448 12453 ...
##  $ Raw.Ethanol    : int  18520 18651 18764 18849 18921 18998 19058 19114 19155 19195 ...
##  $ Pressure.hPa. : num  940 940 940 940 940 ...
##  $ PM1.0          : num  0 0 0 0 0 0 0 0 0 0.9 ...
##  $ PM2.5          : num  0 0 0 0 0 0 0 0 0 3.78 ...
##  $ NC0.5          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NC1.0          : num  0 0 0 0 0 ...
##  $ NC2.5          : num  0 0 0 0 0 0 0 0 0 2.78 ...
##  $ CNT            : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Fire.Alarm     : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
df <- df[,c(3,4,5,6,7,8,9,10,11,12,13,14,16)]
df$Fire.Alarm <- factor(df$Fire.Alarm)
str(df)
```

```
## 'data.frame':    62630 obs. of  13 variables:
##  $ Temperature.C.: num  20 20 20 20 20.1 ...
##  $ Humidity...    : num  57.4 56.7 56 55.3 54.7 ...
##  $ TVOC.ppb.      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ eCO2.ppm.      : int  400 400 400 400 400 400 400 400 400 400 ...
##  $ Raw.H2         : int  12306 12345 12374 12390 12403 12419 12432 12439 12448 12453 ...
```

```
##  $ Raw.Ethanol   : int  18520 18651 18764 18849 18921 18998 19058 19114 19155 19195 ...
##  $ Pressure.hPa. : num  940 940 940 940 940 ...
##  $ PM1.0         : num  0 0 0 0 0 0 0 0 0 0.9 ...
##  $ PM2.5         : num  0 0 0 0 0 0 0 0 0 3.78 ...
##  $ NC0.5         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NC1.0         : num  0 0 0 0 0 ...
##  $ NC2.5         : num  0 0 0 0 0 0 0 0 0 2.78 ...
##  $ Fire.Alarm    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Check for null values.

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

```
## Temperature.C.    Humidity...      TVOC.ppb.      eCO2.ppm.         Raw.H2
##              0              0              0              0              0
##    Raw.Ethanol  Pressure.hPa.          PM1.0          PM2.5          NC0.5
##              0              0              0              0              0
##          NC1.0          NC2.5     Fire.Alarm
##              0              0              0
```

#A. #Divide into 80/20 train/test.

```
set.seed(12345)
i <- sample(1:nrow(df), nrow(df)*.8, replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

#B. & C. #Data Exploration & Graphs

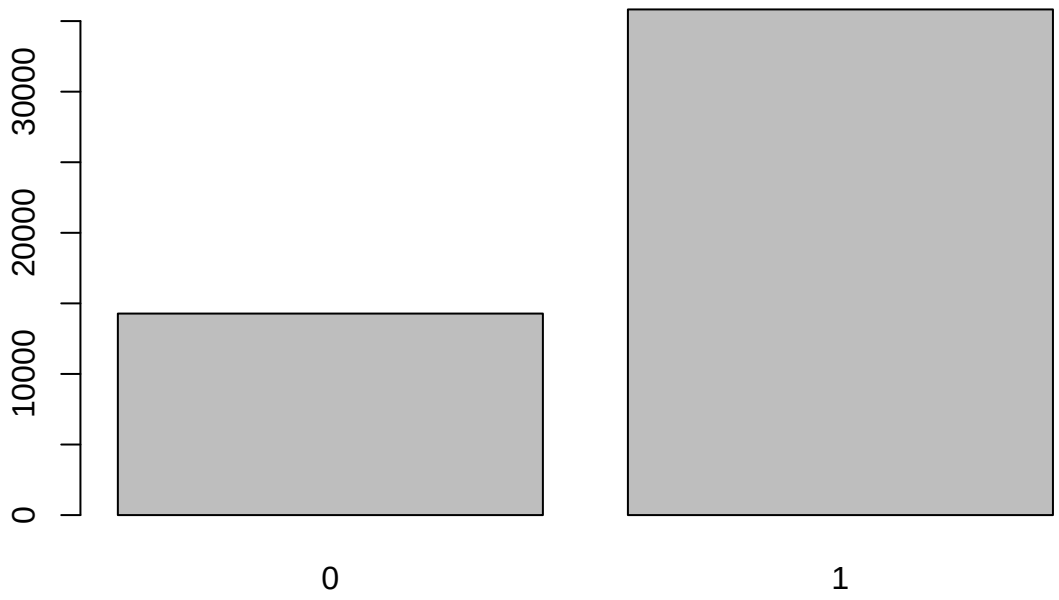
Explore

Exploring distribution of Fire.Alarm

```
summary(train$Fire.Alarm)
```

```
##     0     1
## 14278 35826
```

```
plot(train$Fire.Alarm)
```

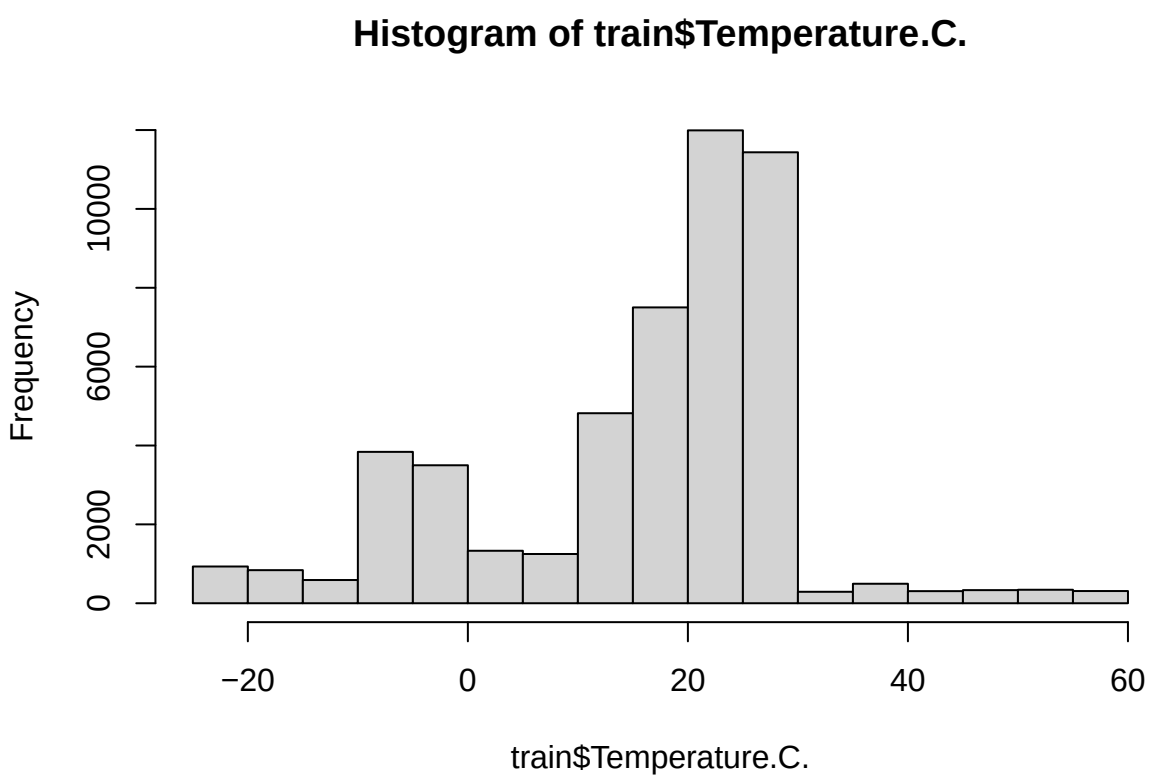


Exploring distribution of temperature

```
summary(train$Temperature.C.)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -22.01   10.91   20.15   15.93   25.41   59.93
```

```
hist(train$Temperature.C.)
```

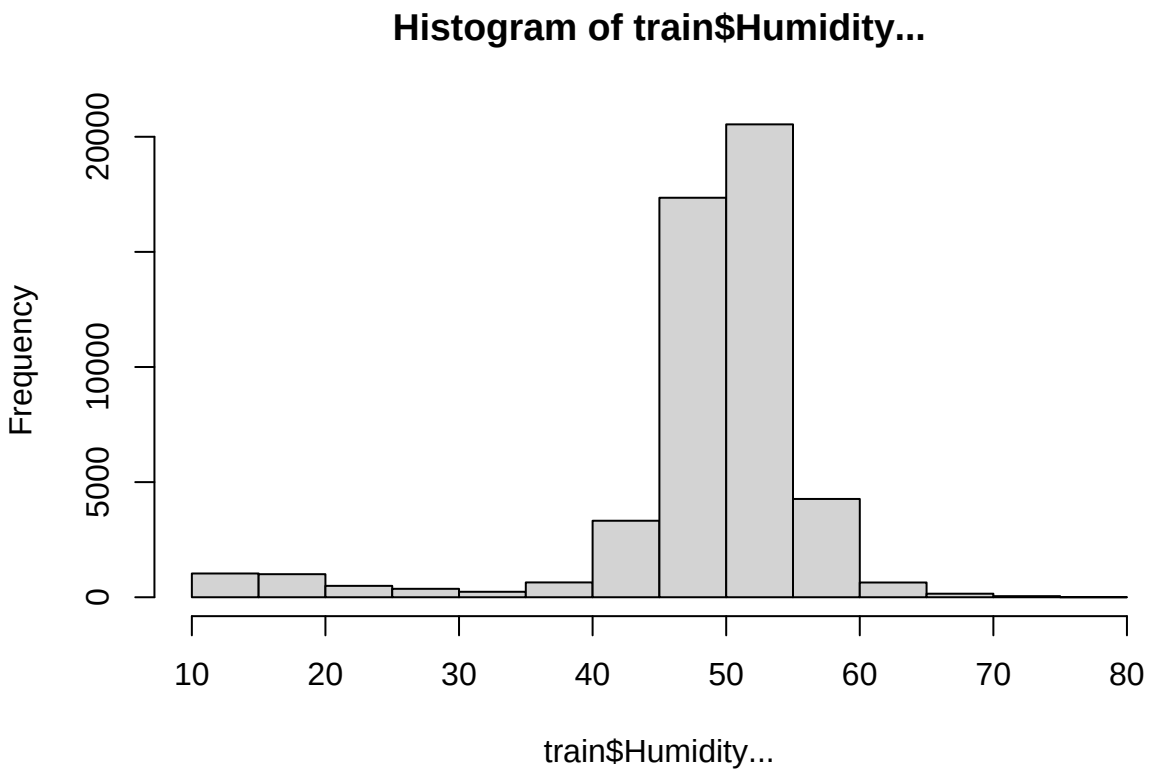


Exploring distribution of humidity

```
summary(train$Humidity...)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.74   47.54   50.16   48.57   53.25   75.20
```

```
hist(train$Humidity...)
```

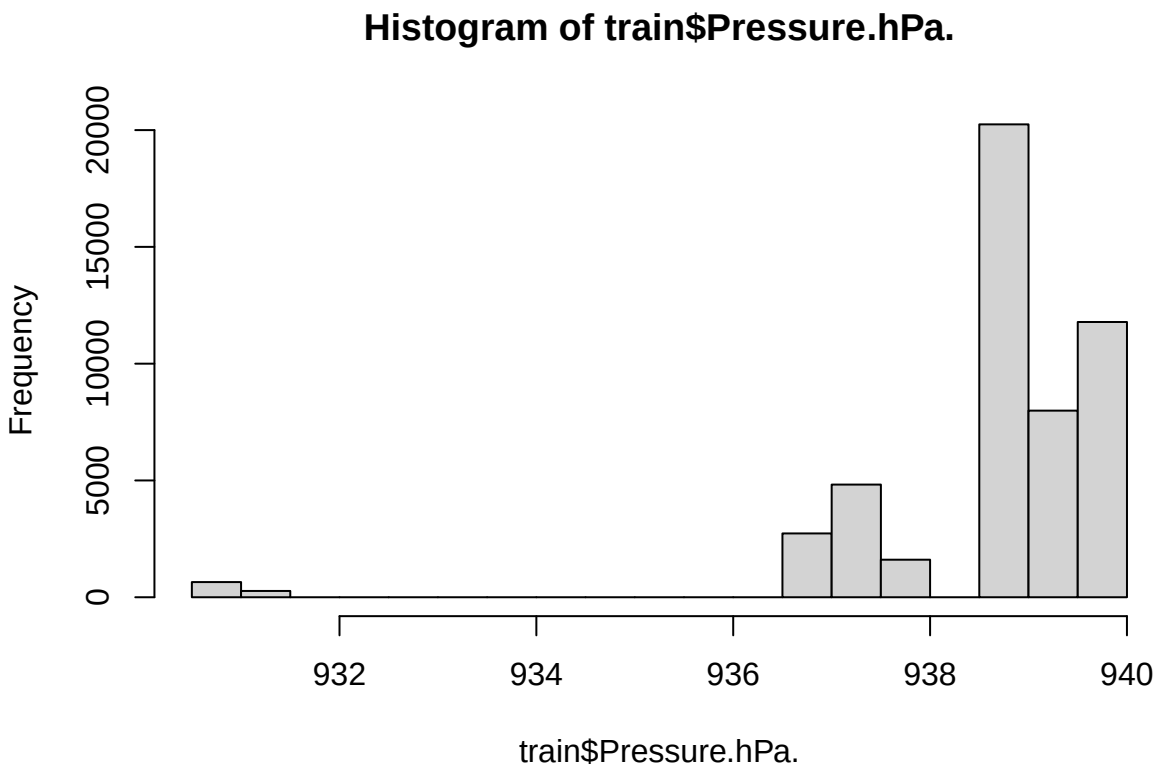


Exploring distribution of pressure

```
summary(train$Pressure.hPa.)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   930.9   938.7   938.8   938.6   939.4   939.9
```

```
hist(train$Pressure.hPa.)
```

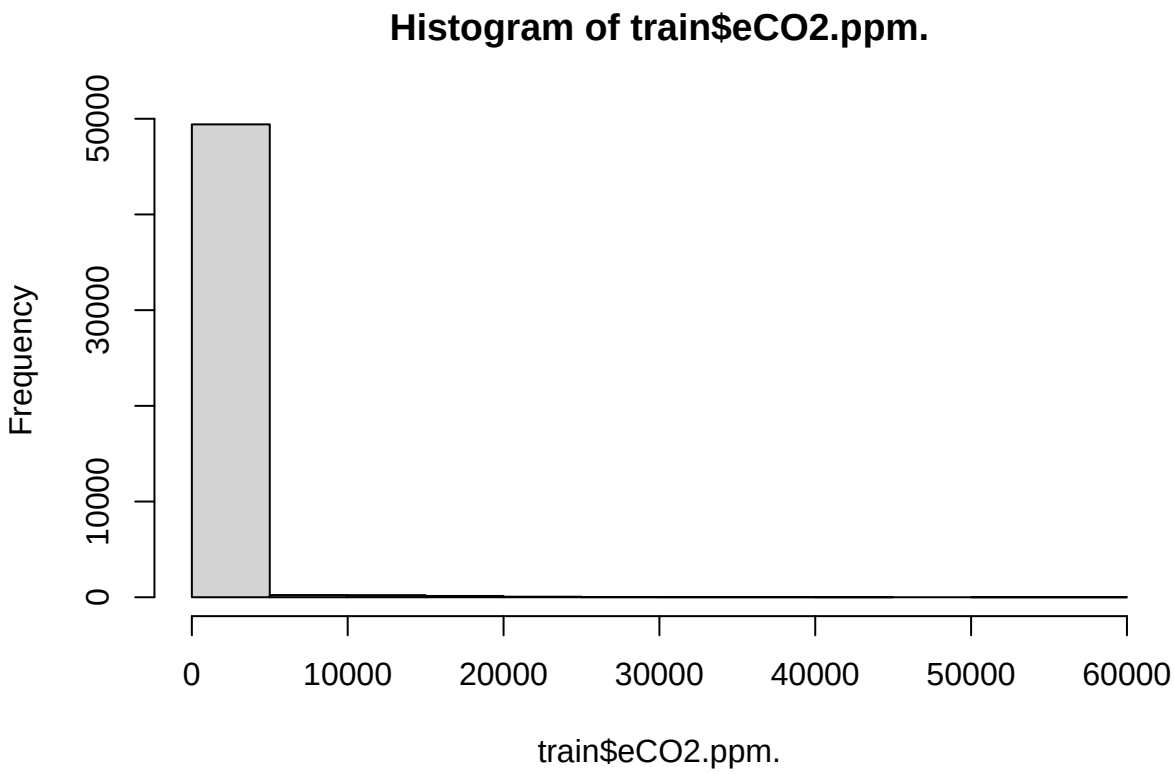


Exploring distribution of eCO2

```
summary(train$eCO2.ppm.)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     400     400     400     661     438   60000
```

```
hist(train$eCO2.ppm.)
```



#D. #Logistic Regression Model Explanation: Deviance Residuals shows a summary of the deviance residuals of the model. Coefficients is similar to Linear regression except it quantifies the difference in the log odds. T value is replaced by Z value which still indicates how significant the variable is, and the p value is the same where it indicates the probability of obtaining the z value. Null deviance measures lack of fit while looking at only the intercept and residual deviance measures lack of fit using the variables. A large difference means that the variables are significant and is similar to the F-statistic. AIC is used to compare algorithms with a lower AIC being preferred. Fisher scoring iterations is the number of iterations the algorithm used to produce the model. This model took 15 Fisher scoring iterations to produce a model with an AIC of 20743. There is a large difference between null and residual difference indicating that the variables used are significant.PM1.0 and PM2.5 were found to be much less significant than the rest of the variables and could possibly be dropped to improve the model.

```
glm1 <- glm(Fire.Alarm~., data=train, family="binomial")
summary(glm1)
```

```
##
## Call:
## glm(formula = Fire.Alarm ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.0023   0.1493   0.3125   4.7619
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.961e+03  3.253e+01  60.284  < 2e-16 ***
```

```
## Temperature.C. -7.877e-02  1.843e-03 -42.737  < 2e-16 ***
## Humidity...     1.553e-01  4.182e-03  37.130  < 2e-16 ***
## TVOC.ppb.      -3.219e-03  8.580e-05 -37.517  < 2e-16 ***
## eCO2.ppm.       4.248e-03  1.344e-04  31.611  < 2e-16 ***
## Raw.H2          1.446e-02  3.010e-04  48.038  < 2e-16 ***
## Raw.Ethanol    -1.519e-02  2.578e-04 -58.910  < 2e-16 ***
## Pressure.hPa.  -1.973e+00  3.380e-02 -58.385  < 2e-16 ***
## PM1.0           7.826e+00  6.133e+00   1.276 0.201954
## PM2.5          -6.711e+00  6.165e+00  -1.089 0.276305
## NC0.5          -6.332e+00  1.204e+00  -5.260 1.44e-07 ***
## NC1.0           4.117e+01  1.129e+01   3.646 0.000267 ***
## NC2.5          -6.020e+01  1.464e+01  -4.112 3.92e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 59883  on 50103  degrees of freedom
## Residual deviance: 20717  on 50091  degrees of freedom
## AIC: 20743
##
## Number of Fisher Scoring iterations: 15
```

#E. #Naive Bayes Model Explanation: A-priori is the odds prior to the training. Conditional probabilities shows the mean and standard deviation of each class for continuous variables and the breakdown of probability for each variable for discrete variables. For this data set approximately 72% of the observations were 1 and 28% were 0. There were no discrete variable so the conditional probabilities only showed the means and standard devations.

```
library(e1071)
nb1 <- naiveBayes(Fire.Alarm~., data=train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.2849673 0.7150327
##
## Conditional probabilities:
##    Temperature.C.
## Y       [,1]     [,2]
##   0 19.66126 14.93973
##   1 14.44886 13.85415
##
##    Humidity...
## Y       [,1]      [,2]
##   0 43.00608 11.967139
##   1 50.78540  5.907294
```

```
##
##    TVOC.ppb.
## Y        [,1]       [,2]
##   0 4414.3555 13960.4423
##   1  882.5488   558.0954
##
##    eCO2.ppm.
## Y       [,1]     [,2]
##   0 935.4802 2844.699
##   1 551.6825 1246.717
##
##    Raw.H2
## Y       [,1]     [,2]
##   0 12901.49 426.9057
##   1 12960.85 167.4425
##
##    Raw.Ethanol
## Y       [,1]     [,2]
##   0 20093.66 940.3744
##   1 19622.90 307.3232
##
##    Pressure.hPa.
## Y       [,1]     [,2]
##   0 938.1023 1.242415
##   1 938.8397 1.304074
##
##    PM1.0
## Y        [,1]      [,2]
##   0 248.07646 1393.2011
##   1  35.22077  583.6592
##
##    PM2.5
## Y        [,1]     [,2]
##   0 424.29364 2729.987
##   1  77.31074 1497.436
##
##    NC0.5
## Y       [,1]     [,2]
##   0 1288.946 6946.211
##   1  140.106 2065.790
##
##    NC1.0
## Y       [,1]     [,2]
##   0 465.4410 3039.898
##   1  86.4982 1694.841
##
##    NC2.5
## Y        [,1]      [,2]
##   0 167.68324 1394.3343
##   1  40.37803  903.8945
```

#F. #Predict and Evaluate Logistic regression performed better with an accuracy of 0.9023 and a kappa of 0.7535 indicating good agreement. Naive Bayes received an accuracy of 0.7649 and a kappa of 0.275 indicating fair agreement. Logistic regression performs better than Naive Bayes on larger data sets. This

data set contained over 60 thousand observations so it is likely that this performance difference contributed to Logistic regression outperforming Naive Bayes.

```
probs <- predict(glm1, newdata=test, type="response")
p1 <- ifelse(probs>0.5, 1, 0)
table(p1, test$Fire.Alarm)
```

```
##
## p1     0    1
##   0 2793  422
##   1  802 8509
```

```
p2 <- predict(nb1, newdata=test, type="class")
table(p2, test$Fire.Alarm)
```

```
##
## p2     0    1
##   0  873  223
##   1 2722 8708
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
confusionMatrix(as.factor(p1), reference=test$Fire.Alarm)
```

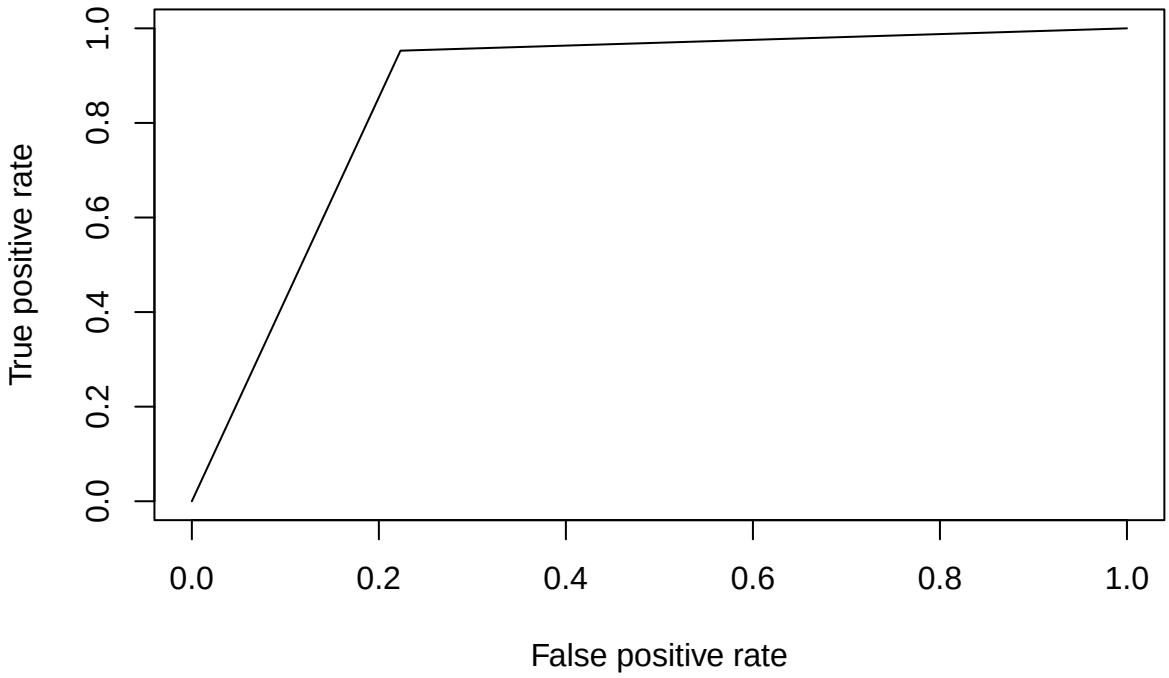
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2793  422
##          1  802 8509
##
##                Accuracy : 0.9023
##                  95% CI : (0.8969, 0.9074)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7535
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7769
##             Specificity : 0.9527
##          Pos Pred Value : 0.8687
##          Neg Pred Value : 0.9139
##              Prevalence : 0.2870
##          Detection Rate : 0.2230
##    Detection Prevalence : 0.2567
```

```
##       Balanced Accuracy : 0.8648
##
##        'Positive' Class : 0
##
```

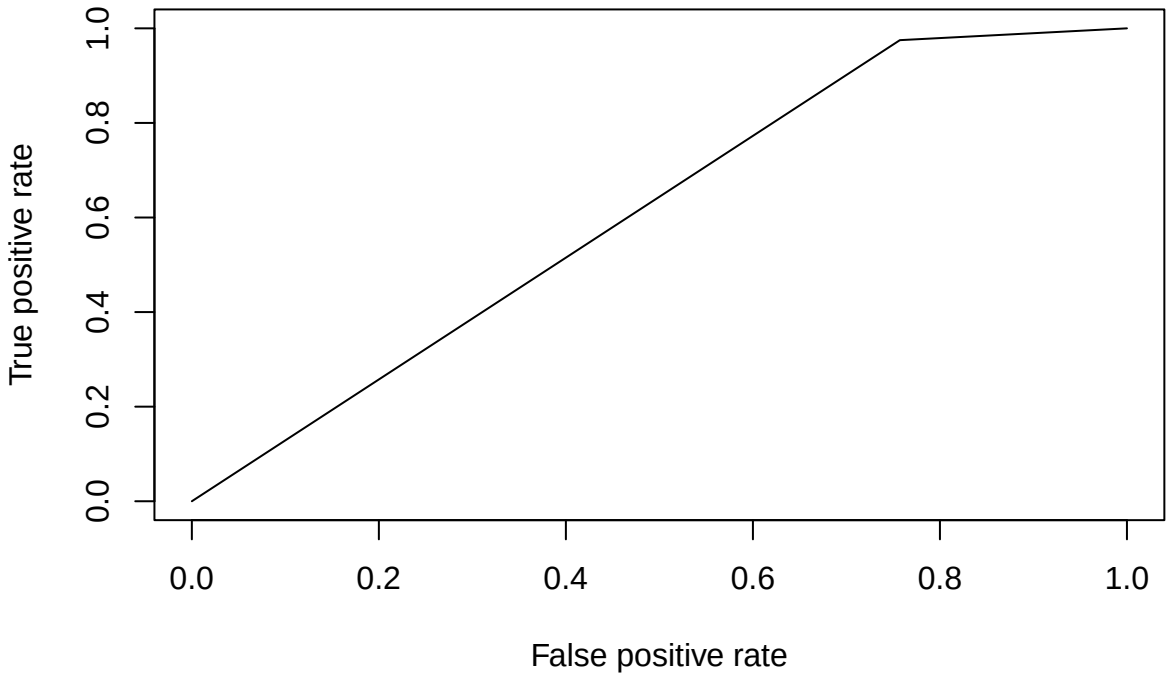
```
confusionMatrix(as.factor(p2), reference=test$Fire.Alarm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0  873  223
##          1 2722 8708
##
##                Accuracy : 0.7649
##                  95% CI : (0.7574, 0.7723)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.275
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.2428
##             Specificity : 0.9750
##          Pos Pred Value : 0.7965
##          Neg Pred Value : 0.7619
##              Prevalence : 0.2870
##          Detection Rate : 0.0697
##    Detection Prevalence : 0.0875
##       Balanced Accuracy : 0.6089
##
##        'Positive' Class : 0
##
```

```
library(ROCR)
pr1 <- prediction(p1, test$Fire.Alarm)
prf1 <- performance(pr1, measure = "tpr", x.measure = "fpr")
plot(prf1)
```




```
p02 <- ifelse(p2=="1", 1, 0)
pr2 <- prediction(p02, test$Fire.Alarm)
prf2 <- performance(pr2, measure = "tpr", x.measure = "fpr")
plot(prf2)
```




```
auc <- performance(pr1, measure="auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8648306
```

```
auc <- performance(pr2, measure="auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.608934
```

#G. #Strengths and Weaknesses Naive Bayes is simple and works well on smaller data sets at the expense of performing worse on larger data sets and has a higher bias and lower variance than logistic regression. Logistic regression is simple and works better than Naive Bayes on larger data sets. It has higher variance and lower bias than Naive Bayes.

#H. #Classification metrics used Accuracy- Is simple and easy to understand but ignores many factors. Simple # correct over # total. Sensitivity- The true positive rate. Quantifies how well it was identified but not much else. Specificity-True negative rate. Quantifies how well it was identified but not much else. Kappa- Adjusts accuracy to account for random chance with values close to 0 being poor agreement and values close to 1 being very good agreement. ROC curve-Visualizes performance of the algorithm with a more vertical line being desirable. Shows how well it fits the data but does not have an easy number to point to. AUC- Measures the predictive value of the model with 0.5 being poor and 1 being perfect. AUC is the easy to point to number from the ROC curve but lacks any other information.