Classification

Charles Wallis

Logistic Regression (Classification)

Classification is a type of supervised learning in machine learning. it is the process of classifying a given data. When data is given, the model determines and predicts which class the data belongs to. There are various algorithms for classification

Logistic regression is a general and effective classification algorithm. It is a powerful algorithm for classifying categorical data, although it is often confusing as the word regression is often used only in regression analysis. It was created based on the linear relationship between the independent variable and the dependent variable.

Naive Bayes The Naive Bayes algorithm is a classification algorithm based on Bayes theorem. P(A|B) means the probability that event A occurs when event B occurs as a conditional probability. The Naive Bayes algorithm is a principle of predicting how this data will behave based on previous events when new data comes in after creating a model that has event data

Data

To perform Classification, we will import a Smoke Detection data set

Import File

```
df <- read.csv("smoke.csv")
df$Fire.Alarm <- factor(df$Fire.Alarm)

X <- vector(mode="integer", length=nrow(df))
count <- 1
for (i in df$eCO2.ppm.) {
if(i >= 30){
X[count] <- 1
}
else{
X[count] <- 0
}
count <- count + 1
}
df$eCO2.ppm. <- as.factor(X)
```

Divide into 80/20 train/test

```
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*0.8, replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Summary of Asteroids

```
summary(train)
```

```
##        X              UTC            Temperature.C.      Humidity
##  Min.   :    0   Min.   :1.655e+09   Min.   :-22.01   Min.   :10.74
##  1st Qu.:15725   1st Qu.:1.655e+09   1st Qu.: 11.13   1st Qu.:47.54
##  Median :31361   Median :1.655e+09   Median : 20.14   Median :50.16
##  Mean   :31339   Mean   :1.655e+09   Mean   : 16.00   Mean   :48.54
##  3rd Qu.:46971   3rd Qu.:1.655e+09   3rd Qu.: 25.41   3rd Qu.:53.24
##  Max.   :62629   Max.   :1.655e+09   Max.   : 59.93   Max.   :75.20
##    TVOC.ppb.     eCO2.ppm.     Raw.H2       Raw.Ethanol    Pressure.hPa.
##  Min.   :    0   1:50104    Min.   :10668   Min.   :15317   Min.   :930.9
##  1st Qu.:  129              1st Qu.:12830   1st Qu.:19435   1st Qu.:938.7
##  Median :  981              Median :12923   Median :19500   Median :938.8
##  Mean   : 1954              Mean   :12942   Mean   :19753   Mean   :938.6
##  3rd Qu.: 1189              3rd Qu.:13109   3rd Qu.:20078   3rd Qu.:939.4
##  Max.   :60000              Max.   :13803   Max.   :21410   Max.   :939.9
##      PM1.0              PM2.5              NC0.5              NC1.0
##  Min.   :    0.00   Min.   :    0.00   Min.   :    0.00   Min.   :    0.00
##  1st Qu.:    1.28   1st Qu.:    1.34   1st Qu.:    8.82   1st Qu.:    1.38
##  Median :    1.81   Median :    1.88   Median :   12.46   Median :    1.94
##  Mean   :  102.00   Mean   :  187.87   Mean   :  496.28   Mean   :  207.42
##  3rd Qu.:    2.10   3rd Qu.:    2.18   3rd Qu.:   14.42   3rd Qu.:    2.25
##  Max.   :14318.17   Max.   :45432.26   Max.   :61482.03   Max.   :51914.68
##      NC2.5                CNT        Fire.Alarm
##  Min.   :    0.000   Min.   :    0   0:14289
##  1st Qu.:    0.033   1st Qu.: 3631   1:35815
##  Median :    0.044   Median : 9348
##  Mean   :   81.966   Mean   :10515
##  3rd Qu.:    0.051   3rd Qu.:17163
##  Max.   :30026.438   Max.   :24993
```

Dimensions of the Data Frame, There are 50104 rows, and 16 columns

```
dim(train)
```

```
## [1] 50104    16
```

Structure of mushroom

```
str(train)
```

```
## 'data.frame':    50104 obs. of  16 variables:
##  $ X             : int  40783 40853 41963 15240 33701 35715 60518 54873 17486 15219 ...
##  $ UTC           : int  1654777132 1654777202 1654778312 1654748571 1654770050 1654772064 1655127940
##  $ Temperature.C.: num  26.7 27 24.6 12.6 19.5 ...
##  $ Humidity      : num  48.8 47.6 53.2 53.5 57.6 ...
##  $ TVOC.ppb.     : int  1117 1163 1176 1130 331 948 0 0 1163 1132 ...
##  $ eCO2.ppm.     : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Raw.H2        : int  12882 12884 12895 12873 13092 12793 13333 13430 12897 12871 ...
##  $ Raw.Ethanol   : int  19452 19452 19438 19454 19928 19492 21200 21252 19438 19454 ...
##  $ Pressure.hPa. : num  939 939 939 939 939 ...
##  $ PM1.0         : num  1.92 1.53 1.65 1.95 0.31 2.21 1.94 2.2 1.78 1.8 ...
##  $ PM2.5         : num  2 1.59 1.72 2.03 0.32 2.3 2.02 2.29 1.85 1.87 ...
##  $ NC0.5         : num  13.24 10.51 11.39 13.42 2.11 ...
##  $ NC1.0         : num  2.065 1.638 1.775 2.092 0.329 ...
##  $ NC2.5         : num  0.047 0.037 0.04 0.047 0.007 0.054 0.047 0.053 0.043 0.044 ...
##  $ CNT           : int  15789 15859 16969 15240 8707 10721 3632 3731 17486 15219 ...
##  $ Fire.Alarm    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 1 2 2 ...
```

First 5 rows of Smoke Detector Data

```
head(train, n=5)
```

```
##           X        UTC Temperature.C. Humidity TVOC.ppb. eCO2.ppm. Raw.H2
## 40784 40783 1654777132         26.670    48.83      1117         1  12882
## 40854 40853 1654777202         26.980    47.62      1163         1  12884
## 41964 41963 1654778312         24.590    53.23      1176         1  12895
## 15241 15240 1654748571         12.554    53.50      1130         1  12873
## 33702 33701 1654770050         19.510    57.59       331         1  13092
##       Raw.Ethanol Pressure.hPa. PM1.0 PM2.5 NC0.5 NC1.0 NC2.5   CNT Fire.Alarm
## 40784       19452       938.806  1.92  2.00 13.24 2.065 0.047 15789          1
## 40854       19452       938.785  1.53  1.59 10.51 1.638 0.037 15859          1
## 41964       19438       938.780  1.65  1.72 11.39 1.775 0.040 16969          1
## 15241       19454       938.813  1.95  2.03 13.42 2.092 0.047 15240          1
## 33702       19928       939.290  0.31  0.32  2.11 0.329 0.007  8707          1
```

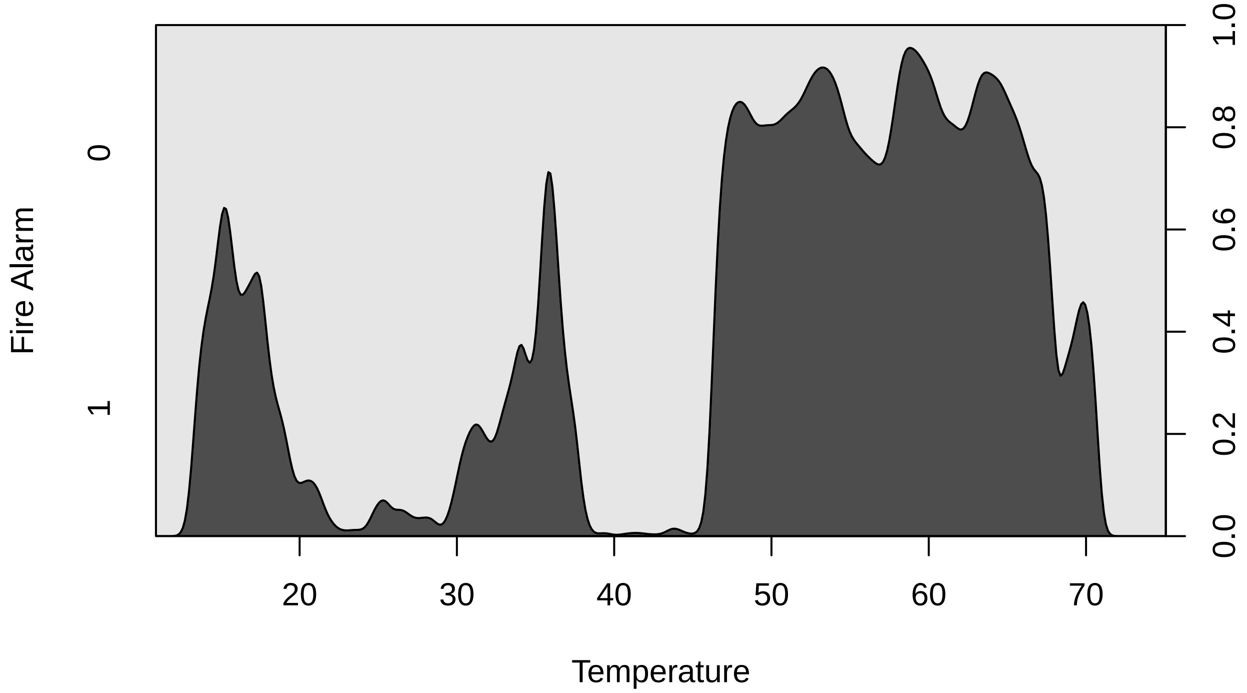
Last 5 rows of Smote Detector Data

```
tail(train, n=5)
```

```
##           X        UTC Temperature.C. Humidity TVOC.ppb. eCO2.ppm. Raw.H2
## 20440 20439 1654753770         11.557    51.62      1174         1  12970
## 46667 46666 1654783015         26.650    48.70      1343         1  12965
## 29662 29661 1654766010         20.060    53.07        62         1  13239
## 47321 47320 1654783669         26.370    50.47      1346         1  12965
## 40084 40083 1654776432         27.140    47.10      1067         1  12867
##       Raw.Ethanol Pressure.hPa. PM1.0 PM2.5 NC0.5 NC1.0 NC2.5   CNT Fire.Alarm
## 20440       19433       938.765  2.17  2.25 14.90 2.324 0.052 20439          1
## 46667       19398       938.704  2.39  2.49 16.47 2.568 0.058 21672          1
## 29662       20157       939.665  2.22  2.31 15.29 2.384 0.054  4667          1
## 47321       19390       938.728  2.10  2.18 14.43 2.251 0.051 22326          1
## 40084       19447       938.818  1.97  2.04 13.54 2.111 0.048 15089          1
```

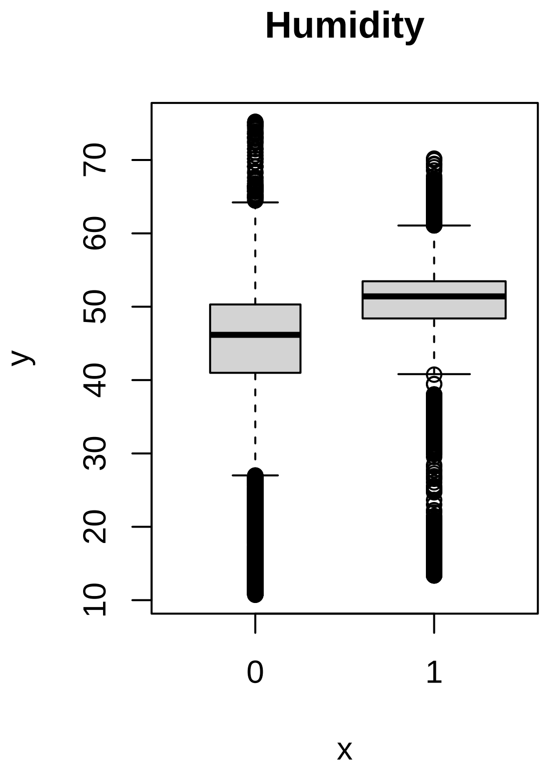
Plot 1, Fire Alarms vs Temperature

```
cdplot(train$Fire.Alarm ~ train$Humidity, ylab = "Fire Alarm", xlab = "Temperature")
```



# Plot 2, Humidity when fire alarm is off / on

```
par(mfrow=c(1,2))
plot(train$Fire.Alarm, train$Humidity, data=train, main="Humidity", varwidth=TRUE)
```



Build a logistic regression model and output the summary.

```r
glm1 <- glm(eCO2.ppm. ~ ., data=train, family="binomial")
```

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

```r
summary(glm1)
```

```
## 
## Call:
## glm(formula = eCO2.ppm. ~ ., family = "binomial", data = train)
## 
## Deviance Residuals: 
##        Min          1Q      Median          3Q         Max  
## -2.409e-06  -2.409e-06  -2.409e-06  -2.409e-06  -2.409e-06  
## 
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -2.657e+01  3.773e+07       0        1
## X               1.546e-17  1.767e-01       0        1
## UTC             1.401e-19  2.264e-02       0        1
## Temperature.C.  7.680e-15  1.636e+02       0        1
## Humidity       -1.808e-14  3.285e+02       0        1
## TVOC.ppb.      -3.924e-17  4.965e-01       0        1
## Raw.H2         -2.316e-16  1.030e+01       0        1
## Raw.Ethanol    -4.594e-16  8.344e+00       0        1
## Pressure.hPa.   1.740e-13  2.167e+03       0        1
## PM1.0          -1.001e-10  5.372e+05       0        1
## PM2.5           3.281e-11  5.460e+05       0        1
## NC0.5           2.458e-12  9.740e+04       0        1
## NC1.0           4.765e-11  9.187e+05       0        1
## NC2.5          -8.615e-11  1.168e+06       0        1
## CNT            -1.402e-17  4.488e-01       0        1
## Fire.Alarm1     4.378e-13  5.710e+03       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 0.0000e+00  on 50103  degrees of freedom
## Residual deviance: 2.9068e-07  on 50088  degrees of freedom
## AIC: 32
##
## Number of Fisher Scoring iterations: 25
```

So in this summary we can see immediately that there are no asterisks to indicate a good predictor of eCO2.ppm. The null deviance is near 0 with a high degree of freedom, and therefore the linear model was not able to find a fit

The residual deviance is very close to the null device, yet higher.

Build a naïve Bayes model and output what the model learned.

```
library(e1071)
nb1 <- naiveBayes(eCO2.ppm. ~ ., data=train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
## 1
## 1
##
## Conditional probabilities:
##    X
## Y       [,1]     [,2]
##   1 31339.27 18056.53
##
##    UTC
## Y         [,1]     [,2]
##   1 1654791796 109540.1
##
##    Temperature.C.
## Y      [,1]     [,2]
##   1 16.0039 14.36201
##
##    Humidity
## Y       [,1]     [,2]
##   1 48.53929 8.868838
##
##    TVOC.ppb.
## Y       [,1]     [,2]
##   1 1954.161 7851.776
##
##    Raw.H2
## Y       [,1]     [,2]
##   1 12942.22 273.2519
##
##    Raw.Ethanol
## Y       [,1]     [,2]
##   1 19753.17 610.6454
##
##    Pressure.hPa.
## Y       [,1]     [,2]
##   1 938.6281 1.333456
##
##    PM1.0
## Y       [,1]     [,2]
##   1 102.0024 933.2956
##
##    PM2.5
## Y       [,1]     [,2]
##   1 187.8704 2000.447
##
##    NC0.5
## Y       [,1]     [,2]
##   1 496.2817 4296.584
##
##    NC1.0
## Y       [,1]     [,2]
##   1 207.4175 2241.599
##
##    NC2.5
## Y       [,1]   [,2]
##   1 81.96595 1094.4
##
##    CNT
## Y       [,1]     [,2]
##   1 10514.92 7599.104
##
##    Fire.Alarm
## Y           0         1
##   1 0.2851868 0.7148132
```

In this Naive Bayes Model for eCO2, the A-priori probability says that there is a 1 chance of being 1 (100%) Then it gives predictors for for different quantities distributions, so in the last row it can be seen that there was a 0.285 in non-alarms and 0.714 in fire alarms

Using these two classification models models, predict and evaluate on the test data using all of the classification metrics discussed in class. Compare the results and indicate why you think these results happened.

```r
probs1 <- predict(glm1, newdata = test, type = "response")
pred1 <- ifelse(probs1>0.5, 1, 0)
acc1 <- mean(pred1==test$eCO2.ppm.)
print(paste("Accuracy = ", acc1))
```

```
## [1] "Accuracy =  0"
```

And as a proof that the model did not find a fit, the accuracy is shown as 0

strengths and weaknesses of Naïve Bayes and Logistic Regression.

Strengths of Naive Bayes and Logistic Regression when used correctly includes being able to classify and separate data using a line that goes between the two data. Naive Bayes works well with many different sizes of data, and the results are easy to understand. Logistic regression also provides coefficient for each predictor in the target variable which makes us know what is making the most and least influence in the target.

The weakness of logistic regression is that it is prone to under fitting because it assumes a linear line between variables, and there are difficulties trying to computer for complex relationships. Naive Bayes assumes that all columns of the data are independent so it is not realistic, and can't be used to compute a probability in a test set since it was not in the training set.

benefits, drawbacks of each of the classification metrics used, and briefly describe what each metric tells you.

Accuracy is how correct the model is, essentially how well it performs with the given data and test data. this metric is the easiest to understand, but it may not always be a good measurement on data sets with bias

Sensitivity is how often the model gave a (correct) true positive result, and this is an important metric to measure how well the model identifies a positive result

Specificity is the opposite of sensitivity, it is how often the model gave a correct true negative result, to measure how well the model identifies a negative result.