Classification

Muhammad Shariq Azeem

09/24/2022

Classification:

Linear models, built for classification, creates a boundary between two types of observations, and decides whether an observation is a part of Class A or Class B, depending on its attributes. There are various different algorithms that allow us to create linear models for classification.

In this document, I am going to look at the the Logistic Regression and the Naive Bayes algorithms. I will also talk about the Strengths and Weaknesses of both of these model later in this document.

Data:

For this assignment, I selected a data set that contains information about the room environment, such as, room temperature, humidity, CO2 levels, etc. I need to use that information to decide if the room is occupied or not.

Source for the data: https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+#

Note: The data sets provided through the link were not meeting the "at least 10K rows" requirement, so I used Excel to combine two data sets into one.

Cleaning the data:

- Got rid of the data column because I don't need that for my model.
- Converted 'Occupancy' attribute to a factor.

```
df <- read.csv("C:/Users/shari/Downloads/data.csv", header=T)
df <- df[,c(2,3,4,5,6,7)]
df$Occupancy <- factor(df$Occupancy)
```

Step A: Divide data into train and test.

```
set.seed(1234)
i <- sample(1:nrow(df), 0.80*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Step B: Five Data Exploration functions on the training data.

1- names():

List the names of all the attributes contained in the data set.

```
names(train)
```

```
## [1] "Temperature"   "Humidity"      "Light"         "CO2"
## [5] "HumidityRatio" "Occupancy"
```

2- dim():

Print the number of rows and the number of columns in the data set.

```
dim(train)
```

```
## [1] 9933    6
```

3- head():

Print the first n rows of the data set.

```
head(train)
```

```
##      Temperature Humidity Light      CO2 HumidityRatio Occupancy
## 7452    20.89000   24.695     0 571.5000   0.003768244         0
## 8016    21.39000   27.790     0 566.0000   0.004376975         0
## 7162    22.63333   24.890   732 587.6667   0.004228062         1
## 8086    21.79000   28.050     0 594.5000   0.004528488         0
## 7269    21.00000   25.390    14 522.0000   0.003901421         0
## 9196    20.60000   26.890     0 498.0000   0.004032247         0
```

4- str():

List the structure of each column of the data set.

```
str(train)
```

```
## 'data.frame':    9933 obs. of  6 variables:
##  $ Temperature  : num  20.9 21.4 22.6 21.8 21 ...
##  $ Humidity     : num  24.7 27.8 24.9 28.1 25.4 ...
##  $ Light        : num  0 0 732 0 14 0 0 454 433 0 ...
##  $ CO2          : num  572 566 588 594 522 ...
##  $ HumidityRatio: num  0.00377 0.00438 0.00423 0.00453 0.0039 ...
##  $ Occupancy    : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 2 2 1 ...
```

5- summary():

Print basic statistics calculated from the values contained in the data set.

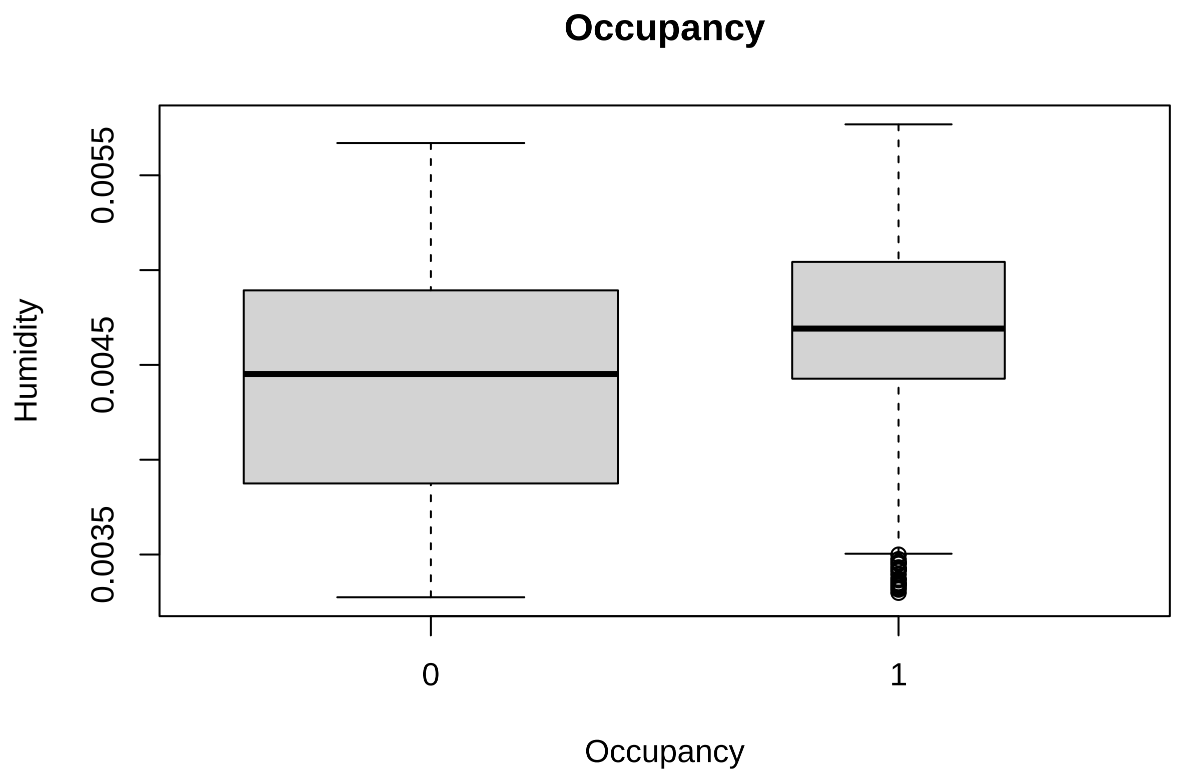
```
summary(train)
```

```
##   Temperature       Humidity         Light             CO2
##  Min.   :19.50   Min.   :21.86   Min.   :   0.0   Min.   : 427.5
##  1st Qu.:20.39   1st Qu.:25.39   1st Qu.:   0.0   1st Qu.: 528.0
##  Median :20.79   Median :28.63   Median :   0.0   Median : 632.7
##  Mean   :21.09   Mean   :28.92   Mean   : 138.1   Mean   : 745.6
##  3rd Qu.:21.67   3rd Qu.:31.86   3rd Qu.: 399.0   3rd Qu.: 857.0
##  Max.   :24.41   Max.   :39.50   Max.   :1581.0   Max.   :2076.5
##  HumidityRatio      Occupancy
##  Min.   :0.003275   0:7512
##  1st Qu.:0.003984   1:2421
##  Median :0.004512
##  Mean   :0.004468
##  3rd Qu.:0.004940
##  Max.   :0.005769
```

Step C: 2 Informative Graphs on the training data.

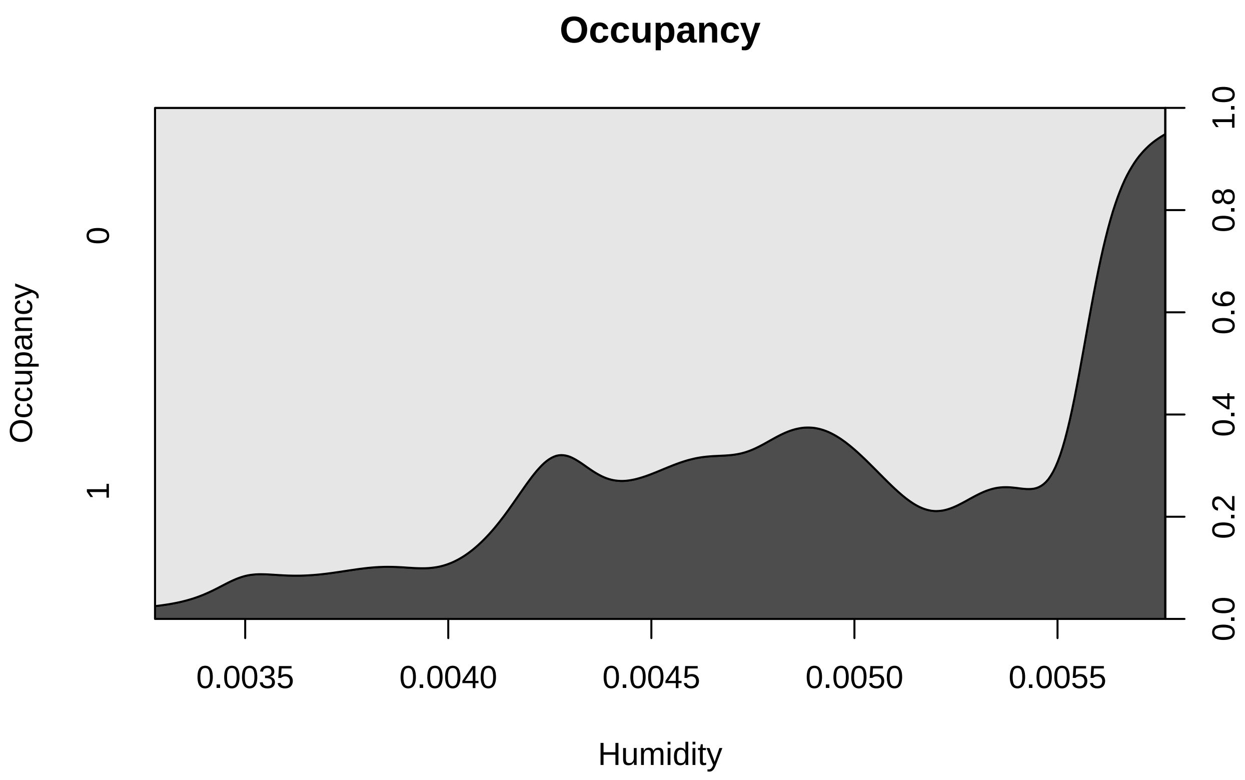
1- plot():

```
plot(HumidityRatio~Occupancy, data=train, main="Occupancy",
     xlab = "Occupancy", ylab = "Humidity", varwidth = TRUE)
```



2- cdplot():

```
cdplot(train$Occupancy~train$HumidityRatio, main="Occupancy",
       ylab = "Occupancy", xlab = "Humidity",)
```



Step D: Build a logistic regression model and print its summary.

Explanation:

First thing I notice in the summary is that all of the predictors are marked with three asterisks and have very low p-value, which means that they are considered very good predictors for my target variable. Next, their coefficients give me the log odds probability of the target variable.

Next, the null deviance and the residual deviance give me the lack of fit of the model based on the intercept, and the lack of fit of the entire model, respectively. Over here, we need to see the Residual Deviance having a much lower value than the Null Deviance, and I do see that. This means that my model, with the predictors, explains the data much better than a null model would.

Lastly, the AIC value is used to compare models, where a model with the lowest AIC is considered to be the best in representing the data. I tried making my model with a different combination of the attributes, and I got the lowest AIC when I used all of the attributes to predict the Occupancy of a room. That means that the model below is the best model that can be made for the given data.

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

```
##
## Call:
## glm(formula = Occupancy ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7642  -0.0375  -0.0239  -0.0131   4.4091
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.409e+02  2.375e+01   5.934 2.96e-09 ***
## Temperature    -7.268e+00  1.107e+00  -6.567 5.13e-11 ***
## Humidity       -3.642e+00  6.538e-01  -5.571 2.54e-08 ***
## Light           2.391e-02  8.591e-04  27.830  < 2e-16 ***
## CO2             2.933e-03  4.207e-04   6.974 3.09e-12 ***
## HumidityRatio   2.411e+04  4.126e+03   5.845 5.06e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11032.48  on 9932  degrees of freedom
## Residual deviance:   994.61  on 9927  degrees of freedom
## AIC: 1006.6
##
## Number of Fisher Scoring iterations: 9
```

Step E: Build a Naive Bayes model and output what the model learned.

Explanation:

The formula for the Bayes Theorem is: Posterior = (likelihood*prior)/(marginal).

Looking at the output of the summary, I can see that the model learned the prior values, marked as A-priori. Prior for Occupied is .244, and prior for Un-Occupied is .756

Similarly, the model also learned the likelihood of each attribute. Since all of my variables are continuous, it gives me the mean and standard deviation values.

- Mean Temperature is 20.7 for Unoccupied, and 22.1 for Occupied.
- Mean Humidity is 29 for Unoccupied, and 28.6 for Occupied.
- Mean Light is 23.6 for Unoccupied, and 493.5 for Occupied.
- Mean CO2 is 683.5 for Unoccupied, and 938 for Occupied.
- Mean HumidityRatio is 0.00439 for Unoccupied, and 0.00472 for Occupied.

```r
library(e1071)

nb1 <- naiveBayes(Occupancy~., data=train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        0        1
## 0.756267 0.243733
##
## Conditional probabilities:
##    Temperature
## Y       [,1]      [,2]
##   0 20.74905 0.8311195
##   1 22.14193 0.8662831
##
##    Humidity
## Y       [,1]     [,2]
##   0 29.02691 4.406892
##   1 28.60673 3.050533
##
##    Light
## Y        [,1]      [,2]
##   0  23.62548  77.62526
##   1 493.45701 110.75047
##
##    CO2
## Y       [,1]     [,2]
##   0 683.5429 276.8469
##   1 938.0636 270.1616
##
##    HumidityRatio
## Y          [,1]         [,2]
##   0 0.004386398 0.0006003845
##   1 0.004720380 0.0005069081
```

Step F: Use both models to predict on the test data. Evaluate the results, using classification metrics.

Explanation:

Note: The system considers an Unoccupied room to be the positive case.

Looking at the confusion matrices and the metrics for both models, I can see that the overall accuracy of the Logistic Regression model is more than the overall accuracy of the Naive Bayes model, by .032. This means that the Logistic Regression model is 3.2% more accurate than the Naive Bayes model.

Second, I can notice that the Logistic Regression model has higher sensitivity than the Naive Bayes model. This means that the Logistic Regression model is better at identifying positive results than the Naive Bayes Result, by 4%.

Lastly, I see that the Naive Bayes model has a specificity of 1, while the Logistic Regression model has a specificity of 0.9967. This means that the Naive Bayes model is 0.3% better at identifying negative results than the Logistic Regression model.

All of these metrics can also be observed in the actual predictions. The Logistic Regression had higher sensitivity, so it had more true positives and less false negatives than the Naive Bayes. Similarly, the Naive Bayes had perfect specificity, so it had all true negatives and zero false positives.

```
probs1 <- predict(glm1, newdata=test, type="response")
pred1 <- ifelse(probs1 > 0.5, 1, 0)

library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
confusionMatrix(as.factor(pred1), reference=test$Occupancy)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1871    2
##          1   13  598
##
##                Accuracy : 0.994
##                  95% CI : (0.9901, 0.9966)
##     No Information Rate : 0.7585
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9836
##
##  Mcnemar's Test P-Value : 0.009823
##
##             Sensitivity : 0.9931
##             Specificity : 0.9967
##          Pos Pred Value : 0.9989
##          Neg Pred Value : 0.9787
##              Prevalence : 0.7585
##          Detection Rate : 0.7532
##    Detection Prevalence : 0.7540
##       Balanced Accuracy : 0.9949
##
##        'Positive' Class : 0
##
```

```
pred2 <- predict(nb1, newdata=test, type="class")

confusionMatrix(pred2, reference=test$Occupancy)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1791    0
##          1   93  600
##
##                Accuracy : 0.9626
##                  95% CI : (0.9543, 0.9697)
##     No Information Rate : 0.7585
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9029
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9506
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.8658
##              Prevalence : 0.7585
##          Detection Rate : 0.7210
##    Detection Prevalence : 0.7210
##       Balanced Accuracy : 0.9753
##
##        'Positive' Class : 0
##
```

Step G: Strengths and Weaknesses.

Logistic Regression:

On one hand, it is easy to understand, implement, and interpret; it can be easily extended to multiple classes; it provides a good model when the data set is linearly separable; and it not only gives us the coefficient of each predictor, but it also gives us its direction. On the other hand, it assumes linearity between dependent and independent variables, which is hard to get in real life scenarios; it can't solve non-linear problems; and it can't capture complex relationships.

Naive Bayes:

On one hand, it gives us a faster training time than some other algorithms; it works well with high scale data sets; and it is not sensitive to irrelevant attributes. On the other hand, it assumes that all the features are independent, which rarely occurs in real-world examples; and it assigns zero probability to any variable in the test set, whose data was not available in the training set.

Step H: Classification Metrics and their benefits:

Accuracy:

Accuracy tells us how many times a model gave a correct prediction, out of all the number of observations. This is a good metric to apply to our model because it gives us a rating on the overall performance of our model. The higher the accuracy, the better the model.

Sensitivity:

Sensitivity tells us how many times a model gave a true positive result, out of all the times that it was supposed to be true. This metric is good at representing if our model can correctly identify a positive result.

If we think about it in a medical testing sense, we want our model to have high sensitivity because that would mean that our test will be able to identify which patient has a certain disease.

Specificity:

Specificity tells us how many times a model gave a true negative result, out of all the times that it was supposed to be negative. This metric is good at representing if our model can correctly identify a negative result. If we think about it in a medical testing sense, we want our model to have high specificity because that would mean that our test will be able to identify which patient does not have a certain disease.