Classification Appliance Energy Data

Yazan Abughazaleh

2/16/2023

Introduction

The purpose of this notebook is to demonstrate the process of performing classification using both logistic regression and Naive Bayes algorithms. The data set I have chosen is the Bank Marketing data set from https://archive.ics.uci.edu/ml/datasets/Bank+Marketing. Performing classification involves identifying which class an observation falls into. Linear models in classification divide the binary classes through the use of a boundary line. Classification is used when target variables are qualitative.

Data Exploration

The first step to performing classification is to do data exploration to better understand our data. We begin by reading in the bank-full.csv file into a data frame and also evaluating the structure.

```
df <- read.csv("bank-full.csv", header = TRUE, sep=";")
str(df)
```

```
## 'data.frame':    45211 obs. of  17 variables:
##  $ age      : int  58 44 33 47 33 35 28 42 58 43 ...
##  $ job      : chr  "management" "technician" "entrepreneur" "blue-collar" ...
##  $ marital  : chr  "married" "single" "married" "married" ...
##  $ education: chr  "tertiary" "secondary" "secondary" "unknown" ...
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  2143 29 2 1506 1 231 447 2 121 593 ...
##  $ housing  : chr  "yes" "yes" "yes" "yes" ...
##  $ loan     : chr  "no" "no" "yes" "no" ...
##  $ contact  : chr  "unknown" "unknown" "unknown" "unknown" ...
##  $ day      : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ month    : chr  "may" "may" "may" "may" ...
##  $ duration : int  261 151 76 92 198 139 217 380 50 55 ...
##  $ campaign : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : chr  "unknown" "unknown" "unknown" "unknown" ...
##  $ y        : chr  "no" "no" "no" "no" ...
```

We can see that many of our variables above are characters, which we would want to convert into factors.

```
for(i in 1:ncol(df)){
  if(is.character(df[,i])){
    df[,i] <- as.factor(df[,i])
```

```
  }
}
```

Next, we want to divide our data into a train and test set.

```
set.seed(3)
i <- sample(1:nrow(df), nrow(df) * 0.80,replace=FALSE)
train <-df[i,]
test <- df[-i,]
```

After the data has been split, we can begin exploring our training set. The first thing we would like to identify is the structure of the set.

```
str(train)
```

```
## 'data.frame':    36168 obs. of  17 variables:
##  $ age      : int  49 26 43 36 32 60 46 44 43 32 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 5 9 5 11 2 6 2 2 2 5 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 3 2 3 2 2 2 1 2 3 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 3 2 3 3 2 2 2 2 1 3 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : int  0 100 0 953 112 5789 870 558 -124 0 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 1 1 1 2 2 2 1 2 2 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 2 1 1 2 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 1 1 1 1 3 3 1 ...
##  $ day      : int  19 26 6 31 20 12 31 7 27 17 ...
##  $ month    : Factor w/ 12 levels "apr","aug","dec",..: 2 9 2 2 1 9 6 9 9 10 ...
##  $ duration : int  78 445 124 102 311 50 87 485 47 107 ...
##  $ campaign : int  6 1 2 2 1 5 2 1 5 2 ...
##  $ pdays    : int  -1 -1 -1 101 321 -1 -1 -1 -1 -1 ...
##  $ previous : int  0 0 0 3 1 0 0 0 0 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 4 4 4 1 2 4 4 4 4 4 ...
##  $ y        : Factor w/ 2 levels "no","yes": 1 2 1 1 1 1 1 1 1 1 ...
```

We can see that all values which should be factors have been properly converted in the set above. To view how many observations we have in the train set, we can use the nrow() function.

```
nrow(train)
```

```
## [1] 36168
```

We can also check how many variables are in the data set with nrow().

```
ncol(train)
```

```
## [1] 17
```

It is also important to make sure we do not have NA's in our data, which we can check with colSums().

```r
colSums(is.na(train))
```

```
##       age       job   marital education   default   balance   housing      loan 
##         0         0         0         0         0         0         0         0 
##   contact       day     month  duration  campaign     pdays  previous  poutcome 
##         0         0         0         0         0         0         0         0 
##         y 
##         0
```

Lastly, we can view a summary of our data with summary().

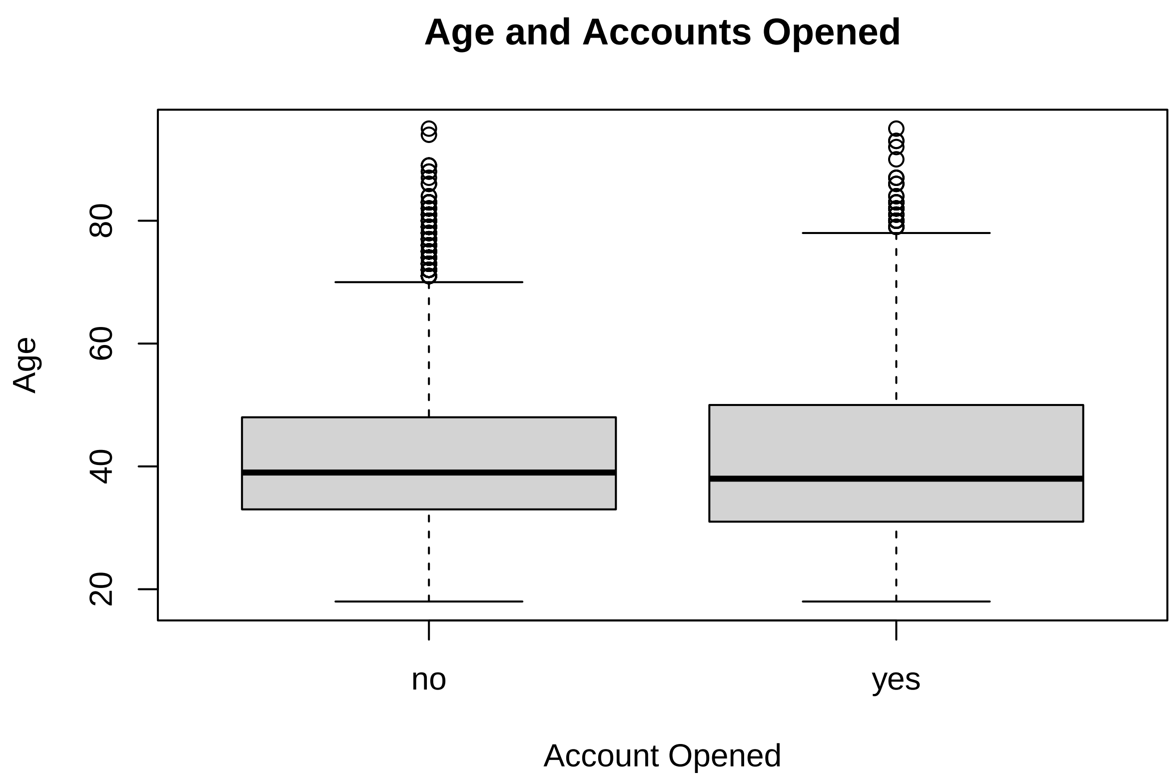
```r
summary(train)
```

```
##       age                 job           marital          education    
##  Min.   :18.00   blue-collar:7808   divorced: 4171   primary  : 5494  
##  1st Qu.:33.00   management :7533   married :21724   secondary:18549  
##  Median :39.00   technician :6067   single  :10273   tertiary :10648  
##  Mean   :40.89   admin.     :4124                    unknown  : 1477  
##  3rd Qu.:48.00   services   :3384                                     
##  Max.   :95.00   retired    :1813                                     
##                  (Other)    :5439                                     
##  default        balance       housing      loan           contact     
##  no :35528   Min.   : -8019   no :15972   no :30426   cellular :23370  
##  yes:  640   1st Qu.:    75   yes:20196   yes: 5742   telephone: 2296  
##              Median :   450                           unknown  :10502  
##              Mean   :  1359                                            
##              3rd Qu.:  1416                                            
##              Max.   :102127                                            
##                                                                        
##       day           month          duration         campaign     
##  Min.   : 1.00   may    :11076   Min.   :   0.0   Min.   : 1.000  
##  1st Qu.: 8.00   jul    : 5480   1st Qu.: 103.0   1st Qu.: 1.000  
##  Median :16.00   aug    : 4919   Median : 179.0   Median : 2.000  
##  Mean   :15.79   jun    : 4306   Mean   : 257.6   Mean   : 2.773  
##  3rd Qu.:21.00   nov    : 3188   3rd Qu.: 318.0   3rd Qu.: 3.000  
##  Max.   :31.00   apr    : 2365   Max.   :4918.0   Max.   :63.000  
##                  (Other): 4834                                    
##      pdays           previous           poutcome       y        
##  Min.   : -1.00   Min.   :  0.0000   failure: 3888   no :31948  
##  1st Qu.: -1.00   1st Qu.:  0.0000   other  : 1484   yes: 4220  
##  Median : -1.00   Median :  0.0000   success: 1199              
##  Mean   : 40.22   Mean   :  0.5758   unknown:29597              
##  3rd Qu.: -1.00   3rd Qu.:  0.0000                              
##  Max.   :871.00   Max.   :275.0000                              
## 
```

Visualizing Data

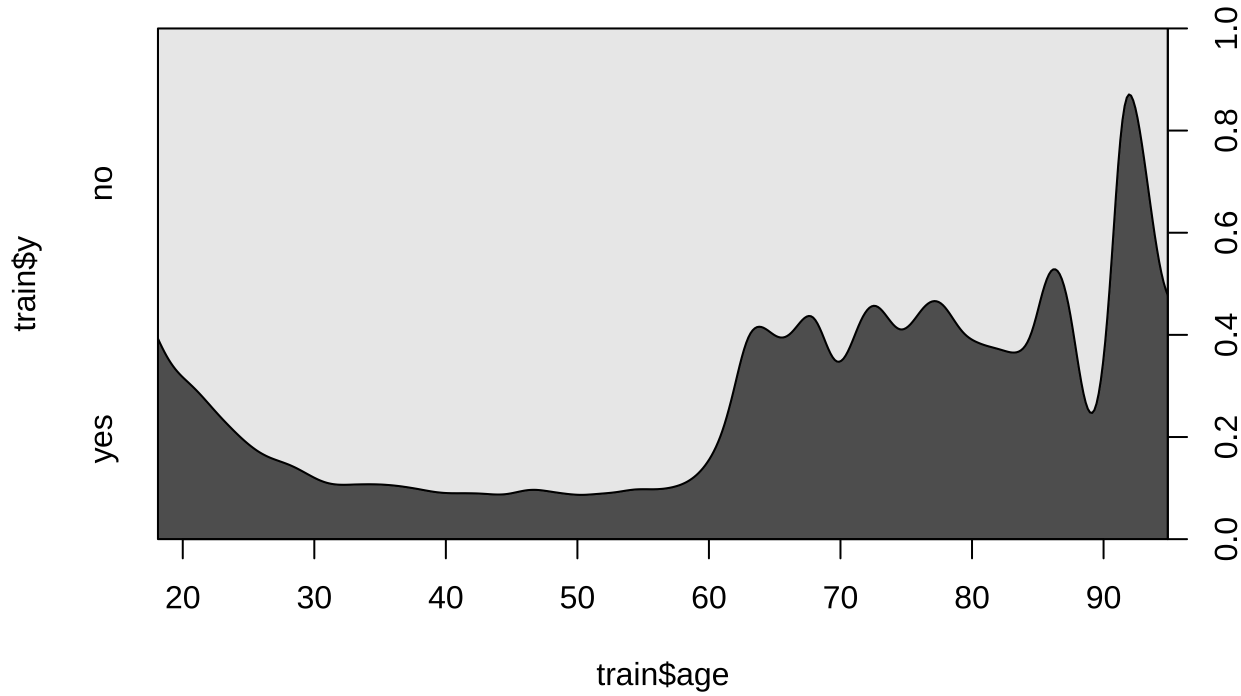
We would like to visualize how age affects the creation of a bank account. The graph below shows age related to the opening of an account and shows that it is more common for people to open an account across a wider age group.

```r
boxplot(train$age ~ train$y, data=train, main="Age and Accounts Opened",varWidth=TRUE, ylab="Age",xlab
```



The next plot is a conditional density plot to visualize how the age affects opening a bank account. The lighter portion indicates accounts not opened while the darker portion indicates new accounts opened.

```r
cdplot(train$y~train$age)
```



Logistic Regression

We can now build our logistic regression model using the glm() function.

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

```
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.7706  -0.3727  -0.2520  -0.1486   3.4054
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.595e+00  2.052e-01 -12.643  < 2e-16 ***
## age                -7.079e-04  2.484e-03  -0.285 0.775699
## jobblue-collar     -3.167e-01  8.216e-02  -3.855 0.000116 ***
## jobentrepreneur    -1.987e-01  1.372e-01  -1.448 0.147586
## jobhousemaid       -5.748e-01  1.570e-01  -3.662 0.000250 ***
## jobmanagement      -1.164e-01  8.259e-02  -1.409 0.158846
## jobretired          2.476e-01  1.097e-01   2.257 0.024014 *
## jobself-employed   -3.231e-01  1.276e-01  -2.531 0.011379 *
```

```
## jobservices        -1.538e-01  9.276e-02  -1.658 0.097340 .
## jobstudent          4.959e-01  1.208e-01   4.107 4.01e-05 ***
## jobtechnician      -1.830e-01  7.782e-02  -2.351 0.018722 *
## jobunemployed      -1.363e-01  1.248e-01  -1.092 0.274615
## jobunknown         -3.816e-01  2.745e-01  -1.390 0.164488
## maritalmarried     -1.869e-01  6.628e-02  -2.821 0.004793 **
## maritalsingle       6.236e-02  7.562e-02   0.825 0.409613
## educationsecondary  1.938e-01  7.270e-02   2.666 0.007687 **
## educationtertiary   3.695e-01  8.447e-02   4.374 1.22e-05 ***
## educationunknown    2.182e-01  1.172e-01   1.862 0.062620 .
## defaultyes         -1.122e-02  1.828e-01  -0.061 0.951040
## balance             1.249e-05  5.806e-06   2.152 0.031434 *
## housingyes         -6.955e-01  4.907e-02 -14.172  < 2e-16 ***
## loanyes            -4.310e-01  6.758e-02  -6.377 1.80e-10 ***
## contacttelephone   -1.645e-01  8.459e-02  -1.944 0.051839 .
## contactunknown     -1.643e+00  8.195e-02 -20.049  < 2e-16 ***
## day                 9.211e-03  2.801e-03   3.288 0.001009 **
## monthaug           -6.912e-01  8.788e-02  -7.865 3.70e-15 ***
## monthdec            6.900e-01  2.052e-01   3.363 0.000770 ***
## monthfeb           -1.726e-01  1.005e-01  -1.717 0.085998 .
## monthjan           -1.157e+00  1.341e-01  -8.629  < 2e-16 ***
## monthjul           -8.307e-01  8.700e-02  -9.548  < 2e-16 ***
## monthjun            5.338e-01  1.044e-01   5.113 3.17e-07 ***
## monthmar            1.725e+00  1.346e-01  12.815  < 2e-16 ***
## monthmay           -3.906e-01  8.075e-02  -4.837 1.32e-06 ***
## monthnov           -8.419e-01  9.445e-02  -8.914  < 2e-16 ***
## monthoct            9.153e-01  1.195e-01   7.662 1.84e-14 ***
## monthsep            8.455e-01  1.350e-01   6.264 3.74e-10 ***
## duration            4.257e-03  7.278e-05  58.491  < 2e-16 ***
## campaign           -8.155e-02  1.105e-02  -7.377 1.62e-13 ***
## pdays               1.361e-04  3.391e-04   0.401 0.688199
## previous            8.063e-03  6.373e-03   1.265 0.205787
## poutcomeother       2.039e-01  1.009e-01   2.020 0.043402 *
## poutcomesuccess     2.278e+00  9.265e-02  24.586  < 2e-16 ***
## poutcomeunknown    -4.025e-02  1.039e-01  -0.387 0.698486
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26059  on 36167  degrees of freedom
## Residual deviance: 17166  on 36125  degrees of freedom
## AIC: 17252
##
## Number of Fisher Scoring iterations: 6
```

The summary output of our logistic regression gives us several key measurements. The first statistic we see is the deviance residuals, which quantify a given point in the data's contribution to the overall likelihood. The deviance residuals are a transformation of the loss function, and they can be used to form an RSS-like statistic. The next metrics we see are null deviance and residual deviance. Typically, we would like to see that the residual deviance is significantly lower than the null deviance. Both the null and residual deviance are a measure of how good the model is fit for the data. Now we can look at the AIC, which stands for Akaike Information Criterion and is based on the deviance. AIC is useful in comparing models to each other. A lower AIC is better and is preferential to models that are less complex with fewer predictors. Lastly, the coefficients quantify the difference in the log odds of our target variable.

Naive Bayes

To use the Naive Bayes algorithm, we must first import the library package e1071. We can then perform the training of a Naive Bayes model with the NaiveBayes() function.

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

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        no       yes
## 0.8833223 0.1166777
##
## Conditional probabilities:
##      age
## Y         [,1]    [,2]
##   no  40.80227 10.1631
##   yes 41.53815 13.3866
##
##      job
## Y          admin. blue-collar entrepreneur   housemaid  management     retired
##   no  0.113622136 0.227119068  0.033335420 0.028202078 0.203205208 0.043977714
##   yes 0.117061611 0.130805687  0.025355450 0.019431280 0.246682464 0.096682464
##      job
## Y     self-employed    services     student  technician  unemployed     unknown
##   no    0.034587455 0.096062351 0.016839865 0.169087267 0.027732565 0.006228872
##   yes   0.033886256 0.074644550 0.052606635 0.157582938 0.039336493 0.005924171
##
##      marital
## Y      divorced   married    single
##   no  0.1152811 0.6111807 0.2735382
##   yes 0.1156398 0.5208531 0.3635071
##
##      education
## Y        primary  secondary   tertiary    unknown
##   no  0.15709904 0.51921873 0.28349192 0.04019031
##   yes 0.11255924 0.46469194 0.37701422 0.04573460
##
##      default
## Y             no        yes
##   no  0.98125078 0.01874922
##   yes 0.99028436 0.00971564
##
##      balance
```

```
## Y         [,1]     [,2]
##   no  1302.084 2982.862
##   yes 1788.949 3243.416
##
##      housing
## Y            no       yes
##   no  0.4162076 0.5837924
##   yes 0.6338863 0.3661137
##
##      loan
## Y             no        yes
##   no  0.83222737 0.16777263
##   yes 0.90947867 0.09052133
##
##      contact
## Y       cellular  telephone    unknown
##   no  0.62263678 0.06210091 0.31526230
##   yes 0.82417062 0.07393365 0.10189573
##
##      day
## Y         [,1]     [,2]
##   no  15.87555 8.301302
##   yes 15.10213 8.501258
##
##      month
## Y             apr         aug         dec         feb         jan         jul
##   no  0.059659447 0.136941280 0.002629273 0.055402529 0.031551271 0.156191311
##   yes 0.108767773 0.128909953 0.017772512 0.081990521 0.027962085 0.116113744
##      month
## Y             jun         mar         may         nov         oct         sep
##   no  0.120664830 0.005321147 0.323463128 0.089583072 0.010892701 0.007700013
##   yes 0.106872038 0.047867299 0.175829384 0.077251185 0.061611374 0.049052133
##
##      duration
## Y         [,1]     [,2]
##   no  220.5720 206.3221
##   yes 537.9088 386.6043
##
##      campaign
## Y         [,1]     [,2]
##   no  2.852917 3.270860
##   yes 2.164455 1.982136
##
##      pdays
## Y         [,1]      [,2]
##   no  36.44375  97.10827
##   yes 68.79076 119.88205
##
##      previous
## Y          [,1]     [,2]
##   no  0.4985915 2.355944
##   yes 1.1606635 2.479261
##
##      poutcome
```

```
## Y        failure      other    success    unknown
##   no  0.10607863 0.03884437 0.01352197 0.84155503
##   yes 0.11824645 0.05758294 0.18175355 0.64241706
```

The data show is broken down into conditional probabilities for each different attribute. The prior for making a bank account, is called Apriori and is .88 for no and .12 for yes. Discrete variables are output as conditional probabilities, while continuous variables output the man and standard deviation of their classes.

Evaluating Data

We now want to use our models to predict and evaluate the test data.

```
probs <- predict(glm1,newdata=test, type="response")
pred <- ifelse(probs>0.5,1,0)
pred <- as.factor(pred)
levels(pred) <- list("no"="0","yes"="1")
levels(test$y) <- list("no"="0","yes"="1")
acc <- mean(as.integer(pred)==as.integer(test$y))
print(paste("glm1 accuracy: ", acc))
```

```
## [1] "glm1 accuracy:  0.901028419772199"
```

The above code snippet calculates the accuracy for the logistic regression model. The predicted accuracy is shown as 88% and the error rate is 12%.

We also can output a confusion matrix by using the table() function to show the number of classifications.

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
confusionMatrix(pred,reference=test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7764  685
##        yes  210  384
##
##                Accuracy : 0.901
##                  95% CI : (0.8947, 0.9071)
##     No Information Rate : 0.8818
##     P-Value [Acc > NIR] : 3.531e-09
##
##                   Kappa : 0.4122
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
##             Sensitivity : 0.9737
##             Specificity : 0.3592
##          Pos Pred Value : 0.9189
##          Neg Pred Value : 0.6465
##              Prevalence : 0.8818
##          Detection Rate : 0.8586
##    Detection Prevalence : 0.9343
##       Balanced Accuracy : 0.6664
##
##        'Positive' Class : no
##
```

Here, we have created a confusion matrix for the logistic regression model. The diagonal represents the true positive and true negative values. Next we will be evaluating the data for the Naive Bayes model.

```r
p1 <- predict(nb1, newdata=test, type="class")
table(p1, test$y)
```

```
##
## p1      no  yes
##   no  7360  473
##   yes  614  596
```

```r
mean(p1==test$y)
```

```
## [1] 0.8797965
```

The results seem to indicate that the logistic regression model outperformed the Naive Bayes model because the accuracy was higher as well as the count of true positives and true negatives in the confusion matrices. This makes sense because Naive Bayes tends to perform better with smaller data sets and the bank data set is a medium sized set.

Strengths and Weaknesses

Logistic regression is an ideal choice to use when data can be linearly separated into two classes. It is computationally inexpensive to perform and has easy to use probabilistic outputs. It does however suffer when trying to fit data, as it tends to under-fit the data especially when decision boundaries are non-linear. Naive Bayes is an ideal algorithm to use when working with small data sets. It is easy to use and implement and handles high dimension data very well. Its weaknesses lie in that it is outperformed by other algorithms for larger data sets, and may work poorly if predictors are not independent of each other.

Evaluation Metrics

There are several important metrics to use when evaluating a classification model. Accuracy, sensitivity, specificity, and kappa were all used in this notebook. Accuracy is a measure of the total percentage of correct classifications performed by the model. It does not however give specifics on the true positive and true negative rates in the model. Sensitivity is used as the measure of the true positive rate, while specificity is indicative of the true negative rate. Lastly, kappa is used to help quantify how closely predictors agree with the actual data.