Homework 8

J. Lucas McKay

3/23/2023

Section: F1 Score and other classifier performance metrics.

We need common measures of classifier performance all the time, but they are actually not built in to base R and implementing them provides a great deal of insight into classifier performance. Create and document utility functions for the following measures of classifier performance. Configure the functions so that they accept either logical vectors of predictions and ground truth values, or numerical arguments with 1s corresponding to positives and 0s corresponding to negatives.

1. Sensitivity(pred,truth) : Sensitivity is defined as # True Positives / # All Positives. (i.e., where # All Positives = # True Positives + # False Negatives.) (1 point)

```
# sample vectors
pred <- c(1, 0, 1, 1, 0, 1, 1, 0)
truth <- c(1, 0, 0, 1, 1, 1, 0, 0)

pred <- as.logical(pred)
truth <- as.logical(truth)

Sensitivity <- function(pred, truth) {
  true_positives <- sum(pred & truth)
  all_positives <- sum(truth) # true positives + false negatives
  sensitivity <- true_positives/all_positives
  return(sensitivity)
}
```

2. Specificity(pred,truth) : Specificity is defined as # True Negatives / # All Negatives. (i.e., where # All Negatives = # True Negatives + # False Positives.) (1 point)

```
Specificity <- function(pred, truth) {
  true_negatives <- sum(!pred & !truth) # !pred=predicted -ve, !truth=actually -ve
  all_negatives <- sum(!truth) # true negatives + false positives
  specificity <- true_negatives/all_negatives
  return(specificity)
}
```

3. Precision(pred,truth) : Precision is defined as # True Positives / (# True Positives + # False Positives). Precision is also referred to as Positive Predictive Value (PPV). (1 point)

```
Precision <- function(pred, truth) {
  true_positives <- sum(pred & truth)
  false_positives <- sum(pred & !truth) # Count the false positives
  precision <- true_positives/(true_positives + false_positives)
  return(precision)
}
```

4. Recall(pred,truth) : Recall is defined identically to Sensitivity. (1 point)

```
Recall <- Sensitivity
```

5. F1(pred,truth) : F1 score is defined as 2 x (precision x recall) / (precision + recall). (1 point)

```
F1 <- function(pred, truth) {
  precision <- Precision(pred, truth)
  recall <- Recall(pred, truth)
  f1_score <- 2 * (precision * recall)/(precision + recall)
  return(f1_score)
}
```

Testing the functions:

```
sensitivity <- Sensitivity(pred, truth)
specificity <- Specificity(pred, truth)
precision <- Precision(pred, truth)
recall <- Recall(pred, truth)
f1 <- F1(pred, truth)

cat("Sensitivity and Recall", sensitivity, "\n")
```

```
## Sensitivity and Recall 0.75
```

```
cat("Specificity:", specificity, "\n")
```

```
## Specificity: 0.5
```

```
cat("Precision:", precision, "\n")
```

```
## Precision: 0.6
```

```
cat("F1 Score:", f1, "\n")
```

```
## F1 Score: 0.6666667
```

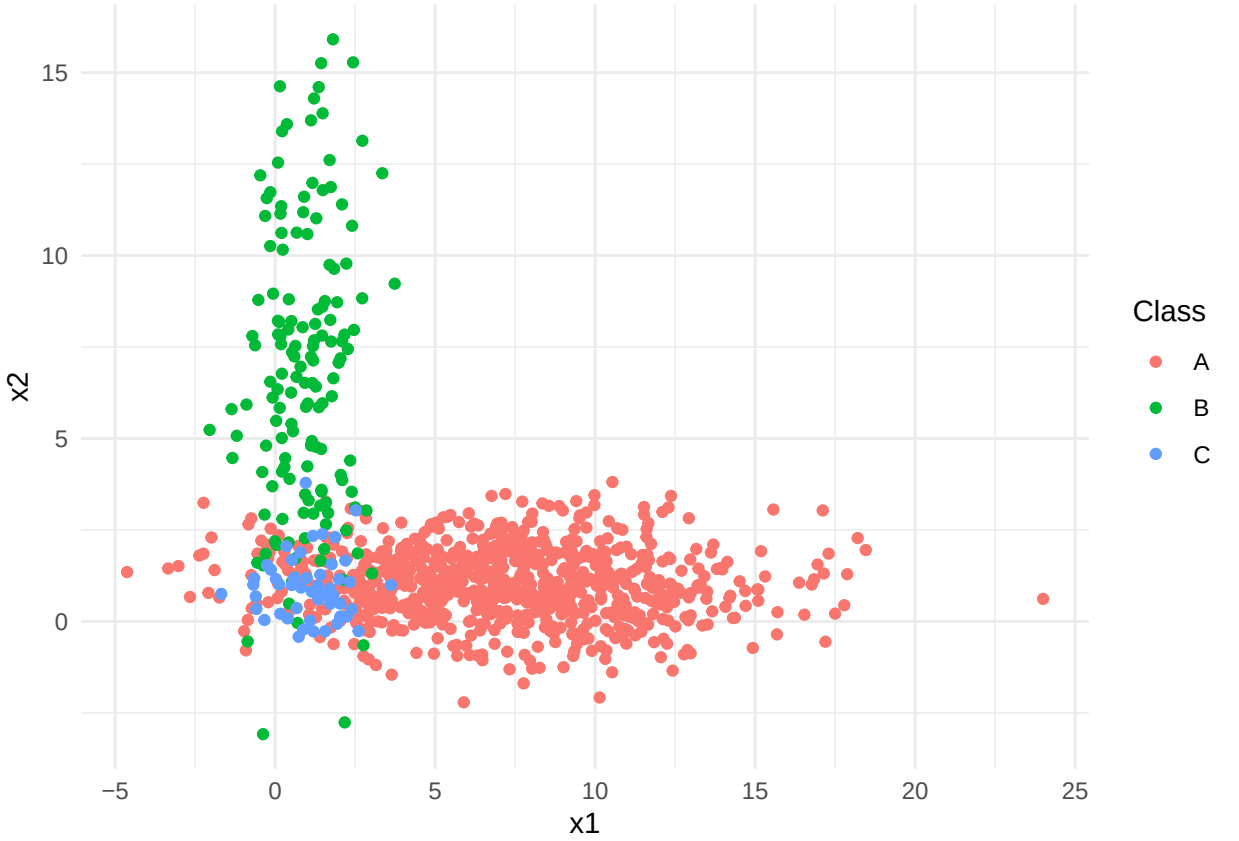
Section 2: Naïve Bayes.

Under the common assumptions of "naïve Bayes" classification, all of the samples $x_i (i = 1, \ldots, n)$ in a vector $\mathbf{x}$ are independent and identically distributed (iid). If there are C_k potential classes from which $\mathbf{x}$ could be drawn, the one for which $p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$ is greatest is the naïve Bayes prediction.

Assume that you have received a dataset with two variables x_1 and x_2, and three classes, A, B, C. You have derived the following summary statistics for each class, where m and s indicate the sample mean and standard deviation of each class, respectively, and n indicates the number of samples corresponding to that class in the dataset. Do the following; make naïve Bayes assumptions for all questions.

Class	n	x_1	x_2
		m, s	m, s
A	800	$7.0, 4.0$	$1.0, 1.0$
B	150	$1.0, 1.0$	$7.0, 4.0$
C	50	$1.0, 1.0$	$1.0, 1.0$

```r
# visualizing the data in a plot
library(ggplot2)

class_A <- data.frame(x1 = rnorm(800, mean = 7.0, sd = 4.0),
                      x2 = rnorm(800, mean = 1.0, sd = 1.0),
                      Class = "A")

class_B <- data.frame(x1 = rnorm(150, mean = 1.0, sd = 1.0),
                      x2 = rnorm(150, mean = 7.0, sd = 4.0),
                      Class = "B")

class_C <- data.frame(x1 = rnorm(50, mean = 1.0, sd = 1.0),
                      x2 = rnorm(50, mean = 1.0, sd = 1.0),
                      Class = "C")

samples <- rbind(class_A, class_B, class_C)

ggplot(samples, aes(x1, x2, color = Class)) +
  geom_point() +
  labs(x = "x1",y = "x2") +
  theme_minimal()
```



1. Calculate the class probabilities $p(C_k)$. (1 point)

```
samples_total <- 800 + 150 + 50

cat("Probability of class A:", 800/samples_total, "\n")
```

```
## Probability of class A: 0.8
```

```
cat("Probability of class B:", 150/samples_total, "\n")
```

```
## Probability of class B: 0.15
```

```
cat("Probability of class C:", 50/samples_total)
```

```
## Probability of class C: 0.05
```

2. Create a function GaussProb (x, m, s) that returns the probability $\prod_{i=1}^{n} p(x_i \mid C_k)$ of a particular sample x_i under a Gaussian process with mean m and standard deviation s. (1 point)

```
# used joint probability to iterate over the x, m, s vectors since they're composed of x1 and x2

GaussProb <- function(x, m, s) {
```

```
  joint_prob <- 1
  for (i in 1:length(x)) {
    joint_prob <- joint_prob * ((1/(sqrt(2 * pi) * s[i])) * exp(-((x[i] - m[i]) ^ 2)/(2 * s[i] ^ 2)))
  }
  return(joint_prob)
}


x <- c(x1 = 9, x2 = 2) # random sample
m <- c(x1 = 7, x2 = 1) # class A example
s <- c(x1 = 4, x2 = 1)

print(GaussProb(x, m, s))
```

```
##         x1
## 0.02129738
```

3. Create functions GaussProbA (x), GaussProbB (x), and GaussProbC (x) that return the probability $\prod_{i=1}^{n} p\left(x_i \mid C_k\right)$ of a particular vector $\mathbf{x} = (x_1, x_2)$ under each class k. (1 point)

```
GaussProbA <- function(x) {
  m <- c(x1 = 7.0, x2 = 1.0)
  s <- c(x1 = 4.0, x2 = 1.0)
  return(GaussProb(x, m, s))
}

GaussProbB <- function(x) {
  m <- c(x1 = 1.0, x2 = 7.0)
  s <- c(x1 = 1.0, x2 = 4.0)
  return(GaussProb(x, m, s))
}

GaussProbC <- function(x) {
  m <- c(x1 = 1.0, x2 = 1.0)
  s <- c(x1 = 1.0, x2 = 1.0)
  return(GaussProb(x, m, s))
}

print(GaussProbA(x))
```

```
##         x1
## 0.02129738
```

```
print(GaussProbB(x))
```

```
##           x1
## 2.306982e-16
```

```
print(GaussProbC(x))
```

```
##           x1
## 1.222502e-15
```

4. Create a function classProb (x) that returns the probabilities of a particular vector $\mathbf{x} = (x_1, x_2)$ under each class k. Make sure to normalize the class predictions using a softmax function so that they can be treated as probabilities. (1 point)

```
softmax <- function(x) {
  exp_x <- exp(x)
  return(exp_x / sum(exp_x))
}

classProb <- function(x) {
  probA <- GaussProbA(x)
  probB <- GaussProbB(x)
  probC <- GaussProbC(x)

  class_probs <- c(A = probA, B = probB, C = probC)
  normalized_probs <- softmax(class_probs)

  return(normalized_probs)
}

class_probs <- classProb(x)
print(class_probs)
```

```
##      A.x1      B.x1      C.x1
## 0.3380828 0.3309586 0.3309586
```

5. Predict the most likely classes for the following vectors: $\mathbf{x} = (x_1, x_2) : (7, 1), (1, 7)$, and $(1, 1)$. Note that these are the centers of the Gaussian processes for each class. What is the most likely class for $(1, 1)$, and why?

```
vector1 <- c(x1 = 7, x2 = 1)
vector2 <- c(x1 = 1, x2 = 7)
vector3 <- c(x1 = 1, x2 = 1)

# calc all probabilities
vector1_probs <- classProb(vector1)
vector2_probs <- classProb(vector2)
vector3_probs <- classProb(vector3)

cat("Probabilites for (7, 1):", vector1_probs, "\n")
```

```
## Probabilites for (7, 1): 0.3422331 0.3288834 0.3288834
```

```
cat("Probabilites for (1, 7):", vector2_probs, "\n")
```

```
## Probabilites for (1, 7): 0.3288834 0.3422331 0.3288834
```

```
cat("Probabilites for (1, 1):", vector3_probs, "\n")
```

```
## Probabilites for (1, 1): 0.3167092 0.3167092 0.3665817
```

```r
# checking for most likely
Class_Mostlikely <- function(class_probs) {
  class_labels <- c("A", "B", "C")
  return(class_labels[which.max(class_probs)])
}

cat("Most likely class for (7, 1):", Class_Mostlikely(vector1_probs), "\n")
```

```
## Most likely class for (7, 1): A
```

```r
cat("Most likely class for (1, 7):", Class_Mostlikely(vector2_probs), "\n")
```

```
## Most likely class for (1, 7): B
```

```r
cat("Most likely class for (1, 1):", Class_Mostlikely(vector3_probs))
```

```
## Most likely class for (1, 1): C
```

The most likely class for (1,1) is C. This is because it's probability is highest among the classes A, B, and C meaning that its samples are closest to the mean and center of the Gaussian process.