DATA 621 - Business Analytics and Data Mining

Fall 2020 - Group 2 - Homework #2

Avraham Adler, Samantha Deokinanan, Amber Ferger, John Kellogg, Bryan Persaud, Jeff Shamp

10/11/2020

0. Preamble

For all the following answers, it is expected that we are only dealing with binary classification in which negative is coded as 0 and positive is coded as 1. It is not difficult to expand the functions to respond to more general cases, but it is an unnecessary complication for this homework assignment.

1. The Data

The data set contains 181 data points, and has three key target columns:

- **class**: the actual class for the observation
- **scored.class**: the predicted class for the observation (based on a threshold of 0.5)
- **scored.probability**: the predicted probability of success for the observation

2. Confusion Matrix

A confusion matrix is a format for summarizing the performance of a classification algorithm. It shows the number of correct and incorrect predictions made by the classification model and compares it to the actual outcomes in the data. The matrix is $N \times N$, where N is the number of target values. The following table displays a 2×2 confusion matrix for two classes—positive and negative. The actual confusion matrix comprises the 4 cells containing `a`, `b`, `c`, and `d`. The enhancements around the matrix display the definitions for many common classification metrics.

```
## -----------------------------------------------------------------------------
##                  |            Target           |
## Confusion Matrix |---------------------------|
##                  |   Positive  |   Negative  |
## -----------------|-------------|-------------|-------------------------------
##       | Positive |a (True Pos) |b (False Pos)| Positive Predictive | a/(a+b)
## Model |----------|-------------|-------------|-------------------------------
##       | Negative |c (False Neg)|d (True Neg) | Negative Predictive | d/(c+d)
## -----------------|-------------|-------------|-------------------------------
##                  | Sensitivity | Specificity |
##                  |-------------|-------------|  Accuracy = (a+d)/(a+b+c+d)
##                  |   a/(a+c)   |   d/(b+d)   |
## -----------------------------------------------------------------------------
```

- **Accuracy**: the proportion of the total number of predictions that were correct.

- **Positive Predictive Value** or **Precision**: the proportion of modeled positive cases that were correct.
- **Negative Predictive Value**: the proportion of modeled negative cases that were correct.
- **Sensitivity** or **Recall**: the proportion of actual positive cases that were properly detected by the model.
- **Specificity**: the proportion of actual negative cases that were properly detected by the model.

Table 1: Raw Confusion Matrix

	0	1
0	119	5
1	30	27

The raw confusion matrix is shown above. In this case, the rows represent the **true** classes and the columns represent the **predicted** classes. This is not the same order as the hand-drawn confusion matrix above, but both are internally consistent.

3. Accuracy

The function `ACC` below takes as input a data frame whose first two columns are the true class and the modeled class and named `class` and `pred_class` respectively. It returns the accuracy of the predictions using the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

```r
ACC <- function(df) {
  if (!is.data.frame(df)) {
    stop("Data set is not a dataframe. Try as.data.frame(dataset).")
  } else {
    sum(df$class == df$pred_class) / dim(df)[[1]]
  }
}
```

For this dataset, the accuracy is 80.663%.

4. Classification Error Rate

The function `CER` below takes as input a data frame whose first two columns are the true class and the modeled class and named `class` and `pred_class` respectively. It returns the classification error rate of the predictions using the following formula:

$$\text{Classification Error Rate} = \frac{FP + FN}{TP + FP + TN + FN}$$

```r
CER <- function(df) {
  if (!is.data.frame(df)) {
    stop("Data set is not a dataframe. Try as.data.frame(dataset).")
  } else {
    sum(df$class != df$pred_class) / dim(df)[[1]]
  }
}
```

For this dataset, the classification error rate is 19.337%.

Mathematically, the classification error rate must be identical to $1 - \text{Accuracy}$. In this data set that is clear as $80.663\% + 19.337\% = 100\%$.

5. Precision

The function PRC below takes as input a data frame whose first two columns are the true class and the modeled class and named class and pred_class respectively. It returns the classification error rate of the predictions using the following formula:

$$\text{Precision} = \frac{TP}{TP + FP}$$

```
PRC <- function(df) {
  if (!is.data.frame(df)) {
    stop("Data set is not a dataframe. Try as.data.frame(dataset).")
  } else {
    sum(df$class == 1 & df$pred_class == 1) / sum(df$pred_class == 1)
  }
}
```

For this dataset, the precision is 84.375%.

6. Sensitivity or Recall

The function SNS below takes as input a data frame whose first two columns are the true class and the modeled class and named class and pred_class respectively. It returns the sensitivity of the predictions using the following formula:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

```
SNS <- function(df) {
  if (!is.data.frame(df)) {
    stop("Data set is not a dataframe. Try as.data.frame(dataset).")
  } else {
    sum(df$class == 1 & df$pred_class == 1) / sum(df$class == 1)
  }
}
```

For this dataset, the sensitivity is 47.368%.

7. Specificity

The function SPC below takes as input a data frame whose first two columns are the true class and the modeled class and named class and pred_class respectively. It returns the specificity of the predictions using the following formula:

$$\text{Specificity} = \frac{TN}{TN + FP}$$

```r
SPC <- function(df) {
  if (!is.data.frame(df)) {
    stop("Data set is not a dataframe. Try as.data.frame(dataset).")
  } else {
    sum(df$class == 0 & df$pred_class == 0) / sum(df$class == 0)
  }
}
```

For this dataset, the specificity is 95.968%.

8. F_1 Score

The function `F1` below takes as input a data frame whose first two columns are the true class and the modeled class and named `class` and `pred_class` respectively. It returns the F_1 score of the predictions using the following formula:

$$\mathrm{F}_1\ \mathrm{Score} = \frac{2 \times \mathrm{Precision} \times \mathrm{Sensitivity}}{\mathrm{Precision} + \mathrm{Sensitivity}}$$

```r
F1 <- function(df) {
  if (!is.data.frame(df)) {
    stop("Data set is not a dataframe. Try as.data.frame(dataset).")
  } else {
    (2 * PRC(df) * SNS(df)) / (PRC(df) + SNS(df))
  }
}
```

For this dataset, the F_1 score is 60.674%.

9. Bounds on F_1

Proof 1

We can prove this by decomposing the metrics into their component as defined above.

$$
\begin{aligned}
F_{1:numerator} &= 2 \cdot P \cdot S\\
&= 2 \cdot \frac{TP}{TP + FP} \cdot \frac{TP}{TP + FN}\\
&= \frac{2 \cdot (TP)^2}{(TP + FP)(TP + FN)}\\
F_{1:denominator} &= P + S\\
&= \frac{TP}{TP + FP} + \frac{TP}{TP + FN}\\
&= \frac{TP(TP + FN) + TP(TP + FP)}{(TP + FP)(TP + FN)}\\
&= \frac{(TP)^2 + TP \cdot FN + (TP)^2 + TP \cdot FP}{(TP + FP)(TP + FN)}\\
&= \frac{2 \cdot (TP)^2 + TP \cdot (FN + FP)}{(TP + FP)(TP + FN)}\\
F_1 &= \frac{F_{1:numerator}}{F_{1:denominator}}\\
&= \frac{2 \cdot (TP)^2}{2 \cdot (TP)^2 + TP \cdot (FN + FP)}\\
&= \frac{2 \cdot TP}{2 \cdot TP + FN + FP}
\end{aligned}
$$

Now, this last equation ***must*** be bounded between 0 and 1. Consider that TP is a non-negative integer. Also, the sum of TP, FP, TN, and FN must equal the number of observations. Excluding the degenerate case where the dataset contains only negative classes and they were all scored properly, the denominator will always be positive. When the dataset consists of only positive classes and they are all coded properly, we have $F_1 = 1$. This is also why there isn't too much to worry about with the all-negative case, as one can simply switch the labels and side-step the division by zero error. In every other case where TP is strictly less than the number of observations, there must be entries in FN or FP. This causes the fraction to be less than one. In the case where no positive entries are coded properly, then the numerator is 0 and the denominator is > 0, so the F_1 score is 0. As the building blocks all represent counts, they cannot be negative, so the score can never fall below 0. ***QED***

Proof 2

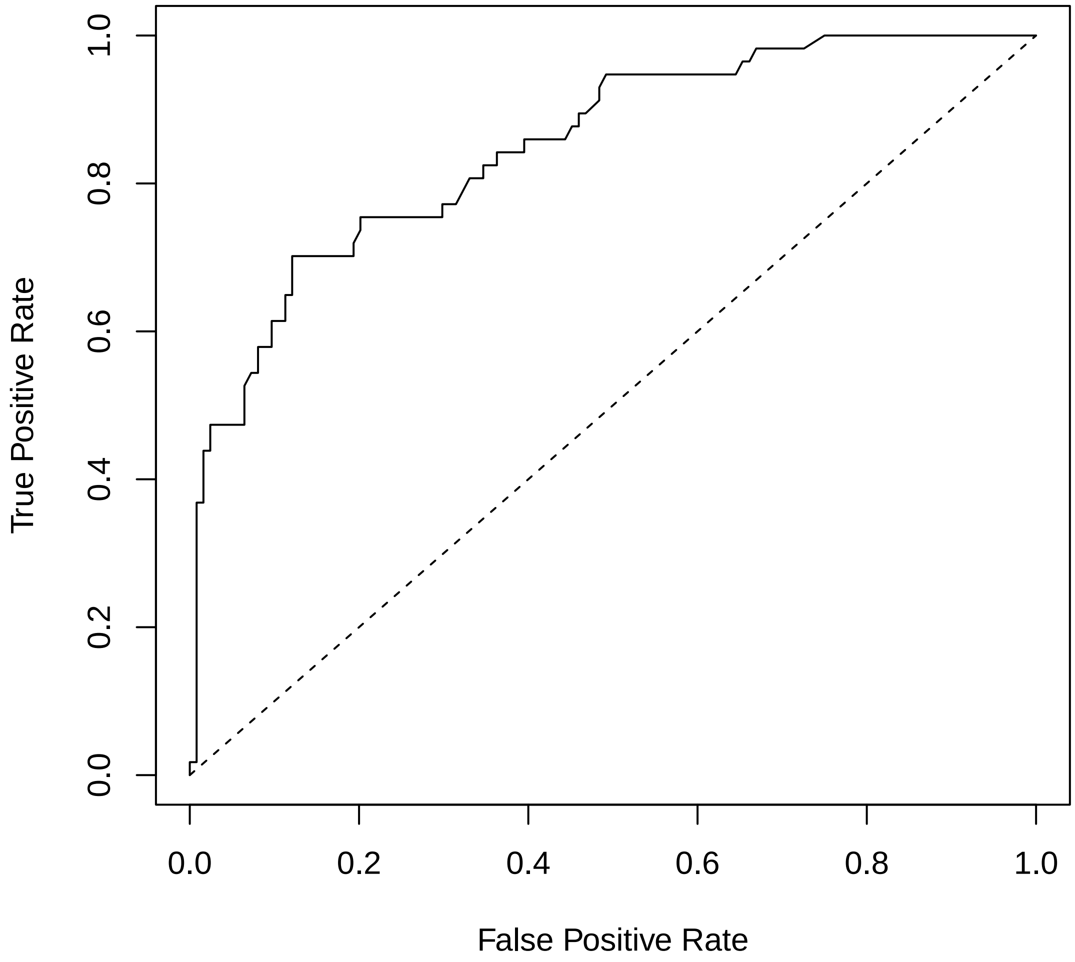
Another proof relies on the assumption that the sensitivity and precision themselves are constrained to $[0, 1]$. Given those assumptions, it ***must*** be true that $2ab < a + b$. This is a corollary of the *AM-GM inequality* since if $a, b \in [0, 1]$ then $\sqrt{ab} > ab$ and this inequality is strict. Thus, the numerator must be strictly less than the denominator. Both numerator and denominator are bounded by $[0, 1]$ as per the initial assumptions. Therefore, excepting the degenerate case where both are 0, this completes the proof that the fraction too is bounded by $[0, 1]$. ***QED***

10. ROC Curve

The receiver operating curve (ROC) is defined as a plot of the true positive rate (TPR) against the false positive rate (FPR) at different probability thresholds. The FPR is usually plotted on the x-axis. The function below takes as input a data frame with columns named `class` and `pred_class` as above, and a third column named `pred_prob` which reflects the predicted probability of being positive. It then steps along a sequence from 0 to 1, assigning classes based on that sequence, calculating the TPR and FPR for each threshold, plotting the results when completed. Lastly, it calculates the area under the ROC (AUC). Trapezoidal quadrature will be used to estimate the AUC from the table created for the ROC, as the intervals on the x-axis are not equidistant.

```
ROC <- function(df) {
  if (!is.data.frame(df)) {
    stop("Data set is not a dataframe. Try as.data.frame(dataset).")
  } else {
    rocTable <- data.frame(thresh = seq(0, 1, 0.001),
                           x = double(1001),
                           y = double(1001))
    for (i in seq_along(rocTable$thresh)) {
      dftmp <- data.frame(class = df$class,
                          pred_class = ifelse(df$pred_prob > rocTable$thresh[i],
                                              1, 0))
      rocTable$x[i] <- 1 - SPC(dftmp)
      rocTable$y[i] <- SNS(dftmp)
    }
  }
  return(list(plot = {plot(rocTable$x, rocTable$y, type = 'l',
                          xlab = "False Positive Rate",
                          ylab = 'True Positive Rate')
    lines(c(0, 1), c(0, 1), lty = 2)},
              AUC = sum((rocTable$y[1:1000] + rocTable$y[2:1001]) * 0.5 *
                          (rocTable$x[1:1000] - rocTable$x[2:1001]))))
}
```

The ROC and AUC for the dataset provided is shown below:



```
## $plot
## NULL
##
## $AUC
## [1] 0.850382
```

11. Metric Production

All the metrics were displayed in their sections but will be reproduced here as well.

```
ACC(class_data)
```

```
## [1] 0.8066298
```

```
CER(class_data)
```

```
## [1] 0.1933702
```

```
PRC(class_data)
```

```
## [1] 0.84375
```

```
SNS(class_data)
```

```
## [1] 0.4736842
```

```
SPC(class_data)
```

```
## [1] 0.9596774
```

```
F1(class_data)
```

```
## [1] 0.6067416
```

12. Using the caret package

```
library(caret)
confusionMatrix(as.factor(class_data$pred_class),
                as.factor(class_data$class),
                positive = '1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 119  30
##          1   5  27
##
##                Accuracy : 0.8066
##                  95% CI : (0.7415, 0.8615)
##     No Information Rate : 0.6851
##     P-Value [Acc > NIR] : 0.0001712
##
##                   Kappa : 0.4916
##
##  Mcnemar's Test P-Value : 4.976e-05
##
##             Sensitivity : 0.4737
##             Specificity : 0.9597
##          Pos Pred Value : 0.8438
##          Neg Pred Value : 0.7987
##              Prevalence : 0.3149
##          Detection Rate : 0.1492
##    Detection Prevalence : 0.1768
##       Balanced Accuracy : 0.7167
##
##        'Positive' Class : 1
##
```

```
# Precision
posPredValue(as.factor(class_data$pred_class),
            as.factor(class_data$class),
            positive = '1')
```

```
## [1] 0.84375
```

```
# Sensitivity
sensitivity(as.factor(class_data$pred_class),
            as.factor(class_data$class),
            positive = '1')
```

```
## [1] 0.4736842
```

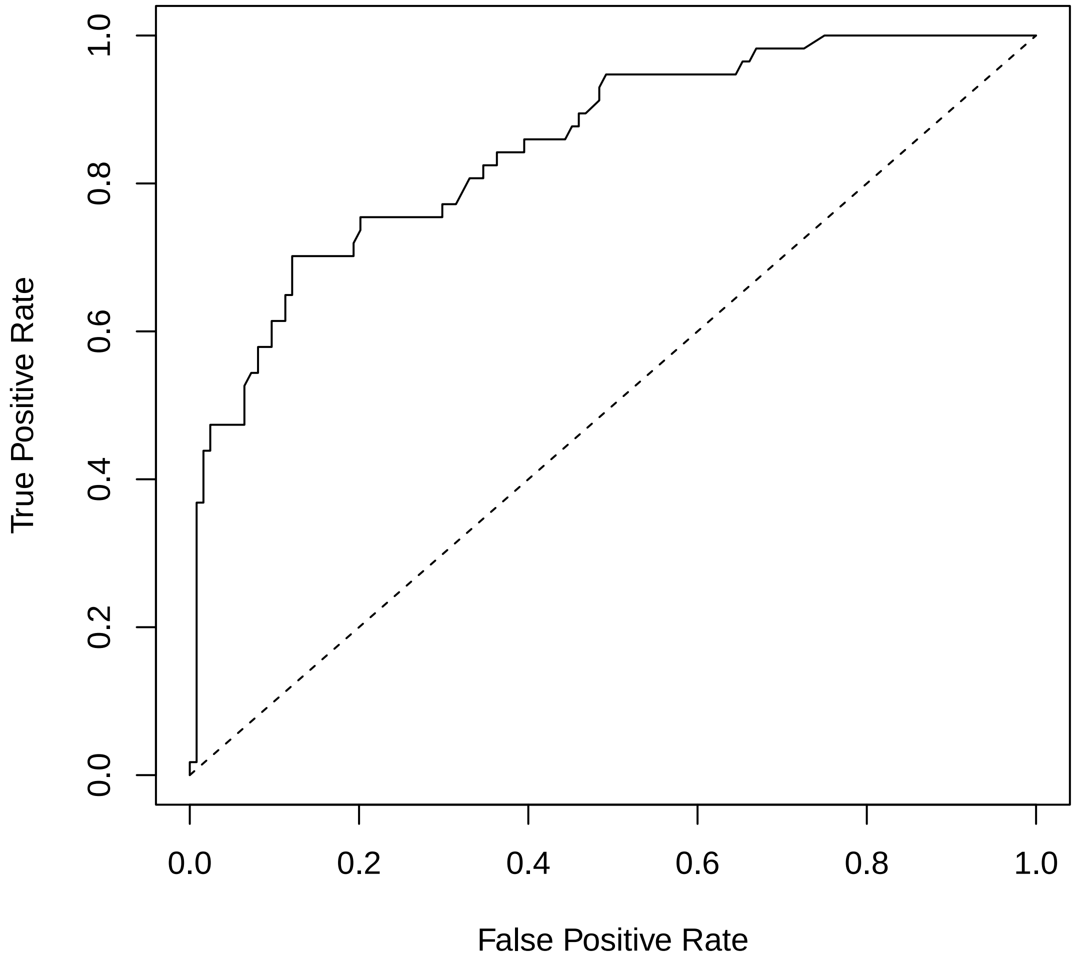
```
# Specificity
specificity(as.factor(class_data$pred_class),
            as.factor(class_data$class),
            negative = '0')
```

```
## [1] 0.9596774
```

As can be seen above, when the functions are called with proper identification of the negative and positive cases, our results match the `caret` package results.

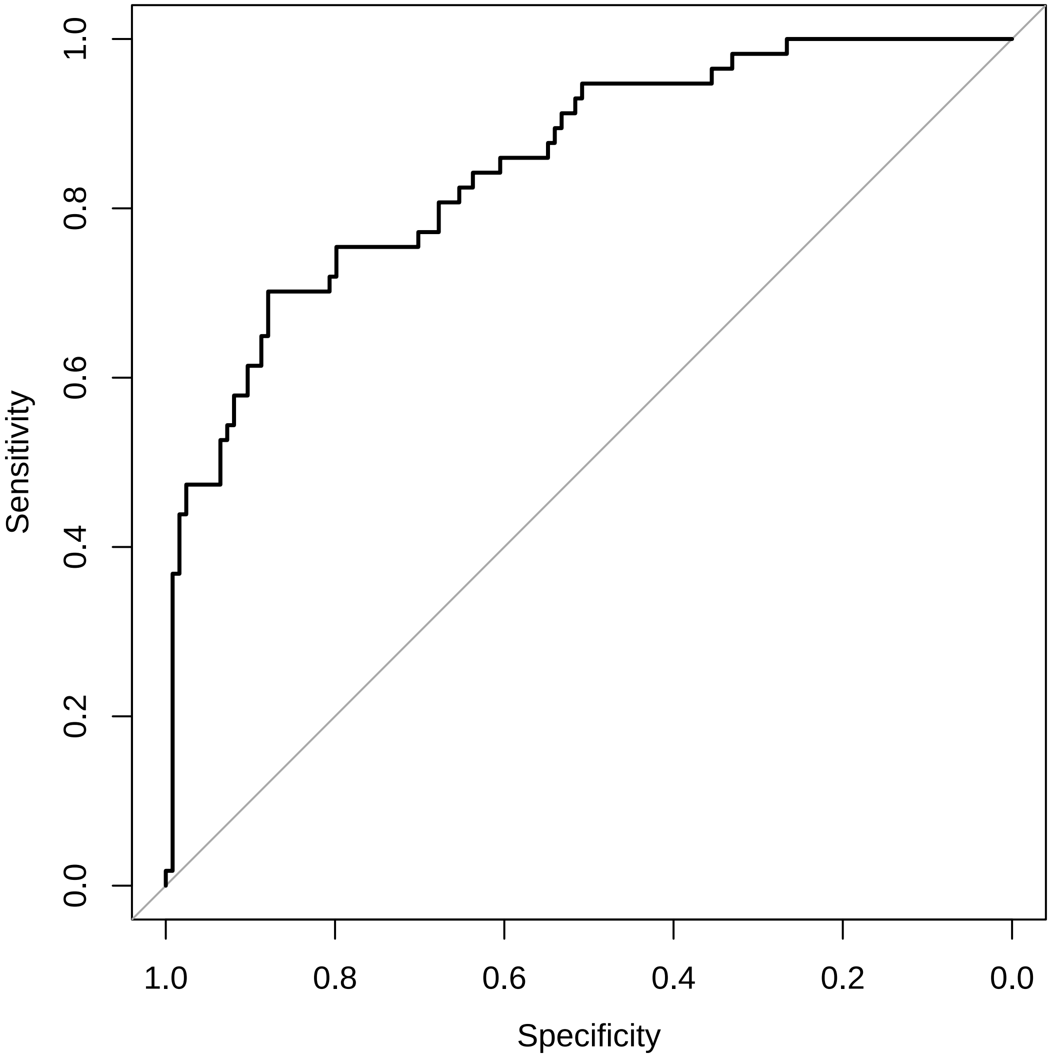
13. Using the `pROC` package

```
ROC(class_data)
```



```
## $plot
## NULL
##
## $AUC
## [1] 0.850382
```

```
library(pROC)
pROCroc <- roc(class_data, class, pred_prob)
plot(pROCroc)
```



```
auc(pROCroc)
```

```
## Area under the curve: 0.8503
```

Visual inspection of the two ROC curves shows extremely similar shapes. The AUC scores are also practically identical. Originally, we used the suggested 101 points between 0 and 1. This led to a difference of less than 1% in the AUC scores. Moving to a 1001-point quadrature all but eliminated that difference.