ROC Analysis

Johnny Lo

June 2021

Table of Contents

Introduction

Receiver Operating Characteristic (ROC) analysis is an evaluation of the diagnostic ability of a binary classifier. The ROC curve is a plot of the true positive rate (sensitivity) against the true negative rate (specificity) with its scale reversed, at various discrimination thresholds of a continuous feature. Some ROC functions plot the false positive rate (100-specificity) instead. The area under the ROC curve (AUC or AUROC) is a measure of how well a feature can distinguish two groups, e.g. normal (0)/case (1).

ROC analysis plays an important role in many research areas such as medicine and health, radiology and biometrics. In machine learning, AUC is often used as a performance measure to evaluate and compare classification models or classifiers.

To start, we will need to install and load the relevant packages for this workshop.

```
#De-comment to install the packages below
#install.packages(c("tidyverse","timeDate","caret","e1071","pROC"))
library(tidyverse)  #For ggplot2 and dplyr
library(caret)  #For penalised regression
library(pROC)  #To perform ROC analysis
```

ROC Analysis with One Feature

We are going to use the *Complication.csv* dataset to demonstrate ROC analysis in R. This dataset was used as an example in the previous module.

To start, import the data as follows. Note that we will need to re-order or re-set the reference category for the outcome variable, **Complication**.

```
#Note that you may need to change directory path
Comp <- read.csv("Complication.csv",
                 header=TRUE,
                 stringsAsFactors=TRUE,
                 na.strings="NA") %>%  #Read in the data
          na.omit()  #Complete cases only.

#Change the reference leve to "No complication"
Comp$Complication <- relevel(Comp$Complication,ref="No complication")

View(Comp)  #Examine its structure
```

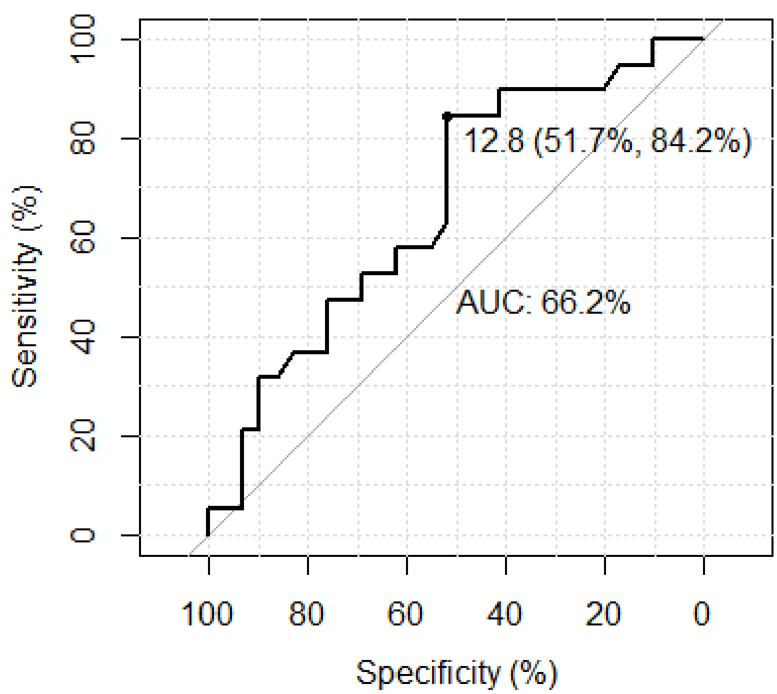
The initial goal was to derive a predictive model to determine whether a patient is likely to have complication during an appendicitis surgery based on the patient characteristics (gender and age) and blood tests (white blood cell count and C-reactive protein). However, given that this is a relatively small dataset, we will not partition it into training and test sets to derive a prediction model. Instead, we will focus on the application on ROC.

Suppose we want to determine the diagnostic ability of white blood cell count (WCC) and C-reactive protein (CRP) for determining whether a complication is likely to ensue during a appendicitis surgery.

Method 1

We begin with WCC.

```
roc.wcc <- roc(Complication~WCC,  #Formula specifying the outcome and feature of interest
               data=Comp,  #Relevant dataset
               plot=TRUE,  #Show the ROC curve
               percent=TRUE,  #Display metrics in percent
               print.auc=TRUE,  #Display area under the curve in ROC curve
               grid=TRUE,  #Show major grids in ROC curve
               print.thres="best"  #Show the threshold with the best overall accuracy
               )
```



On the ROC curve, we note that the AUC = 0.662 or 66.2%, which shows WCC is not a great diagnostic feature on its own, but it is better than a 50/50 guess. The optimal threshold or cut-off on WCC is 12.8, with a specificity of 51.7% and a sensitivity of 84.2%.

If you recall the variable **roc.wcc**, the following is displayed.

```
roc.wcc
```

```
##
## Call:
## roc.formula(formula = Complication ~ WCC, data = Comp, plot = TRUE,
percent = TRUE, print.auc = TRUE, grid = TRUE, print.thres = "best")
##
## Data: WCC in 29 controls (Complication No complication) < 19 cases
(Complication Complication).
## Area under the curve: 66.15%
```

The important information to note here is the direction “<”, which shows how the samples are classified according WCC. In this instance, we note that the *controls* are on the left of “<”, and *cases* are on the right, i.e. controls are less than cases. In other words, patients whose WCC are less than said threshold are classified as *controls*, i.e. reference outcome level or **No complication**; and those whose WCC is greater than said threshold are classified as *cases*, i.e. **Complication**. This relationship also illustrates that there is a positive relationship between WCC and probability of complication, i.e. the higher the WCC, the more likely it is that a complication will be encountered in the surgery. The reverse is true if the direction is given as “>” instead.

In a ROC curve, the optimal threshold is given as the point that is closest to the top left hand corner, i.e. specificity = sensitivity = 100%. It is not necessary unique, and may not be the optimal choice in practice. For instance in this example, the specificity is quite low in comparison to sensitivity. In certain studies, one may value specificity higher than sensitivity or prefer the two measures to be as similar as possible. In these instances, other cut-offs are preferred..

To view the results of other cut-offs, we use the *coords(.)* function. We can also specify the relevant measures to display, i.e. specificity, sensitivity, accuracy, and etc. Note that the cut-offs are given as the **average** between each pair of ordered feature values.

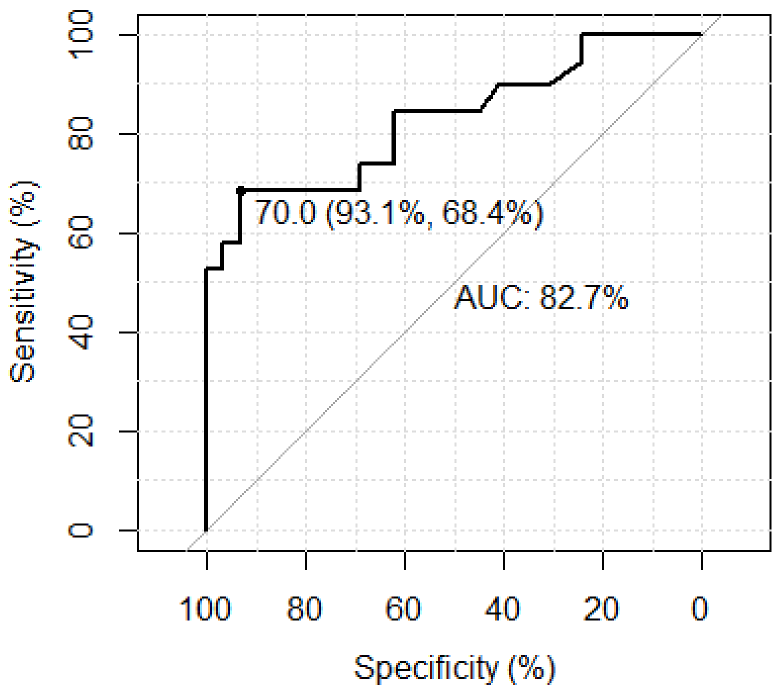
```
coords(roc.wcc,  #relevant ROC object
       "all",  #Show results of all cut-offs.
       ret=c("threshold","specificity","sensitivity","accuracy") #Measures to
diplay
       )%>%
  round(digits=3)  #Round values to 3 decimal places
```

```
##    threshold specificity sensitivity accuracy
## 1       -Inf       0.000     100.000   39.583
## 2      5.300       3.448     100.000   41.667
## 3      6.820       6.897     100.000   43.750
## 4      8.140      10.345     100.000   45.833
## 5      8.690      10.345      94.737   43.750
## 6      9.025      13.793      94.737   45.833
## 7      9.455      17.241      94.737   47.917
## 8      9.785      20.690      89.474   47.917
## 9     10.335      24.138      89.474   50.000
## 10    11.000      27.586      89.474   52.083
## 11    11.250      34.483      89.474   56.250
## 12    11.400      37.931      89.474   58.333
```

```
## 13     11.550      41.379      89.474   60.417
## 14     11.850      41.379      84.211   58.333
## 15     12.200      44.828      84.211   60.417
## 16     12.500      48.276      84.211   62.500
## 17     12.800      51.724      84.211   64.583
## 18     12.950      51.724      78.947   62.500
## 19     13.050      51.724      73.684   60.417
## 20     13.450      51.724      68.421   58.333
## 21     13.950      51.724      63.158   56.250
## 22     14.150      55.172      57.895   56.250
## 23     14.400      58.621      57.895   58.333
## 24     14.750      62.069      57.895   60.417
## 25     15.400      62.069      52.632   58.333
## 26     16.150      68.966      52.632   62.500
## 27     16.450      68.966      47.368   60.417
## 28     16.550      72.414      47.368   62.500
## 29     16.800      75.862      47.368   64.583
## 30     17.650      75.862      42.105   62.500
## 31     18.350      75.862      36.842   60.417
## 32     18.450      79.310      36.842   62.500
## 33     18.700      82.759      36.842   64.583
## 34     19.050      86.207      31.579   64.583
## 35     19.400      89.655      31.579   66.667
## 36     20.000      89.655      26.316   64.583
## 37     20.650      89.655      21.053   62.500
## 38     21.050      93.103      21.053   64.583
## 39     22.050      93.103      10.526   60.417
## 40     23.450      93.103       5.263   58.333
## 41     24.250      96.552       5.263   60.417
## 42     26.300     100.000       5.263   62.500
## 43        Inf     100.000       0.000   60.417
```

Now, we can repeat the process for CRP.

```
roc.crp <- roc(Complication~CRP,  #Formula specifying the outcome and feature
of interest
               data=Comp,  #Relevant dataset
               plot=TRUE,  #Show the ROC curve
               percent=TRUE,  #Display metrics in percent
               print.auc=TRUE,  #Display area under the curve in ROC curve
               grid=TRUE,  #Show major grids in ROC curve
               print.thres="best"  #Show the threshold with the best overall
accuracy
               );
```



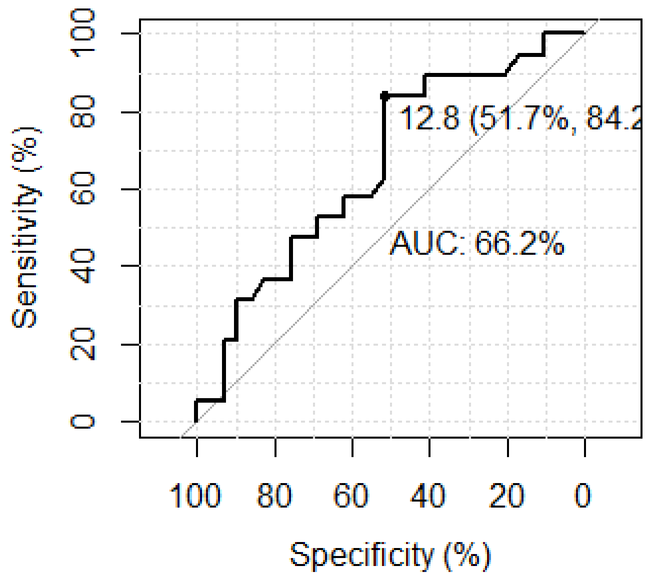
For CPR, the AUC = 0.827 or 82.7%, which is much better than WCC, whose AUC = 66.2%. The optimal cut-off is set a CRP = 70, resulting in a specificity of 93.1% and sensitivity of 68.4%. Unlike for WCC, the cut-off for CRP favour specificity more than sensitivity. However, other cut-offs can be referred to in order to increase the sensitivity, but this will be at a cost to the overall accuracy.

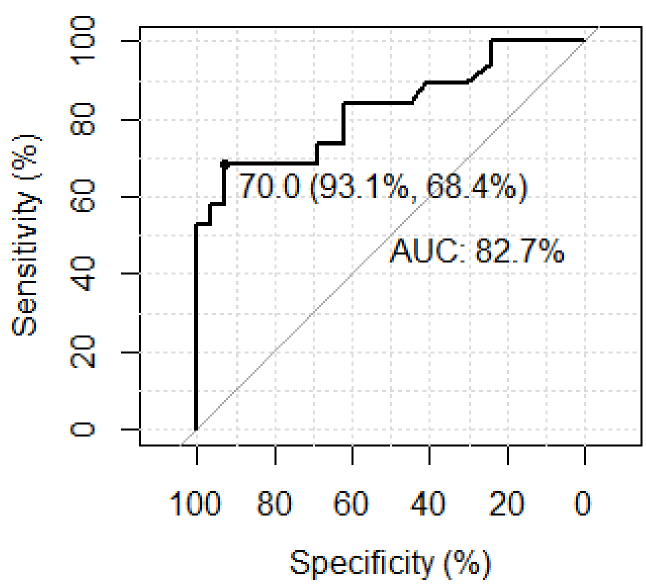
```
coords(roc.crp,  #relevant ROC object
       "all",  #Show results of all cut-offs.
       ret=c("threshold","specificity","sensitivity","accuracy") #Measures to
diplay
       )%>%
  round(digits=3)  #Round values to 3 decimal places

##    threshold specificity sensitivity accuracy
## 1       -Inf       0.000     100.000   39.583
## 2       1.40       3.448     100.000   41.667
## 3       1.95      10.345     100.000   45.833
## 4       2.35      13.793     100.000   47.917
## 5       2.60      20.690     100.000   52.083
## 6       2.85      24.138     100.000   54.167
## 7       3.95      24.138      94.737   52.083
## 8       5.50      31.034      89.474   54.167
## 9       6.50      37.931      89.474   58.333
## 10      7.50      41.379      89.474   60.417
## 11      9.50      44.828      84.211   60.417
## 12     11.50      48.276      84.211   62.500
## 13     12.50      51.724      84.211   64.583
```

```
## 14     13.50      55.172      84.211   66.667
## 15     14.50      62.069      84.211   70.833
## 16     15.50      62.069      73.684   66.667
## 17     18.50      65.517      73.684   68.750
## 18     23.50      68.966      73.684   70.833
## 19     27.00      68.966      68.421   68.750
## 20     29.00      72.414      68.421   70.833
## 21     38.00      75.862      68.421   72.917
## 22     47.00      79.310      68.421   75.000
## 23     58.50      86.207      68.421   79.167
## 24     70.00      93.103      68.421   83.333
## 25     73.00      93.103      63.158   81.250
## 26     76.50      93.103      57.895   79.167
## 27     79.00      96.552      57.895   81.250
## 28     82.50      96.552      52.632   79.167
## 29     89.50     100.000      52.632   81.250
## 30    117.00     100.000      47.368   79.167
## 31    150.00     100.000      42.105   77.083
## 32    165.00     100.000      36.842   75.000
## 33    202.00     100.000      31.579   72.917
## 34    278.00     100.000      26.316   70.833
## 35    332.00     100.000      21.053   68.750
## 36    346.00     100.000      15.789   66.667
## 37    370.00     100.000      10.526   64.583
## 38    392.50     100.000       5.263   62.500
## 39       Inf     100.000       0.000   60.417
```

If we have multiple continuous features to assess via ROC, we can actually analyse them together in one go.

```
roc.all <- roc(Complication~WCC+CRP,  #Analyse both WCC and CRP
               data=Comp,  #Relevant dataset
               plot=TRUE,  #Show the ROC curve
               percent=TRUE,  #Display metrics in percent
               print.auc=TRUE,  #Display area under the curve in ROC curve
               grid=TRUE,  #Show major grids in ROC curve
               print.thres="best"  #Show the threshold with the best overall 
accuracy
               )
```

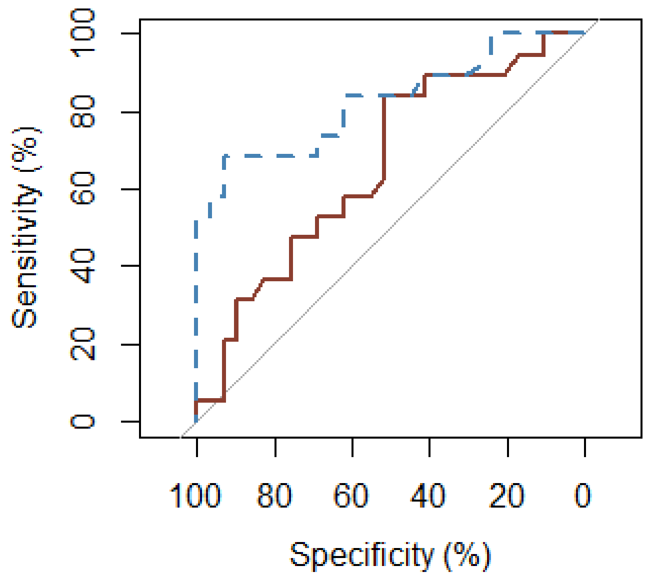




Although the formulation *Complication~WCC+CRP* typically refers to a multivariable model where the outcome is modelled as a function of the two variables conditioned on one another; however with the *roc(.)* function, the features are modelled independently. The results for the WCC and CRP in this instance are stored in list. You can check this using the *str(.)* function.

We can also plot the ROC curves on the same plot.

```
plot(roc.all$WCC,col="coral4")  #Plot ROC curve for WCC first
plot(roc.all$CRP,col="steelblue",lty=2,add=TRUE)  #Add ROC curve for CRP
```



Method 2

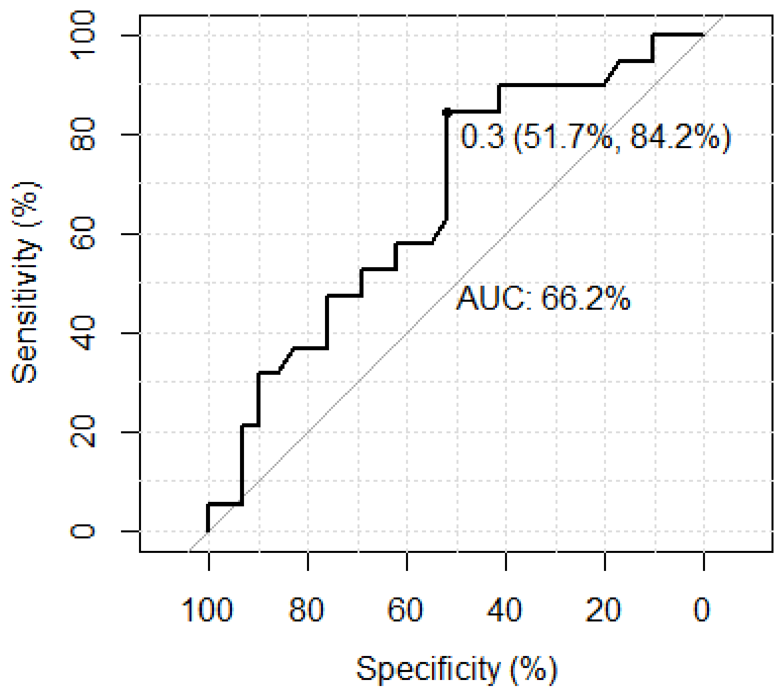
The alternate method to perform ROC analysis is by:

1) Performing logistic regression with the continuous feature and obtain the predicted probabilities corresponding to each of the sampled values of the continuous feature.
2) Perform ROC analysis on the predicted probabilities.

```
#Alternate approach
mod.wcc <- glm(Complication~WCC, data=Comp, family="binomial"); #Logistic
regression
summary(mod.wcc)  #Summary of the model

##
## Call:
## glm(formula = Complication ~ WCC, family = "binomial", data = Comp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4611  -0.9643  -0.7645   1.1833   1.7042
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.16562    1.02572  -2.111   0.0347 *
## WCC          0.11476    0.06365   1.803   0.0714 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 64.443  on 47  degrees of freedom
```

```
## Residual deviance: 60.889  on 46  degrees of freedom
## AIC: 64.889
##
## Number of Fisher Scoring iterations: 4
```

```
pred.wcc <- predict(mod.wcc,type="response");  #Predicted probabilities
roc.wcc.alt <- roc(Complication~pred.wcc,
                   data=Comp,
                   plot=TRUE,
                   percent=TRUE,
                   print.auc=TRUE,
                   grid=TRUE,
                   print.thres="best")
```




```
summ <- coords(roc.wcc.alt,
               "all",  #Show results of all cut-offs.
               ret=c("threshold","specificity","sensitivity","accuracy")
              )
summ[order(summ$accuracy,decreasing=TRUE),]
```

```
##    threshold specificity sensitivity accuracy
## 35 0.5151782   89.655172   31.578947 66.66667
## 17 0.3325534   51.724138   84.210526 64.58333
## 29 0.4408761   75.862069   47.368421 64.58333
## 33 0.4951026   82.758621   36.842105 64.58333
## 34 0.5051428   86.206897   31.578947 64.58333
## 36 0.5323368   89.655172   26.315789 64.58333
```

```
## 38 0.5621953   93.103448   21.052632 64.58333
## 16 0.3249716   48.275862   84.210526 62.50000
## 18 0.3363815   51.724138   78.947368 62.50000
## 26 0.4225854   68.965517   52.631579 62.50000
## 28 0.4338092   72.413793   47.368421 62.50000
## 30 0.4650826   75.862069   42.105263 62.50000
## 32 0.4879317   79.310345   36.842105 62.50000
## 37 0.5508606   89.655172   21.052632 62.50000
## 42 0.6993211  100.000000    5.263158 62.50000
## 13 0.3015084   41.379310   89.473684 60.41667
## 15 0.3174505   44.827586   84.210526 60.41667
## 19 0.3389481   51.724138   73.684211 60.41667
## 24 0.3839388   62.068966   57.894737 60.41667
## 27 0.4309927   68.965517   47.368421 60.41667
## 31 0.4850648   75.862069   36.842105 60.41667
## 39 0.5900080   93.103448   10.526316 60.41667
## 41 0.6495828   96.551724    5.263158 60.41667
## 43       Inf  100.000000    0.000000 60.41667
## 12 0.2978998   37.931034   89.473684 58.33333
## 14 0.3088404   41.379310   84.210526 58.33333
## 20 0.3493623   51.724138   68.421053 58.33333
## 25 0.4017978   62.068966   52.631579 58.33333
## 40 0.6283183   93.103448    5.263158 58.33333
## 23 0.3744911   58.620690   57.894737 58.33333
## 11 0.2943078   34.482759   89.473684 56.25000
## 21 0.3624687   51.724138   63.157895 56.25000
## 22 0.3677807   55.172414   57.894737 56.25000
## 10 0.2884059   27.586207   89.473684 52.08333
## 9  0.2731062   24.137931   89.473684 50.00000
## 8  0.2606368   20.689655   89.473684 47.91667
## 7  0.2534378   17.241379   94.736842 47.91667
## 4  0.2260260   10.344828  100.000000 45.83333
## 6  0.2442004   13.793103   94.736842 45.83333
## 3  0.2010704    6.896552  100.000000 43.75000
## 5  0.2371680   10.344828   94.736842 43.75000
## 2  0.1742442    3.448276  100.000000 41.66667
## 1       -Inf    0.000000  100.000000 39.58333
```

Note that the AUC = 66.2% is **exactly** the same as the previous approach, and with identical specificity and sensitivity at the optimal cut-off. The difference here is that the cut-off is given at a particular probability, i.e. 0.333 (although the plot shows 0.3), instead of a WCC value, but we can easily map the probability back to the actual WCC value.

```r
#Add the predicted probabilities to data frame
df <- cbind(Comp,Probability=pred.wcc)

#Sort the data frame in accordance to the probabilities in ascending order
df[order(df$Probability,decreasing=FALSE),]
```

```
##    Subject Gender Age   WCC   CRP    Complication Probability
## 22      23   Male  23  4.70   8.0 No complication   0.1643458
## 42      43   Male  44  5.90   2.3 No complication   0.1841425
## 8        8 Female  48  7.74  28.0 No complication   0.2179983
## 37      38   Male  71  8.54 160.0    Complication   0.2340537
## 12      12   Male  23  8.84   1.6 No complication   0.2402822
## 2        2   Male  22  9.21  11.0 No complication   0.2481186
## 3        3   Male  37  9.70  75.0    Complication   0.2587570
## 21      22   Male  16  9.70   6.0 No complication   0.2587570
## 14      14 Female  44  9.87   2.4 No complication   0.2625165
## 33      34   Male  31 10.80  48.0 No complication   0.2836958
## 24      25 Female  29 11.20  48.0 No complication   0.2931160
## 38      39 Female  18 11.20  85.0 No complication   0.2931160
## 27      28   Male  21 11.30  78.0 No complication   0.2954995
## 10      10 Female  14 11.50   5.0 No complication   0.3003000
## 32      33   Male  18 11.60  71.0    Complication   0.3027169
## 44      45   Male  44 12.10  30.0 No complication   0.3149639
## 25      26 Female  26 12.30   6.0 No complication   0.3199370
## 18      18 Female  21 12.70  69.0 No complication   0.3300062
## 11      11 Female  33 12.90 234.0    Complication   0.3351006
## 41      42 Female  38 13.00 395.0    Complication   0.3376624
## 29      30   Male  15 13.10  15.0    Complication   0.3402338
## 36      37   Male  45 13.80   5.0    Complication   0.3584909
## 17      17 Female  42 14.10  12.0 No complication   0.3664465
## 45      46   Male  65 14.10 350.0    Complication   0.3664465
## 4        4   Male  55 14.20  46.0 No complication   0.3691149
## 50      51 Female  16 14.60   1.6 No complication   0.3798673
## 39      40   Male  43 14.90   2.9    Complication   0.3880103
## 6        6 Female  15 15.90  14.0 No complication   0.4155853
## 16      16   Male  21 15.90   7.0 No complication   0.4155853
## 49      50 Female  60 16.40 140.0    Complication   0.4295855
## 30      31   Male  32 16.50   1.2 No complication   0.4323999
## 47      48   Male  18 16.60  14.0 No complication   0.4352186
## 1        1 Female  18 17.00 390.0    Complication   0.4465337
## 43      44   Male  51 18.30 342.0    Complication   0.4836316
## 26      27 Female  84 18.40  69.0 No complication   0.4864980
## 34      35 Female  33 18.50  16.0 No complication   0.4893653
## 7        7   Male  17 18.90   2.4 No complication   0.5008398
## 48      49   Male  39 18.90  26.0    Complication   0.5008398
## 40      41   Male  37 19.20  13.0 No complication   0.5094457
## 15      15   Male  53 19.60  15.0    Complication   0.5209107
## 9        9 Female  56 20.40  80.0    Complication   0.5437628
## 20      21   Male  40 20.90   5.0 No complication   0.5579584
## 13      13   Male  42 21.20 322.0    Complication   0.5664321
## 46      47   Male  20 21.20 170.0    Complication   0.5664321
## 23      24   Male  26 22.90  94.0    Complication   0.6135838
## 35      36   Male  43 24.00   2.8 No complication   0.6430527
## 5        5   Male  48 24.50  21.0 No complication   0.6561128
## 31      32   Male  17 28.10   8.0    Complication   0.7425295
```

In the above table, we can see that the optimal probability cut-off of 0.333, is between 0.330 and 0.335 for Subjects 18 and 11, respectively. Their corresponding WCCs are 12.7 and 12.9, and the average between the two is 12.8, i.e. the WCC cut-off.

Exercise: Try this approach for CRP and see if you can get the same result as the previous method. What is the optimal probability cut-off for CRP in this instance?

Thus far, it seems the alternate method is more arduous than modelling the features directly with the *roc(.)* function, so why bother?

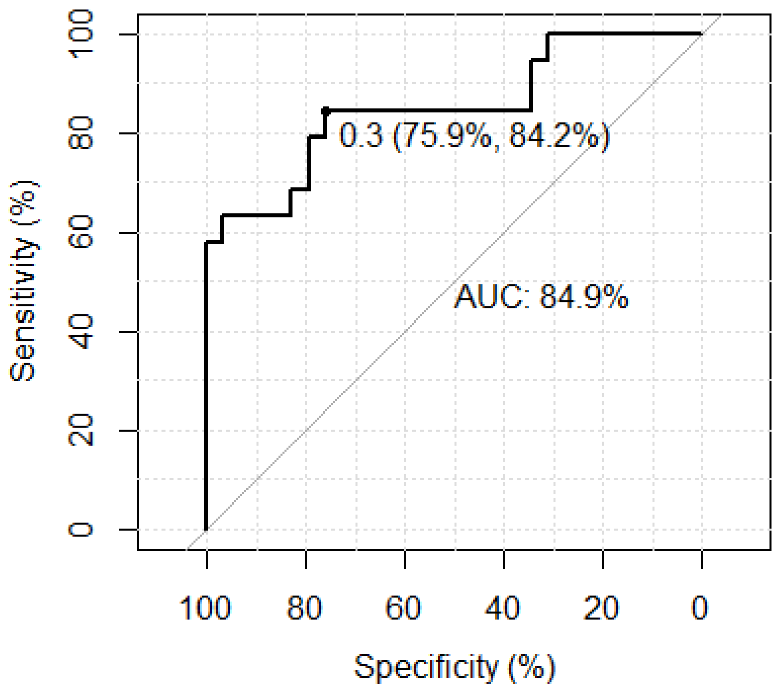
The advantage of this method is that it actually allows us to model the binary outcome in a multivariable manner with 2 or more features.

ROC Analysis with Two or More Features

Suppose we want to model utilise WCC and CRP **jointly** and see if the two features can improve on the previous ROC models modelled on the features individually.

```
#Logistic regression model with WCC and CRP
mod.joint <- glm(Complication~WCC+CRP, data=Comp, family="binomial");
summary(mod.joint)  #Summary of the joint model

##
## Call:
## glm(formula = Complication ~ WCC + CRP, family = "binomial",
##     data = Comp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3938  -0.6920  -0.3864   0.2710   2.0687
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.19990    1.51851  -2.766  0.00568 **
## WCC          0.14801    0.08067   1.835  0.06656 .
## CRP          0.02858    0.01080   2.645  0.00816 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 64.443  on 47  degrees of freedom
## Residual deviance: 39.314  on 45  degrees of freedom
## AIC: 45.314
##
## Number of Fisher Scoring iterations: 6
```

```
pred.joint <- predict(mod.joint,type="response");  #Predicted probabilities
roc.joint <- roc(Complication~pred.joint,
                 data=Comp,
                 plot=TRUE,
                 percent=TRUE,
                 print.auc=TRUE,
                 grid=TRUE,
                 print.thres="best")
```



For the multivariable ROC model, the AUC = 84.9% has improved as compared to CRP alone (AUC = 82.7%). This improvement may be marginally, but depending on the application, this may amount to saving one more life, and/or millions of dollars. In particular here, the sensitivity has increased substantially to 84.2% as compared to the CRP ROC model at 68.4%. Furthermore, there is a greater balance between the specificity (75.9%) and sensitivity (84.2%) at the optimal cut-off. Again, be careful with the lack of decimal places for the optimal cut-off in the above plot. We should examine the full result to identify this threshold more precisely.

```
res <- coords(roc.joint,  #relevant ROC object
              "all",  #Show results of all cut-offs.
              ret=c("threshold","specificity","sensitivity","accuracy")
              )%>%
         round(digits=3)  #Round values to 3 decimal places

#Order the results according to the accuracy
res[order(res$accuracy,decreasing=TRUE),]
```

```
##    threshold specificity sensitivity accuracy
## 36     0.527      96.552      63.158   83.333
## 38     0.686     100.000      57.895   83.333
## 35     0.489      93.103      63.158   81.250
## 37     0.584      96.552      57.895   81.250
## 39     0.794     100.000      52.632   81.250
## 26     0.286      75.862      84.211   79.167
## 28     0.327      79.310      78.947   79.167
## 34     0.449      89.655      63.158   79.167
## 40     0.852     100.000      47.368   79.167
## 25     0.274      72.414      84.211   77.083
## 27     0.304      75.862      78.947   77.083
## 29     0.345      79.310      73.684   77.083
## 31     0.375      82.759      68.421   77.083
## 33     0.420      86.207      63.158   77.083
## 41     0.885     100.000      42.105   77.083
## 24     0.270      68.966      84.211   75.000
## 30     0.356      79.310      68.421   75.000
## 32     0.401      82.759      63.158   75.000
## 42     0.940     100.000      36.842   75.000
## 23     0.252      65.517      84.211   72.917
## 43     0.983     100.000      31.579   72.917
## 22     0.232      62.069      84.211   70.833
## 44     0.994     100.000      26.316   70.833
## 21     0.217      58.621      84.211   68.750
## 45     1.000     100.000      21.053   68.750
## 20     0.208      55.172      84.211   66.667
## 46     1.000     100.000      15.789   66.667
## 19     0.199      51.724      84.211   64.583
## 47     1.000     100.000      10.526   64.583
## 18     0.183      48.276      84.211   62.500
## 48     1.000     100.000       5.263   62.500
## 17     0.168      44.828      84.211   60.417
## 49       Inf     100.000       0.000   60.417
## 10     0.108      31.034     100.000   58.333
## 12     0.124      34.483      94.737   58.333
## 16     0.157      41.379      84.211   58.333
## 9      0.097      27.586     100.000   56.250
## 11     0.119      31.034      94.737   56.250
## 13     0.133      34.483      89.474   56.250
## 15     0.148      37.931      84.211   56.250
## 8      0.091      24.138     100.000   54.167
## 14     0.142      34.483      84.211   54.167
## 7      0.080      20.690     100.000   52.083
## 6      0.072      17.241     100.000   50.000
## 5      0.067      13.793     100.000   47.917
## 4      0.060      10.345     100.000   45.833
## 3      0.046       6.897     100.000   43.750
## 2      0.037       3.448     100.000   41.667
## 1       -Inf       0.000     100.000   39.583
```

From the above results, it appears that the optimal probability cut-off is given at 0.286, with an overall accuracy of 79.2. The other candidate cut-off could be 0.327 as it has better specificity-sensitivity balance, with the same accuracy.

Notice how this cut-off is not the threshold that maximises the overall accuracy. These other cut-offs with higher overall accuracy tend to favour specificity (93-100%) substantially more than sensitivity (52-63%).

Evaluating and Comparing ML Models Using AUC

In cases where you are dealing with *imbalanced* data, it often inappropriate to evaluate and compare the predictive power of your machine learning models based on their overall accuracy. In such cases, one metric (typically specificity) will dominate the other just on the fact that the overwhelming majority of the samples (e.g. 99%) fall in one class. For example, in the detection of rare diseases or fraud cases, most of the cases will be “normal”. Given this, it is often difficult to distinguish your models if one has 99.1% accuracy and the other has 99.3% accuracy. Furthermore, accuracy is based on a **single** cut-off.

On the other hand, AUC is *insensitive* to imbalanced data and it is an aggregate measure across all cut-offs and therefore, is a better measure for these situations..

Here, we will revisit the COVID-19 data and import the **cleaned** training and test sets. Note that the proportion of cases resulting in deaths is less than 10%.

```
COVID.train <- read.csv("COVID-19_Train.csv", header=TRUE,
stringsAsFactors=TRUE);
COVID.test <- read.csv("COVID-19_Test.csv", header=TRUE,
stringsAsFactors=TRUE)
```

To start, we will construct a binary logistic regression model based on the available features in the training set.

```
mod.covid.lg <- glm(death~., family="binomial",data=COVID.train);
summary(mod.covid.lg)  #Summarise the model

##
## Call:
## glm(formula = death ~ ., family = "binomial", data = COVID.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8092  -0.3058  -0.1438  -0.0509   3.5150
##
## Coefficients:
##                   Estimate Std. Error z value        Pr(>|z|)
## (Intercept)       -9.90071    1.41131  -7.015 0.0000000000023 ***
## age                0.11617    0.01873   6.203 0.0000000005542 ***
```

```
## regionHong Kong   -3.06508    0.82298  -3.724        0.000196 ***
## regionJapan       -3.37803    0.65596  -5.150 0.0000002608459 ***
## regionSouth Korea  0.66035    0.66078   0.999        0.317625
## gendermale         1.48519    0.49471   3.002        0.002681 **
## hosp_visitYes      0.77433    0.56429   1.372        0.169999
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 268.85  on 420  degrees of freedom
## Residual deviance: 157.30  on 414  degrees of freedom
## AIC: 171.3
##
## Number of Fisher Scoring iterations: 7
```

Next, we will determine *accuracy* and *AUC* of the binary logistic regression for both the **training** and **test** sets. The approach here is identical to "Method 2" in the previous section whereby the probabilities are initially predicted from the model, and later analysed with ROC curve. This approach is also applicable to other probability-based binary classifiers, such as penalised logistic regression and partial least squares discriminant analyses.

We begin with the **training** set.

```
#Predicted probabilites on the training set
pred.prob.lg.train <- predict(mod.covid.lg,type="response")
pred.class.lg.train <- ifelse(pred.prob.lg.train<0.5,"No","Yes")  #Predicted
classes
acc.lg.train <- mean(pred.class.lg.train==COVID.train$death);  #Accuracy

#ROC analysis of the model on the training set.
roc.lg.train <- roc(death~pred.prob.lg.train, data=COVID.train)
auc.lg.train <- roc.lg.train$auc

#Output the accuracy and AUC of the model on the training set
c(Acc_Train=acc.lg.train, AUC_Train=auc.lg.train)

## Acc_Train AUC_Train
## 0.9216152 0.9192555
```

There is little difference between the accuracy and the AUC of the model for the training set, which are 92.2% and 91.9%, respectively.

For the **test** set, the results are as follows:

```
#Predicted probabilites on the test set
pred.prob.lg.test <- predict(mod.covid.lg, newdata=COVID.test,
type="response")
pred.class.lg.test <- ifelse(pred.prob.lg.test<0.5,"No","Yes")  #Predicted
classes
acc.lg.test <- mean(pred.class.lg.test==COVID.test$death);  #Accuracy
```

```
#ROC analysis of the model on the test set.
roc.lg.test <- roc(death~pred.prob.lg.test, data=COVID.test)
auc.lg.test <- roc.lg.test$auc

#Output the accuracy and AUC of the model on the test set
c(Acc_Test=acc.lg.test, AUC_Test=auc.lg.test)

##  Acc_Test  AUC_Test
## 0.9280576 0.8898046
```

Here, the difference between the accuracy and the AUC of the model, which are 92.8% and 89.0%, respectively, for the test set is just under 4%. Overall, the performance of the model across both training and test sets is similar.

Note: if the accuracy/AUC of the model for the training set is significantly greater than those for the test set, then this is an indication of *overfitting*.

Let us now compare the logistic regression model to the LASSO logistic regression model. First, we will tune the LASSO regression model as before.

```
set.seed(1)
lambdas <- 10^seq(-3,3,length=100) #A sequence 100 lambda values
mod.covid.LASSO <- train(death ~., #Formula
                   data = COVID.train, #Training data
                   method = "glmnet",  #Penalised regression modelling
                   #Set preProcess to c("center", "scale") to standardise
data
                   preProcess = NULL,
                   #Perform 10-fold CV, 5 times over.
                   trControl = trainControl("repeatedcv",
                                            number = 10,
                                            repeats = 5),
                   tuneGrid = expand.grid(alpha = 1, #LASSO regression
                                          lambda = lambdas)
                  )
# Model coefficients
coef(mod.covid.LASSO$finalModel, mod.covid.LASSO$bestTune$lambda)

## 7 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)       -6.92364206
## age                0.08095509
## regionHong Kong   -1.85046732
## regionJapan       -2.25404598
## regionSouth Korea  .
## gendermale         0.92335028
## hosp_visitYes      0.09817193
```

Now, we are ready to evaluate the performance of the LASSO regression model.

```
#Predicted probabilities of death on the training and test sets
#Note that the probabilities for both classes are provided. We use the "Yes"
only.
pred.prob.LASSO.train <- predict(mod.covid.LASSO, type="prob")$Yes
pred.prob.LASSO.test <- predict(mod.covid.LASSO, newdata=COVID.test,
type="prob")$Yes

#Predicted classes of death on the training and test sets
pred.class.LASSO.train <- predict(mod.covid.LASSO)
pred.class.LASSO.test <- predict(mod.covid.LASSO,newdata=COVID.test)

#Accuracy of the LASSO model on the training and test sets
acc.LASSO.train <- mean(pred.class.LASSO.train==COVID.train$death);
acc.LASSO.test <- mean(pred.class.LASSO.test==COVID.test$death);

#ROC analysis of the LASSO model on the training and test sets
roc.LASSO.train <- roc(death~pred.prob.LASSO.train, data=COVID.train)
auc.LASSO.train <- roc.LASSO.train$auc

roc.LASSO.test <- roc(death~pred.prob.LASSO.test, data=COVID.test)
auc.LASSO.test <- roc.LASSO.test$auc

#Output the accuracy and AUC of the LASSO regression model
c(Acc_Train=acc.LASSO.train,
  AUC_Train=auc.LASSO.train,
  Acc_Test=acc.LASSO.test,
  AUC_Test=auc.LASSO.test)

## Acc_Train AUC_Train  Acc_Test  AUC_Test
## 0.9192399 0.9137035 0.9136691 0.8949939

#Recall the accuracy and AUC of the binary logistic regression model
c(Acc_Train=acc.lg.train,
  AUC_Train=auc.lg.train,
  Acc_Test=acc.lg.test,
  AUC_Test=auc.lg.test)

## Acc_Train AUC_Train  Acc_Test  AUC_Test
## 0.9216152 0.9192555 0.9280576 0.8898046
```

The accuracy of the LASSO regression model on the test set (91.4%) is marginally lower than the binary logistic regression model (92.8%). The AUCs of the models (LASSO logistic 89.5% VS binary logistic 89.0%) are also similar, suggesting that the LASSO model is comparable to binary logistic regression model when all other probability cut-offs are considered instead of just the default cut-off of 0.5.