Project 2

Achraf Cherkaoui

Classification

In this classification setting, we use a **wine** dataset of chemical measurement of two variables, `Color_intensity` and `Alcalinity_of_ash`, on 130 wines from two cultivars in a region in Italy.

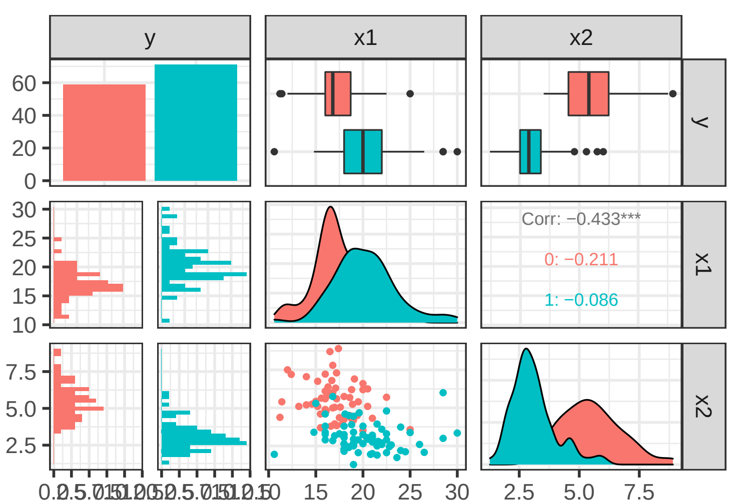
The data set is a subset of a data set from https://archive.ics.uci.edu/ml/datasets/Wine, see that page or http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.names for information of the source of the data.

First, read the original data in, keep the response `class`, and predictors `Alcalinity_of_ash` and `Color_intensity`. We only use 2 classes (there are 3 classes in the original dataset) and we re-code them to be $y = 0$ or 1. Also, we rename `Alcalinity_of_ash` and `Color_intensity` to be x_1 and x_2.

Then, we make plot and visualize the relation between the variables. Look at **the pairwise correlation between x_1, x_2, and y.**

```
library(ggplot2)
library(GGally)

wine = read.table(file = "http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",
                  sep = ",", head=F)
colnames(wine) = c("class", "Alcohol", "Malic_acid", "Ash", "Alcalinity_of_ash", "Magnesium",
                   "Total_phenols", "Flavanoids", "Nonflavanoid_phenols", "Proanthocyanins",
                   "Color_intensity", "Hue", "OD280/OD315_of_diluted wines", "Proline")
wine = wine[which(wine$class!=3),c(1,5,11)]
wine$class=as.factor(wine$class-1)
colnames(wine)=c("y","x1","x2")
ggpairs(wine, ggplot2::aes(color=y)) + theme_bw(18)
```



Obviously, the data is a roughly **balanced** dataset.

Then, we would like to use **Logistic regression**, **LDA**, and **KNN** methods to estimate the test error rates. To do so, we will use the **Validation Set** approach. So, now we split the dataset to be the `train` and `test` datasets.

```
n <- nrow(wine)
set.seed(43855385) # You can change the seed to your favorite number
reorder = sample(1:n) # shuffle
test = wine[reorder[1:(n/2)],]
train = wine[reorder[((n/2)+1):n],]
```

a) Logistic Regression

Question 1: Fit a **logistic regression** model on $y \sim x1 + x2$ to the training set and name the result **LR.Wine**. Show the summary of **LR.Wine**.

```
LR.Wine <- glm(y~x1+x2, data=train, family="binomial")
summary(LR.Wine)
```

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7076  -0.2771   0.0715   0.3319   2.5712
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1543     2.7727   0.056 0.955619
## x1            0.4309     0.1636   2.634 0.008441 **
## x2           -1.8542     0.5030  -3.686 0.000228 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 89.354  on 64  degrees of freedom
## Residual deviance: 39.921  on 62  degrees of freedom
## AIC: 45.921
##
## Number of Fisher Scoring iterations: 6
```

Question 2: Based on your estimates of β_1 and β_2. Give their specific interpretation in words.

Answer: β_1= 0.4309, with one unite increase in x_1 leads to an increases of 0.4309 in the log-odds of class. β_1= 0.4309, then eβ_1 = 1.5386 and the interpretation becomes: An increase of one unite in x_1 multiplies the odds of class by 1.53886. an increase of one unite in x_1 is associated with an increase of 53.8% in the odds of class. increases by 53.8% (1.5386 - 1 = 0.5386).

β_2= -1.8542 , with one unite increase in x_2 leads to a decreases of 1.8542 in the log-odds of class. β_2= -1.8542 , then eβ_1 = 0.1566 and the interpretation becomes : an increase of one unite in x_2 is associated with a decrease of 84.3% in the odds of class. decreases by 84.3% (1- 0.1566 = 0.8434).

Bonus Question 3: (Bonus question for both Math 4385 and 5385)**: Use the estimates $\hat{\beta}_1$ and $\hat{\beta}_2$ to manually find** the equation of the decision boundary** in the form of $x_2 = \text{intercept} + \text{slope} \times x_1$ using the decision rule $\hat{P}(Y = 1|x) = 0.5$ (Hint: at 0.5 decision boundary, the logit $= \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \hat{\beta}_2 \cdot x_2 = 0$). Then make a plot for (x_1, x_2) of both the training data with classes (i.e., y) color-coded and the decision boundary.

Answer:

The decision boundary can be derived as follows:

$\hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \hat{\beta}_2 \cdot x_2 = 0$

$x_2 = \hat{\beta}_0/(-\hat{\beta}_2) + \hat{\beta}_1/(-\hat{\beta}_2) \cdot x_1$

$x_2 = 0.154/-(-1.854) + 0.431/-(-1.854) \cdot x_1$

$x_2 = 0.831 + 0.232 \cdot x_1$

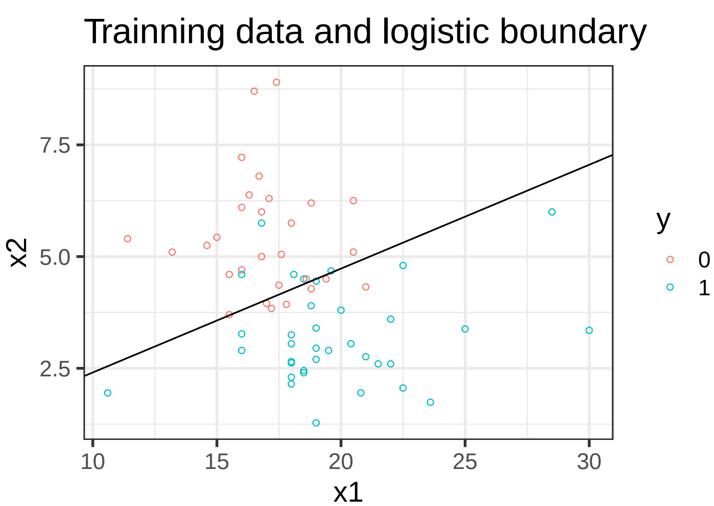
-intercept = 0.0831.

-slope = 0.232 .

The plot is shown below (**Note**: To let R run the following chunk, change the following setting `eval=FALSE` to be `eval=TRUE`)

```
slope_0 <-coef(LR.Wine)[2]/(-coef(LR.Wine)[3])
intercept_0 <- coef(LR.Wine)[1]/(-coef(LR.Wine)[3])

g1 = ggplot(data=train, aes(x=x1, y=x2, colour=y)) +
       geom_point(pch = 1) + theme_bw(18) # Plot the training data
g1 + geom_abline(slope = slope_0,intercept=intercept_0) +
     ggtitle("Trainning data and logistic boundary") +
     theme_bw(18) # Add the decision boundary
```




```
     # Change the slope_0 and intercept_0 to be the values found in your derived decision boundary.
```

Question 4: Use the `predict()` function to calculate the predicted probabilities for all observations in the test set. Name the results **'LR.pred.prob'.**

```
LR.pred.prob <- predict(LR.Wine,newdata = test ,type="response")
LR.pred.prob
```

```
##           55           96          127           51          108          128
## 0.0259336623 0.9712645472 0.8921708092 0.0003885656 0.9765848491 0.9989649873
##           87           89           79            4          129           20
## 0.9956620141 0.9862104779 0.5565972244 0.0008501691 0.9988654669 0.0599523997
##          119           83           73           81          104           41
```

```
## 0.6779656079 0.9983684907 0.9723793956 0.9760111518 0.9913127254 0.0143254812
##            6          110          105          114           45           32
## 0.0029831974 0.9793394884 0.9717056454 0.9704665580 0.1340613408 0.0120162650
##           88            1          125           97           52          112
## 0.9985528884 0.0270860454 0.9822144495 0.9916996643 0.0563764023 0.9959085869
##           35           18          103           80            2          121
## 0.6350130004 0.0303144970 0.9822144495 0.9950315947 0.0414491571 0.9396954240
##           15           13          101           93           27           67
## 0.0001875512 0.0343980574 0.8286656104 0.9682811173 0.1408412591 0.0388137292
##           21           31           69           38          106          115
## 0.0314481930 0.3275368183 0.8322690729 0.5075442076 0.9903178003 0.9887084990
##          123           70           25           47           23           34
## 0.9995547181 0.8917702519 0.9042633964 0.1153901646 0.5649570120 0.1891134577
##           53            3           17           63           76           46
## 0.0010218686 0.0860571212 0.0615907259 0.7036122486 0.5006918345 0.1950302365
##           16          116            9           26           75
## 0.0025482984 0.9972561091 0.0306238908 0.9864636542 0.9627165917
```

Question 5: Use 0.5 as cutoff to make predictions for class (i.e., y) on test data and name the prediction object as `LR.pred`.

```
LR.pred <- ifelse(LR.pred.prob > 0.5 , 1,0)
LR.pred
```

```
##  55  96 127  51 108 128  87  89  79   4 129  20 119  83  73  81 104  41   6 110
##   0   1   1   0   1   1   1   1   1   0   1   0   1   1   1   1   1   0   0   1
## 105 114  45  32  88   1 125  97  52 112  35  18 103  80   2 121  15  13 101  93
##   1   1   0   0   1   0   1   1   0   1   1   0   1   1   0   1   0   0   1   1
##  27  67  21  31  69  38 106 115 123  70  25  47  23  34  53   3  17  63  76  46
##   0   0   0   0   1   1   1   1   1   1   1   0   1   0   0   0   0   1   1   0
##  16 116   9  26  75
##   0   1   0   1   1
```

Question 6: Use make the **confusion matrix**, find the test error rate, sensitivity, specificity.

Answer 6: the test error rate is **9.23** , the sensitivity **97.1** , the specificity **83.3**

```
LR.table <- table(test$y,LR.pred )
LR.table
```

```
##    LR.pred
##      0  1
##   0 25  5
##   1  1 34
```

```
sensitivity <- (34/35)*100
specificity <- (25/30)*100
sensitivity
```

```
## [1] 97.14286
```

```
specificity
```

```
## [1] 83.33333
```

```
testerrorrate <- (6/65)*100
testerrorrate
```

```
## [1] 9.230769
```

b) LDA

Question 1: Perform LDA method (use `lda()`) to the training data and name the resulting object `LDA.Wine`. Show `LDA.Wine` (Note: Not the summary of `LDA.Wine` as we did for linear or logistic regression.)

```
library(MASS)
LDA.Wine <- lda(y~ x1+x2 , data = train)
LDA.Wine
```

```
## Call:
## lda(y ~ x1 + x2, data = train)
##
## Prior probabilities of groups:
##         0         1
## 0.4461538 0.5538462
##
## Group means:
##         x1       x2
## 0 17.01724 5.434828
## 1 19.71389 3.233056
##
## Coefficients of linear discriminants:
##           LD1
## x1  0.1711455
## x2 -0.7482680
```

Question 2: Use the `predict()` function to calculate the predicted class for the test set (0.5 is the default cutoff).

```
la.pre <- predict(LDA.Wine ,test)
la.pre
```

```
## $class
##  [1] 0 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 0 1 1 0 1 1 0 1 0 0
## [39] 1 1 0 0 0 0 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 0 0 1 0 1 1
## Levels: 0 1
##
## $posterior
##               0            1
## 55  0.947138910 0.0528610904
## 96  0.039864163 0.9601358375
## 127 0.122465021 0.8775349787
## 51  0.998437730 0.0015622701
## 108 0.034216158 0.9657838419
## 128 0.002523983 0.9974760168
## 87  0.008244483 0.9917555169
## 89  0.021786295 0.9782137051
## 79  0.400567551 0.5994324493
## 4   0.997037450 0.0029625503
## 129 0.002666466 0.9973335336
## 20  0.894289748 0.1057102521
## 119 0.302326635 0.6976733653
## 83  0.003620178 0.9963798220
## 73  0.039646624 0.9603533760
## 81  0.034243685 0.9657563146
## 104 0.014567904 0.9854320963
```

```
## 41  0.967699631 0.0323003686
## 6   0.991330452 0.0086695476
## 110 0.030364600 0.9696354000
## 105 0.039678341 0.9603216593
## 114 0.041249121 0.9587508791
## 45  0.800740242 0.1992597582
## 32  0.972568214 0.0274317863
## 88  0.003307467 0.9966925328
## 1   0.944962672 0.0550373278
## 125 0.026912688 0.9730873124
## 97  0.014177937 0.9858220631
## 52  0.900472530 0.0995274698
## 112 0.007764684 0.9922353164
## 35  0.341503827 0.6584961727
## 18  0.941029890 0.0589701102
## 103 0.026912688 0.9730873124
## 80  0.009260001 0.9907399985
## 2   0.920039688 0.0799603120
## 121 0.074690960 0.9253090403
## 15  0.999157143 0.0008428574
## 13  0.933119644 0.0668803562
## 101 0.177192865 0.8228071347
## 93  0.043727755 0.9562722452
## 27  0.792001818 0.2079981819
## 67  0.925748386 0.0742516144
## 21  0.937878851 0.0621211486
## 31  0.609986094 0.3900139061
## 69  0.173623233 0.8263767668
## 38  0.446025666 0.5539743343
## 106 0.016196355 0.9838036449
## 115 0.018496731 0.9815032694
## 123 0.001217960 0.9987820405
## 70  0.119942766 0.8800572336
## 25  0.110006428 0.8899935717
## 47  0.822144077 0.1778559230
## 23  0.396142366 0.6038576338
## 34  0.742054082 0.2579459178
## 53  0.996480149 0.0035198514
## 3   0.860973997 0.1390260029
## 17  0.894607622 0.1053923781
## 63  0.283554603 0.7164453965
## 76  0.448930282 0.5510697176
## 46  0.735117408 0.2648825925
## 16  0.992499788 0.0075002120
## 116 0.005548505 0.9944514951
## 9   0.938559812 0.0614401884
## 26  0.021865288 0.9781347118
## 75  0.050170969 0.9498290309
##
## $x
##             LD1
## 55  -1.58437909
## 96   1.29247031
## 127  0.71765490
```

```
## 51  -3.27912294
## 108  1.36769194
## 128  2.61904535
## 87   2.05506342
## 89   1.58779501
## 79  -0.02495521
## 4   -2.97504356
## 129  2.59293922
## 20  -1.22855267
## 119  0.18041939
## 83   2.44750502
## 73   1.29517225
## 81   1.36729712
## 104  1.78210780
## 41  -1.82812324
## 6   -2.46319492
## 110  1.42620241
## 105  1.29477743
## 114  1.27559266
## 45  -0.87559469
## 32  -1.90796768
## 88   2.49048881
## 1   -1.56415921
## 125  1.48510771
## 97   1.79516087
## 52  -1.26039569
## 112  2.08372213
## 35   0.09524146
## 18  -1.52945632
## 103  1.48510771
## 80   1.99950035
## 2   -1.37438168
## 121  0.97724159
## 15  -3.57206155
## 13  -1.46577028
## 101  0.51196444
## 93   1.24669705
## 27  -0.85004131
## 67  -1.41243537
## 21  -1.50318369
## 31  -0.42815134
## 69   0.52366654
## 38  -0.11331744
## 106  1.73108001
## 115  1.66699915
## 123  2.96516094
## 70   0.72888321
## 25   0.77520922
## 47  -0.94198266
## 23  -0.01620053
## 34  -0.71710743
## 53  -2.89304993
## 3   -1.08065343
## 17  -1.23014911
```

```
## 63   0.22340317
## 76  -0.11888782
## 46  -0.70007184
## 16  -2.53245134
## 116  2.24412169
## 9   -1.50875407
## 26   1.58604064
## 75   1.17831781
```

Question 3: Make the **confusion matrix** and find the test error rate, sensitivity, specificity.

Answer 3: the test error rate is **9.23** , the sensitivity **97.1** , the specificity **83.3**

```
la.class <- la.pre$class
table(test$y ,la.class )
```

```
##    la.class
##      0  1
##   0 25  5
##   1  1 34
```

```
sensitivity <- (34/35)*100
specificity <-(25/30)*100
sensitivity
```

```
## [1] 97.14286
```

```
specificity
```

```
## [1] 83.33333
```

```
testerrorrate <- (6/65)*100
testerrorrate
```

```
## [1] 9.230769
```

3) KNN

Question 1: Perform KNN method (use `knn()`) to the training data with $K = 3$ and make prediction for the test set and name your results `KNN.Wine`. (**Note**: To let R run the following chunk, change the following setting `eval=FALSE` to be `eval=TRUE`) (0.5 is the default cutoff).

```
library(class)
KNN.Wine <- knn(train[,c("x1","x2")], test[,c("x1","x2")], cl=train$y, k= 3,prob=TRUE)
KNN.Wine
```

```
##  [1] 0 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 1 1 0 0 1 0 1 1 0 1 1 0 1 1 0 1 0 0
## [39] 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 1 1
## attr(,"prob")
##  [1] 0.6666667 1.0000000 0.6666667 1.0000000 1.0000000 1.0000000 1.0000000
##  [8] 1.0000000 0.6666667 1.0000000 1.0000000 1.0000000 0.6666667 1.0000000
## [15] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [22] 1.0000000 0.6666667 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [29] 0.6666667 1.0000000 0.6666667 1.0000000 1.0000000 1.0000000 0.6666667
## [36] 1.0000000 1.0000000 0.6666667 0.6666667 1.0000000 0.6666667 1.0000000
## [43] 0.6666667 0.6666667 0.6666667 0.6666667 1.0000000 1.0000000 1.0000000
## [50] 0.6666667 1.0000000 0.6666667 0.6666667 0.6666667 1.0000000 1.0000000
## [57] 1.0000000 0.6666667 0.6666667 0.6666667 1.0000000 1.0000000 1.0000000
## [64] 1.0000000 1.0000000
```

```
## Levels: 0 1
```

Question 2: Make the **confusion matrix** and find the test error rate, sensitivity, specificity.

Answer 2: the test error rate is **9.23** , the sensitivity **94.3** , the specificity **86.7**

```
KNNtable <-table(test$y,KNN.Wine)
KNNtable
```

```
##    KNN.Wine
##      0  1
##   0 26  4
##   1  2 33
```

```
testerrorrate <- ((4+2)/65)*100
testerrorrate
```

```
## [1] 9.230769
```

```
sensitivity <- ((33)/35)*100
sensitivity
```

```
## [1] 94.28571
```

```
specificity <- (26/(26+4))*100
specificity
```

```
## [1] 86.66667
```

4) Compare classifiers

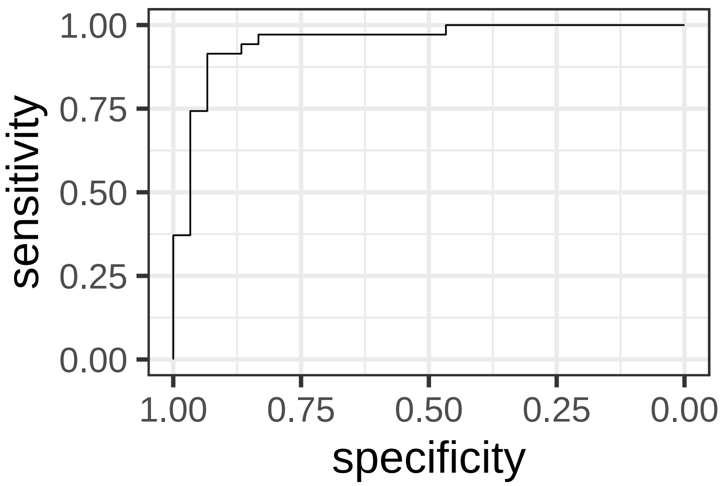
Question 1: Compare the test error rates of the above three classifiers based on the 0.5 cut-off on posterior probability classification rule and which method would you prefer? Why? Which one is the most flexible classifier among the three?

Solution: All the test error rate are the same for all the classification methods TER = 9.23.Moreover, the sensitivity and the specificity for logistic regression and LDA are the same compared to KNN witch is slightly different.KNN with K = 3 gave the worst results out of all methods. When the true decision boundaries are linear, then the LDA and logistic regression approaches will tend to perform well. I prefer to use logistic regression because it is easy to interpret the results from it.In this exercise the logistic regression and LDA methods tend to be more flexible because they give the best results.

Question 2 (For Math 5385): Plot the ROC curve for your prediction of Logistic regression on the test set and calculate the AUC value. (**Note**: To let R run the following chunk, change the following setting `eval=FALSE` to be `eval=TRUE`). Based on the AUC value, do you think if the Logistic regression is a good model for the `wine` data set ?

Answer 2: The area under the curve is between 1 and 0.9 witch means that the model is an excellent fit for our data.

```
library(pROC)
LR.roc <- roc(test$y, LR.pred.prob,legacy.axes=TRUE)
ggroc(LR.roc)+theme_bw(28)
```




```
auc(LR.roc)
```

```
## Area under the curve: 0.9524
```