MIS 545 Naïve Bayes –
Balance and Scale Predictions and Predicting Abalone Male or Female
By Kirsten Fure
July 26, 2019

R Code:

```
Setwd(“ ")
install.packages("e1071")
library(e1071)
BalanceScale = read.csv('Balance_Scale.csv')
summary(BalanceScale)
head(BalanceScale)
nrow(BalanceScale[!complete.cases(BalanceScale), ])
nrow(BalanceScale)
class(BalanceScale$classes)

sample_size = floor(.7 *nrow(BalanceScale))
training_index = sample(nrow(BalanceScale), size = sample_size,
 replace = FALSE)
train = BalanceScale[training_index, ]
test = BalanceScale[-training_index, ]
Balance.model = naiveBayes(classes ~ . , data = train)
Balance.model
Balance.predict = predict(Balance.model, test, type = 'class')
results = data.frame(actual = test[ , 'classes'], predicted =
 Balance.predict)
table(results)
nrow(results[results$predicted == results$actual,])/nrow(results)
```

Screenshots/Output:

```
> setwd("/Users/kirsten 1/Documents/Masters Programs/MIS 545 Data Mining - UofA/Lab 3")
> install.packages("e1071")
Installing package into '/Users/kirsten 1/Library/R/3.6/library'
(as 'lib' is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://cran.cnr.berkeley.edu/bin/macosx/el-capitan/contrib/3.6/e1071_1.7-2.tgz
Content type 'application/x-gzip' length 900872 bytes (879 KB)
==================================================
downloaded 879 KB


The downloaded binary packages are in
    /var/folders/cl/1l5wpk_s6xxf5kb67qx2gbdw0000gp/T//RtmpOE4KYs/downloaded_packages
> library(e1071)
> BalanceScale = read.csv('Balance_Scale.csv')
> summary(BalanceScale)
 classes  Left_weight Left_distance  Right_weight Right_distance
 B: 49   Min.   :1    Min.   :1     Min.   :1     Min.   :1
 L:288   1st Qu.:2    1st Qu.:2     1st Qu.:2     1st Qu.:2
 R:288   Median :3    Median :3     Median :3     Median :3
         Mean   :3    Mean   :3     Mean   :3     Mean   :3
         3rd Qu.:4    3rd Qu.:4     3rd Qu.:4     3rd Qu.:4
         Max.   :5    Max.   :5     Max.   :5     Max.   :5
> head(BalanceScale)
  classes Left_weight Left_distance Right_weight Right_distance
1       B           1             1            1              1
2       R           1             1            1              2
3       R           1             1            1              3
4       R           1             1            1              4
5       R           1             1            1              5
6       R           1             1            2              1
> nrow(BalanceScale[!complete.cases(BalanceScale), ])
[1] 0
> nrow(BalanceScale)
[1] 625

> class(BalanceScale$classes)
[1] "factor"
```

This shows that there is no NULL or missing data and a total of 625 rows in our source data. Classes is a categorical type variable.

```
> sample_size = floor(.7 *nrow(BalanceScale))
> training_index = sample(nrow(BalanceScale), size = sample_size, replace = FALSE)
> train = BalanceScale[training_index, ]
> test = BalanceScale[-training_index, ]
> Balance.model = naiveBayes(classes ~ . , data = train)
> Balance.model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
         B          L          R
0.08237986 0.47139588 0.44622426

Conditional probabilities:
   Left_weight
Y       [,1]     [,2]
  B 2.972222 1.443925
  L 3.665049 1.193250
  R 2.523077 1.348245

   Left_distance
Y       [,1]     [,2]
  B 3.027778 1.275844
  L 3.611650 1.227593
  R 2.317949 1.244135

   Right_weight
Y       [,1]     [,2]
  B 3.000000 1.393864
  L 2.407767 1.342999
  R 3.630769 1.200119

   Right_distance
Y       [,1]     [,2]
  B 3.027778 1.362479
  L 2.432039 1.318811
  R 3.589744 1.237681

> Balance.predict = predict(Balance.model, test, type = 'class')
> results = data.frame(actual = test[ , 'classes'], predicted = Balance.predict)
> table(results)
      predicted
actual  B  L  R
     B  0  7  6
     L  0 79  3
     R  0  4 89
> nrow(results[results$predicted == results$actual, ]) / nrow(results)
[1] 0.8670213
```

I successfully predicted L 79 times, but mistakenly predicted R (instead of L) 3 times. I successfully predicted R 89 times, but mistakenly predicted L (instead of R) 4 times. I never successfully predicted B, but mistakenly predicted L (instead of B) 7 times and mistakenly predicted R (instead of B) 6 times. Overall prediction rate of .867 is high.

R Code:

```
abalone = read.csv('abalone.csv')
summary(abalone)
head(abalone)
nrow(abalone[!complete.cases(abalone), ])
nrow(abalone)
class(abalone$Sex)
a_sample_size = floor(.7 *nrow(abalone))
a_training_index = sample(nrow(abalone), size = a_sample_size,
 replace = FALSE)
a_train = abalone[a_training_index, ]
a_test = abalone[-a_training_index, ]

abalone.model = naiveBayes(as.factor(Sex) ~ . , data = a_train)
abalone.model
abalone.predict = predict(abalone.model, a_test, type = 'class')
abalone.results = data.frame(actual = a_test[ , 'Sex'], predicted
= abalone.predict)
table(abalone.results)
nrow(abalone.results[abalone.results$predicted ==
abalone.results$actual,])/nrow(abalone.results)
aggregate(abalone[, -1], by = list(abalone$Sex), mean)
```

Screenshots:

```
abalone = read.csv('abalone.csv')
summary(abalone)
ex          Length         Diameter          Height        Wholeweight     Shuckedweight    Visceraweight     Shellweight         Rings
:1307   Min.   :0.075   Min.   :0.0550   Min.   :0.0000   Min.   :0.0020   Min.   :0.0010   Min.   :0.0005   Min.   :0.0015   Min.   : 1
:1342   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150   1st Qu.:0.4415   1st Qu.:0.1860   1st Qu.:0.0935   1st Qu.:0.1300   1st Qu.: 8
:1528   Median :0.545   Median :0.4250   Median :0.1400   Median :0.7995   Median :0.3360   Median :0.1710   Median :0.2340   Median : 9
        Mean   :0.524   Mean   :0.4079   Mean   :0.1395   Mean   :0.8287   Mean   :0.3594   Mean   :0.1806   Mean   :0.2388   Mean   : 9
        3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650   3rd Qu.:1.1530   3rd Qu.:0.5020   3rd Qu.:0.2530   3rd Qu.:0.3290   3rd Qu.:11
        Max.   :0.815   Max.   :0.6500   Max.   :1.1300   Max.   :2.8255   Max.   :1.4880   Max.   :0.7600   Max.   :1.0050   Max.   :29
head(abalone)
Sex Length Diameter Height Wholeweight Shuckedweight Visceraweight Shellweight Rings
  M  0.455    0.365  0.095      0.5140        0.2245        0.1010       0.150    15
  M  0.350    0.265  0.090      0.2255        0.0995        0.0485       0.070     7
  F  0.530    0.420  0.135      0.6770        0.2565        0.1415       0.210     9
  M  0.440    0.365  0.125      0.5160        0.2155        0.1140       0.155    10
  I  0.330    0.255  0.080      0.2050        0.0895        0.0395       0.055     7
  I  0.425    0.300  0.095      0.3515        0.1410        0.0775       0.120     8
nrow(abalone[!complete.cases(abalone), ])
] 0
nrow(abalone)
] 4177
```

```
> class(abalone$Sex)
[1] "factor"
```

This shows that there are no rows with NULL or missing data, and the data has 4177 rows. Also, Sex is a categorical variable.

```
> a_sample_size = floor(.7 *nrow(abalone))
> a_training_index = sample(nrow(abalone), size = a_sample_size, replace = FALSE)
> a_train = abalone[a_training_index, ]
> a_test = abalone[-a_training_index, ]
> abalone.model = naiveBayes(as.factor(Sex) ~ . , data = a_train)
> abalone.model

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
        F         I         M
0.3113240 0.3246664 0.3640096

Conditional probabilities:
   Length
Y        [,1]       [,2]
  F 0.5811374 0.08578216
  I 0.4283930 0.10795190
  M 0.5606485 0.10200646

   Diameter
Y        [,1]       [,2]
  F 0.4564505 0.07088654
  I 0.3270811 0.08736934
  M 0.4385432 0.08385436

   Height
Y        [,1]       [,2]
  F 0.1591648 0.04373783
  I 0.1079979 0.03173509
  M 0.1511278 0.03407695

   Wholeweight
Y        [,1]      [,2]
  F 1.0558456 0.4346116
  I 0.4318130 0.2845968
  M 0.9857599 0.4684888

   Shuckedweight
Y        [,1]      [,2]
  F 0.4506209 0.2017293
  I 0.1920364 0.1280147
  M 0.4291715 0.2202954

   Visceraweight
Y         [,1]       [,2]
  F 0.23285165 0.09843879
  I 0.09245364 0.06229418
  M 0.21445160 0.10462865

   Shellweight
Y        [,1]       [,2]
  F 0.3042703 0.12664673
  I 0.1279241 0.08454545
  M 0.2813449 0.13234618

   Rings
Y        [,1]     [,2]
  F 11.217582 3.138343
  I  7.884089 2.448467
```

```
> abalone.predict = predict(abalone.model, a_test, type = 'class')
> abalone.results = data.frame(actual = a_test[ , 'Sex'], predicted = abalone.predict)
> table(abalone.results)
      predicted
actual   F   I   M
     F 232  88  77
     I  25 312  56
     M 267 110  87
```

```
> nrow(abalone.results[abalone.results$predicted == abalone.results$actual, ]) / nrow(abalone.results)
[1] 0.5318979
```

I successfully predicted Female 232 times, Male only 87 times and "I" 312 times. However there were many times the predictions were wrong. The overall prediction rate is only.53. I tried to determine if there was a certain combination of attributes that would be better predictors for male vs. female, but it seems that males and females are similar in the attributes given. See below; the mean values are very similar.

```
> aggregate(abalone[ , -1], by = list(abalone$Sex), mean)
  Group.1    Length  Diameter    Height Wholeweight Shuckedweight Visceraweight Shellweight     Rings
1       F 0.5790933 0.4547322 0.1580107   1.0465321     0.4461878    0.23068860   0.3020099 11.129304
2       I 0.4277459 0.3264940 0.1079955   0.4313625     0.1910350    0.09201006   0.1281822  7.890462
3       M 0.5613907 0.4392866 0.1513809   0.9914594     0.4329460    0.21554450   0.2819692 10.705497
```