MA5810

Naïve Bayes Classifier and Discriminant Analysis

Assessment 1

Bienvenido Hiyas Jr.
13824819

15-Nov-2020

MA5810_Asessment1_Hiyas

Bien

15/11/2020

Assessment Task 1: Comparison of classifiers

In this task compare the performance of the supervised learning algorithms Linear Discriminant Analysis, Quadratic Discriminant Analysis and the Naïve Bayes Classifier using a publicly available Blood Pressure Data. The data to be used for this task is provided in the HBblood.csv file in the Assessment 1 folder.

The HBblood.csv dataset contains values of the percent HbA1c (a measure of the amount of glucose and haemoglobin joined together in blood) and systolic blood pressure (SBP) (in mm/Hg) for 1,200 clinically healthy female patients within the ages 60 to 70 years. Additionally, the ethnicity, Ethno, for each patient was recorded and discombobulated into three groups, A, B or C, for analysis.

1. Discuss and justify which of the supervised learning algorithms (i.e. Linear Discriminant Analysis, Quadratic Discriminant Analysis and the Naïve Bayes Classifier) would you choose for predicting the response Ethno using HbA1c and SBP as the feature variables. Provide any plots/images needed to support your discussion. Hint: Base your answer on the empirical properties of the data.

Installing library packages and libraries for all tasks (1-3)

library("ggpubr")

library("psych")

library("gridExtra")

library("dplyr")

library("tidyr")

library("ggplot2")

library("naivebayes")

Accessing the data "HBblood.csv"

```
data = read.csv('HBblood.csv')
```

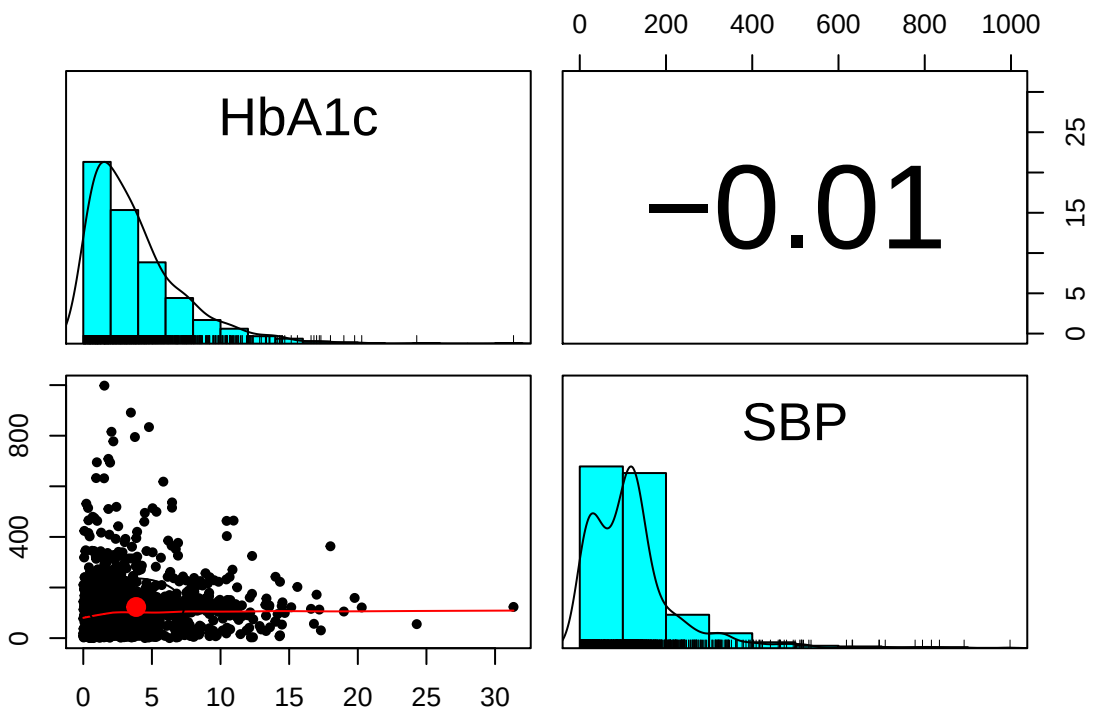
```
str(data)
```

```
## 'data.frame':    1200 obs. of  3 variables:
##  $ Ethno: chr  "C" "B" "B" "C" ...
##  $ HbA1c: num  0.00129 0.00206 0.01245 0.01292 0.01448 ...
##  $ SBP  : num  9.41 77.39 18.94 144.15 210.62 ...
```

```
#data Transformation
data$Ethno = as.factor(data$Ethno)
```

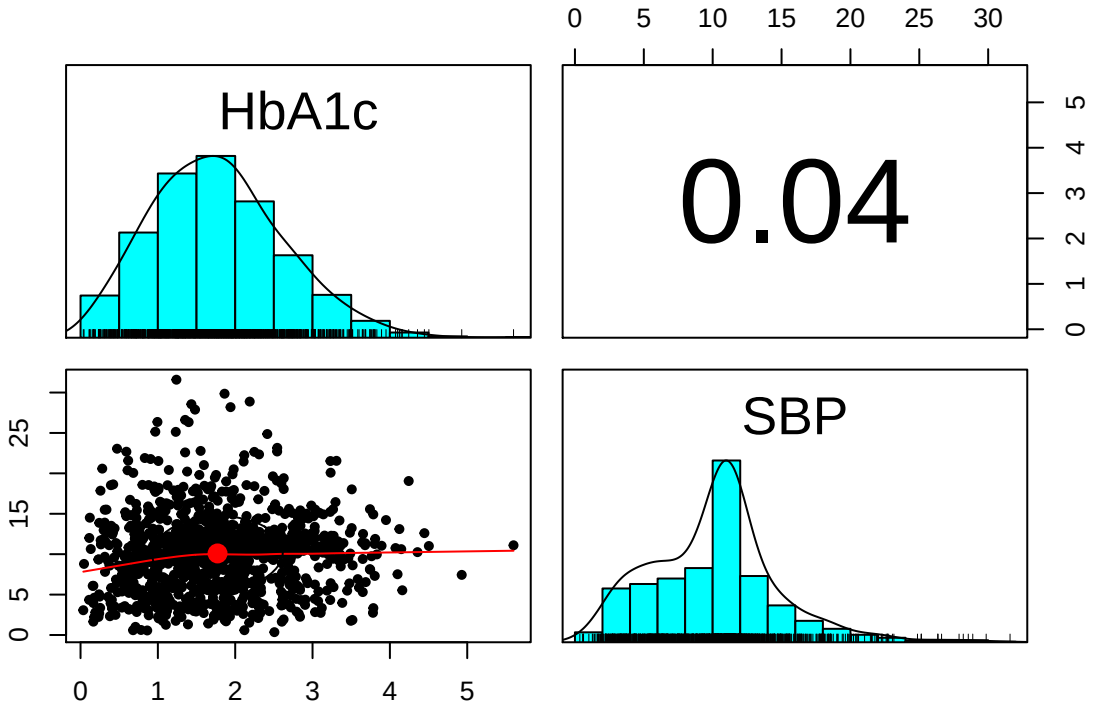
Note: Ethno has been transformed from CHar to Factor variable. Note that data has 1200 observations and the variables are Ethno = Character(Categorical 4=levels), HbA1c = Numerical, and SBP = Numerical

```r
#visualisation
pairs.panels(data[-1])
```



The graph shown above shows that HbA1c and SBP are not normally distributed and are positively skewed. It also shows that HbA1c and SBP are not related or independent with each other with Correlation coefficient is only -0.01

```r
data$HbA1c = sqrt(data$HbA1c)
data$SBP = sqrt(data$SBP)
#visualisation of data
pairs.panels(data[-1])
```



After data transformation using "sqrt", data is now normally distributed

---

```
#covariance matrix
cov(data [-1])
```

```
##           HbA1c       SBP
## HbA1c 0.7132699  0.156952
## SBP   0.1569520 21.172435
```

Note that covariance matrix are different in all class.

Naïve Bayes

- Assumption that the predictor variables are conditionally independent of each other

LDA

- LDA assumes normally distributed data and a class-specific mean vector.
- LDA assumes a common covariance matrix. So, a covariance matrix that is common to all classes in a data set.

QDA

- Observation of each class is drawn from a normal distribution (same as LDA).
- QDA assumes that each class has its own covariance matrix (different from LDA).

Naive Bayes can work either numerical and categorical Both LDA and QDA require numerical predictors // in this case our predictors are numerical

//Since our data predictors are numeric, follows normal distribution, and different covariance...

Conclusion: Therefore, the supervised learning algorithm that should be used in the case is QDA

Assessment Task 2: Application of a classifier

The Mushroom dataset, available from the UCI Data Repository https://archive.ics.uci.edu/ml/datasets/Mushroom, contains 8124 observations describing mushrooms belonging to classes edible or potentially poisonous (first variable encoded 'e' or 'p', respectively). There are 22 categorical predictors (variables 2 to 23), one of them with missing values ('?').

#read mushroom data -> data2

```
data2 <- read.table(file ="mushroom2.csv", header=FALSE, sep = ",")
```

```
#data transformation
str(data2)
```

```
## 'data.frame':    8124 obs. of  23 variables:
##  $ V1 : chr  "p" "e" "e" "p" ...
##  $ V2 : chr  "x" "x" "b" "x" ...
##  $ V3 : chr  "s" "s" "s" "y" ...
##  $ V4 : chr  "n" "y" "w" "w" ...
##  $ V5 : chr  "t" "t" "t" "t" ...
##  $ V6 : chr  "p" "a" "l" "p" ...
##  $ V7 : chr  "f" "f" "f" "f" ...
##  $ V8 : chr  "c" "c" "c" "c" ...
##  $ V9 : chr  "n" "b" "b" "n" ...
##  $ V10: chr  "k" "k" "n" "n" ...
##  $ V11: chr  "e" "e" "e" "e" ...
##  $ V12: chr  "e" "c" "c" "e" ...
##  $ V13: chr  "s" "s" "s" "s" ...
##  $ V14: chr  "s" "s" "s" "s" ...
##  $ V15: chr  "w" "w" "w" "w" ...
##  $ V16: chr  "w" "w" "w" "w" ...
##  $ V17: chr  "p" "p" "p" "p" ...
##  $ V18: chr  "w" "w" "w" "w" ...
##  $ V19: chr  "o" "o" "o" "o" ...
##  $ V20: chr  "p" "p" "p" "p" ...
##  $ V21: chr  "k" "n" "n" "k" ...
##  $ V22: chr  "s" "n" "n" "s" ...
##  $ V23: chr  "u" "g" "m" "u" ...
```

```
data2$V1 = as.factor(data2$V1)
data2$V2 = as.factor(data2$V2)
data2$V3 = as.factor(data2$V3)
data2$V4 = as.factor(data2$V4)
data2$V5 = as.factor(data2$V5)
data2$V6 = as.factor(data2$V6)
data2$V7 = as.factor(data2$V7)
data2$V8 = as.factor(data2$V8)
data2$V9 = as.factor(data2$V9)
data2$V10 = as.factor(data2$V10)
data2$V11 = as.factor(data2$V11)
data2$V12 = as.factor(data2$V12)
data2$V13 = as.factor(data2$V13)
data2$V14 = as.factor(data2$V14)
data2$V15 = as.factor(data2$V15)
data2$V16 = as.factor(data2$V16)
data2$V17 = as.factor(data2$V17)
```

```
data2$V18 = as.factor(data2$V18)
data2$V19 = as.factor(data2$V19)
data2$V20 = as.factor(data2$V20)
data2$V21 = as.factor(data2$V21)
data2$V22 = as.factor(data2$V22)
data2$V23 = as.factor(data2$V23)
summary(data2)
```

```
##  V1       V2       V3             V4       V5             V6       V7
##  e:4208   b: 452   f:2320   n      :2284   f:4748   n      :3528   a: 210
##  p:3916   c:   4   g:   4   g      :1840   t:3376   f      :2160   f:7914
##           f:3152   s:2556   e      :1500            s      : 576
##           k: 828   y:3244   y      :1072            y      : 576
##           s:  32            w      :1040            a      : 400
##           x:3656            b      : 168            l      : 400
##                             (Other): 220            (Other): 484
##  V8       V9            V10       V11      V12      V13      V14
##  c:6812   b:5612   b      :1728   e:3516   ?:2480   f: 552   f: 600
##  w:1312   n:2512   p      :1492   t:4608   b:3776   k:2372   k:2304
##                    w      :1202            c: 556   s:5176   s:4936
##                    n      :1048            e:1120   y:  24   y: 284
##                    g      : 752            r: 192
##                    h      : 732
##                    (Other):1170
##       V15            V16       V17      V18      V19      V20
##  w      :4464   w      :4384   p:8124   n:  96   n:  36   e:2776
##  p      :1872   p      :1872            o:  96   o:7488   f:  48
##  g      : 576   g      : 576            w:7924   t: 600   l:1296
##  n      : 448   n      : 512            y:   8            n:  36
##  b      : 432   b      : 432                              p:3968
##  o      : 192   o      : 192
##  (Other): 140   (Other): 156
##       V21       V22      V23
##  w      :2388   a: 384   d:3148
##  n      :1968   c: 340   g:2148
##  k      :1872   n: 400   l: 832
##  h      :1632   s:1248   m: 292
##  r      :  72   v:4040   p:1144
##  b      :  48   y:1712   u: 368
##  (Other): 144            w: 192
```

```
#NA's : 2480 are found in V12
```

As noticed, the variables V1-V2 are transformed from character class to factor class and the variable V12 with ?:2480 are removed.

*PLEASE NOT THAT I WAS ABLE TO DELETE V12 ROWS WITH ?/NAS IN R STUDIO BUT IT IS DIFFERENT IN R MARKDOWN. HOWEVER, RESULT SHOWS LITTLE DIFFERENCE. R RESULTS WITH 5633 OBS. OF 23 VARIABLES.

—-CODES AND RESULT IN R —-

#predicting data

nb_predict2_train = predict(nb_model2, train.data2)

table(nb_predict2_train, train.data2$V1)

nb_predict2_train

#nb_predict2_train e p #e 2779 216 #p 12 1509

#Accuracy: 95%

nb_predict2_test = predict(nb_model2, test.data2)

table(nb_predict2_test, test.data2$V1)

#nb_predict2_test e p #e 694 72 #p 3 359

#Accuracy: 93.35%

#predicting data

confusionMatrix(predict(nb_model2,newdata = train.data2), train.data2$V1)

#Accuracy: 94.31

confusionMatrix(predict(nb_model2,newdata = test.data2), test.data2$V1)

#Accuracy: 94.21

```
#removing rows with NA's
data2 = na.omit(data2)
summary(data2)
```

```
##  V1       V2       V3             V4       V5             V6       V7
##  e:4208   b: 452   f:2320   n      :2284   f:4748   n      :3528   a: 210
##  p:3916   c:   4   g:   4   g      :1840   t:3376   f      :2160   f:7914
##           f:3152   s:2556   e      :1500            s      : 576
##           k: 828   y:3244   y      :1072            y      : 576
##           s:  32            w      :1040            a      : 400
##           x:3656            b      : 168            l      : 400
##                             (Other): 220            (Other): 484
##  V8       V9            V10       V11      V12      V13      V14
##  c:6812   b:5612   b      :1728   e:3516   ?:2480   f: 552   f: 600
##  w:1312   n:2512   p      :1492   t:4608   b:3776   k:2372   k:2304
##                    w      :1202            c: 556   s:5176   s:4936
##                    n      :1048            e:1120   y:  24   y: 284
##                    g      : 752            r: 192
##                    h      : 732
##                    (Other):1170
##       V15            V16       V17      V18      V19      V20
##  w      :4464   w      :4384   p:8124   n:  96   n:  36   e:2776
##  p      :1872   p      :1872            o:  96   o:7488   f:  48
##  g      : 576   g      : 576            w:7924   t: 600   l:1296
##  n      : 448   n      : 512            y:   8            n:  36
##  b      : 432   b      : 432                              p:3968
##  o      : 192   o      : 192
##  (Other): 140   (Other): 156
##       V21       V22      V23
##  w      :2388   a: 384   d:3148
##  n      :1968   c: 340   g:2148
##  k      :1872   n: 400   l: 832
##  h      :1632   s:1248   m: 292
##  r      :  72   v:4040   p:1144
##  b      :  48   y:1712   u: 368
```

```
##  (Other): 144            w: 192
```

1. Randomly split the dataset into a training subset and a test subset containing 80% and 20% of the data. Provide and comment your R-code.

```
##partition data set into train and set and randomly split it
set.seed(123)
# Creating index for randomly splitting the dataset
ind2= createDataPartition(data2$V1, p=0.8, list=FALSE)
train.data2 <- data2[ind2,]
test.data2 <- data2[-ind2,]

# print number of observations in test vs. train
c(nrow(train.data2), nrow(test.data2))
```

```
## [1] 6500 1624
```

```
# Proportions of V1 EDIBLE and POISONOUS
table(train.data2$V1) %>% prop.table()
```

```
##
##     e     p
## 0.518 0.482
```

2. Define and justify a classifier – from Week 2- to classify the mushroom population into edible or poisonous. The classifier needs to use all 22 predictors (variables V2 toV23) to model the dependent variable (V1) .

Ans: Since the data predictors are not numericals, LDA and QdA cannot be used. It also shows that the predictors are not related or independent to each other. Therefore, Naive Bayes is used in this situation

3. Implement the proposed classifier from Question 2 using the training data subset from Question 1. Provide and comment R code. Display the most relevant parts of the results

```
#Training the model using Naive Bayes
nb_model2 = naive_bayes(V1 ~., data = train.data2)
nb_model2
```

```
##
## ================================== Naive Bayes ==================================
##
##  Call:
## naive_bayes.formula(formula = V1 ~ ., data = train.data2)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 0
##
## ---------------------------------------------------------------------------------
##
##  A priori probabilities:
##
##     e     p
## 0.518 0.482
##
## ---------------------------------------------------------------------------------
##
```

```
##  Tables:
##
## ------------------------------------------------------------------------------
##  ::: V2 (Categorical)
## ------------------------------------------------------------------------------
##
## V2            e           p
##   b 0.091773092 0.012767316
##   c 0.000000000 0.001276732
##   f 0.384615385 0.396744335
##   k 0.050787051 0.150973508
##   s 0.007722008 0.000000000
##   x 0.465102465 0.438238110
##
## ------------------------------------------------------------------------------
##  ::: V3 (Categorical)
## ------------------------------------------------------------------------------
##
## V3             e            p
##   f 0.3739233739 0.1966166613
##   g 0.0000000000 0.0009575487
##   s 0.2628452628 0.3609958506
##   y 0.3632313632 0.4414299394
##
## ------------------------------------------------------------------------------
##  ::: V4 (Categorical)
## ------------------------------------------------------------------------------
##
## V4            e           p
##   b 0.012177012 0.029364826
##   c 0.007722008 0.003830195
##   e 0.149391149 0.222470476
##   g 0.242946243 0.210660709
##   n 0.299376299 0.254707948
##   p 0.013662014 0.023619534
##   r 0.003861004 0.000000000
##   u 0.004752005 0.000000000
##   w 0.171369171 0.081710820
##   y 0.094743095 0.173635493
##
## ------------------------------------------------------------------------------
##  ::: V5 (Bernoulli)
## ------------------------------------------------------------------------------
##
## V5          e         p
##   f 0.3436293 0.8436004
##   t 0.6563707 0.1563996
##
## ------------------------------------------------------------------------------
##  ::: V6 (Categorical)
## ------------------------------------------------------------------------------
##
## V6            e           p
##   a 0.096525097 0.000000000
```

```
##   c 0.000000000 0.048515800
##   f 0.000000000 0.551228854
##   l 0.091773092 0.000000000
##   m 0.000000000 0.008937121
##   n 0.811701812 0.032556655
##   p 0.000000000 0.063836578
##   s 0.000000000 0.152569422
##   y 0.000000000 0.142355570
##
## ------------------------------------------------------------------------------
##
## # ... and 17 more tables
##
## ------------------------------------------------------------------------------
```

```
#predicting data
confusionMatrix(predict(nb_model2,newdata = train.data2),
                train.data2$V1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    e    p
##          e 3346  360
##          p   21 2773
##
##                Accuracy : 0.9414
##                  95% CI : (0.9354, 0.947)
##     No Information Rate : 0.518
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8822
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9938
##             Specificity : 0.8851
##          Pos Pred Value : 0.9029
##          Neg Pred Value : 0.9925
##              Prevalence : 0.5180
##          Detection Rate : 0.5148
##    Detection Prevalence : 0.5702
##       Balanced Accuracy : 0.9394
##
##        'Positive' Class : e
##
```

```
#Accuracy: 94.31

confusionMatrix(predict(nb_model2,newdata = test.data2),
                test.data2$V1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   e   p
```

```
##          e 833  91
##          p   8 692
##
##                Accuracy : 0.939
##                  95% CI : (0.9263, 0.9502)
##     No Information Rate : 0.5179
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8775
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9905
##             Specificity : 0.8838
##          Pos Pred Value : 0.9015
##          Neg Pred Value : 0.9886
##              Prevalence : 0.5179
##          Detection Rate : 0.5129
##    Detection Prevalence : 0.5690
##       Balanced Accuracy : 0.9371
##
##        'Positive' Class : e
##
```

```
#Accuracy: 94.21
```

4. Interpret and discuss the relationships between the predictors and response variables of the fitted model based on the summary shown in Question 3.

Ans: The relationship between the predictors and variables are good and correlated since the model has high accuracy using the predictors. The predictors can identify whether the mushrooms are poisonous or edible.

5. Discuss the performance of the proposed classifier for both training and test data.

Ans: By using the Naive Bayes model on training data, we got a high accuracy rate of 94.31%. This means that our model performs well using the training data. Thus as we used our model to our test dta, we also have a high accuracy of 94.31%

Assessment Task 3: Implementation of classifiers

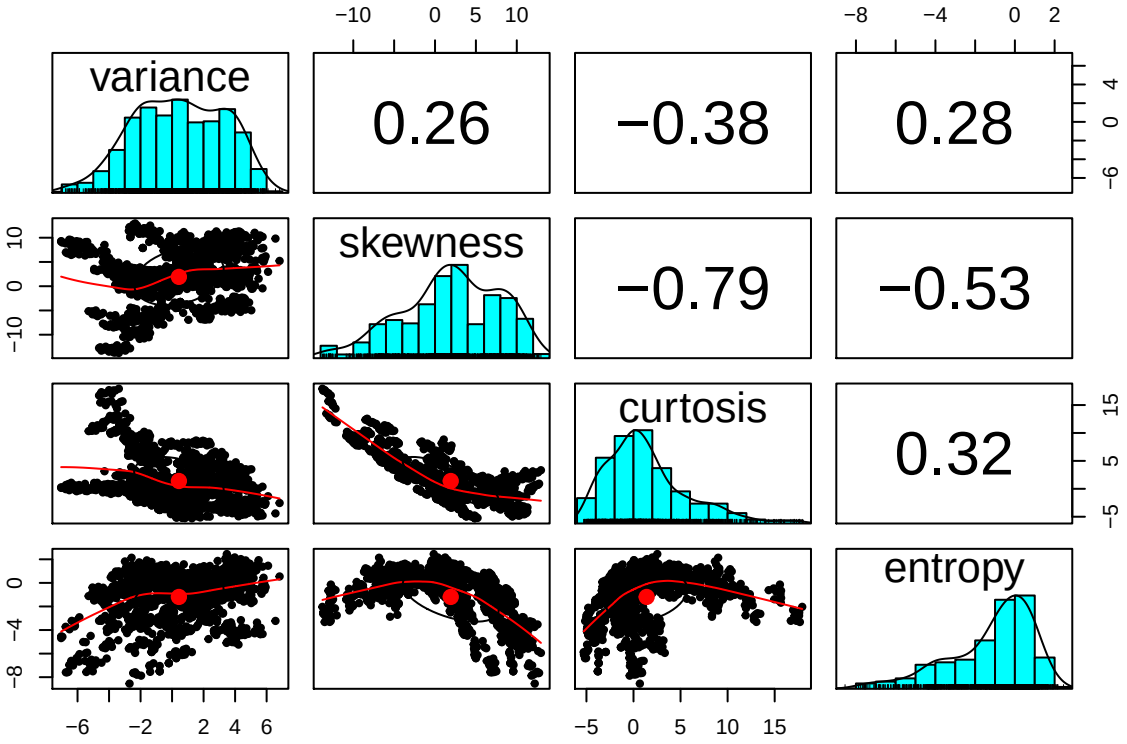
In this task, compare the performance of the supervised learning algorithms Linear Discriminant Analysis and the Naïve Bayes Classifier using the banknote authentication dataset, which is publicly available at https://archive.ics.uci.edu/ml/datasets/banknote+authentication. The data were extracted from images using Wavelet transform tool. The data contain 1372 observations with 4 inputs (variance of wavelet transformed image, skewness of wavelet transformed image, curtosis of wavelet transformed image and entropy of image) and 1 output (Class), which is used as target and only has two values of 0 (False) and 1 (True).

1. Implement both the LDA and Naïve Bayes classifiers to classify the authentication of bank notes. Display the R-code, code comments and model summaries for both models.

```
#read data "banknote.txt" from file location
data3 =read.table(file ="banknote.txt", header=FALSE, sep = ",", na.strings = "?",
```

```
                  col.names = c("variance","skewness","curtosis","entropy","target"))
#checking data classes
str(data3)
#transform target variable from character to factor
data3$target = as.factor(data3$target)

#visualisation of data3
pairs.panels(data3[-5])
```




```
#checking on covariances
cov(data3[,1:4])


##partition data set into train and set and randomly split it
set.seed(123)
# Creating index for randomly splitting the dataset
ind3= createDataPartition(data3$target, p=0.8, list=FALSE)
train.data3 <- data3[ind3,]
test.data3 <- data3[-ind3,]
# Training the model
c(nrow(train.data3), nrow(test.data3))

#using Naive Bayes
nb_model3 = naive_bayes(target ~., data = train.data3)

#calculate how to perform the data
nb_predict3 = predict(nb_model3, train.data3)
table(nb_predict3, train.data3$target)

#using LDA
```

```
lda3 <- train(target~.,
                      data=train.data3,
                      trControl = trainControl(method = "cv", number = 5),
                      method = "lda")
```

```
#confusion matrix
confusionMatrix(predict(lda3,newdata = train.data3),
                train.data3$target)
confusionMatrix(predict(lda3,newdata = test.data3),
                test.data3$target)
```

2. Compare and Discuss the performance of the LDA and Naïve Bayes classifiers obtained in Question 1.

```
table(nb_predict3, train.data3$target)
```

```
##
## nb_predict3   0   1
##           0 530 107
##           1  80 381
```

```
confusionMatrix(predict(lda3,newdata = train.data3),
                train.data3$target)
```

As the result above, we are able to classify 530 out of 610 for "0" cases correctly and 381 out of 488 for"1" cases correctly. This means the ability of our Naive Bayes Algorithm to predict "0" cases is about 87% and 78% for "1" cases, resulting in an overall accuracy of 82%.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 583   0
##          1  27 488
##
##                Accuracy : 0.9754
##                  95% CI : (0.9644, 0.9837)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9505
##
##  Mcnemar's Test P-Value : 5.624e-07
##
##             Sensitivity : 0.9557
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9476
##              Prevalence : 0.5556
##          Detection Rate : 0.5310
##    Detection Prevalence : 0.5310
##       Balanced Accuracy : 0.9779
##
##        'Positive' Class : 0
##
```

Shown on the above result, the accuracy of LDA is 97.5 %.

Therefore, LDA has performed better than Naive bayes.

3. Discuss your findings from Question 1 and Question 2 by using the assumptions of LDA and Naïve Bayes classifiers as the basis of your discussion. Provide any plots/images or analysis needed to support your discussion.

As shown in Question 1, we illustrated that the data seems to follows normal distribution although with a little skewed to the left or right. It was also illustrated that the data doesn't have strong conditionally independently with each other. It can also be noticed that the data or predictors are numericals which is a strong idication of using LDA.

With this, LDA assumptions has been met better than Naive Bayes, thus resulting in a better model accuracy of LDA 97.5% than naive bayes 82%.

Appendix:

R code:

#installing libraries

library("ggpubr") library("psych") library("gridExtra") library("dplyr") library("tidyr") library("ggplot2") library("naivebayes")

task1 ############3

library(caret, warn.conflicts = F, quietly = T)

data = read.csv('HBblood.csv')

str(data)

#dta transformation

data$Ethno = as.factor(data$Ethno)$

#visualisation of data

pairs.panels(data[-1])

#normalizing the data

data$HbA1c = sqrt(data$HbA1c)$

data$SBP = sqrt(data$SBP)$

#visualisation

pairs.panels(data[-1])

#covariance matrix

cov(data [-1])

task2

rm(list=ls())

data2 <- read.table(file ="mushroom2.csv", header=FALSE, sep = ",", na.strings = "?")

str(data2)

is.na.data.frame(data2)

summary(data2)

$data2V1 = as.factor(data2V1)$ $data2V2 = as.factor(data2V2)$ $data2V3 = as.factor(data2V3)$
$data2V4 = as.factor(data2V4)$ $data2V5 = as.factor(data2V5)$ $data2V6 = as.factor(data2V6)$
$data2V7 = as.factor(data2V7)$ $data2V8 = as.factor(data2V8)$ $data2V9 = as.factor(data2V9)$
$data2V10 = as.factor(data2V10)$ $data2V11 = as.factor(data2V11)$ $data2V12 = as.factor(data2V12)$
$data2V13 = as.factor(data2V13)$ $data2V14 = as.factor(data2V14)$ $data2V15 = as.factor(data2V15)$
$data2V16 = as.factor(data2V16)$ $data2V17 = as.factor(data2V17)$ $data2V18 = as.factor(data2V18)$
$data2V19 = as.factor(data2V19)$ $data2V20 = as.factor(data2V20)$ $data2V21 = as.factor(data2V21)$
$data2V22 = as.factor(data2V22)$ $data2V23 = as.factor(data2V23)$

summary(data2)

#NA's : 2480 are found in V12

#removing rows with NA's

data2 = na.omit(data2)

summary(data2)

#Visualization

cov(data2[-1])

ggplot(data2) + geom_bar(aes(x = V1)) ggplot(data2) + geom_col(aes(x = V1, y = n, fill = V1), position = "fill") + scale_fill_manual(values = V1)

##partition data set into train and set and randomly split it

set.seed(123)

Creating index for randomly splitting the dataset

ind2= createDataPartition(data2$V1, p=0.8, list=FALSE)

train.data2 <- data2[ind2,]

test.data2 <- data2[-ind2,]

c(nrow(train.data2), nrow(test.data2))

summary(data2)

#Training the model using Naive Bayes

nb_model2 = naive_bayes(V1 ~., data = train.data2)

nb_model2

#the probability of e = 51.8% and p = 48.2%

#predicting data

nb_predict2_train = predict(nb_model2, train.data2)

table(nb_predict2_train, train.data2$V1)

nb_predict2_train

#nb_predict2_train e p #e 2779 216 #p 12 1509 #Accuracy: 95%

nb_predict2_test = predict(nb_model2, test.data2)

table(nb_predict2_test, test.data2$V1)

#nb_predict2_test e p #e 694 72 #p 3 359

#Accuracy: 93.35%

#predicting data

confusionMatrix(predict(nb_model2,newdata = train.data2), train.data2$V1) #Accuracy: 94.31

confusionMatrix(predict(nb_model2,newdata = test.data2), test.data2$V1)

#Accuracy: 94.21

Task 3

#read data "banknote.txt" from file location data3 =read.table(file ="banknote.txt", header=FALSE, sep = ",", na.strings = "?", col.names = c("variance","skewness","curtosis","entropy","target"))

head(data3)

#checking data classes

str(data3)

#transform target variable from character to factor

data3$target = as.factor(data3$target)

str(data3)

#visualisation of data3

pairs.panels(data3[-5])

#checking on covariances

cov(data3[,1:4])

pairs(data3[,1:4])

##partition data set into train and set and randomly split it

set.seed(123)

Creating index for randomly splitting the dataset

ind3= createDataPartition(data3$target, p=0.8, list=FALSE)

train.data3 <- data3[ind3,]

test.data3 <- data3[-ind3,]

Training the model

c(nrow(train.data3), nrow(test.data3))

summary(data3)

#using Naive Bayes

naive.Bayes3 <- train(target~., data=train.data3, trControl = trainControl(method = "cv", number = 3), method = "nb") naive.Bayes3

#predicting the data and confusion matrix

confusionMatrix(predict(naive.Bayes3,newdata = train.data3), train.data3$target)

confusionMatrix(predict(naive.Bayes3,newdata = test.data3), test.data3$target)

nb_model3 = naive_bayes(target ~., data = train.data3)

nb_model3

#calculate how to perform the data

nb_predict3 = predict(nb_model3, train.data3)

table(nb_predict3, train.data3$target)

nb_predict3

#nb_predict3 0 1 #0 530 107 #1 80 381

#As the result, we are able to classify 530 out of 610 for "0" cases correctly and #381 out of 488 for"1" cases correctly. This means the ability of our Naive Bayes Algorithm #to predict "0" cases is about 87% and 78% for "1" cases, resulting in an overall accuracy of 82%

#using LDA

lda3 <- train(target~., data=train.data3, trControl = trainControl(method = "cv", number = 5), method = "lda") lda3

confusionMatrix(predict(lda3,newdata = train.data3), train.data3$target)

confusionMatrix(predict(lda3,newdata = test.data3), test.data3$target)

#covariance checking

library(heplots) library(ggplot2) library(dplyr) install.packages("gridExtra") library(gridExtra)

#note this code runs on R but not in RMarkdown.

plot <- list() box_variables3 <- c("variance","skewness","curtosis","entropy") for(i in box_variables3) { plot[[i]] <- ggplot(data3, aes_string(x = "target", y = i, col = "target", fill = "target")) + geom_boxplot(alpha = 0.2) + theme(legend.position = "none") + scale_color_manual(values = c("blue", "red")) + scale_fill_manual(values = c("blue", "red")) }

do.call(grid.arrange, c(plot, nrow = 1))

#The four different boxplots show us that the length of each plot clearly the same .This is an indication of equal variances. THus LDA is bet to used.