4) Naive Bayes

2023-04-04

Pre-processing the data-set

```
data <- read.csv("Naive_Bayes_Dataset.csv", header = TRUE)
processed_data <- na.omit(data)
head(processed_data)
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0
```

```
summary(processed_data)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin          BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
str(processed_data)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

```
nrow(processed_data)
```

```
## [1] 768
```

Splitting the model

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
indexs = createDataPartition(processed_data$Outcome, times = 1, p = 0.8, list = F)
#times = no. of times to be split
#p = percentage of data to be used for training, here 80% is used of training and 20%
for testing

train = processed_data[indexs, ]
nrow(train)
```

```
## [1] 615
```

```
test = processed_data[-indexs, ]
nrow(test)
```

```
## [1] 153
```

Creating the model

```
library(e1071)

model <- naiveBayes(Outcome ~ ., data = train)
model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.6601626 0.3398374
##
## Conditional probabilities:
##    Pregnancies
## Y       [,1]     [,2]
##   0 3.184729 2.914055
##   1 4.770335 3.564634
##
##    Glucose
## Y       [,1]     [,2]
##   0 109.9064 26.77914
##   1 141.5550 32.67274
##
##    BloodPressure
## Y       [,1]     [,2]
##   0 68.24877 17.84376
##   1 70.88038 21.47808
##
##    SkinThickness
## Y       [,1]     [,2]
##   0 19.57143 14.82519
##   1 22.53589 17.21275
##
##    Insulin
## Y        [,1]     [,2]
##   0  69.69212 101.7504
##   1 100.33014 134.1832
##
##    BMI
## Y       [,1]     [,2]
##   0 30.47882 7.688905
##   1 35.33684 7.539577
##
##    DiabetesPedigreeFunction
## Y        [,1]      [,2]
##   0 0.4374335 0.3042306
##   1 0.5494258 0.3749720
##
##    Age
## Y       [,1]     [,2]
##   0 30.87192 11.42915
##   1 36.92823 10.86675
```

Predicting the values using the model and the Confusion matrix

```
Predict <- predict(model, newdata = test)
Predict
```

```
##   [1] 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1
##  [38] 1 0 0 1 1 1 0 0 0 1 0 1 0 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0
##  [75] 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0
## [112] 0 1 0 1 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 1 1 0 0 0 0 0 0
## [149] 1 1 0 1 1
## Levels: 0 1
```

```
#table(test$Outcome, predict(model, test)), sometimes if you get an error of values o
verlapping use this
cm <- table(test$Outcome, Predict)
confusionMatrix(cm)
```

```
## Confusion Matrix and Statistics
##
##    Predict
##      0  1
##   0 78 16
##   1 23 36
##
##                Accuracy : 0.7451
##                  95% CI : (0.6684, 0.812)
##     No Information Rate : 0.6601
##     P-Value [Acc > NIR] : 0.0149
##
##                   Kappa : 0.4499
##
##  Mcnemar's Test P-Value : 0.3367
##
##             Sensitivity : 0.7723
##             Specificity : 0.6923
##          Pos Pred Value : 0.8298
##          Neg Pred Value : 0.6102
##              Prevalence : 0.6601
##          Detection Rate : 0.5098
##    Detection Prevalence : 0.6144
##       Balanced Accuracy : 0.7323
##
##        'Positive' Class : 0
##
```

Conclusion: The accuracy of the model is, 83.66% which can be regarded as an acceptable solution for the dataset. In conclusion, Naive Bayes is a simple yet powerful algorithm for classification tasks. It is based on Bayes' theorem, which allows us to calculate the probability of a certain class given the data we have. Despite its simplicity, Naive Bayes has been shown to be highly effective in many real-world applications, such as spam detection, sentiment analysis, and medical diagnosis. During the course of this lab report, we have implemented and evaluated the Naive Bayes algorithm on a given dataset. We have seen how the algorithm works and how to tune its parameters for better performance. We have also discussed some of the limitations of Naive Bayes, such as the assumption of independence between features, and how to address these limitations. Overall, Naive Bayes is a useful algorithm to

have in your machine learning toolbox. It is easy to implement, fast to train, and can achieve good results even with limited data. However, it is important to keep in mind its limitations and to choose the appropriate algorithm for your specific task.