# agaricus_lepiota

Shiva Sankar Modala

2023-02-10

```
# Installing the package caret
# install.packages('caret')

# Reading the data in csv format for the agaricus lepiota from the mushroom
package
mush_data = read.csv('https://archive.ics.uci.edu/ml/machine-learning-
databases/mushroom/agaricus-
lepiota.data',header=FALSE,sep=",",stringsAsFactors = TRUE)

# Loading the library for the statistics and the probability package.
library(e1071)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

# Finding the missing values in our dataset.
values_missing = which(mush_data$V12=='?')

# Assigning a new variable for the dataset excluding the missing data
MisingsVal_Mushroom = mush_data[-c(values_missing)]

# We can replace the missing values with either removing those
# or we can replace missing values with mode of the data
mush_mode = (table(as.vector(MisingsVal_Mushroom$V12)))
replaceVal_mush = mush_data

# Replacing the missing value with the character 'b'
replaceVal_mush$V12[values_missing]= 'b'

# To train the data, we need to split the given dataset.
# So, I have applied 80% for the training and the rest for the testing.
miss_index = sample(1:nrow(MisingsVal_Mushroom),size =
0.8*nrow(MisingsVal_Mushroom))

# This is the train and test for the data without replacing
miss_train = mush_data[miss_index,]
miss_test = mush_data[-miss_index,]

# To train the data we need to split the given dataset with the replaced
data.
# So, I have applied 80% for the training and the rest for the testing.
Replace_index=sample(1:nrow(replaceVal_mush),size =
```

```r
0.8*nrow(replaceVal_mush))
Replace_train = mush_data[Replace_index,]
Replace_test = mush_data[-Replace_index,]

# Apply the naive bayes classifier for both our data with missing values
miss_naiveBayes = naiveBayes(V1~.,data=miss_train)
# Apply the naive bayes classifier for both our data with replaced values
replace_naiveBayes = naiveBayes(V1~.,data=Replace_train)

# Apply the predict function for our classifier in both the test and train
data.
Miss_test_pred= predict(miss_naiveBayes,miss_test)
Miss_train_pred = predict(miss_naiveBayes,miss_train)
# Similarly apply the same predict function for the train and test for the
replaced data.
Replace_test_pred = predict(replace_naiveBayes,Replace_test)
Replace_train_pred = predict(replace_naiveBayes,Replace_train)

# With the confusion matrix we can find the false positives that the model
produced.

### These output values are subjective and can change when we re-run the
program.
# So, there can be a slight change in the values everytime we re-run the
program.

# Confusion Matrix
confusionMatrix(table(Miss_test_pred,miss_test$V1),dnn=c("Predicted","Actual"
))

## Confusion Matrix and Statistics
##
##
## Miss_test_pred   e   p
##              e 867  87
##              p   1 670
##
##                Accuracy : 0.9458
##                  95% CI : (0.9337, 0.9563)
##     No Information Rate : 0.5342
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8904
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9988
##             Specificity : 0.8851
##          Pos Pred Value : 0.9088
##          Neg Pred Value : 0.9985
##              Prevalence : 0.5342
```

```
##          Detection Rate : 0.5335
##    Detection Prevalence : 0.5871
##       Balanced Accuracy : 0.9420
##
##        'Positive' Class : e
##

cat("\n The accuracy for the missing values for the test is 0.9458")

##
##  The accuracy for the missing values for the test is 0.9458

cat("\n The false positive for the test is 94.")

##
##  The false positive for the test is 94.

# Similarly for the training data
confusionMatrix(table(Miss_train_pred,miss_train$V1),dnn=c("Predicted","Actua
l"))

## Confusion Matrix and Statistics
##
##
## Miss_train_pred    e    p
##               e 3315  361
##               p   25 2798
##
##                Accuracy : 0.9406
##                  95% CI : (0.9346, 0.9462)
##     No Information Rate : 0.5139
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8808
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9925
##             Specificity : 0.8857
##          Pos Pred Value : 0.9018
##          Neg Pred Value : 0.9911
##              Prevalence : 0.5139
##          Detection Rate : 0.5101
##    Detection Prevalence : 0.5656
##       Balanced Accuracy : 0.9391
##
##        'Positive' Class : e
##

cat("\n The accuracy for the missing values for the training is 0.9406")
```

```
##
##  The accuracy for the missing values for the training is 0.9406
```

```r
cat("\n The false positive for the train is 358.")
```

```
##
##  The false positive for the train is 358.
```

```r
# With the confusion matrix we can find the false positives that the model produced.

confusionMatrix(table(Replace_test_pred,Replace_test$V1),dnn=c("Predicted","Actual"))
```

```
## Confusion Matrix and Statistics
##
##
## Replace_test_pred   e   p
##                 e 841  81
##                 p   8 695
##
##                Accuracy : 0.9452
##                  95% CI : (0.933, 0.9558)
##     No Information Rate : 0.5225
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8898
##
##  Mcnemar's Test P-Value : 2.312e-14
##
##             Sensitivity : 0.9906
##             Specificity : 0.8956
##          Pos Pred Value : 0.9121
##          Neg Pred Value : 0.9886
##              Prevalence : 0.5225
##          Detection Rate : 0.5175
##    Detection Prevalence : 0.5674
##       Balanced Accuracy : 0.9431
##
##        'Positive' Class : e
##
```

```r
cat("\n The accuracy for the replaced values for the test is 0.9452.")
```

```
##
##  The accuracy for the replaced values for the test is 0.9452.
```

```r
cat("\n The false positive for the test is 103.")
```

```
##
##  The false positive for the test is 103.
```

```r
# With the confusion matrix we can find the false positives that the model produced.
confusionMatrix(table(Replace_train_pred,Replace_train$V1),dnn=c("Predicted","Actual"))
```

```
## Confusion Matrix and Statistics
##
##
## Replace_train_pred    e    p
##                  e 3335  367
##                  p   24 2773
##
##                Accuracy : 0.9398
##                  95% CI : (0.9338, 0.9455)
##     No Information Rate : 0.5168
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8791
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9929
##             Specificity : 0.8831
##          Pos Pred Value : 0.9009
##          Neg Pred Value : 0.9914
##              Prevalence : 0.5168
##          Detection Rate : 0.5132
##    Detection Prevalence : 0.5696
##       Balanced Accuracy : 0.9380
##
##        'Positive' Class : e
##
```

```r
cat("\n The accuracy for the replaced values for the training is 0.9398.")
```

```
##
##  The accuracy for the replaced values for the training is 0.9398.
```

```r
cat("\n The false positive for the train is 350.")
```

```
##
##  The false positive for the train is 350.
```

```r
## Once again these output values are subjective and can change when we re-run the program
# So, there can be a slight change in the values that are different from what you can see in the cat statement.
```