Assignment_3

Tirumal Achina

2023-10-13

Summary:

I have stored the data into the data frame ‘buried’, so that we can call it and use it where ever we need it. After that we have created a dummy variable “Injury” that classifies max severe injury i.e.; Max_Sev_IR equals 1 or 2 assumed that their is some degree of injury(Injury = Yes).”No injury” is implied if MaX_Sev_IR equals 0.

1: we have a attribute called Injury that is a categorical variable having classifiers yes or no. we have only information that accident is reported, so the reported accident would be predicted as INJURY = YES. This is because the number of “Injury= yes” records are more than the “Injury = No” records which would return it has “yes” that means it has more probability to be classified as an accident.

2: we are going to select the first 24 records in the dataset and considering two predicting factors WEATHER_R AND TRAF_CON_R. We stored this dataset into a variable names “Sub_Buried”. By creating a pivot table of these records we can clearly understand the data according to the levels of weather and traffic.

##Bayes Theorem : P(A/B) = (P(B/A)P(A))/P(B) where P(A),P(B) are events and P(B) not equal to 0.

2.1: we successfully computed the bayes conditional probabilities for six predictors of Injury would be yes. We got the following values for different combinations.

P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 0): 0.6666667

P(INJURY = Yes | WEATHER_R = 2 and TRAF_CON_R = 0): 0.1818182

P(INJURY = Yes | WEATHER_R = 2 and TRAF_CON_R = 2): 1

The other 3 combinations pf probability of injury=yes is 0.

P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 1): 0

P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 2): 0

P(INJURY = Yes | WEATHER_R = 2 and TRAF_CON_R = 1): 0

2.2: In this we have set a cutoff of 0.5 that is probability greater than 0.5 would be classified as “Yes” and less than 0.5 as “No”. Here we created a new attribute so that we can store the predicted injury and we can use it to compare between the actual and predicted. .

2.3: We computed the naive bayes conditional probability of injury with given attributes WEATHER_R = 1 and TRAF_CON_R = 1.

P(Injury=Yes | WEATHER_R = 1, TRAF_CON_R = 1) = (P(WEATHER_R = 1, TRAF_CON_R = 1 | Injury=yes) * P(Injury))/ P(WEATHER_R = 1, TRAF_CON_R = 1)

The results are as follows:

-If INJURY = YES, the probability is 0.

-If INJURY - NO , the probability is 1.

2.4: The Naive Bayes model's predictions and the exact Bayes classification has following observations.

-The primary and important observation is that both classifications showing "yes" at same indexes. By this we can say the Ranking(=Ordering) of Observations is "Equivalent".

-If the ranking is equivalent then that shows both classifications given equivalent importance to all the factors and also the similar understanding of data. Here models are making consistent decisions about the relative importance of datapoints.

-To conclude this evaluation is based on a subset that included only three attributes. To get the overall performance and equivalence of the model, we would typically evaluate it on a whole dataset and use standard evaluation metrics such as accuracy, precision, recall, and F1-score. These metrics provide a more comprehensive understanding of the model's classification performance.

3: Next thing here is we are dividing our whole dataset into training(60%) and validation sets(40%). After diving the sets we train the model with the training data that is used to identify the future accidents(new or unseen data) with the provided information.

-Validation set: This set is used to valid the data in it by using reference as training dataset so that we can know how well our model is trained when we give the unknown data(new data). It would classify the validation set by considering the training set.

-After partitioning of the data frame, we normalize the data so that all the data will be on same line. we perform operations on these normalized data so that we can get accurate values which we use for decision making.

-The important thing is that the attributes that we compare should have same levels so that we don't encounter an error and also should be of numeric or integer.

3.1: The confusion matrix shows the various metrics like Accuracy,Sensitivity and the matrix has True positives,False Positives, False Negatives and True Negatives.

-Accuracy: The model has an Accuracy of 0.5228 or 52.2%. This shows that our model identifies 52% cases rightly and 48% are wrongly identified.A Low accuracy in a model is due to Improper data quality, data leakage, Data pre-processing and many other factors to be considered.

-Sensitivity: The sensitivity is 0.15 that is 15%. This says that the model is unable to identify the True Positive rate or recall(yes cases).

-Specificity: The specificity is 0.87 or 87%. the model is identifying the Negative cases correctly.

3.2: Error rate is used to know how well the model is identifying the instances that shows lower error rate the more the model is efficient.

Error Rate = (Number of Misclassified Instances) / (Total Number of Instances)

-The error rate is 0.47 or 47%

```
#Loading the libraries that are required for the task
library(class)
library(caret)
```

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

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

```
library(e1071)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#Loading the data set and assigning it to buried variable.
buried <- read.csv("accidentsFull.csv")
dim(buried)
```

```
## [1] 42183    24
```

#1.Using the information in this dataset, if an accident has just been reported and no further information is available, what should the prediction be? (INJURY = Yes or No?) Why?s

```
#Here we added a dummy variable called injury.
buried$INJURY = ifelse(buried$MAX_SEV_IR %in% c(1,2),"yes","no")
table(buried$INJURY) # as yes is greater then no
```

```
##
##    no   yes
## 20721 21462
```

```
#To know the column names so that we can have clear idea of columns that we are working on.
t(names(buried))
```

```
##      [,1]       [,2]      [,3]      [,4]        [,5]       [,6]       [,7]
## [1,] "HOUR_I_R" "ALCHL_I" "ALIGN_I" "STRATUM_R" "WRK_ZONE" "WKDY_I_R" "INT_HWY"
##      [,8]         [,9]         [,10]       [,11]        [,12]
## [1,] "LGTCON_I_R" "MANCOL_I_R" "PED_ACC_R" "RELJCT_I_R" "REL_RWY_R"
##      [,13]        [,14]     [,15]      [,16]        [,17]      [,18]
## [1,] "PROFIL_I_R" "SPD_LIM" "SUR_COND" "TRAF_CON_R" "TRAF_WAY" "VEH_INVL"
##      [,19]       [,20]          [,21]      [,22]            [,23]
## [1,] "WEATHER_R" "INJURY_CRASH" "NO_INJ_I" "PRPTYDMG_CRASH" "FATALITIES"
##      [,24]        [,25]
## [1,] "MAX_SEV_IR" "INJURY"
```

#2.Select the first 24 records in the dataset and look only at the response (INJURY) and the two predictors WEATHER_R and TRAF_CON_R. Create a pivot table that examines INJURY as a function of the two predictors for these 24 records. Use all three variables in the pivot table as rows/columns.

```
#Creating the pivot tables
sub_buried <- buried[1:24,c("INJURY","WEATHER_R","TRAF_CON_R")]
sub_buried
```

```
##    INJURY WEATHER_R TRAF_CON_R
## 1     yes         1          0
## 2      no         2          0
## 3      no         2          1
## 4      no         1          1
## 5      no         1          0
## 6     yes         2          0
## 7      no         2          0
## 8     yes         1          0
## 9      no         2          0
## 10     no         2          0
## 11     no         2          0
## 12     no         1          2
## 13    yes         1          0
## 14     no         1          0
## 15    yes         1          0
## 16    yes         1          0
## 17     no         2          0
## 18     no         2          0
## 19     no         2          0
## 20     no         2          0
## 21    yes         1          0
## 22     no         1          0
## 23    yes         2          2
## 24    yes         2          0
```

```
pi_table1 <- ftable(sub_buried)
pi_table1
```

```
##                 TRAF_CON_R 0 1 2
## INJURY WEATHER_R
## no     1                   3 1 1
##        2                   9 1 0
## yes    1                   6 0 0
##        2                   2 0 1
```

```
pi_table2 <- ftable(sub_buried[,-1])
pi_table2
```

```
##           TRAF_CON_R  0  1  2
## WEATHER_R
## 1                      9  1  1
## 2                     11  1  1
```

#2.1.Compute the exact Bayes conditional probabilities of an injury (INJURY = Yes) given the six possible combinations of the predictors.

```
#bayes
#INJURY = YES
pair_1 = pi_table1[3,1]/pi_table2[1,1]
cat("P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 0):", pair_1, "\n")
```

```
## P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 0): 0.6666667
```

```
pair_2 = pi_table1[3,2]/pi_table2[1,2]
cat("P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 1):", pair_2, "\n")
```

```
## P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 1): 0
```

```
pair_3 = pi_table1[3,3]/pi_table2[1,3]
cat("P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 2):", pair_3, "\n")
```

```
## P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 2): 0
```

```
pair_4 = pi_table1[4,1]/pi_table2[2,1]
cat("P(INJURY = Yes | WEATHER_R = 2 and TRAF_CON_R = 0):", pair_4, "\n")
```

```
## P(INJURY = Yes | WEATHER_R = 2 and TRAF_CON_R = 0): 0.1818182
```

```
pair_5 = pi_table1[4,2]/pi_table2[2,2]
cat("P(INJURY = Yes | WEATHER_R = 2 and TRAF_CON_R = 1):", pair_5, "\n")
```

```
## P(INJURY = Yes | WEATHER_R = 2 and TRAF_CON_R = 1): 0
```

```
pair_6 = pi_table1[4,3]/pi_table2[2,3]
cat("P(INJURY = Yes | WEATHER_R = 2 and TRAF_CON_R = 2):", pair_6, "\n")
```

```
## P(INJURY = Yes | WEATHER_R = 2 and TRAF_CON_R = 2): 1
```

```
#Now we check the condition whether Injury = no

dual_1 = pi_table1[1,1]/pi_table2[1,1]
cat("P(INJURY = no | WEATHER_R = 1 and TRAF_CON_R = 0):", dual_1, "\n")
```

```
## P(INJURY = no | WEATHER_R = 1 and TRAF_CON_R = 0): 0.3333333
```

```
dual_2 = pi_table1[1,2]/pi_table2[1,2]
cat("P(INJURY = no | WEATHER_R = 1 and TRAF_CON_R = 1):", dual_2, "\n")
```

```
## P(INJURY = no | WEATHER_R = 1 and TRAF_CON_R = 1): 1
```

```
dual_3 = pi_table1[1,3]/pi_table2[1,3]
cat("P(INJURY = no | WEATHER_R = 1 and TRAF_CON_R = 2):", dual_3, "\n")
```

```
## P(INJURY = no | WEATHER_R = 1 and TRAF_CON_R = 2): 1
```

```
dual_4 = pi_table1[2,1]/pi_table2[2,1]
cat("P(INJURY = no | WEATHER_R = 2 and TRAF_CON_R = 0):", dual_4, "\n")
```

```
## P(INJURY = no | WEATHER_R = 2 and TRAF_CON_R = 0): 0.8181818
```

```
dual_5 = pi_table1[2,2]/pi_table2[2,2]
cat("P(INJURY = no | WEATHER_R = 2 and TRAF_CON_R = 1):", dual_5, "\n")
```

```
## P(INJURY = no | WEATHER_R = 2 and TRAF_CON_R = 1): 1
```

```
dual_6 = pi_table1[2,3]/pi_table2[2,3]
cat("P(INJURY = no | WEATHER_R = 2 and TRAF_CON_R = 2):", dual_6, "\n")
```

```
## P(INJURY = no | WEATHER_R = 2 and TRAF_CON_R = 2): 0
```

```
#Now probability of the total occurences.
```

#2.2 Classify the 24 accidents using these probabilities and a cutoff of 0.5.

```
# we have calculated the conditional probabilities already, we can use them to classify the 24 a
ccidents.
prob_injury <- rep(0,24)
for(i in 1:24){
  print(c(sub_buried$WEATHER_R[i],sub_buried$TRAF_CON_R[i]))

  if(sub_buried$WEATHER_R[i] == "1" && sub_buried$TRAF_CON_R[i] == "0"){
    prob_injury[i] = pair_1

  } else if (sub_buried$WEATHER_R[i] == "1" && sub_buried$TRAF_CON_R[i] == "1"){
    prob_injury[i] = pair_2

  } else if (sub_buried$WEATHER_R[i] == "1" && sub_buried$TRAF_CON_R[i] == "2"){
    prob_injury[i] = pair_3

  }
  else if (sub_buried$WEATHER_R[i] == "2" && sub_buried$TRAF_CON_R[i] == "0"){
    prob_injury[i] = pair_4

  } else if (sub_buried$WEATHER_R[i] == "2" && sub_buried$TRAF_CON_R[i] == "1"){
    prob_injury[i] = pair_5

  }
  else if(sub_buried$WEATHER_R[i] == "2" && sub_buried$TRAF_CON_R[i] == "2"){
    prob_injury[i] = pair_6
  }

}
```

```
## [1] 1 0
## [1] 2 0
## [1] 2 1
## [1] 1 1
## [1] 1 0
## [1] 2 0
## [1] 2 0
## [1] 1 0
## [1] 2 0
## [1] 2 0
## [1] 2 0
## [1] 1 2
## [1] 1 0
## [1] 1 0
## [1] 1 0
## [1] 1 0
## [1] 2 0
## [1] 2 0
## [1] 2 0
## [1] 2 0
## [1] 1 0
## [1] 1 0
## [1] 2 2
## [1] 2 0
```

```
#cutoff 0.5

sub_buried$prob_injury = prob_injury
sub_buried$pred.prob  = ifelse(sub_buried$prob_injury>0.5, "yes","no")


head(sub_buried)
```

```
##   INJURY WEATHER_R TRAF_CON_R prob_injury pred.prob
## 1    yes         1          0   0.6666667       yes
## 2     no         2          0   0.1818182        no
## 3     no         2          1   0.0000000        no
## 4     no         1          1   0.0000000        no
## 5     no         1          0   0.6666667       yes
## 6    yes         2          0   0.1818182        no
```

#2.3Compute manually the naive Bayes conditional probability of an injury given WEATHER_R = 1 and TRAF_CON_R = 1.

```
IY = pi_table1[3,2]/pi_table2[1,2]
I = (IY * pi_table1[3, 2]) / pi_table2[1, 2]
cat("P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 1):", IY, "\n")
```

```
## P(INJURY = Yes | WEATHER_R = 1 and TRAF_CON_R = 1): 0
```

```
IN = pi_table1[1,2]/pi_table2[1,2]
N = (IY * pi_table1[3, 2]) / pi_table2[1, 2]
cat("P(INJURY = no | WEATHER_R = 1 and TRAF_CON_R = 1):", IN, "\n")
```

```
## P(INJURY = no | WEATHER_R = 1 and TRAF_CON_R = 1): 1
```

#2.4.Run a naive Bayes classifier on the 24 records and two predictors. Check the model output to obtain probabilities and classifications for all 24 records. Compare this to the exact Bayes classification. Are the resulting classifications equivalent? Is the ranking (= ordering) of observations equivalent?

```
new_b <- naiveBayes(INJURY ~ TRAF_CON_R + WEATHER_R,
                 data = sub_buried)


new_buried <- predict(new_b, newdata = sub_buried, type = "raw")
sub_buried$nbpred.prob <- new_buried[,2]
```

```
new_c <- train(INJURY ~ TRAF_CON_R + WEATHER_R,
      data = sub_buried, method = "nb")
```

```
## Warning: model fit failed for Resample02: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R
```

```
## Warning: model fit failed for Resample06: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R
```

```
## Warning: model fit failed for Resample08: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R
```

```
## Warning: model fit failed for Resample09: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R
```

```
## Warning: model fit failed for Resample10: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R, WEATHER_R
```

```
## Warning: model fit failed for Resample11: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R, WEATHER_R
```

```
## Warning: model fit failed for Resample14: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R
```

```
## Warning: model fit failed for Resample16: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R
```

```
## Warning: model fit failed for Resample21: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R
```

```
## Warning: model fit failed for Resample24: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: WEATHER_R
```

```
## Warning: model fit failed for Resample25: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBaye
s.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
##   Zero variances for at least one class in variables: TRAF_CON_R
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
predict(new_c, newdata = sub_buried[,c("INJURY", "WEATHER_R", "TRAF_CON_R")])
```

```
##  [1] no  no  no  no  no  no  no  no  no  no  no  yes no  no  no  no  no  no  no
## [20] no  no  no  no  no
## Levels: no yes
```

```
predict(new_c, newdata = sub_buried[,c("INJURY", "WEATHER_R", "TRAF_CON_R")],
                                    type = "raw")
```

```
##  [1] no  no  no  no  no  no  no  no  no  no  no  yes no  no  no  no  no  no  no
## [20] no  no  no  no  no
## Levels: no yes
```

#Let us now return to the entire dataset. Partition the data into training (60%) and validation (40%).

```
accident = buried[c(-24)]

set.seed(1)
acc.index = sample(row.names(accident), 0.6*nrow(accident)[1])
valid.index = setdiff(row.names(accident), acc.index)


acc.df = accident[acc.index,]
valid.df= accident[valid.index,]

dim(acc.df)
```

```
## [1] 25309    24
```

```
dim(valid.df)
```

```
## [1] 16874    24
```

```
norm.values <- preProcess(acc.df[,], method = c("center", "scale"))
acc.norm.df <- predict(norm.values, acc.df[, ])
valid.norm.df <- predict(norm.values, valid.df[, ])


class(acc.norm.df$INJURY)
```

```
## [1] "character"
```

```
acc.norm.df$INJURY <- as.factor(acc.norm.df$INJURY)

class(acc.norm.df$INJURY)
```

```
## [1] "factor"
```

##3.1.Run a naive Bayes classifier on the complete training set with the relevant predictors (and INJURY as the response). Note that all predictors are categorical. Show the confusion matrix.

```
nb_model <- naiveBayes(INJURY ~ WEATHER_R + TRAF_CON_R, data = acc.norm.df)

predictions <- predict(nb_model, newdata = valid.norm.df)

#Checking factor levels in validation dataset and training dataset
valid.norm.df$INJURY <- factor(valid.norm.df$INJURY, levels = levels(acc.norm.df$INJURY))

# Show the confusion matrix
confusionMatrix(predictions, valid.norm.df$INJURY)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  1285 1118
##        yes 6934 7537
##
##                Accuracy : 0.5228
##                  95% CI : (0.5152, 0.5304)
##     No Information Rate : 0.5129
##     P-Value [Acc > NIR] : 0.005162
##
##                   Kappa : 0.0277
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.15635
##             Specificity : 0.87083
##          Pos Pred Value : 0.53475
##          Neg Pred Value : 0.52083
##              Prevalence : 0.48708
##          Detection Rate : 0.07615
##    Detection Prevalence : 0.14241
##       Balanced Accuracy : 0.51359
##
##        'Positive' Class : no
##
```

#3.2.What is the overall error of the validation set?

```
error_rate <- 1 - sum(predictions == valid.norm.df$INJURY) / nrow(valid.norm.df)
error_rate
```

```
## [1] 0.4771838
```