FML ASSIGNMENT.3

SANIA FATIMA

2023-10-15

Problem Statement

The file accidentsFull.csv contains information on 42,183 actual automobile accidents in 2001 in the United States that involved one of three levels of injury: NO INJURY, INJURY, or FATALITY. For each accident, additional information is recorded, such as day of week, weather conditions, and road type. A firm might be interested in developing a system for quickly classifying the severity of an accident based on initial reports and associated data in the system (some of which rely on GPS-assisted reporting).

Our goal here is to predict whether an accident just reported will involve an injury (MAX_SEV_IR = 1 or 2) or will not (MAX_SEV_IR = 0). For this purpose, create a dummy variable called INJURY that takes the value "yes" if MAX_SEV_IR = 1 or 2, and otherwise "no."

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?
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 12 records. Use all three variables in the pivot table as rows/columns.

- Compute the exact Bayes conditional probabilities of an injury (INJURY = Yes) given the six possible combinations of the predictors.
- Classify the 24 accidents using these probabilities and a cutoff of 0.5.
- Compute manually the naive Bayes conditional probability of an injury given WEATHER_R = 1 and TRAF_CON_R = 1.
- 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?

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

- 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.
- What is the overall error of the validation set?

Summary

1. Data Input and Cleaning: • I start by loading the necessary R libraries: e1071 and caret. These libraries provide functions and tools for working with machine learning algorithms and classification. • Next, we read the dataset from a CSV file using the read.csv function and store it in a variable called accidents. The dataset appears to contain information about automobile accidents. • After loading the dataset, you create a binary dummy variable called "INJURY" based on the "MAX_SEV_IR" column. If "MAX_SEV_IR" is greater than 0, it's considered an injury (INJURY = "yes"), otherwise, it's not an injury (INJURY = "no"). • We convert all the variables in the dataset to factors using a loop. This step is often necessary when working with categorical data in R. • Finally, I calculate the probabilities of "yes" and "no" in the "INJURY" variable and print the results.
   These probabilities indicate the likelihood of an accident resulting in an injury or not. The code demonstrates the importance of data preprocessing and cleaning. Variables are converted to the appropriate data types (in this case, factors) to ensure they can be used effectively in the analysis.
2. Data Exploration and Analysis: • In this section, we address several questions: • I select the first 24 records in the dataset and create a subset called accidents24 that includes the "INJURY" response variable and the two predictor variables "WEATHER_R" and "TRAF_CON_R." • I also create a pivot table (dt1) that examines the relationship between "INJURY" and the two predictors, "WEATHER_R" and "TRAF_CON_R." This provides a summary of how different combinations of these predictors relate to injuries. • I compute the exact Bayes conditional probabilities of an injury given six possible combinations of the predictors. These conditional probabilities are based on the frequencies observed in the dataset. • I classify the 24 accidents using these conditional probabilities and a cutoff of 0.5. If the probability of injury is greater than or equal to 0.5, the accident is classified as "yes"; otherwise, it's classified as "no." • I manually calculate the naive Bayes conditional probability of an injury given "WEATHER_R = 1" and "TRAF_CON_R = 1." This is done using the formula for conditional probability. • I also run a Naive Bayes classifier on the 24 records using the naiveBayes function and compare the model's output to the exact Bayes classification. I check whether the resulting classifications are equivalent and whether the ranking of observations is the same.
3. Training and Validation Set: • I move on to using the entire dataset and partition it into training (60%) and validation (40%) sets. This is a common practice in machine learning to train and evaluate models on different datasets. • Then,I train a Naive Bayes classifier on the training data using the "WEATHER_R" and "TRAF_CON_R" as predictors and "INJURY" as the response variable. The naiveBayes function is used for this purpose. • You use the trained classifier to predict "INJURY" for the validation set. The predictions are stored in the variable predicted_classifications.
4. Confusion Matrix and Overall Error: • I create a confusion matrix (confusion_matrix) to assess the performance of the Naive Bayes classifier on the validation set. A confusion matrix shows the number of true positives, true negatives, false positives, and false negatives. • I calculate the overall error (misclassification rate) by comparing the predicted classifications to the true labels in the validation set. • I repeat the same steps for the training set as well, creating another confusion matrix and calculating the overall error for the training set.

The code provides a comprehensive analysis of the dataset, including data cleaning, data exploration, and model evaluation using Naive Bayes classification. It's a common workflow in predictive modeling to preprocess data, build models, and evaluate their performance using various metrics such as confusion matrices and misclassification rates.

Data Input and Cleaning

Load the required libraries and read the input file

```r
library(e1071)
library(caret)
```

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

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

Load the file in.

```r
accidents <- read.csv("C:/Users/Sania fatima/Desktop/ass.FML/accidentsFull (2).csv")
head(accidents,n=24)
```

```
##    HOUR_I_R ALCHL_I ALIGN_I STRATUM_R WRK_ZONE WKDY_I_R INT_HWY LGTCON_I_R
## 1         0       2       2         1        0        1       0          3
## 2         1       2       1         0        0        1       1          3
## 3         1       2       1         0        0        1       0          3
## 4         1       2       1         1        0        0       0          3
## 5         1       1       1         0        0        1       0          3
## 6         1       2       1         1        0        1       0          3
## 7         1       2       1         0        0        1       1          3
## 8         1       2       1         1        0        1       0          3
## 9         1       2       1         1        0        1       0          3
## 10        0       2       1         0        0        0       0          3
## 11        1       2       1         0        0        1       0          3
## 12        1       2       1         1        0        1       0          3
## 13        1       2       1         1        0        1       0          3
## 14        1       2       2         0        0        1       0          3
## 15        1       2       2         1        0        1       0          3
## 16        1       2       2         1        0        1       0          3
## 17        1       2       1         1        0        1       0          3
## 18        1       2       1         1        0        0       0          3
## 19        1       2       1         1        0        1       0          3
## 20        1       2       1         0        0        1       0          3
## 21        1       2       1         1        0        1       0          3
## 22        1       2       2         0        0        1       0          3
## 23        1       2       1         0        0        1       0          3
## 24        1       2       1         1        0        1       9          3
##    MANCOL_I_R PED_ACC_R RELJCT_I_R REL_RWY_R PROFIL_I_R SPD_LIM SUR_COND
## 1           0         0          1         0          1      40        4
## 2           2         0          1         1          1      70        4
## 3           2         0          1         1          1      35        4
## 4           2         0          1         1          1      35        4
## 5           2         0          0         1          1      25        4
## 6           0         0          1         0          1      70        4
## 7           0         0          0         0          1      70        4
## 8           0         0          0         0          1      35        4
## 9           0         0          1         0          1      30        4
## 10          0         0          1         0          1      25        4
## 11          0         0          0         0          1      55        4
## 12          2         0          0         1          1      40        4
## 13          1         0          0         1          1      40        4
## 14          0         0          0         0          1      25        4
```

```
## 15          0         0          0         0          1      35        4
## 16          0         0          0         0          1      45        4
## 17          0         0          0         0          1      20        4
## 18          0         0          0         0          1      50        4
## 19          0         0          0         0          1      55        4
## 20          0         0          1         1          1      55        4
## 21          0         0          1         0          0      45        4
## 22          0         0          1         0          0      65        4
## 23          0         0          0         0          0      65        4
## 24          2         0          1         1          0      55        4
##    TRAF_CON_R TRAF_WAY VEH_INVL WEATHER_R INJURY_CRASH NO_INJ_I PRPTYDMG_CRASH
## 1           0        3        1         1            1        1              0
## 2           0        3        2         2            0        0              1
## 3           1        2        2         2            0        0              1
## 4           1        2        2         1            0        0              1
## 5           0        2        3         1            0        0              1
## 6           0        2        1         2            1        1              0
## 7           0        2        1         2            0        0              1
## 8           0        1        1         1            1        1              0
## 9           0        1        1         2            0        0              1
## 10          0        1        1         2            0        0              1
## 11          0        1        1         2            0        0              1
## 12          2        1        2         1            0        0              1
## 13          0        1        4         1            1        2              0
## 14          0        1        1         1            0        0              1
## 15          0        1        1         1            1        1              0
## 16          0        1        1         1            1        1              0
## 17          0        1        1         2            0        0              1
## 18          0        1        1         2            0        0              1
## 19          0        1        1         2            0        0              1
## 20          0        1        1         2            0        0              1
## 21          0        3        1         1            1        1              0
## 22          0        3        1         1            0        0              1
## 23          2        2        1         2            1        2              0
## 24          0        2        2         2            1        1              0
##    FATALITIES MAX_SEV_IR
## 1           0          1
## 2           0          0
## 3           0          0
## 4           0          0
## 5           0          0
## 6           0          1
## 7           0          0
## 8           0          1
## 9           0          0
## 10          0          0
## 11          0          0
## 12          0          0
## 13          0          1
## 14          0          0
## 15          0          1
## 16          0          1
## 17          0          0
## 18          0          0
```

```
## 19          0          0
## 20          0          0
## 21          0          1
## 22          0          0
## 23          0          1
## 24          0          1
```

Create a Dummy Variable called Injury

This code is a common step in binary classification tasks, where you convert a continuous or multi-class target variable (in this case, "MAX_SEV_IR") into a binary outcome variable (injury or no injury). The table function is then used to summarize and show the distribution of injuries in the dataset.The value consists of "no" is 20721 and "yes" is 21462 If this condition accident$MAX-SEV_IR is greater than 0.It has three levels 0,1,2 (1&2 indicates that it was either an accident or fatality ,so we wnt the injury to be YES.If injury is YES then take YES and vice versa)

```
accidents$INJURY =ifelse(accidents$MAX_SEV_IR>0,"yes","no")
table(accidents$INJURY)
```

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

VARIABLE TO FACTORS: Converting variables to factors is common when dealing with categorical or nominal data, as it allows us to work with these variables more effectively in statistical analysis or modeling. Factors are used to represent categorical variables, and they can be particularly useful in R for tasks like creating statistical models, making data summaries, and generating visualizations that involve categorical data.

```
# Convert variables to factor
for (i in c(1:dim(accidents)[2])){
  accidents[,i] <- as.factor(accidents[,i])
}
head(accidents,n=24)
```

```
##    HOUR_I_R ALCHL_I ALIGN_I STRATUM_R WRK_ZONE WKDY_I_R INT_HWY LGTCON_I_R
## 1         0       2       2         1        0        1       0          3
## 2         1       2       1         0        0        1       1          3
## 3         1       2       1         0        0        1       0          3
## 4         1       2       1         1        0        0       0          3
## 5         1       1       1         0        0        1       0          3
## 6         1       2       1         1        0        1       0          3
## 7         1       2       1         0        0        1       1          3
## 8         1       2       1         1        0        1       0          3
## 9         1       2       1         1        0        1       0          3
## 10        0       2       1         0        0        0       0          3
## 11        1       2       1         0        0        1       0          3
## 12        1       2       1         1        0        1       0          3
## 13        1       2       1         1        0        1       0          3
## 14        1       2       2         0        0        1       0          3
## 15        1       2       2         1        0        1       0          3
## 16        1       2       2         1        0        1       0          3
## 17        1       2       1         1        0        1       0          3
```

```
## 18        1       2       1         1        0        0       0          3
## 19        1       2       1         1        0        1       0          3
## 20        1       2       1         0        0        1       0          3
## 21        1       2       1         1        0        1       0          3
## 22        1       2       2         0        0        1       0          3
## 23        1       2       1         0        0        1       0          3
## 24        1       2       1         1        0        1       9          3
##    MANCOL_I_R PED_ACC_R RELJCT_I_R REL_RWY_R PROFIL_I_R SPD_LIM SUR_COND
## 1           0         0          1         0          1      40        4
## 2           2         0          1         1          1      70        4
## 3           2         0          1         1          1      35        4
## 4           2         0          1         1          1      35        4
## 5           2         0          0         1          1      25        4
## 6           0         0          1         0          1      70        4
## 7           0         0          0         0          1      70        4
## 8           0         0          0         0          1      35        4
## 9           0         0          1         0          1      30        4
## 10          0         0          1         0          1      25        4
## 11          0         0          0         0          1      55        4
## 12          2         0          0         1          1      40        4
## 13          1         0          0         1          1      40        4
## 14          0         0          0         0          1      25        4
## 15          0         0          0         0          1      35        4
## 16          0         0          0         0          1      45        4
## 17          0         0          0         0          1      20        4
## 18          0         0          0         0          1      50        4
## 19          0         0          0         0          1      55        4
## 20          0         0          1         1          1      55        4
## 21          0         0          1         0          0      45        4
## 22          0         0          1         0          0      65        4
## 23          0         0          0         0          0      65        4
## 24          2         0          1         1          0      55        4
##    TRAF_CON_R TRAF_WAY VEH_INVL WEATHER_R INJURY_CRASH NO_INJ_I PRPTYDMG_CRASH
## 1           0        3        1         1            1        1              0
## 2           0        3        2         2            0        0              1
## 3           1        2        2         2            0        0              1
## 4           1        2        2         1            0        0              1
## 5           0        2        3         1            0        0              1
## 6           0        2        1         2            1        1              0
## 7           0        2        1         2            0        0              1
## 8           0        1        1         1            1        1              0
## 9           0        1        1         2            0        0              1
## 10          0        1        1         2            0        0              1
## 11          0        1        1         2            0        0              1
## 12          2        1        2         1            0        0              1
## 13          0        1        4         1            1        2              0
## 14          0        1        1         1            0        0              1
## 15          0        1        1         1            1        1              0
## 16          0        1        1         1            1        1              0
## 17          0        1        1         2            0        0              1
## 18          0        1        1         2            0        0              1
## 19          0        1        1         2            0        0              1
## 20          0        1        1         2            0        0              1
## 21          0        3        1         1            1        1              0
```

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

PROBABILITIES:

probability_yes <- mean(accidents$INJURY == "yes"): This calculates the probability of "yes" in the "INJURY" variable. It does so by comparing each element of the "INJURY" variable to the string "yes" and then taking the mean of the resulting logical values. In other words, it calculates the proportion of "yes" values in the "INJURY" variable.So, the probability of yes is 50.8%

probability_no <- mean(accidents$INJURY == "no"): Similarly, this line calculates the probability of "no" in the "INJURY" variable by comparing each element to the string "no" and taking the meana and the probability of no is 40.12%

These probabilities provide insights into the distribution of injuries in dataset. It's important information for understanding the prevalence of injuries in the context of the analysis.

```r
probability_yes <- mean(accidents$INJURY == "yes")
probability_no <- mean(accidents$INJURY == "no")

cat("Probability of 'yes':", probability_yes, "\n")
```

```
## Probability of 'yes': 0.5087832
```

```r
cat("Probability of 'no':", probability_no, "\n")
```

```
## Probability of 'no': 0.4912168
```

Questions

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 12 records. Use all three variables in the pivot table as rows/columns.

- Compute the exact Bayes conditional probabilities of an injury (INJURY = Yes) given the six possible combinations of the predictors.
- Classify the 24 accidents using these probabilities and a cutoff of 0.5.
- Compute manually the naive Bayes conditional probability of an injury given WEATHER_R = 1 and TRAF_CON_R = 1.
- 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?

dt1 <- ftable(accidents24): This creates a pivot table (ftable) using the entire "accidents24" dataset. The pivot table examines the relationships between "INJURY," "WEATHER_R," and "TRAF_CON_R." It shows how these variables are distributed in the first 24 records.

dt2 <- ftable(accidents24[, -1]): This line creates another pivot table using only the variables "WEATHER_R" and "TRAF_CON_R" (excluding "INJURY"). This table focuses on the relationships between the two predictor variables "WEATHER_R" and "TRAF_CON_R" without considering "INJURY."

The resulting pivot tables provide a visual summary of how the variables are distributed and related in the first 24 records of the dataset, which can be useful for initial exploratory data analysis and understanding the data's structure.

```
accidents24 <- accidents[1:24,c("INJURY","WEATHER_R","TRAF_CON_R")]
#head(accidents24)
```

#generating pivot table

```
dt1 <- ftable(accidents24)
dt2 <- ftable(accidents24[,-1]) # print table only for conditions
dt1
```

```
##                  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
```

```
dt2
```

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

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 12 records. Use all three variables in the pivot table as rows/columns.

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

The code calculates these conditional probabilities using the pivot tables and the joint probabilities provided in the tables. These conditional probabilities are important in Bayesian statistics and machine learning for making predictions and classifications based on the available data.

```
# Injury = yes
p1 = dt1[3,1] / dt2[1,1] # Injury, Weather=1 and Traf=0
p2 = dt1[4,1] / dt2[2,1] # Injury, Weather=2, Traf=0
p3 = dt1[3,2] / dt2[1,2] # Injury, W=1, T=1
p4 = dt1[4,2] / dt2[2,2] # I, W=2,T=1
p5 = dt1[3,3] / dt2[1,3] # I, W=1,T=2
p6 = dt1[4,3]/ dt2[2,3] #  I,W=2,T=2

# Injury = no
n1 = dt1[1,1] / dt2[1,1] # Weather=1 and Traf=0
n2 = dt1[2,1] / dt2[2,1] # Weather=2, Traf=0
n3 = dt1[1,2] / dt2[1,2] # W=1, T=1
n4 = dt1[2,2] / dt2[2,2] # W=2,T=1
n5 = dt1[1,3] / dt2[1,3] # W=1,T=2
n6 = dt1[2,3] / dt2[2,3] # W=2,T=2
print(c(p1,p2,p3,p4,p5,p6))
```

```
## [1] 0.6666667 0.1818182 0.0000000 0.0000000 0.0000000 1.0000000
```

```
print(c(n1,n2,n3,n4,n5,n6))
```

```
## [1] 0.3333333 0.8181818 1.0000000 1.0000000 1.0000000 0.0000000
```

2. Let us now compute

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

After calculating the conditional probabilities, the code assigns these probabilities to a new variable called "prob.inj" in the "accidents24" dataset.

accidents24$pred.prob <- ifelse(accidents24$prob.inj > 0.5, "yes", "no"): This line classifies the records in the "accidents24" dataset into "Yes" or "No" based on the calculated probabilities and a threshold of 0.5. If the probability of injury ("prob.inj") is greater than 0.5, it assigns "yes" to "accidents24$pred.prob"; otherwise, it assigns "no."

Finally, prob.inj contains the calculated probabilities of injury for the first 24 records, and "accidents24$pred.prob" contains the predicted classifications based on these probabilities.

```
prob.inj <- rep(0,24)

for (i in 1:24) {
  print(c(accidents24$WEATHER_R[i],accidents24$TRAF_CON_R[i]))
  if (accidents24$WEATHER_R[i] == "1") {
    if (accidents24$TRAF_CON_R[i]=="0"){
      prob.inj[i] = p1
```

```
    }
    else if (accidents24$TRAF_CON_R[i]=="1") {
      prob.inj[i] = p3
    }
    else if (accidents24$TRAF_CON_R[i]=="2") {
      prob.inj[i] = p5
    }
  }
  else {
    if (accidents24$TRAF_CON_R[i]=="0"){
      prob.inj[i] = p2
    }
    else if (accidents24$TRAF_CON_R[i]=="1") {
      prob.inj[i] = p4
    }
    else if (accidents24$TRAF_CON_R[i]=="2") {
      prob.inj[i] = p6
    }
  }
}
```

```
## [1] 1 0
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
## [1] 2 1
## Levels: 1 2 0
## [1] 1 1
## Levels: 1 2 0
## [1] 1 0
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
## [1] 1 0
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
## [1] 1 2
## Levels: 1 2 0
## [1] 1 0
## Levels: 1 2 0
## [1] 1 0
## Levels: 1 2 0
## [1] 1 0
## Levels: 1 2 0
## [1] 1 0
## Levels: 1 2 0
```

```
## [1] 2 0
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
## [1] 1 0
## Levels: 1 2 0
## [1] 1 0
## Levels: 1 2 0
## [1] 2 2
## Levels: 1 2 0
## [1] 2 0
## Levels: 1 2 0
```

```
accidents24$prob.inj <- prob.inj

accidents24$pred.prob <- ifelse(accidents24$prob.inj>0.5, "yes", "no")

prob.inj
```

```
##  [1] 0.6666667 0.1818182 0.0000000 0.0000000 0.6666667 0.1818182 0.1818182
##  [8] 0.6666667 0.1818182 0.1818182 0.1818182 0.0000000 0.6666667 0.6666667
## [15] 0.6666667 0.6666667 0.1818182 0.1818182 0.1818182 0.1818182 0.6666667
## [22] 0.6666667 1.0000000 0.1818182
```

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

This code essentially demonstrates the calculation of conditional probabilities using the Naive Bayes approach, specifically for the given combination of "WEATHER_R" and "TRAF_CON_R" values.

The function calculates the conditional probability of injury given the specific values of "WEATHER_R" and "TRAF_CON_R" based on the Naive Bayes theorem. The numerator represents the joint probabilities of the predictors given injury, and the denominator includes the Marginal probability (joint probabilities given no injury).

Example probabilities are provided for each of the required parameters, which are used to demonstrate the calculation of the conditional probability.

The code then calculates the conditional probability for the specific case where "WEATHER_R = 1" and "TRAF_CON_R = 1" and assigns it to the variable result.

So the naivye bayes probability of this is 0.When a Naive Bayes probability is calculated as exactly 0, it indicates that the model, based on the available training data, is certain that the event being predicted cannot occur under the given conditions. In the context of classification, a Naive Bayes classifier assigns probabilities to different classes based on the available features (predictor variables) in the training data. When the classifier assigns a probability of 0 to a certain class, it's saying that there is no chance or no evidence in the training data that the data point belongs to that class under the given conditions.

NAIVE BAYES: #p(I=Y|W=1,T=1)=P(W=1|Y)P(T=1|Y)P(Y)/P(W=1|Y)P(T=1|Y)P(Y)+P(W=1|N)P(T=1|N)P(N)

```r
# Define a function to calculate conditional probability
naive_bayes_probability <- function(P_W1_Y, P_T1_Y, P_Y, P_W1_N, P_T1_N, P_N) {
  numerator <- P_W1_Y * P_T1_Y * P_Y
  denominator <- P_W1_Y * P_T1_Y * P_Y + P_W1_N * P_T1_N * P_N
  conditional_probability <- numerator / denominator
  return(conditional_probability)
}
# Example probabilities
P_W1_Y <- 0.7  # P(W=1|Y)
P_T1_Y <- 0.8  # P(T=1|Y)
P_Y <- 0.4     # P(Y)
P_W1_N <- 0.3  # P(W=1|N)
P_T1_N <- 0.2  # P(T=1|N)
P_N <- 0.6     # P(N)
# Calculate conditional probability
result <- naive_bayes_probability(P_W1_Y, P_T1_Y, P_Y, P_W1_N, P_T1_N, P_N)
cat("P(I=Y|W=1, T=1) =", result, "\n")
```

```
## P(I=Y|W=1, T=1) = 0.8615385
```

```r
# Calculate the naive Bayes conditional probability
p_injury_yes_given_weather1_traf1 <- p3

# Display the result
cat("The Naive Bayes conditional probability of an injury given WEATHER_R = 1 and TRAF_CON_R = 1 is:", p
```

```
## The Naive Bayes conditional probability of an injury given WEATHER_R = 1 and TRAF_CON_R = 1 is: 0
```

2.

- 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?

The result is a data frame where we have the original data, and for each observation, we also have the predicted probability of the "INJURY = Yes" class according to the Naive Bayes model. This can be useful for further analysis or for making decisions based on these probabilities.

```r
nb <- naiveBayes(INJURY ~ TRAF_CON_R + WEATHER_R,
                 data = accidents24)

nbt <- predict(nb, newdata = accidents24,type = "raw")
accidents24$nbpred.prob <- nbt[,2] # Transfer the "Yes" nb prediction
head(accidents24)
```

```
##   INJURY WEATHER_R TRAF_CON_R  prob.inj pred.prob nbpred.prob
## 1    yes         1          0 0.6666667       yes 0.571428571
## 2     no         2          0 0.1818182        no 0.250000000
## 3     no         2          1 0.0000000        no 0.002244949
## 4     no         1          1 0.0000000        no 0.008919722
## 5     no         1          0 0.6666667       yes 0.571428571
## 6    yes         2          0 0.1818182        no 0.250000000
```

Let us use klaR

The result is a trained Naive Bayes model stored in the nb2 object. This model can be used for making predictions on new data or for further analysis of the classification task. The train function is often used for training and evaluating classification models, and it may provide additional information about the model's performance, depending on the specific implementation and settings used.

```
library(klaR)
```

```
## Loading required package: MASS
```

```
nb2 <- train(INJURY ~ TRAF_CON_R + WEATHER_R,
             data = accidents24, method = "nb")
```

```
## Warning: model fit failed for Resample01: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample02: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample03: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample04: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample05: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample06: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample07: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample08: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample09: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample10: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample11: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample12: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2, WEATHER_R2
```

```
## Warning: model fit failed for Resample13: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample14: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample15: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample16: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample17: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample18: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample19: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample20: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample21: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample22: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample23: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

## Warning: model fit failed for Resample24: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1

## Warning: model fit failed for Resample25: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.defaul
##   Zero variances for at least one class in variables: TRAF_CON_R1, TRAF_CON_R2

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

## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

The result is a set of predictions based on the Naive Bayes model for the selected data. The predictions can be used for evaluating the model's performance or for making decisions based on the classification results. Additionally, inspecting the model's details can provide insights into how the classifier was trained and configured.So,it provide the levels as no and yes.

```
library(klaR)
predict(nb2, newdata = accidents24[,c("INJURY", "WEATHER_R", "TRAF_CON_R")])
```

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

```
nb2
```

```
## Naive Bayes
##
## 24 samples
##  2 predictor
##  2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 24, 24, 24, 24, 24, 24, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE            NaN          NaN
##    TRUE      0.5523752  -0.03414305
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = TRUE and adjust
##  = 1.
```

The result is a set of raw class probabilities for each observation in the selected data. These probabilities represent the likelihood of each class (e.g., "Yes" and "No" for "INJURY") based on the model and the predictor variables

```
predict(nb2, newdata = accidents24[,c("INJURY", "WEATHER_R", "TRAF_CON_R")],
        type = "raw")
```

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

```
nb2
```

```
## Naive Bayes
##
## 24 samples
##  2 predictor
##  2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 24, 24, 24, 24, 24, 24, ...
```

```
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE            NaN          NaN
##    TRUE      0.5523752  -0.03414305
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = TRUE and adjust
##  = 1.
```

```r
# Define a range of threshold values
thresholds <- seq(0.1, 0.9, by = 0.1)

# Create a data frame to store results
results <- data.frame(Threshold = thresholds, Agreement = NA)

# Loop through threshold values and calculate agreement
for (i in 1:length(thresholds)) {
  threshold <- thresholds[i]
  accidents24$pred.prob <- ifelse(accidents24$prob.inj > threshold, "yes", "no")
  agreement <- sum(accidents24$pred.prob == accidents24$INJURY)
  results[i, "Agreement"] <- agreement
}

# Find the threshold with the highest agreement
best_threshold <- results$Threshold[which.max(results$Agreement)]
```

Load the required library for Naive Bayes classification

The code essentially evaluates how well the Naive Bayes classifier performs compared to the "exact" Bayes classification. It checks if the classifications and the ranking of observations are similar, which is a common approach for evaluating the performance of classification models.

The discrepancy in the ranking of the Naive Bayes and exact Bayes probabilities is likely due to the simplifying assumptions made by the Naive Bayes classifier, which may not fully capture the complexities of the data. Here's why the rankings may differ:

1. Independence Assumption: Naive Bayes assumes that the predictor variables (in this case, "WEATHER_R" and "TRAF_CON_R") are conditionally independent given the class variable ("INJURY"). This assumption might not hold in reality. Variables can be correlated, and Naive Bayes may underestimate the joint influence of correlated predictors.

2. Conditional Probability Estimation: Naive Bayes estimates conditional probabilities based on observed frequencies in the training data. If the training data is limited or not representative of the population, the estimated probabilities may not accurately reflect the true conditional probabilities. In contrast, the exact Bayes approach uses the actual frequencies in the data.

3. Simplicity of the Model: Naive Bayes is a simplified model designed for efficiency and ease of computation. It makes a strong independence assumption to make calculations tractable. In contrast, the exact Bayes approach does not make the same independence assumptions and calculates probabilities based on the full joint distribution.

4. Cutoff Threshold: In your Naive Bayes implementation, you used a cutoff threshold of 0.5 to classify accidents as "yes" or "no." This is a common threshold, but it might not be optimal for all datasets. Choosing a different threshold could potentially result in different rankings.

5. Data Characteristics: The exact Bayes approach considers all the observed data, while the Naive Bayes model estimates probabilities based on the data available. If the dataset is small or unrepresentative, the Naive Bayes model might not capture the true underlying relationships.

```
# Create a data frame with only the relevant columns
data_for_naive_bayes <- accidents24[, c("WEATHER_R", "TRAF_CON_R", "INJURY")]

# Convert the columns to factors
data_for_naive_bayes$WEATHER_R <- as.factor(data_for_naive_bayes$WEATHER_R)
data_for_naive_bayes$TRAF_CON_R <- as.factor(data_for_naive_bayes$TRAF_CON_R)
data_for_naive_bayes$INJURY <- as.factor(data_for_naive_bayes$INJURY)

# Train a Naive Bayes classifier
nb_classifier <- naiveBayes(INJURY ~ WEATHER_R + TRAF_CON_R, data = data_for_naive_bayes)

# Use the classifier to predict probabilities and classifications
predicted_probs <- predict(nb_classifier, data_for_naive_bayes, type = "raw")
predicted_classifications <- predict(nb_classifier, data_for_naive_bayes)

# Compare exact Bayes classification and Naive Bayes classification
exact_classification <- accidents24$INJURY
exact_probabilities <- prob.inj

cat("Exact Bayes Probabilities: ", exact_probabilities, "\n")
```

```
## Exact Bayes Probabilities:  0.6666667 0.1818182 0 0 0.6666667 0.1818182 0.1818182 0.6666667 0.181818
```

```
cat("Naive Bayes Probabilities: ", predicted_probs, "\n")
```

```
## Naive Bayes Probabilities:  0.4285714 0.75 0.9977551 0.9910803 0.4285714 0.75 0.75 0.4285714 0.75 0.
```

```
cat("Exact Bayes Classification: ", exact_classification, "\n")
```

```
## Exact Bayes Classification:  2 1 1 1 1 2 1 2 1 1 1 1 2 1 2 2 1 1 1 1 2 1 2 2
```

```
cat("Naive Bayes Classification: ", predicted_classifications, "\n")
```

```
## Naive Bayes Classification:  2 1 1 1 2 1 1 2 1 1 1 2 2 2 2 2 1 1 1 1 2 2 1 1
```

```
# Check if the classifications are equivalent
classification_match <- all(exact_classification == predicted_classifications)
cat("Are the classifications equivalent is ", classification_match, "\n")
```

```
## Are the classifications equivalent is  FALSE
```

```r
# Check if the ranking of observations is equivalent
ranking_match <- all(order(-exact_probabilities) == order(-predicted_probs))
cat("Is the ranking of observations equivalent is ", ranking_match, "\n")
```

```
## Is the ranking of observations equivalent is  FALSE
```

Let us now return to the entire dataset. Partition the data into training (60%) and validation (40%). 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. What is the overall error of the validation set?

The code performs model training, validation, and evaluation of a Naive Bayes classifier using a common approach, including the calculation of a confusion matrix and misclassification rate. The results of this evaluation can provide insights into the classifier's performance on new, unseen data.

```r
# Load the required library for Naive Bayes classification
library(e1071)

# Assuming "accidents" is your complete dataset
# Split the data into training (60%) and validation (40%)
set.seed(123)  # Set a seed for reproducibility
sample_size <- floor(0.6 * nrow(accidents))
training_set <- sample(seq_len(nrow(accidents)), size = sample_size)

# Create training and validation datasets
train_data <- accidents[training_set, ]
validation_data <- accidents[-training_set, ]

# Train a Naive Bayes classifier on the training data
nb_classifier <- naiveBayes(INJURY ~ WEATHER_R + TRAF_CON_R, data = train_data)

# Use the classifier to predict INJURY for the validation set
predicted_classifications <- predict(nb_classifier, validation_data)
```

```r
library(caret)
# Create a confusion matrix for validation set
confusion_matrix <- confusionMatrix(predicted_classifications, validation_data$INJURY)
confusion_matrix
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction   no  yes
##        no  1326 1072
##        yes 6984 7492
##
##                Accuracy : 0.5226
##                  95% CI : (0.515, 0.5301)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : 4.725e-05
##
##                   Kappa : 0.0348
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.15957
##             Specificity : 0.87482
##          Pos Pred Value : 0.55296
##          Neg Pred Value : 0.51755
##              Prevalence : 0.49247
##          Detection Rate : 0.07858
##    Detection Prevalence : 0.14211
##       Balanced Accuracy : 0.51720
##
##        'Positive' Class : no
##
```

```
# Print the confusion matrix and overall error
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```
print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  1326 1072
##        yes 6984 7492
##
##                Accuracy : 0.5226
##                  95% CI : (0.515, 0.5301)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : 4.725e-05
##
##                   Kappa : 0.0348
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.15957
##             Specificity : 0.87482
##          Pos Pred Value : 0.55296
##          Neg Pred Value : 0.51755
##              Prevalence : 0.49247
##          Detection Rate : 0.07858
##    Detection Prevalence : 0.14211
##       Balanced Accuracy : 0.51720
##
##        'Positive' Class : no
##
```

```
# Create a confusion matrix for training set
#confusion_matrix2 <- confusionMatrix(predicted_classifications, train_data$INJURY)
#print(confusion_matrix2)
```

```r
# Calculate the overall error (misclassification rate)
total_samples <- sum(confusion_matrix$INJURY)
correctly_classified <- sum(diag(confusion_matrix$INJURY))
misclassification_rate <- 1 - correctly_classified / total_samples

#Use the classifier to predict INJURY for the train set
predicted_classifications <- predict(nb_classifier, train_data)
```

```r
# Calculate the overall error (misclassification rate)
total_samples <- sum(confusion_matrix$INJURY)
correctly_classified <- sum(diag(confusion_matrix$INJURY))
misclassification_rate <- 1 - correctly_classified / total_samples
cat("Confusion Matrix :\n")
```

```
## Confusion Matrix :
```

```r
# Print the confusion matrix and overall error
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```r
print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  1326 1072
##        yes 6984 7492
##
##                Accuracy : 0.5226
##                  95% CI : (0.515, 0.5301)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : 4.725e-05
##
##                   Kappa : 0.0348
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.15957
##             Specificity : 0.87482
##          Pos Pred Value : 0.55296
##          Neg Pred Value : 0.51755
##              Prevalence : 0.49247
##          Detection Rate : 0.07858
##    Detection Prevalence : 0.14211
##       Balanced Accuracy : 0.51720
##
##        'Positive' Class : no
##
```

```r
# Create a confusion matrix for training set
confusion_matrix2 <- confusionMatrix(predicted_classifications, train_data$INJURY)
print(confusion_matrix2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    no   yes
##        no   1971  1654
##        yes 10440 11244
##
##                Accuracy : 0.5221
##                  95% CI : (0.516, 0.5283)
##     No Information Rate : 0.5096
##     P-Value [Acc > NIR] : 3.439e-05
##
##                   Kappa : 0.031
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.15881
##             Specificity : 0.87176
##          Pos Pred Value : 0.54372
##          Neg Pred Value : 0.51854
##              Prevalence : 0.49038
##          Detection Rate : 0.07788
##    Detection Prevalence : 0.14323
##       Balanced Accuracy : 0.51529
##
##        'Positive' Class : no
##
```

```r
cat("Overall Error (Misclassification Rate): ", misclassification_rate, "\n")
```

```
## Overall Error (Misclassification Rate):  NaN
```

An overall error (misclassification rate) of 0.477 indicates that the Naive Bayes classifier, when applied to the validation set, is making incorrect predictions for approximately 47.7% of the cases in the validation set.

Here are some inferences we can make from an overall error of 0.477:

Accuracy: The accuracy of the Naive Bayes classifier on the validation set is approximately 52.3% (100% - 47.7%). This means that it correctly classifies a little over half of the cases.

Misclassification: The misclassification rate is relatively high, which suggests that the classifier is not very effective in accurately predicting the "INJURY" variable based on the features "WEATHER_R" and "TRAF_CON_R."

Model Improvement: An overall error of 0.477 may indicate that there is room for improvement in the model. We might want to consider refining the features used for classification, using different classification algorithms, or fine-tuning the Naive Bayes model parameters to improve its performance.

Consider Additional Metrics: While the misclassification rate provides a broad view of model performance, it's a good practice to consider additional evaluation metrics like precision, recall, F1 score, or the ROC curve to gain a more comprehensive understanding of the model's strengths and weaknesses.

To conclude, an overall error of 0.477 suggests that the current Naive Bayes classifier could benefit from improvements to better predict the "INJURY" variable.