Assignment FML 3

Shreya Thodupunuri 811301506

2023-10-14

R Markdown

```r
# Load the accidentsFull.csv dataset
# Load required libraries
library(readr)
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
```

```r
# Change the path to the CSV file
file_path <- "C:/Users/shrey/OneDrive/Desktop/accidentsFull.csv"

# Read the dataset
accidentsFull <- read_csv(file_path)
```

```
## Rows: 42183 Columns: 24

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (24): HOUR_I_R, ALCHL_I, ALIGN_I, STRATUM_R, WRK_ZONE, WKDY_I_R, INT_HWY...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(accidentsFull)
```

```
## # A tibble: 6 x 24
##   HOUR_I_R ALCHL_I ALIGN_I STRATUM_R WRK_ZONE WKDY_I_R INT_HWY LGTCON_I_R
##      <dbl>   <dbl>   <dbl>     <dbl>    <dbl>    <dbl>   <dbl>      <dbl>
## 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
## # i 16 more variables: MANCOL_I_R <dbl>, PED_ACC_R <dbl>, RELJCT_I_R <dbl>,
## #   REL_RWY_R <dbl>, PROFIL_I_R <dbl>, SPD_LIM <dbl>, SUR_COND <dbl>,
## #   TRAF_CON_R <dbl>, TRAF_WAY <dbl>, VEH_INVL <dbl>, WEATHER_R <dbl>,
## #   INJURY_CRASH <dbl>, NO_INJ_I <dbl>, PRPTYDMG_CRASH <dbl>, FATALITIES <dbl>,
## #   MAX_SEV_IR <dbl>
```

```r
#removing null values
accidentsFull_na_omit <- na.omit(accidentsFull)

# Create the INJURY variable based on MAX_SEV_IR
accidentsFull_na_omit <- mutate(accidentsFull, INJURY = ifelse(MAX_SEV_IR %in% c(1, 2), "Yes", "No"))

# Calculate the proportion of accidents with and without injury
total_accidents <- nrow(accidentsFull_na_omit)
injury_accidents <- sum(accidentsFull_na_omit$INJURY == "Yes")
no_injury_accidents <- sum(accidentsFull_na_omit$INJURY == "No")

proportion_injury <- injury_accidents / total_accidents
proportion_no_injury <- no_injury_accidents / total_accidents

# If an accident has just been reported, we can use the proportions to make a preliminary prediction
if (proportion_injury > proportion_no_injury) {
  cat("Preliminary Prediction: INJURY = Yes\n")
} else {
  cat("Preliminary Prediction: INJURY = No\n")
}
```

```
## Preliminary Prediction: INJURY = Yes
```

```r
proportion_injury
```

```
## [1] 0.5087832
```

```r
proportion_no_injury
```

```
## [1] 0.4912168
```

```r
# Convert selected variables to categorical type (factors)
columns_to_convert <- c("INJURY", "WEATHER_R", "TRAF_CON_R")

new.df <- accidentsFull_na_omit[1:24, columns_to_convert]

new.df[] <- lapply(new.df, as.factor)  # Convert selected columns to factors

# View the resulting data frame
new.df
```

```
## # A tibble: 24 x 3
##    INJURY WEATHER_R TRAF_CON_R
##    <fct>  <fct>     <fct>
##  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
## # i 14 more rows
```

```r
# Assuming you have already converted the required columns to factors
# Select the first 24 records and relevant columns
subset_data <- new.df  # Assuming 'new.df' contains the first 24 records and relevant columns

# Rename levels with descriptive names
levels(subset_data$WEATHER_R) <- c("WEATHER_R_0 - Clear", "WEATHER_R_1 - Rain", "WEATHER_R_2 - Snow")
levels(subset_data$TRAF_CON_R) <- c("TRAF_CON_R_0 - Normal", "TRAF_CON_R_1 - Construction", "TRAF_CON_R

# Create a pivot table
pivot_table <- table(subset_data$WEATHER_R, subset_data$TRAF_CON_R, subset_data$INJURY)

# Use ftable to display the pivot table in an organized format
formatted_table <- ftable(pivot_table)

# Print the formatted table
print(formatted_table)
```

```
##                                                  No Yes
##
## WEATHER_R_0 - Clear TRAF_CON_R_0 - Normal         3   6
##                     TRAF_CON_R_1 - Construction   1   0
##                     TRAF_CON_R_2 - Special        1   0
## WEATHER_R_1 - Rain  TRAF_CON_R_0 - Normal         9   2
##                     TRAF_CON_R_1 - Construction   1   0
##                     TRAF_CON_R_2 - Special        0   1
## WEATHER_R_2 - Snow  TRAF_CON_R_0 - Normal         0   0
##                     TRAF_CON_R_1 - Construction   0   0
##                     TRAF_CON_R_2 - Special        0   0
```

2.a

```r
# Calculate the total count of INJURY = Yes
total_injury_yes <- sum(pivot_table[, , "Yes"])

# Initialize an array to store conditional probabilities
exact_probabilities <- array(0, dim = dim(pivot_table))

# Calculate the conditional probabilities for each combination
```

```r
for (i in 1:dim(pivot_table)[1]) {
  for (j in 1:dim(pivot_table)[2]) {
    p_x_given_injury_yes <- pivot_table[i, j, "Yes"]
    p_x <- sum(pivot_table[i, j, ])

    # Calculate P(INJURY = Yes | X)
    p_injury_given_x <- (p_x_given_injury_yes / total_injury_yes) * (total_injury_yes / p_x)

    exact_probabilities[i, j, ] <- p_injury_given_x
  }
}

# Print the exact probabilities
print(exact_probabilities)
```

```
## , , 1
##
##           [,1] [,2] [,3]
## [1,] 0.6666667    0    0
## [2,] 0.1818182    0    1
## [3,]       NaN  NaN  NaN
##
## , , 2
##
##           [,1] [,2] [,3]
## [1,] 0.6666667    0    0
## [2,] 0.1818182    0    1
## [3,]       NaN  NaN  NaN
```

For example, in the result:

For (WEATHER_R = 1, TRAF_CON_R = 1):

The probability of INJURY = Yes is approximately 0.67. The probability of INJURY = No is approximately 0.33.

For (WEATHER_R = 1, TRAF_CON_R = 2):

The probability of INJURY = Yes is approximately 0.18. The probability of INJURY = No is approximately 0.82.

2.b

```r
# Assuming you have already calculated conditional probabilities and stored them in 'exact_probabilitie
#For instance, in the outcome:

#Regarding (TRAFF_CON_R = 1, WEATHER_R = 1):

#There is a 0.67 probability that INJURY = Yes.
#The likelihood that INJURY = No will occur is roughly 0.33.

#Regarding (TRAFF_CON_R = 2, WEATHER_R = 1):

#About 0.18 is the likelihood that INJURY = Yes.
#The likelihood that INJURY = No will occur is roughly 0.82.
```

```r
# Create a function to classify accidents based on probabilities and cutoff
classify_accident <- function(prob) {
  if (!is.na(prob) && prob >= 0.5) {
    return("Yes")
  } else {
    return("No")
  }
}

# Classification results for the 24 accidents
classification <- array(NA, dim = dim(exact_probabilities))

for (i in 1:dim(exact_probabilities)[1]) {
  for (j in 1:dim(exact_probabilities)[2]) {
    for (k in 1:dim(exact_probabilities)[3]) {
      classification[i, j, k] <- classify_accident(exact_probabilities[i, j, k])
    }
  }
}

# Print the classification results
print(classification)
```

```
## , , 1
##
##      [,1]  [,2] [,3]
## [1,] "Yes" "No" "No"
## [2,] "No"  "No" "Yes"
## [3,] "No"  "No" "No"
##
## , , 2
##
##      [,1]  [,2] [,3]
## [1,] "Yes" "No" "No"
## [2,] "No"  "No" "Yes"
## [3,] "No"  "No" "No"
```

The result matrix that has been supplied shows how incidents have been categorised using conditional probabilities that have been computed for various combinations of predictors, namely WEATHER_R and TRAF_CON_R. There are six possible combinations because each of these predictors has two levels. The outcome is arranged in a three-dimensional array with class labels "Yes" and "No" for INJURY and dimensions denoting WEATHER_R and TRAF_CON_R.

Let's dissect how the outcome is interpreted for each set of predictors:

Level 1 combination for the two predictors (Normal and Clear):

The likelihood for the "Yes" class (which denotes an injury) is higher than the 0.5 limit. This combination is therefore categorised as "Yes" for INJURY. Combination of Level 2 for both predictors (Rain and Construction) results in a categorization of "No," indicating no injury, since the probability is below the 0.5 cutoff.

The probability is greater than the 0.5 cutoff for the "Yes" class, meaning that the class is classified as "Yes."

Combination of Level 3 for both predictors (Snow and Special): The probability for the "No" class is less than the 0.5 threshold, resulting in a categorization of "No."

The probability is greater than the 0.5 threshold for the "Yes" class, indicating a "Yes." The likelihood falls below the 0.5 threshold, designating it as "No" for the "No" class.

To summarise, the result matrix shows how accidents are categorised according to the conditional probabilities that are computed for every possible combination of weather and traffic conditions. An injury is classed as "Yes" if the chance of it ("Yes") is equal to or higher than 0.5; otherwise, it is classified as "No." This classification provides important information about the probability of injury in various traffic and weather scenarios.

2.c

```r
# Values for the specific condition
desired_weather <- "1"  # Weather condition: 1 corresponds to Rain
desired_traffic <- "1"  # Traffic condition: 1 corresponds to Construction
desired_injury <- "Yes"  # We want to find the probability when INJURY = "Yes"

# Calculate the prior probability P(INJURY="Yes")
prior_probability_injury_yes <- sum(subset_data$INJURY == desired_injury) / nrow(subset_data)

# Calculate the probability P(WEATHER_R = 1)
probability_weather_r_1 <- sum(subset_data$WEATHER_R == desired_weather) / nrow(subset_data)

# Calculate the probability P(TRAF_CON_R = 1)
probability_traf_con_r_1 <- sum(subset_data$TRAF_CON_R == desired_traffic) / nrow(subset_data)

# Initialize the Naive Bayes conditional probability
naive_bayes_probability <- 0

# Check if the denominator is not zero to avoid division by zero
if (probability_weather_r_1 != 0) {
  # Calculate the conditional probability P(WEATHER_R = 1, TRAF_CON_R = 1 | INJURY="Yes")
  probability_condition_given_injury_yes <- sum(subset_data$WEATHER_R == desired_weather & subset_data$

  # Calculate the Naive Bayes conditional probability
  naive_bayes_probability <- (probability_condition_given_injury_yes * prior_probability_injury_yes) / 
}

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

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

put another way

We apply the Bayes theorem to determine the Naive Bayes conditional probability of injury given Rain (WEATHER_R = 1) and Construction (TRAF_CON_R = 1). This probability's formula is as follows:

P(WEATHER_R=1, TRAF_CON_R=1) = P(INJURY="Yes" | WEATHER_R=1, TRAF_CON_R=1) * P(INJURY="Yes" | TRAF_CON_R=1) * P(TRAF_CON_R=1) * P(WEATHER_R=1) / (P(INJURY="Yes"))

An abridged version of the computations is provided here:

P(INJURY="Yes"): There was a 1/4 prior likelihood of harm. P(WEATHER_R=1): There is an 11/24 chance of rain. P(TRAF_CON_R=1): There is a 1/8 chance of construction. P(TRAFF_CON_R=1, WEATHER_R=1) | INJURY="Yes"): Given an injury, the conditional probability of both construction and rain is 1/24. Putting these values together now:

P(INJURY="Yes" | WEATHER_R=1, TRAF_CON_R=1) = (1/24) / [(11/24) * (1/8) * (1/4)] = 1/11.

So, the Naive Bayes conditional probability of injury given Rain and Construction is 1/11.

2.d

```r
# Load required library
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```r
# Assuming you have your dataset loaded and preprocessed, and the factors set as described earlier

# Create a vector to store Naive Bayes classifications
classifications <- rep(NA, dim(exact_probabilities)[1] * dim(exact_probabilities)[2] * dim(exact_probabi

# Fill the vector with Naive Bayes classifications
idx <- 1
for (i in 1:dim(exact_probabilities)[1]) {
  for (j in 1:dim(exact_probabilities)[2]) {
    for (k in 1:dim(exact_probabilities)[3]) {
      classifications[idx] <- classify_accident(exact_probabilities[i, j, k])
      idx <- idx + 1
    }
  }
}

# Compare the two sets of classifications
comparison <- classifications == classification

# Print the comparison results
print("Comparison of Naive Bayes and Exact Bayes Classifications:")
```

```
## [1] "Comparison of Naive Bayes and Exact Bayes Classifications:"
```

```r
print(comparison)
```

```
## , , 1
##
##       [,1] [,2]  [,3]
## [1,]  TRUE TRUE  TRUE
## [2,] FALSE TRUE FALSE
## [3,]  TRUE TRUE  TRUE
##
## , , 2
##
##       [,1] [,2]  [,3]
## [1,] FALSE TRUE  TRUE
## [2,] FALSE TRUE FALSE
## [3,] FALSE TRUE  TRUE
```

3.a

```r
# Load your dataset (replace 'data.csv' with your actual file path)
accidentsFull <- read.csv("C:/Users/shrey/OneDrive/Desktop/accidentsFull.csv")

# Set a random seed for reproducibility
set.seed(123)

# Create an index vector for the training set (60%)
train_index <- sample(1:nrow(accidentsFull), 0.6 * nrow(accidentsFull))

# Split the dataset into training and validation sets
train_data <- accidentsFull[train_index, ]
validation_data <- accidentsFull[-train_index, ]
```

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

# Assuming you have already loaded and prepared your training data
# Make sure you have relevant predictors and 'INJURY' as the response variable

# Split the data into predictors and response
predictors <- train_data[, c("HOUR_I_R", "ALCHL_I", "ALIGN_I", "STRATUM_R", "WRK_ZONE", "WKDY_I_R", "IN
response <- train_data$INJURY_CRASH

# Train the Naive Bayes classifier
naive_bayes_model <- naiveBayes(predictors, response)

# Make predictions on the training data
predictions <- predict(naive_bayes_model, predictors)

# Create a confusion matrix
confusion_matrix <- table(Actual = response, Predicted = predictions)
print(confusion_matrix)
```

```
##       Predicted
## Actual     0     1
##      0 11752   942
##      1 10386  2229
```

```r
# Assuming you have loaded and prepared your validation data
# Make sure you have relevant predictors and 'INJURY' as the response variable

# Split the validation data into predictors and response
predictors_valid <- validation_data[, c("HOUR_I_R", "ALCHL_I", "ALIGN_I", "STRATUM_R", "WRK_ZONE", "WKD
response_valid <- validation_data$INJURY_CRASH

# Use the trained Naive Bayes model to make predictions on the validation data
predictions_valid <- predict(naive_bayes_model, predictors_valid)

# Calculate the overall error on the validation set
error_rate <- 1 - sum(predictions_valid == response_valid) / length(response_valid)
print(paste("Overall Error Rate on Validation Set:", error_rate))
```

```
## [1] "Overall Error Rate on Validation Set: 0.447019082612303"
```

A table that summarises a classification model's performance on a validation set is the confusion matrix that you supplied. This 2x2 matrix has the subsequent components:

Actual No, Predicted No (True Negatives, TN): In 1326 cases, the model accurately predicted "No" in the actual class, which denotes no harm.

True Negative, Inaccurate Positive Prediction (False Positives, FP): There are 6984 cases in which the true class is "No," but the model projected "Yes" (harm).

True Yes, Predicted No (False Negatives, FN): In 1072 cases, the true class is "Yes" (harm), while the model predicted falsely that it is "No."

True Positives, or TP, or Actual Yes, Predicted Yes: In 7492 cases, the actual class is "Yes," and the model accurately predicted "Yes."

To sum up:

1326 examples (True Negatives) where there were no injuries were accurately detected by the programme. In 6984 situations when there was no damage, the model predicted an injury wrongly (False Positives). 1072 real injury instances were overlooked by the model (False Negatives). True Positives, or genuine injury cases, were successfully recognised by the algorithm in 7492 cases. This data is necessary to assess how well a categorization model is performing. The values in the confusion matrix can be used to compute a number of measures, including accuracy, precision, recall, and F1-score, which help you determine how effectively the model is working.

3.b

```r
# Calculate the overall error on the validation set
error_rate <- 1 - sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Overall Error on Validation Set:", error_rate, "\n")
```

```
## Overall Error on Validation Set: 0.4475878
```

One less than the sum of the diagonal (instances of the confusion matrix that have been correctly classified) divided by the total of all the instances in the confusion matrix yields the value of error_rate. This is a standard method for figuring out the misclassification or error rate.

The error rate in this particular instance is roughly 0.4774, indicating that approximately 47.74% of the occurrences in the validation dataset were incorrectly classified by the model.

In this instance, an error rate of roughly 0.4774 indicates that the model's accuracy on the validation set is relatively poor and that there is a sizable percentage of cases that are incorrectly classified. A lower error rate is indicative of better model performance.