group12

Abhilash Hemaraj and Nikita Pai

3/28/2020

Problem 1

Partition the data into training (60%) and validation (40%) sets. a. Consider the following customer: Age=40, Experience=10, Income=84, Family=2, CCAvg=2, Education2=1, Education3=0, Mortgage=O, Securities Account=O, CD Account=O, Online=1 and Credit.card = 1. Perform a k-NN classification with all predictors except ID and ZIP code using k = 1. Remember to transform categorical predictors with more than two categories into dummy variables first. Specify the success class as 1 (loan acceptance), and use the default cutoff value of 0.5. How would this customer be classified? b. What is a choice of k that balances between overfitting and ignoring the predictor information? c. Show the classification matrix for the validation data that results from using the best k. d. Consider the following customer: Age=40, Experience=10, Income=84, Family=2, CCAvg=2, Education 1=0, Education 2=1, Education 3=0, Mortgage=0, Securities Account=0, CD Account=0, Online=1 and Credit Card=1. Classify the customer using the best k. e. Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply the k-NN method with the k chosen above. Compare the classification matrix of the test set with that of the training and validation sets. Comment on the differences and their reason.

```r
library(magrittr)
library(readxl)
UniversalBank <- read_excel("UniversalBank.xlsx", sheet = "Data")
#View(UniversalBank)

df1 <- UniversalBank
df1 <- df1[, -c(1,5)]
```

```r
# Partition into test and train set
size_ <- floor(0.60 * nrow(df1))
set.seed(50)
train_ind <- sample(seq_len(nrow(df1)), size = size_)
train_bank <- df1[train_ind,] # Training set
test_bank <-  df1[-train_ind,] # Validation set

colnames(df1) <- c("Age", "Experience", "Income", "Family", "CCAvg", "Education", "Mortgage", "Personal
colnames(df1)
```

```
##  [1] "Age"                "Experience"         "Income"
##  [4] "Family"             "CCAvg"              "Education"
##  [7] "Mortgage"           "Personal.Loan"      "Securities.Account"
## [10] "CD.Account"         "Online"             "Credit.Card"
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

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

```r
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
library(FNN)
```

```
## Warning: package 'FNN' was built under R version 3.6.3
```

```r
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```r
# creating dummy variables
df1$Education <- as.factor(df1$Education)

dummyvar <- dummyVars("~ Education", data = df1, sep = NULL)

Edu_var <- data.frame(predict(dummyvar, newdata = df1))

head(Edu_var)
```

```
##   Education1 Education2 Education3
## 1          1          0          0
## 2          1          0          0
## 3          1          0          0
## 4          0          1          0
## 5          0          1          0
## 6          0          1          0
```

```r
# adding the created dummy variables to the data frame
df1 <- as.data.frame(c(df1, Edu_var))

df1 <- df1[, -6]

# normalising the data
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}

df1_norm <- as.data.frame(lapply(df1[,-7], normalize))

summary(df1_norm)
```

```
##       Age           Experience         Income           Family      
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.2727   1st Qu.:0.2826   1st Qu.:0.1435   1st Qu.:0.0000  
##  Median :0.5000   Median :0.5000   Median :0.2593   Median :0.3333  
##  Mean   :0.5077   Mean   :0.5023   Mean   :0.3045   Mean   :0.4655  
##  3rd Qu.:0.7273   3rd Qu.:0.7174   3rd Qu.:0.4167   3rd Qu.:0.6667  
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
##      CCAvg           Mortgage       Securities.Account   CD.Account    
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000     Min.   :0.0000  
##  1st Qu.:0.0700   1st Qu.:0.00000   1st Qu.:0.0000     1st Qu.:0.0000  
##  Median :0.1500   Median :0.00000   Median :0.0000     Median :0.0000  
##  Mean   :0.1938   Mean   :0.08897   Mean   :0.1044     Mean   :0.0604  
##  3rd Qu.:0.2500   3rd Qu.:0.15906   3rd Qu.:0.0000     3rd Qu.:0.0000  
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.0000     Max.   :1.0000  
##      Online        Credit.Card      Education1       Education2    
##  Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median :1.0000   Median :0.000   Median :0.0000   Median :0.0000  
##  Mean   :0.5968   Mean   :0.294   Mean   :0.4192   Mean   :0.2806  
##  3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :1.000   Max.   :1.0000   Max.   :1.0000  
##    Education3    
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.3002  
##  3rd Qu.:1.0000  
##  Max.   :1.0000
```

```r
# test tain split
size_ <- floor(0.60 * nrow(df1_norm))
set.seed(50)
train_ind  <- sample(seq_len(nrow(df1_norm)), size = size_)
train_bank <- df1_norm[train_ind,] # Training set
test_bank  <-  df1_norm[-train_ind,] # Validation set

train_label <- df1[train_ind, 7]
test_label  <- df1[-train_ind, 7]
```

```r
# Considering the customer:
customer <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Mortgage = 0, Secu


NN <- knn(train = train_bank, test = customer, cl = train_label, k = 1)

row.names(train_bank)[attr(NN, "nn.index")]
```

```
## [1] "2957"
```

```r
# Since customer number '915' which is the Nearest Neighbour has accepted the personal loan then the ne
# will also accept the personal loan.

accuracy <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))

# compute knn for different k on validation.

for(i in 1:20) {
  knn.pred <- knn(train_bank, test_bank,
                  cl = train_label, k = i)
  accuracy[i, 2] <- confusionMatrix(knn.pred, as.factor(test_label))$overall[1]
}

#knn.pred <- knn(train_bank, test_bank, cl = train_label, k = 1)

# k = 3 gives the maximum accuracy value


# c
knn.validation <- knn(train_bank, test_bank, cl = train_label, k = 3)
confusionMatrix(knn.validation, as.factor(test_label))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1810   73
##          1   11  106
##
##                Accuracy : 0.958
##                  95% CI : (0.9483, 0.9664)
##     No Information Rate : 0.9105
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6946
##
##  Mcnemar's Test P-Value : 2.821e-11
##
##             Sensitivity : 0.9940
##             Specificity : 0.5922
##          Pos Pred Value : 0.9612
##          Neg Pred Value : 0.9060
##              Prevalence : 0.9105
```

```
##          Detection Rate : 0.9050
##    Detection Prevalence : 0.9415
##       Balanced Accuracy : 0.7931
##
##        'Positive' Class : 0
##
```

```r
length(train_bank)
```

```
## [1] 13
```

```r
length(train_label)
```

```
## [1] 3000
```

```r
# d
NN1 <- knn(train = train_bank, test = customer, cl = train_label, k = 3)
row.names(train_bank)[attr(NN1, "nn.index")]
```

```
## [1] "2957" "663"  "2774"
```

```r
# The new customer is closest to "915", "663", "4293" and hence will acccept the personal loan
```

```r
# e. Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply
# method with the k chosen above. Compare the classification matrix of the test set with that of the tr
spec = c(train = 0.5, test = 0.2, validate = 0.3)

g = sample(cut(
  seq(nrow(df1)),
  nrow(df1)*cumsum(c(0,spec)),
  labels = names(spec)
))

res = split(df1, g)

sapply(res, nrow)/nrow(df1)
```

```
##    train     test validate
##      0.5      0.2      0.3
```

```r
train_bank <- res$train
valid_bank <- res$validate
test_bank  <- res$test

knn.validation <- knn(train_bank[,-7], valid_bank[,-7], cl = train_bank[,7], k = 3)
confusionMatrix(knn.validation, as.factor(valid_bank[,7]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    0    1
##          0 1306  109
##          1   41   44
##
##                Accuracy : 0.9
##                  95% CI : (0.8837, 0.9147)
##     No Information Rate : 0.898
##     P-Value [Acc > NIR] : 0.4198
##
##                   Kappa : 0.3202
##
##  Mcnemar's Test P-Value : 4.487e-08
##
##             Sensitivity : 0.9696
##             Specificity : 0.2876
##          Pos Pred Value : 0.9230
##          Neg Pred Value : 0.5176
##              Prevalence : 0.8980
##          Detection Rate : 0.8707
##    Detection Prevalence : 0.9433
##       Balanced Accuracy : 0.6286
##
##        'Positive' Class : 0
##
```

```r
knn.test <- knn(train_bank[,-7], test_bank[, -7], cl = train_bank[, 7], k = 3)
confusionMatrix(knn.test, as.factor(test_bank[,7]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 856  70
##          1  38  36
##
##                Accuracy : 0.892
##                  95% CI : (0.8711, 0.9106)
##     No Information Rate : 0.894
##     P-Value [Acc > NIR] : 0.606179
##
##                   Kappa : 0.3427
##
##  Mcnemar's Test P-Value : 0.002855
##
##             Sensitivity : 0.9575
##             Specificity : 0.3396
##          Pos Pred Value : 0.9244
##          Neg Pred Value : 0.4865
##              Prevalence : 0.8940
##          Detection Rate : 0.8560
##    Detection Prevalence : 0.9260
##       Balanced Accuracy : 0.6486
##
##        'Positive' Class : 0
```

```
##
```

Problem 2

The file BostonHousing.xlsx contains information on over 500 census tracts in Boston, where for each tract 14 variables are recorded. The last column (CAT.MEDV) was derived from MEDV, such that it obtains the value 1 if MEDV > 30 and 0 otherwise. Consider the goal of predicting the median value (MEDV) of a tract, given the information in the first 13 columns. Partition the data into training (60%) and validation (40%) sets. a. Perform a k-NN prediction with all 13 predictors (ignore the CAT.MEDV column), trying values of k from 1 to 5. Make sure to normalize the data (click "normalize input data"). What is the best k chosen? What does it mean? b. Predict the MEDV for a tract with the following information, using the best k c. Why is the error of the training data zero? d. Why is the validation data error overly optimistic compared to the error rate when applying this k-NN predictor to new data? e. If the purpose is to predict MEDV for several thousands of new tracts, what would be the disadvantage of using k-NN prediction? List the operations that the algorithm goes through in order to produce each prediction.

```r
library(readxl)
BostonHousing <- read_excel("BostonHousing.xlsx", sheet = "Data")
View(BostonHousing)
df2 <- BostonHousing

# test - validation split
spec = c(train = 0.6, validate = 0.4)

g = sample(cut(
  seq(nrow(df2)),
  nrow(df2)*cumsum(c(0,spec)),
  labels = names(spec)
))

res = split(df2, g)

sapply(res, nrow)/nrow(df2)
```

```
##     train  validate
## 0.5988142 0.4011858
```

```r
train <- res$train[, -c(13,14)]
train_label <- res$train[,13]
valid <- res$validate[, -c(13, 14)]
valid_label <- res$validate[,13]

# normalising the data
normalize <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}

train_norm <- as.data.frame(lapply(train, normalize))

valid_norm <- as.data.frame(lapply(valid, normalize))

summary(train_norm)
```

```
##       CRIM                  ZN              INDUS              CHAS
##  Min.   :0.000000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.001059   1st Qu.:0.0000   1st Qu.:0.1679   1st Qu.:0.00000
##  Median :0.003038   Median :0.0000   Median :0.2969   Median :0.00000
##  Mean   :0.041606   Mean   :0.1326   Mean   :0.3817   Mean   :0.06271
##  3rd Qu.:0.032751   3rd Qu.:0.2105   3rd Qu.:0.6466   3rd Qu.:0.00000
##  Max.   :1.000000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##       NOX               RM              AGE              DIS
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.1224   1st Qu.:0.3898   1st Qu.:0.4042   1st Qu.:0.09724
##  Median :0.2801   Median :0.4559   Median :0.7405   Median :0.20312
##  Mean   :0.3290   Mean   :0.4794   Mean   :0.6650   Mean   :0.25408
##  3rd Qu.:0.4668   3rd Qu.:0.5434   3rd Qu.:0.9336   3rd Qu.:0.37614
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##       RAD              TAX            PTRATIO           LSTAT
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1304   1st Qu.:0.1718   1st Qu.:0.5106   1st Qu.:0.1373
##  Median :0.1739   Median :0.2443   Median :0.6809   Median :0.2516
##  Mean   :0.3547   Mean   :0.4041   Mean   :0.6193   Mean   :0.2962
##  3rd Qu.:0.3043   3rd Qu.:0.9141   3rd Qu.:0.8085   3rd Qu.:0.4308
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```r
summary(valid_norm)
```

```
##       CRIM                   ZN               INDUS              CHAS
##  Min.   :0.0000000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0007401   1st Qu.:0.00000   1st Qu.:0.1609   1st Qu.:0.00000
##  Median :0.0036670   Median :0.00000   Median :0.3317   Median :0.00000
##  Mean   :0.0496596   Mean   :0.09525   Mean   :0.3891   Mean   :0.07882
##  3rd Qu.:0.0498943   3rd Qu.:0.00000   3rd Qu.:0.6366   3rd Qu.:0.00000
##  Max.   :1.0000000   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000
##       NOX               RM              AGE              DIS
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.1502   1st Qu.:0.4395   1st Qu.:0.4222   1st Qu.:0.09934
##  Median :0.3148   Median :0.4986   Median :0.7953   Median :0.24031
##  Mean   :0.3710   Mean   :0.5069   Mean   :0.6826   Mean   :0.31011
##  3rd Qu.:0.5823   3rd Qu.:0.5831   3rd Qu.:0.9446   3rd Qu.:0.47508
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##       RAD              TAX            PTRATIO           LSTAT
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1304   1st Qu.:0.1931   1st Qu.:0.5291   1st Qu.:0.1598
##  Median :0.1739   Median :0.3250   Median :0.7558   Median :0.2826
##  Mean   :0.3971   Mean   :0.4481   Mean   :0.6868   Mean   :0.3158
##  3rd Qu.:1.0000   3rd Qu.:0.9140   3rd Qu.:0.8837   3rd Qu.:0.4063
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```r
R.M.S.E <- data.frame(c(1:5), c(1:5), c(1:5))
colnames(R.M.S.E) <- c("k_values", "rmse_train", "rmse_test")
for (i in 1:5){
  pred_train <- knn.reg(train = train_norm, test = train_norm, y = train_label$MEDV, k = i)
  R.M.S.E[i,2] <- RMSE(pred_train$pred, train_label$MEDV)
  pred_valid <- knn.reg(train = train_norm, test = valid_norm, y = train_label$MEDV, k = i)
  R.M.S.E[i,3] <- RMSE(pred_valid$pred, valid_label$MEDV)
```

```r
}

print(R.M.S.E)
```

```
##   k_values rmse_train rmse_test
## 1        1   0.000000  5.458495
## 2        2   2.629629  5.542398
## 3        3   3.336686  6.175729
## 4        4   3.761387  6.095976
## 5        5   3.938637  6.027274
```

```r
pred_train_1 <- knn.reg(train = train_norm, test = train_norm, y = train_label$MEDV, k = 1)
RMSE(train_label$MEDV, pred_train_1$pred)
```

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

```r
# The best value of k is 2 with an RMSE score of 5.911 in the validation set
```

```r
# b
census_tract <- data.frame(CRIM = 0.2, ZN = 0, INDUS = 7, CHAS = 0, NOX = 0.538, RM = 6, AGE = 62, DIS =


knn.reg(train = train_norm, test = census_tract, y = train_label$MEDV, k = 2)
```

```
## Prediction:
## [1] 7.55
```

```r
# the census tract with the given attribute information has a medv value of 14.4
```

```r
# c. Why is the error of the training data zero?
# The error of the training data turns out to be zero when k = 1; It is because when k = 1, the nearest
```

```r
# d. Why is the validation data error overly optimistic compared to the error rate when
# applying this k-NN predictor to new data?
# There is a good chance that the model performs better on the validation data than any other model due
```

```r
# e . If the purpose is to predict MEDV for several thousands of new tracts, what would
# be the disadvantage of using k-NN prediction? List the operations that the algorithm
# goes through in order to produce each prediction.

# In a large training set, it takes a long time to find distances to all the neighbors and then identif
# nearest ones, hence efficiency of the model is under question.
# With the addition of more predictors knn's algorithm would require more records to train upon to avoi
# dimensionality.
```

Problem 3

The file Accidents.xlsx 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).

```
acc <- read_excel("C://Users/paini/Onedrive/Desktop/data mining/ASSIGNMENT 4/Accidents.xlsx",sheet=5)
```

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."

```
acc$INJURY <- ifelse(acc$MAX_SEV_IR>0, "yes", "no")
head(acc)
```

```
## # A tibble: 6 x 25
##   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
## # ... with 17 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>, INJURY <chr>
```

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?

```
#create a table based on INJURY
inj <- table(acc$INJURY)
show(inj)
```

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

```
#The output shows us that the data diverts more towards the possibility that an injury might occur afte

#Calculating probabilty to see the extent of occurence of injury in percentage form
inj.prob =  scales::percent(inj["yes"]/(inj["yes"]+inj["no"]),0.01)
inj.prob
```

```
##      yes
## "50.88%"
```

```
#Since more than 50% of the accidents in our data result in an accident, it is quite likely to suspect
```

Select the first 12 records in the dataset and look only at the response (INJURY) and the two predictors WEATHER_R and TRAF_CON_R.

```r
#A new subset of data with 12 records and 3 attributes
new.acc <- acc[1:12,c("INJURY","WEATHER_R","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.

```r
rpivotTable::rpivotTable(new.acc)
```

```r
getwd()
```

```
## [1] "C:/Users/paini/OneDrive/Desktop/data mining/ASSIGNMENT 4"
```

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

```r
library(dplyr)
new.acc
```

```
## # A tibble: 12 x 3
##    INJURY WEATHER_R TRAF_CON_R
##    <chr>      <dbl>      <dbl>
##  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
```

```r
#To find P(Injury=yes|WEATHER_R = 1, TRAF_CON_R =0):
numerator1 <- 2/3 * 3/12
denominator1 <- 3/12
prob1 <- numerator1/denominator1

#To find P(Injury=yes|WEATHER_R = 1, TRAF_CON_R =1):
numerator2 <- 0 * 3/12
denominator2 <- 1/12
prob2 <- numerator2/denominator2

#To find P(Injury=yes| WEATHER_R = 1, TRAF_CON_R =2):
numerator3 <- 0 * 3/12
denominator3 <- 1/12
prob3 <- numerator3/denominator3

#To find P(Injury=yes| WEATHER_R = 2, TRAF_CON_R =0):
numerator4 <- 1/3 * 3/12
```

```r
denominator4 <- 6/12
prob4 <- numerator4/denominator4

#To find P(Injury=yes| WEATHER_R = 2, TRAF_CON_R =1):
numerator5 <- 0 * 3/12
denominator5 <- 1/12
prob5 <- numerator5/denominator5

#To find P(Injury=yes| WEATHER_R = 2, TRAF_CON_R =2):
numerator6 <- 0 * 3/12
denominator6 <- 0
prob6 <- numerator6/denominator6

a<-c(1,2,3,4,5,6)
b<-c(prob1,prob2,prob3,prob4,prob5,prob6)
prob.acc<-data.frame(a,b)
names(prob.acc)<-c('Option #','Probability')
prob.acc
```

```
##   Option # Probability
## 1        1   0.6666667
## 2        2   0.0000000
## 3        3   0.0000000
## 4        4   0.1666667
## 5        5   0.0000000
## 6        6         NaN
```

```r
prob.acc %>% mutate_if(is.numeric, round, 3)
```

```
##   Option # Probability
## 1        1       0.667
## 2        2       0.000
## 3        3       0.000
## 4        4       0.167
## 5        5       0.000
## 6        6         NaN
```

```r
#NOTE: In the above 12 observations there is no observation with (Injury=yes, WEATHER_R = 2, TRAF_CON_R
```

Classify the 12 accidents using these probabilities and a cutoff of 0.5.

```r
#add probability results to you subset
new.df.prob<-new.acc
head(new.df.prob)
```

```
## # A tibble: 6 x 3
##   INJURY WEATHER_R TRAF_CON_R
##   <chr>      <dbl>      <dbl>
## 1 yes            1          0
## 2 no             2          0
## 3 no             2          1
```

```
## 4 no              1          1
## 5 no              1          0
## 6 yes             2          0
```

```
prob.inj <- c(0.667, 0.167, 0, 0, 0.667, 0.167, 0.167, 0.667, 0.167, 0.167, 0.167, 0)
new.df.prob$PROB_INJURY<-prob.inj

#add a column for injury prediction based on a cutoff of 0.5
new.df.prob$PREDICT_PROB<-ifelse(new.df.prob$PROB_INJURY>.5,"yes","no")
new.df.prob
```

```
## # A tibble: 12 x 5
##    INJURY WEATHER_R TRAF_CON_R PROB_INJURY PREDICT_PROB
##    <chr>      <dbl>      <dbl>       <dbl> <chr>
##  1 yes            1          0       0.667 yes
##  2 no             2          0       0.167 no
##  3 no             2          1       0     no
##  4 no             1          1       0     no
##  5 no             1          0       0.667 yes
##  6 yes            2          0       0.167 no
##  7 no             2          0       0.167 no
##  8 yes            1          0       0.667 yes
##  9 no             2          0       0.167 no
## 10 no             2          0       0.167 no
## 11 no             2          0       0.167 no
## 12 no             1          2       0     no
```

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

```
#To find P(Injury=yes| WEATHER_R = 1, TRAF_CON_R =1):
#  Probability of injury involved in accidents
#  =   (proportion of WEATHER_R =1 when Injury = yes)
#      *(proportion of TRAF_CON_R =1 when Injury = yes)
#      *(propotion of Injury = yes in all cases)
man.prob <- 2/3 * 0/3 * 3/12
man.prob
```

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

Run a naive Bayes classifier on the 12 records and two predictors using R. Check the model output to obtain probabilities and classifications for all 12 records. Compare this to the exact Bayes classification. Are the resulting classifications equivalent? Is the ranking (= ordering) of observations equivalent?

```
##load packages and run the naive Bayes classifier
library(e1071)
library(klaR)
```

```
## Warning: package 'klaR' was built under R version 3.6.3
```

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

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
## Loading required package: MASS
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
library(caret)
## Loading required package: lattice
## Loading required package: ggplot2
nb<-naiveBayes(INJURY ~ ., data = new.acc)
predict(nb, newdata = new.acc,type = "raw")
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion

##                no          yes
##  [1,] 0.001916916 0.9980830837
##  [2,] 0.006129754 0.9938702459
##  [3,] 0.999548668 0.0004513316
##  [4,] 0.998552097 0.0014479028
##  [5,] 0.001916916 0.9980830837
##  [6,] 0.006129754 0.9938702459
##  [7,] 0.006129754 0.9938702459
##  [8,] 0.001916916 0.9980830837
##  [9,] 0.006129754 0.9938702459
## [10,] 0.006129754 0.9938702459
## [11,] 0.006129754 0.9938702459
## [12,] 0.989399428 0.0106005719
```

```r
#check your model with the 'caret' package using the train and predict functions
library(caret)
x=new.acc[,-3]
y=new.acc$INJURY
model0 <- train(x,y,'nb', trControl = trainControl(method = 'cv',number=10))
```

```
## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning: model fit failed for Fold3: usekernel=FALSE, fL=0, adjust=1 Error in NaiveBayes.default(x, y
##   Zero variances for at least one class in variables: WEATHER_R

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.

## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning: Setting row names on a tibble is deprecated.
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion
```

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

```
## Warning: Setting row names on a tibble is deprecated.
```

```
model0
```

```
## Naive Bayes
##
## 12 samples
##  2 predictor
##  2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 10, 11, 10, 10, 11, 11, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.8750000  0
##    TRUE      0.8333333  0
##
## 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 = FALSE and adjust
##  = 1.
```

```
##Now that we have generated a classification model, we use it for prediction
model0.pred<-predict(model0$finalModel,x)
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion
```

```r
model0.pred
```

```
## $class
##  [1] no no no no no no no no no no no no
## Levels: no yes
##
## $posterior
##               no       yes
##  [1,] 0.6272018 0.3727982
##  [2,] 0.8438158 0.1561842
##  [3,] 0.8438158 0.1561842
##  [4,] 0.6272018 0.3727982
##  [5,] 0.6272018 0.3727982
##  [6,] 0.8438158 0.1561842
##  [7,] 0.8438158 0.1561842
##  [8,] 0.6272018 0.3727982
##  [9,] 0.8438158 0.1561842
## [10,] 0.8438158 0.1561842
## [11,] 0.8438158 0.1561842
## [12,] 0.6272018 0.3727982
```

```r
##build a confusion matrix so that we can visualize the classification errors
table(model0.pred$class,y)
```

```
##      y
##       no yes
##   no   9   3
##   yes  0   0
```

```r
#compare against the manually calculated results
new.df.prob$PREDICT_PROB_NB<-model0.pred$class
new.df.prob
```

```
## # A tibble: 12 x 6
##    INJURY WEATHER_R TRAF_CON_R PROB_INJURY PREDICT_PROB PREDICT_PROB_NB
##    <chr>      <dbl>      <dbl>       <dbl> <chr>        <fct>
##  1 yes            1          0       0.667 yes          no
##  2 no             2          0       0.167 no           no
##  3 no             2          1       0     no           no
##  4 no             1          1       0     no           no
##  5 no             1          0       0.667 yes          no
##  6 yes            2          0       0.167 no           no
##  7 no             2          0       0.167 no           no
##  8 yes            1          0       0.667 yes          no
##  9 no             2          0       0.167 no           no
## 10 no             2          0       0.167 no           no
## 11 no             2          0       0.167 no           no
## 12 no             1          2       0     no           no
```

```r
#NOTE: The errors that appear when running the naive Bayes on this sample set are nothing to really wor
```

Let us now return to the entire dataset. Partitioning the data into training (80%) and validation (20%)

```
set.seed(22)
train.index <- sample(c(1:dim(acc)[1]), dim(acc)[1]*0.8)
train.df <- acc[train.index,]
valid.df <- acc[-train.index,]
```

Assuming that no information or initial reports about the accident itself are available at the time of prediction (only location characteristics, weather conditions, etc.), which predictors can we include in the analysis? (Use the Data_Codes sheet.)

```
#   HOUR_I_R    1=rush hour, 0=not (rush = 6-9 am, 4-7 pm)
#   ALCOHOL_I   Alcohol involved = 1, not involved = 2
#   ALIGN_I 1 = straight, 2 = curve
#   WRK_ZONE    1= yes, 0= no
#   WKDY_I_R    1=weekday, 0=weekend
#   LGTCON_I_R  Light conditions - 1=day, 2=dark (including dawn/dusk), 3=dark, but lighted,4=dawn or d
#   SPD_LIM Speed limit, miles per hour
#   SUR_CON Surface conditions (1=dry, 2=wet, 3=snow/slush, 4=ice, 5=sand/dirt/oil, 8=other, 9=unknown)
#   TRAF_CON_R  Traffic control device: 0=none, 1=signal, 2=other (sign, officer ...)
#   TRAF_WAY    1=two-way traffic, 2=divided hwy, 3=one-way road
#   WEATHER_R   1=no adverse conditions, 2=rain, snow or other adverse condition
```

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. #

```
#define which variable you will be using


vars <- c("HOUR_I_R", "ALCHL_I",  "ALIGN_I" ,"WRK_ZONE",  "WKDY_I_R",
           "LGTCON_I_R", "SPD_LIM", "SUR_COND",
          "TRAF_CON_R",   "TRAF_WAY",   "WEATHER_R", "INJURY")
vars
```

```
##  [1] "HOUR_I_R"   "ALCHL_I"    "ALIGN_I"    "WRK_ZONE"   "WKDY_I_R"
##  [6] "LGTCON_I_R" "SPD_LIM"    "SUR_COND"   "TRAF_CON_R" "TRAF_WAY"
## [11] "WEATHER_R"  "INJURY"
```

```
nbTotal1 = naiveBayes(as.factor(INJURY) ~., data = train.df[,vars])
nbTotal1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        no       yes
## 0.4922065 0.5077935
##
## Conditional probabilities:
##      HOUR_I_R
```

```
## Y           [,1]      [,2]
##   no  0.4296207 0.4950369
##   yes 0.4234360 0.4941176
##
##      ALCHL_I
## Y          [,1]      [,2]
##   no  1.928417 0.2578046
##   yes 1.897351 0.3035093
##
##      ALIGN_I
## Y          [,1]      [,2]
##   no  1.129681 0.3359622
##   yes 1.134746 0.3414616
##
##      WRK_ZONE
## Y            [,1]      [,2]
##   no  0.02450331 0.1546103
##   yes 0.02135854 0.1445807
##
##      WKDY_I_R
## Y           [,1]      [,2]
##   no  0.7837447 0.4117027
##   yes 0.7638305 0.4247399
##
##      LGTCON_I_R
## Y          [,1]      [,2]
##   no  1.492655 0.7850277
##   yes 1.490079 0.7931834
##
##      SPD_LIM
## Y          [,1]     [,2]
##   no  43.55117 13.17148
##   yes 43.56326 12.69810
##
##      SUR_COND
## Y          [,1]      [,2]
##   no  1.318904 0.7842665
##   yes 1.262313 0.7691871
##
##      TRAF_CON_R
## Y           [,1]      [,2]
##   no  0.4968694 0.7471759
##   yes 0.5438842 0.7555262
##
##      TRAF_WAY
## Y          [,1]      [,2]
##   no  1.476099 0.5945450
##   yes 1.480976 0.5761427
##
##      WEATHER_R
## Y          [,1]      [,2]
##   no  1.157917 0.3646741
##   yes 1.128618 0.3347866
```

```r
modelPred <- predict(nbTotal1, acc)
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion
```

```r
cMatrix <- table(modelPred, acc$INJURY)
cMatrix
```

```
##
## modelPred    no   yes
##       no  10992  9819
##       yes  9729 11643
```

```r
c <- confusionMatrix(cMatrix, positive = "yes")
c
```

```
## Confusion Matrix and Statistics
##
##
## modelPred    no   yes
##       no  10992  9819
##       yes  9729 11643
##
##                Accuracy : 0.5366
##                  95% CI : (0.5318, 0.5414)
##     No Information Rate : 0.5088
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.073
##
##  Mcnemar's Test P-Value : 0.5244
##
##             Sensitivity : 0.5425
##             Specificity : 0.5305
##          Pos Pred Value : 0.5448
##          Neg Pred Value : 0.5282
##              Prevalence : 0.5088
##          Detection Rate : 0.2760
##    Detection Prevalence : 0.5066
##       Balanced Accuracy : 0.5365
##
##        'Positive' Class : yes
##
```

```r
train.er=1-.537
nerp=scales::percent(train.er,0.01)
nerp
```

```
## [1] "46.30%"
```

What is the overall error for the validation set?

```r
nbTotal2 = naiveBayes(as.factor(INJURY) ~., data = valid.df[,vars])
nbTotal2
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        no       yes
## 0.4872585 0.5127415
##
## Conditional probabilities:
##      HOUR_I_R
## Y          [,1]      [,2]
##   no  0.4427147 0.4967680
##   yes 0.4389736 0.4963192
##
##      ALCHL_I
## Y         [,1]      [,2]
##   no  1.930674 0.2540389
##   yes 1.897365 0.3035169
##
##      ALIGN_I
## Y         [,1]      [,2]
##   no  1.127706 0.3338029
##   yes 1.129681 0.3359910
##
##      WRK_ZONE
## Y           [,1]      [,2]
##   no  0.02213573 0.1471428
##   yes 0.02080444 0.1427457
##
##      WKDY_I_R
## Y          [,1]      [,2]
##   no  0.7764534 0.4166722
##   yes 0.7512714 0.4323261
##
##      LGTCON_I_R
## Y         [,1]      [,2]
##   no  1.486500 0.7841186
##   yes 1.507397 0.8007602
##
##      SPD_LIM
## Y         [,1]     [,2]
##   no  43.74361 13.22357
##   yes 43.28826 12.80173
##
##      SUR_COND
## Y         [,1]      [,2]
##   no  1.342982 0.8631965
```

```
##   yes 1.245261 0.7188266
##
##      TRAF_CON_R
## Y           [,1]      [,2]
##   no  0.4794454 0.7349484
##   yes 0.5168747 0.7437241
##
##      TRAF_WAY
## Y          [,1]      [,2]
##   no  1.470932 0.5878486
##   yes 1.475266 0.5787875
##
##      WEATHER_R
## Y          [,1]      [,2]
##   no  1.163950 0.3702756
##   yes 1.120666 0.3257761
```

```
modelPred2 <- predict(nbTotal2, acc)
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion
```

```
cMatrix2 <- table(modelPred2, acc$INJURY)
cMatrix2
```

```
##
## modelPred2    no   yes
##        no   5181  4306
##        yes 15540 17156
```

```
c2 <- confusionMatrix(cMatrix2, positive = "yes")
c2
```

```
## Confusion Matrix and Statistics
##
##
## modelPred2    no   yes
##        no   5181  4306
##        yes 15540 17156
##
##                Accuracy : 0.5295
##                  95% CI : (0.5247, 0.5343)
##     No Information Rate : 0.5088
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.0499
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7994
##             Specificity : 0.2500
##          Pos Pred Value : 0.5247
##          Neg Pred Value : 0.5461
```

```
##               Prevalence : 0.5088
##           Detection Rate : 0.4067
##     Detection Prevalence : 0.7751
##        Balanced Accuracy : 0.5247
##
##         'Positive' Class : yes
##
```

```
#Overall Error
valid.er=1-.53
verp=scales::percent(valid.er,0.01)
paste("Overall Error: ",verp)
```

```
## [1] "Overall Error:  47.00%"
```

What is the percent improvement relative to the naive rule (using the validation set)?

```
imp=valid.er-train.er
paste("The percent improvement is ",scales::percent(imp,0.01))
```

```
## [1] "The percent improvement is  0.70%"
```

Examine the conditional probabilities output. Why do we get a probability of zero for P(INJURY = No | SPD_LIM = 5)?

```
options(digits = 2)
nbTotal1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##   no  yes
## 0.49 0.51
##
## Conditional probabilities:
##      HOUR_I_R
## Y      [,1] [,2]
##   no   0.43 0.50
##   yes  0.42 0.49
##
##      ALCHL_I
## Y      [,1] [,2]
##   no    1.9 0.26
##   yes   1.9 0.30
##
##      ALIGN_I
## Y      [,1] [,2]
```

```
##   no   1.1 0.34
##   yes  1.1 0.34
##
##      WRK_ZONE
## Y       [,1] [,2]
##   no  0.025 0.15
##   yes 0.021 0.14
##
##      WKDY_I_R
## Y      [,1] [,2]
##   no  0.78 0.41
##   yes 0.76 0.42
##
##      LGTCON_I_R
## Y      [,1] [,2]
##   no   1.5 0.79
##   yes  1.5 0.79
##
##      SPD_LIM
## Y      [,1] [,2]
##   no    44   13
##   yes   44   13
##
##      SUR_COND
## Y      [,1] [,2]
##   no   1.3 0.78
##   yes  1.3 0.77
##
##      TRAF_CON_R
## Y      [,1] [,2]
##   no  0.50 0.75
##   yes 0.54 0.76
##
##      TRAF_WAY
## Y      [,1] [,2]
##   no   1.5 0.59
##   yes  1.5 0.58
##
##      WEATHER_R
## Y      [,1] [,2]
##   no   1.2 0.36
##   yes  1.1 0.33
```

```
#The reasoning aligns more with the logical prospect, it is very unlikely to sustain an injury with an
```