IE7275: Data Mining in Engineering

Homework-4

Prerit Samria
Seungyeon Ko

```
library (readxl)
library(caret)
```

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

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

```
# install.packages ("FNN")
library(FNN)
```

Problem-1

```
bank.df <- read_excel ("UniversalBank.xlsx", sheet = "Data", col_names = TRUE)
```

```
str(bank.df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    5000 obs. of  14 variables:
##  $ ID                : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age               : num  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : num  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : num  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP Code          : num  91107 90089 94720 94112 91330 ...
##  $ Family            : num  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : num  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : num  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal Loan     : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities Account: num  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD Account        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : num  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : num  0 0 0 0 1 0 0 1 0 0 ...
```

We see that predictors such as ID and ZIP code have no correlation with the outcome (Personal Loan). Thus we delete that from the data set.

```
bank.df <- bank.df [, -c(1, 5)]
str(bank.df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    5000 obs. of  12 variables:
##  $ Age               : num  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : num  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : num  49 34 11 100 45 29 72 22 81 180 ...
##  $ Family            : num  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : num  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : num  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal Loan     : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities Account: num  1 1 0 0 0 0 0 0 0 0 ...
```

```
##  $ CD Account        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : num  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : num  0 0 0 0 1 0 0 1 0 0 ...
```

Since Education has 3 categories, we need to convert it to 3 dummies for each category. "1" refers to undergraduate, "2" refers to graduate and "3" refers to advanced/professional.

```
bank.df$Education <- as.factor(bank.df$Education)
bank.df[,c("Education1", "Education2", "Education3")] <- model.matrix( ~ Education - 1,
                                                                        data = bank.df)
str(bank.df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    5000 obs. of  15 variables:
##  $ Age               : num  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : num  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : num  49 34 11 100 45 29 72 22 81 180 ...
##  $ Family            : num  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : num  0 0 0 0 0 155 0 0 104 0 ...
##  $ Personal Loan     : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ Securities Account: num  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD Account        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : num  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : num  0 0 0 0 1 0 0 1 0 0 ...
##  $ Education1        : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ Education2        : num  0 0 0 1 1 1 1 0 1 0 ...
##  $ Education3        : num  0 0 0 0 0 0 0 1 0 1 ...
```

```
# select the predictors and outcome (Personal Loan)
# Removing education and putting outcome as last column
bank.df <- bank.df [, c(1,2,3,4,5,13,14,15,7,9,10,11,12,8)]
```

We are going to change names of predictors and response variables such that they don't have space in between.

```
colnames(bank.df) [c(10, 11, 14)] <- c ("Securities.Account", "CD.Account",
                                        "Personal_Loan")
```

We will first standardize the data since the scales are different for the predictors.

```
# Partitioning the data into training (60%) and validation (40%) sets.
set.seed(101)
train.index <- sample(row.names(bank.df), 0.6*dim(bank.df)[1])
valid.index <- setdiff(row.names(bank.df), train.index)
train.df <- bank.df[train.index, ]
valid.df <- bank.df[valid.index, ]
```

Part-a

```
new.df <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2,
                     Education1 = 0, Education2 = 1, Education3 = 0, Mortgage = 0,
                     Securities.Account = 0, CD.Account = 0, Online = 1,
                     CreditCard = 1)
```

```
# Normalizing the data
bank.norm.df <- bank.df
train.norm.df <- train.df
valid.norm.df <- valid.df
```

```r
new.norm.df <- new.df
```

```r
# normalizing using preProcess
library(caret)
norm.values <- preProcess(train.df [, -14], method = c ("center", "scale"))
# preProcess: normalizes the data -> (x - mean(x))/ sd(x)
# method = "center" subtracts the mean of the predictor's data (again
# from the data in x) from the predictor values while method = "scale"
# divides by the standard deviation

bank.norm.df [,-14] <- predict (norm.values, bank.df [, -14])
train.norm.df [, -14] <- predict (norm.values, train.df [, -14])
valid.norm.df [, -14] <- predict (norm.values, valid.df [, -14])
new.norm.df <- predict(norm.values, new.df)
```

```r
# Applying kNN with k = 1
knn_1 <- knn(train = train.norm.df [, -14], test = new.norm.df,
             cl = train.norm.df$Personal_Loan, k = 1)
knn_1
```

```
## [1] 0
## attr(,"nn.index")
##      [,1]
## [1,]  336
## attr(,"nn.dist")
##           [,1]
## [1,] 0.4954934
## Levels: 0
```

This customer would be classified as "0", i.e., the customer won't be accepting the personal loan offer.

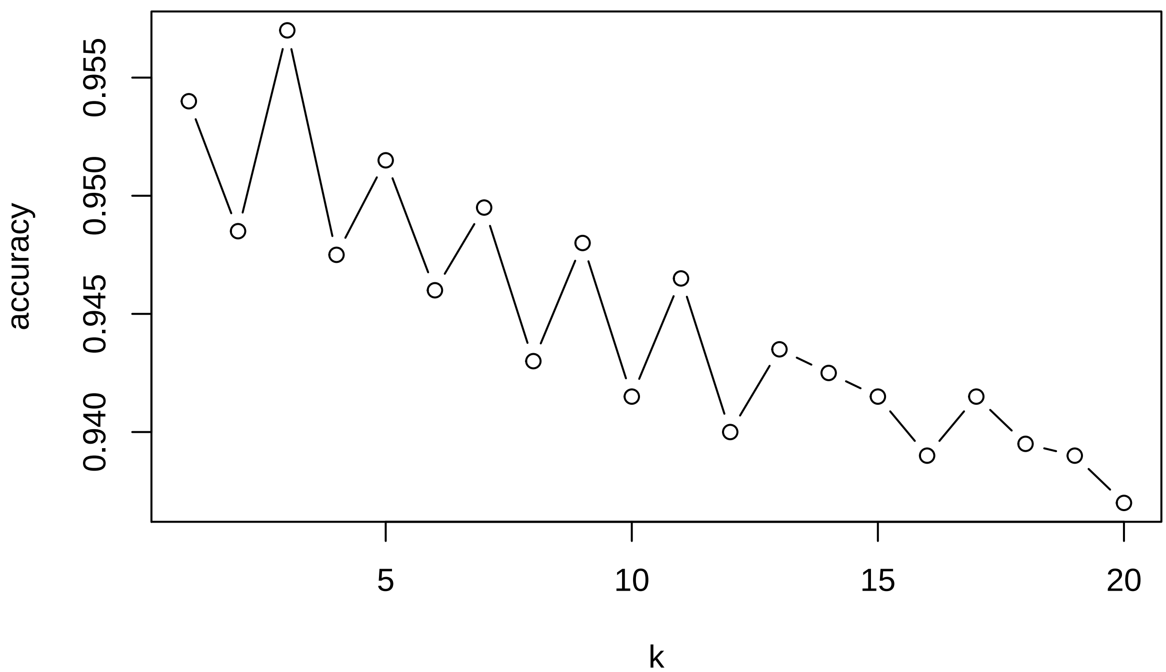
Part-b

```r
# initialize a data frame with two columns: k, and accuracy.
accuracy.df <- 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 = train.norm.df [, -14], test= valid.norm.df [, -14],
                  cl = train.norm.df$Personal_Loan, k = i)
  accuracy.df[i, 2] <- confusionMatrix(table(knn.pred,
                                             valid.norm.df$Personal_Loan))$overall[1]
}

accuracy.df
```

```
##     k accuracy
## 1   1   0.9540
## 2   2   0.9485
## 3   3   0.9570
## 4   4   0.9475
## 5   5   0.9515
## 6   6   0.9460
## 7   7   0.9495
## 8   8   0.9430
## 9   9   0.9480
```

```
## 10 10   0.9415
## 11 11   0.9465
## 12 12   0.9400
## 13 13   0.9435
## 14 14   0.9425
## 15 15   0.9415
## 16 16   0.9390
## 17 17   0.9415
## 18 18   0.9395
## 19 19   0.9390
## 20 20   0.9370
```

```
plot(accuracy.df, type = "b")
```



As we know that a low k value would result in overfitting. To reduce overfiiting, we choose k such that the accuracy is still high. Thus, we choose k = 5 whose accuracy is better only to k = 3 (which would result in overfitting).

Part-c

```
knn_5 <- knn(train = train.norm.df [, -14], test = valid.norm.df [, -14],
             cl = train.norm.df$Personal_Loan, k = 5)
table(valid.norm.df$Personal_Loan, knn_5)
```

```
##    knn_5
##         0    1
##   0 1795    8
##   1   89  108
```

Part-d

```
nn_5 <- knn(train = train.norm.df [, -14], test = new.norm.df,
            cl = train.norm.df$Personal_Loan, k = 5)
as.data.frame(nn_5)
```

```
##   nn_5
```

```
## 1     0
```

The customer would be classified as "0", i.e., the customer won't be accepting the personal loan offer.

Part-e

```r
# Partitioning the data into training (50%), validation (30%) and test (20%) sets.
set.seed(110)
train1.index <- sample(row.names(bank.df), 0.5*dim(bank.df)[1])
valid1.index <- sample(setdiff(row.names(bank.df), train1.index), 0.3*dim(bank.df)[1])
test1.index <- setdiff(row.names(bank.df), union(train1.index, valid1.index))
train1.df <- bank.df[train1.index, ]
valid1.df <- bank.df[valid1.index, ]
test1.df <- bank.df[test1.index, ]
```

```r
# Normalizing the data
bank1.norm.df <- bank.df
train1.norm.df <- train1.df
valid1.norm.df <- valid1.df
test1.norm.df <- test1.df
new1.norm.df <- new.df
```

```r
# normalizing using preProcess
library(caret)
norm1.values <- preProcess(train1.df [, -14], method = c ("center", "scale"))
# preProcess: normalizes the data -> (x - mean(x))/ sd(x)
# method = "center" subtracts the mean of the predictor's data (again
# from the data in x) from the predictor values while method = "scale"
# divides by the standard deviation

bank1.norm.df [,-14] <- predict (norm1.values, bank.df [, -14])
train1.norm.df [, -14] <- predict (norm1.values, train1.df [, -14])
valid1.norm.df [, -14] <- predict (norm1.values, valid1.df [, -14])
test1.norm.df [, -14] <- predict (norm1.values, test1.norm.df [, -14])
new1.norm.df <- predict(norm1.values, new.df)
```

```r
# compute knn
knn_train <- knn(train = train1.norm.df [, -14], test = train1.norm.df [, -14],
                 cl = train1.norm.df$Personal_Loan, k = 5)
table(train1.norm.df$Personal_Loan, knn_train)
```

```
##    knn_train
##        0    1
##   0 2250    3
##   1   73  174
```

Error rate for training set is 100*(73 + 3)/2500 = 3.04 %.

```r
knn_valid <- knn(train = train1.norm.df [, -14], test = valid1.norm.df [, -14],
                 cl = train1.norm.df$Personal_Loan, k = 5)
table(valid1.norm.df$Personal_Loan, knn_valid)
```

```
##    knn_valid
##        0    1
##   0 1356    4
##   1   61   79
```

Error rate for validation set is 100*(61 + 4)/1500 = 4.33 %.

```r
knn_test <- knn(train = train1.norm.df [, -14], test = test1.norm.df [, -14],
                cl = train1.norm.df$Personal_Loan, k = 5)
table(test1.norm.df$Personal_Loan, knn_test)
```

```
##    knn_test
##       0   1
##   0 901   6
##   1  33  60
```

Error rate for test set is 100*(33 + 6)/1000 = 3.9 %.

The error rates for validation and test sets are higher than that for the training set. Also, the difference between validation and test error rate is not high, which implies that the model does not overfit or ignore predictors.

The error rates obtained for training set is lower because we used training set itself as new data set's neighbors. There is small difference between validation and test error rates becuase we chose the value of k based on the validation error rate. Since the two error rates are similar, it means that the model is good.

Problem-2

```r
housing.df <- read_excel("BostonHousing.xlsx", sheet = "Data", col_names = TRUE)
str(housing.df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    506 obs. of  14 variables:
##  $ CRIM     : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ ZN       : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ INDUS    : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ CHAS     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NOX      : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ RM       : num  6.58 6.42 7.18 7 7.15 ...
##  $ AGE      : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ DIS      : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ RAD      : num  1 2 2 3 3 3 5 5 5 5 ...
##  $ TAX      : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ PTRATIO  : num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ LSTAT    : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ MEDV     : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
##  $ CAT. MEDV: num  0 0 1 1 1 0 0 0 0 0 ...
```

```r
# remove CAT.MEDV
housing.df <- housing.df [, -14]
str(housing.df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    506 obs. of  13 variables:
##  $ CRIM   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ ZN     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ INDUS  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ CHAS   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NOX    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ RM     : num  6.58 6.42 7.18 7 7.15 ...
##  $ AGE    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ DIS    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ RAD    : num  1 2 2 3 3 3 5 5 5 5 ...
##  $ TAX    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ PTRATIO: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
```

```
## $ LSTAT  : num  4.98 9.14 4.03 2.94 5.33 ...
## $ MEDV   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

We will first standardize the data since the scales are different for the predictors.

```r
# Partitioning the data into training (60%) and validation (40%) sets.
set.seed(201)
train2.index <- sample(row.names(housing.df), 0.6*dim(housing.df)[1])
valid2.index <- setdiff(row.names(housing.df), train2.index)
train2.df <- housing.df[train2.index, ]
valid2.df <- housing.df[valid2.index, ]
```

Part-a

```r
# Normalizing the data
housing.norm.df <- housing.df
train2.norm.df <- train2.df
valid2.norm.df <- valid2.df
```

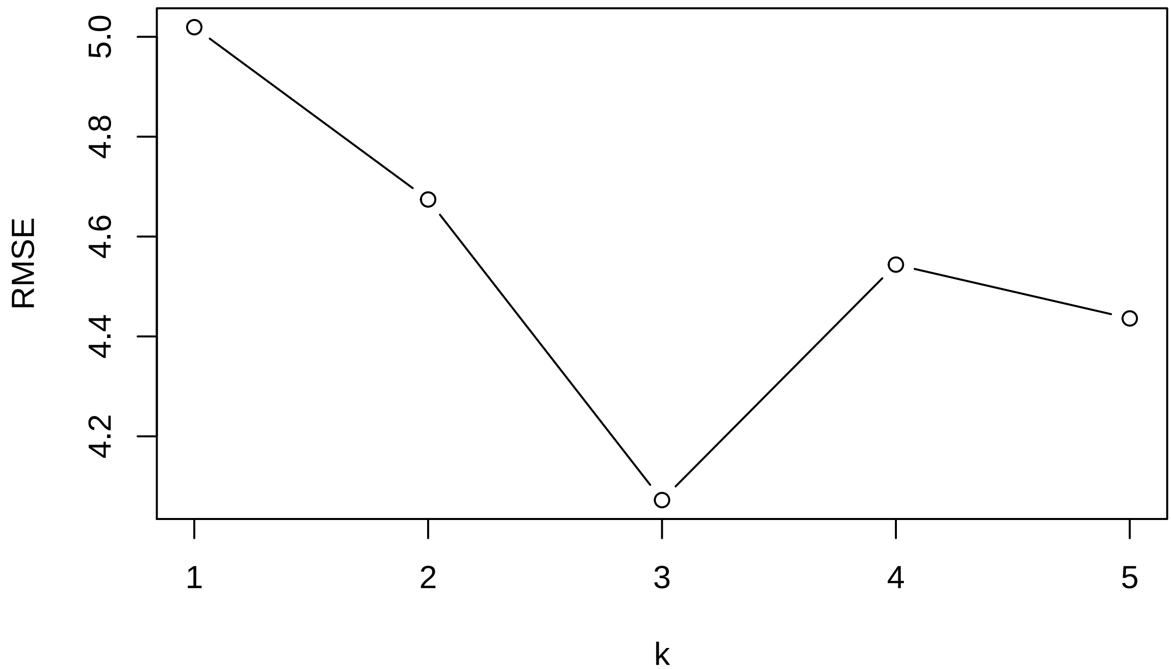
```r
# normalizing using preProcess
library(caret)
norm2.values <- preProcess(train2.df [, -13], method = c ("center", "scale"))
# preProcess: normalizes the data -> (x - mean(x))/ sd(x)
# method = "center" subtracts the mean of the predictor's data (again
# from the data in x) from the predictor values while method = "scale"
# divides by the standard deviation

housing.norm.df [,-13] <- predict (norm2.values, housing.df [, -13])
train2.norm.df [, -13] <- predict (norm2.values, train2.df [, -13])
valid2.norm.df [, -13] <- predict (norm2.values, valid2.df [, -13])
```

```r
RMSE.df <- data.frame(k = seq(1, 5, 1), RMSE = rep(0, 5))
```

```r
# compute knn for different k on validation.
# Since our response MEDV is numerical but not categorical, we want to use
# knn.reg() to do the prediction.
for(i in 1:5) {
  knn.pred <- knn.reg(train = train2.norm.df [, -13], test = valid2.norm.df [, -13],
                      y = train2.norm.df$MEDV, k = i)
  RMSE.df[i, 2] <- sqrt(sum((valid2.norm.df$MEDV -
                               as.array(knn.pred$pred))^2)/nrow(as.array(knn.pred$pred)))
}
RMSE.df
```

```
##   k     RMSE
## 1 1 5.019268
## 2 2 4.674206
## 3 3 4.072329
## 4 4 4.543815
## 5 5 4.436046
```

```r
plot(RMSE.df, type = "b")
```



We can observe that RMSE drops after k = 1, and rises up after k = 3, which means that the model might be overfitting at the beginning and then starts to ignore predictors after k = 3. Thus, k = 4 would be the best k.

Part-b

```
new2.df <- data.frame(CRIM = 0.2, ZN = 0, INDUS = 7, CHAS = 0, NOX = 0.538,
                      RM = 6, AGE = 62, DIS = 4.7, RAD = 4, TAX = 307,
                      PTRATIO = 21, LSTAT = 10)
```

```
# Normalizing the new data
new2.norm.df <- new2.df
new2.norm.df <- predict(norm2.values, new2.df)
```

```
knn_new2 <- knn.reg(train = train2.norm.df [, -13], test = new2.norm.df,
                    y = train2.norm.df$MEDV, k = 4)
knn_new2
```

```
## Prediction:
## [1] 18.9
```

The predicted MEDV is 18.9.

Part-c

```
knn_train2_1 <- knn.reg(train = train2.norm.df [, -13], test = train2.norm.df [, -13],
                      y = train2.norm.df$MEDV, k = 1)
knn_train2_1 <- as.array(knn_train2_1$pred)
RMSE_1 <- sqrt(sum((train2.norm.df$MEDV - knn_train2_1)^2)/nrow(knn_train2_1))
RMSE_1
```

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

The error of the training set would be 0 at k = 1 because we used training set itself as new data set's neighbors.

```
knn_train2_3 <- knn.reg(train = train2.norm.df [, -13], test = train2.norm.df [, -13],
                      y = train2.norm.df$MEDV, k = 3)
```

```r
knn_train2_3 <- as.array(knn_train2_3$pred)
RMSE_3 <- sqrt(sum((train2.norm.df$MEDV - knn_train2_3)^2)/nrow(knn_train2_3))
RMSE_3
```

```
## [1] 3.118639
```

The error of the training set would be 3.12 at k = 3.

Part-d

The model was chosen which performs best on the validation set. So the validation data error is overly optimistic compared to new data error rate.

Part-e

Algorithm for each prediction in k-NN are the following: (i) compute the distances of the new record from each record in the entire training set. (ii) find k numbers of nearest neighbors (smallest distances), take the weighted average response value of all the neighbors as the prediction value. The disadvantage of using k-NN prediction is that the process of prediction would be time-consuming since for each record, we need to compute its distances from the entire training set to predict it.

Problem-3

```r
accidents.df <- read_excel("Accidents.xlsx", sheet = "Data", col_names = TRUE)
```

```r
# Creating dummy variable called INJURY that takes the value "yes"
# if MAX_SEV_IR = 1 or 2, and otherwise "no."
accidents.df$INJURY <- ifelse(accidents.df$MAX_SEV_IR == 0, "No","Yes")
str(accidents.df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    42183 obs. of  25 variables:
##  $ HOUR_I_R      : num  0 1 1 1 1 1 1 1 1 0 ...
##  $ ALCHL_I       : num  2 2 2 2 1 2 2 2 2 2 ...
##  $ ALIGN_I       : num  2 1 1 1 1 1 1 1 1 1 ...
##  $ STRATUM_R     : num  1 0 0 1 0 1 0 1 1 0 ...
##  $ WRK_ZONE      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ WKDY_I_R      : num  1 1 1 0 1 1 1 1 1 0 ...
##  $ INT_HWY       : num  0 1 0 0 0 0 1 0 0 0 ...
##  $ LGTCON_I_R    : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ MANCOL_I_R    : num  0 2 2 2 2 0 0 0 0 0 ...
##  $ PED_ACC_R     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ RELJCT_I_R    : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ REL_RWY_R     : num  0 1 1 1 1 0 0 0 0 0 ...
##  $ PROFIL_I_R    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ SPD_LIM       : num  40 70 35 35 25 70 70 35 30 25 ...
##  $ SUR_COND      : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ TRAF_CON_R    : num  0 0 1 1 0 0 0 0 0 0 ...
##  $ TRAF_WAY      : num  3 3 2 2 2 2 2 1 1 1 ...
##  $ VEH_INVL      : num  1 2 2 2 3 1 1 1 1 1 ...
##  $ WEATHER_R     : num  1 2 2 1 1 2 2 1 2 2 ...
##  $ INJURY_CRASH  : num  1 0 0 0 0 1 0 1 0 0 ...
##  $ NO_INJ_I      : num  1 0 0 0 0 1 0 1 0 0 ...
##  $ PRPTYDMG_CRASH: num  0 1 1 1 1 0 1 0 1 1 ...
##  $ FATALITIES    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ MAX_SEV_IR    : num  1 0 0 0 0 1 0 1 0 0 ...
##  $ INJURY        : chr  "Yes" "No" "No" "No" ...
```

Part-a

```
table (accidents.df$INJURY)
```

```
##
##    No   Yes
## 20721 21462
```

If an accident has just been reported and no further information is available, we should predict INJURY = "Yes". Based on Naive Rule, we should predict "Yes" because probability of "Yes" in the data set is higher.

Part-b

```
subset <- accidents.df [1:12, c(16,19,25)]
subset$TRAF_CON_R <- factor(subset$TRAF_CON_R)
subset$WEATHER_R <- factor(subset$WEATHER_R)
subset$INJURY <- factor(subset$INJURY)
str(subset)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    12 obs. of  3 variables:
##  $ TRAF_CON_R: Factor w/ 3 levels "0","1","2": 1 1 2 2 1 1 1 1 1 1 ...
##  $ WEATHER_R : Factor w/ 2 levels "1","2": 1 2 2 1 1 2 2 1 2 2 ...
##  $ INJURY    : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 2 1 2 1 1 ...
```

(i)

```
table(subset$WEATHER_R, subset$TRAF_CON_R, subset$INJURY,
      dnn = c("WEATHER_R","TRAF_CON_R", "INJURY"))
```

```
## , , INJURY = No
##
##          TRAF_CON_R
## WEATHER_R 0 1 2
##         1 1 1 1
##         2 5 1 0
##
## , , INJURY = Yes
##
##          TRAF_CON_R
## WEATHER_R 0 1 2
##         1 2 0 0
##         2 1 0 0
```

(ii)

```
subset
```

```
## # A tibble: 12 x 3
##    TRAF_CON_R WEATHER_R INJURY
##    <fct>      <fct>     <fct>
##  1 0          1         Yes
##  2 0          2         No
##  3 1          2         No
##  4 1          1         No
##  5 0          1         No
##  6 0          2         Yes
##  7 0          2         No
##  8 0          1         Yes
```

```
##   9 0           2          No
## 10 0           2          No
## 11 0           2          No
## 12 2           1          No
```

Using Exact Bayes:

P(INJURY = “Yes” | TRAF_CON_R = 0, WEATHER_R = 1) = 2 / 3 = 0.667

P(INJURY = “Yes” | TRAF_CON_R = 0, WEATHER_R = 2) = 1 / 6 = 0.167

P(INJURY = “Yes” | TRAF_CON_R = 1, WEATHER_R = 1) = 0 / 1 = 0

P(INJURY = “Yes” | TRAF_CON_R = 1, WEATHER_R = 2) = 0 / 1 = 0

P(INJURY = “Yes” | TRAF_CON_R = 2, WEATHER_R = 1) = 0 / 1 = 0

P(INJURY = “Yes” | TRAF_CON_R = 2, WEATHER_R = 2) = 0 / 0 = 0

(iii)

Using Cutoff = 0.5, accident would be considered INJURY = “Yes” with TRAF_CON_R = 0 and WEATHER_R = 1 since only P(INJURY = “Yes” | TRAF_CON_R = 0, WEATHER_R = 1) > 0.5

```r
subset$predicted <- ifelse(subset$TRAF_CON_R == 0 & subset$WEATHER_R == 1, "Yes", "No")
subset
```

```
## # A tibble: 12 x 4
##    TRAF_CON_R WEATHER_R INJURY predicted
##    <fct>      <fct>     <fct>  <chr>
##  1 0          1         Yes    Yes
##  2 0          2         No     No
##  3 1          2         No     No
##  4 1          1         No     No
##  5 0          1         No     Yes
##  6 0          2         Yes    No
##  7 0          2         No     No
##  8 0          1         Yes    Yes
##  9 0          2         No     No
## 10 0          2         No     No
## 11 0          2         No     No
## 12 2          1         No     No
```

(iv)

Using Naives Bayes: P(INJURY = “Yes” | TRAF_CON_R = 1, WEATHER_R = 1) = Numerator / Denominator, where

Numerator = P (1 | “Yes”) . P (1 | “Yes”) . P (“Yes”) = (0/3)(2/3)(3/12) = 0 Denominator = P (1 | “Yes”) . P (1 | “Yes”) . P (“Yes”) + P (1 | “No”) . P (1 | “No”) . P (“No”) = (0/3)(2/3)(3/12) + (2/9)(3/9)(9/12) = 0 + 1/18 = 1/18

Thus, P(INJURY = “Yes” | TRAF_CON_R = 1, WEATHER_R = 1) = 0 / (1/18) = 0.

(v)

```r
library(e1071)
```

```r
subset.nb <- naiveBayes(INJURY ~ ., data = subset)
subset.nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##   No  Yes
## 0.75 0.25
##
## Conditional probabilities:
##      TRAF_CON_R
## Y             0         1         2
##   No  0.6666667 0.2222222 0.1111111
##   Yes 1.0000000 0.0000000 0.0000000
##
##      WEATHER_R
## Y             1         2
##   No  0.3333333 0.6666667
##   Yes 0.6666667 0.3333333
##
##      predicted
## Y            No       Yes
##   No  0.8888889 0.1111111
##   Yes 0.3333333 0.6666667
```

```r
# predict probabilities
pred.prob <- predict(subset.nb, newdata = subset, type = "raw")
pred.prob
```

```
##              No          Yes
##  [1,] 0.1428571 0.8571428571
##  [2,] 0.9142857 0.0857142857
##  [3,] 0.9997188 0.0002811709
##  [4,] 0.9988763 0.0011237358
##  [5,] 0.1428571 0.8571428571
##  [6,] 0.9142857 0.0857142857
##  [7,] 0.9142857 0.0857142857
##  [8,] 0.1428571 0.8571428571
##  [9,] 0.9142857 0.0857142857
## [10,] 0.9142857 0.0857142857
## [11,] 0.9142857 0.0857142857
## [12,] 0.9977551 0.0022449489
```

```r
# predict class membership, cutoff = 0.5
pred.class <- predict (subset.nb, newdata = subset)
pred.class
```

```
##  [1] Yes No  No  No  Yes No  No  Yes No  No  No  No
## Levels: No Yes
```

```r
df <- data.frame(actual = subset$INJURY, predicted = pred.class, pred.prob)
df
```

```
##    actual predicted        No          Yes
```

```
## 1     Yes       Yes 0.1428571 0.8571428571
## 2      No        No 0.9142857 0.0857142857
## 3      No        No 0.9997188 0.0002811709
## 4      No        No 0.9988763 0.0011237358
## 5      No       Yes 0.1428571 0.8571428571
## 6     Yes        No 0.9142857 0.0857142857
## 7      No        No 0.9142857 0.0857142857
## 8     Yes       Yes 0.1428571 0.8571428571
## 9      No        No 0.9142857 0.0857142857
## 10     No        No 0.9142857 0.0857142857
## 11     No        No 0.9142857 0.0857142857
## 12     No        No 0.9977551 0.0022449489
```

The resulting classifications are equivalent. The ranking (= ordering) of observations are also equivalent.

Part-c

```r
# Partitioning the data into training (60%) and validation (40%) sets.
set.seed(401)
train.index <- sample(nrow(accidents.df), 0.6*nrow(accidents.df))
```

(i)

After looking at the Data_Codes sheet, we observe that all the predictors are important. However, we can remove MAX_SEV_IR since it is a redundant variable.

```r
accidents.df <- accidents.df [, -24]
str(accidents.df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    42183 obs. of  24 variables:
##  $ HOUR_I_R      : num  0 1 1 1 1 1 1 1 1 0 ...
##  $ ALCHL_I       : num  2 2 2 2 1 2 2 2 2 2 ...
##  $ ALIGN_I       : num  2 1 1 1 1 1 1 1 1 1 ...
##  $ STRATUM_R     : num  1 0 0 1 0 1 0 1 1 0 ...
##  $ WRK_ZONE      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ WKDY_I_R      : num  1 1 1 0 1 1 1 1 1 0 ...
##  $ INT_HWY       : num  0 1 0 0 0 0 1 0 0 0 ...
##  $ LGTCON_I_R    : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ MANCOL_I_R    : num  0 2 2 2 2 0 0 0 0 0 ...
##  $ PED_ACC_R     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ RELJCT_I_R    : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ REL_RWY_R     : num  0 1 1 1 1 0 0 0 0 0 ...
##  $ PROFIL_I_R    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ SPD_LIM       : num  40 70 35 35 25 70 70 35 30 25 ...
##  $ SUR_COND      : num  4 4 4 4 4 4 4 4 4 4 ...
##  $ TRAF_CON_R    : num  0 0 1 1 0 0 0 0 0 0 ...
##  $ TRAF_WAY      : num  3 3 2 2 2 2 2 1 1 1 ...
##  $ VEH_INVL      : num  1 2 2 2 3 1 1 1 1 1 ...
##  $ WEATHER_R     : num  1 2 2 1 1 2 2 1 2 2 ...
##  $ INJURY_CRASH  : num  1 0 0 0 0 1 0 1 0 0 ...
##  $ NO_INJ_I      : num  1 0 0 0 0 1 0 1 0 0 ...
##  $ PRPTYDMG_CRASH: num  0 1 1 1 1 0 1 0 1 1 ...
##  $ FATALITIES    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ INJURY        : chr  "Yes" "No" "No" "No" ...
```

Now, we need to convert all the variables into categorical.

```r
accidents.df$HOUR_I_R <- factor(accidents.df$HOUR_I_R)
accidents.df$ALCHL_I <- factor(accidents.df$ALCHL_I)
accidents.df$ALIGN_I <- factor(accidents.df$ALIGN_I)
accidents.df$STRATUM_R <- factor(accidents.df$STRATUM_R)
accidents.df$WRK_ZONE <- factor(accidents.df$WRK_ZONE)
accidents.df$WKDY_I_R <- factor(accidents.df$WKDY_I_R)
accidents.df$INT_HWY <- factor(accidents.df$INT_HWY)
accidents.df$LGTCON_I_R <- factor(accidents.df$LGTCON_I_R)
accidents.df$MANCOL_I_R <- factor(accidents.df$MANCOL_I_R)
accidents.df$PED_ACC_R <- factor(accidents.df$PED_ACC_R)
accidents.df$RELJCT_I_R <- factor(accidents.df$RELJCT_I_R)
accidents.df$REL_RWY_R <- factor(accidents.df$REL_RWY_R)
accidents.df$PROFIL_I_R <- factor(accidents.df$PROFIL_I_R)
accidents.df$SPD_LIM <- factor(accidents.df$SPD_LIM)
accidents.df$SUR_COND <- factor(accidents.df$SUR_COND)
accidents.df$TRAF_CON_R <- factor(accidents.df$TRAF_CON_R)
accidents.df$TRAF_WAY <- factor(accidents.df$TRAF_WAY)
accidents.df$VEH_INVL <- factor(accidents.df$VEH_INVL)
accidents.df$WEATHER_R <- factor(accidents.df$WEATHER_R)
accidents.df$INJURY_CRASH <- factor(accidents.df$INJURY_CRASH)
accidents.df$NO_INJ_I <- factor(accidents.df$NO_INJ_I)
accidents.df$PRPTYDMG_CRASH <- factor(accidents.df$PRPTYDMG_CRASH)
accidents.df$FATALITIES <- factor(accidents.df$FATALITIES)
accidents.df$INJURY <- factor(accidents.df$INJURY)
str(accidents.df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    42183 obs. of  24 variables:
##  $ HOUR_I_R      : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 1 ...
##  $ ALCHL_I       : Factor w/ 2 levels "1","2": 2 2 2 2 1 2 2 2 2 2 ...
##  $ ALIGN_I       : Factor w/ 2 levels "1","2": 2 1 1 1 1 1 1 1 1 1 ...
##  $ STRATUM_R     : Factor w/ 2 levels "0","1": 2 1 1 2 1 2 1 2 2 1 ...
##  $ WRK_ZONE      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ WKDY_I_R      : Factor w/ 2 levels "0","1": 2 2 2 1 2 2 2 2 2 1 ...
##  $ INT_HWY       : Factor w/ 3 levels "0","1","9": 1 2 1 1 1 1 2 1 1 1 ...
##  $ LGTCON_I_R    : Factor w/ 3 levels "1","2","3": 3 3 3 3 3 3 3 3 3 3 ...
##  $ MANCOL_I_R    : Factor w/ 3 levels "0","1","2": 1 3 3 3 3 1 1 1 1 1 ...
##  $ PED_ACC_R     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ RELJCT_I_R    : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 1 2 2 ...
##  $ REL_RWY_R     : Factor w/ 2 levels "0","1": 1 2 2 2 2 1 1 1 1 1 ...
##  $ PROFIL_I_R    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ SPD_LIM       : Factor w/ 15 levels "5","10","15",..: 8 14 7 7 5 14 14 7 6 5 ...
##  $ SUR_COND      : Factor w/ 5 levels "1","2","3","4",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ TRAF_CON_R    : Factor w/ 3 levels "0","1","2": 1 1 2 2 1 1 1 1 1 1 ...
##  $ TRAF_WAY      : Factor w/ 3 levels "1","2","3": 3 3 2 2 2 2 2 1 1 1 ...
##  $ VEH_INVL      : Factor w/ 11 levels "1","2","3","4",..: 1 2 2 2 3 1 1 1 1 1 ...
##  $ WEATHER_R     : Factor w/ 2 levels "1","2": 1 2 2 1 1 2 2 1 2 2 ...
##  $ INJURY_CRASH  : Factor w/ 2 levels "0","1": 2 1 1 1 1 2 1 2 1 1 ...
##  $ NO_INJ_I      : Factor w/ 15 levels "0","1","2","3",..: 2 1 1 1 1 2 1 2 1 1 ...
##  $ PRPTYDMG_CRASH: Factor w/ 2 levels "0","1": 1 2 2 2 2 1 2 1 2 2 ...
##  $ FATALITIES    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ INJURY        : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 2 1 2 1 1 ...
```

```r
# Partining into training and validation sets
train.df <- accidents.df [train.index, ]
```

```r
valid.df <- accidents.df [-train.index, ]
```

(ii)

```r
train.nb <- naiveBayes(INJURY ~ ., data = train.df)
train.nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        No       Yes
## 0.4940535 0.5059465
##
## Conditional probabilities:
##      HOUR_I_R
## Y             0         1
##   No  0.5691779 0.4308221
##   Yes 0.5732136 0.4267864
##
##      ALCHL_I
## Y              1          2
##   No  0.07229687 0.92770313
##   Yes 0.10464662 0.89535338
##
##      ALIGN_I
## Y             1         2
##   No  0.8704415 0.1295585
##   Yes 0.8684108 0.1315892
##
##      STRATUM_R
## Y             0         1
##   No  0.5451855 0.4548145
##   Yes 0.4752050 0.5247950
##
##      WRK_ZONE
## Y              0          1
##   No  0.97472809 0.02527191
##   Yes 0.97852401 0.02147599
##
##      WKDY_I_R
## Y             0         1
##   No  0.2183301 0.7816699
##   Yes 0.2381882 0.7618118
##
##      INT_HWY
## Y                0            1            9
##   No  0.8498080614 0.1495521433 0.0006397953
##   Yes 0.8601327606 0.1390081999 0.0008590394
##
```

```
##      LGTCON_I_R
## Y             1         2         3
##   No  0.6926583 0.1255598 0.1817818
##   Yes 0.6961343 0.1115970 0.1922686
##
##      MANCOL_I_R
## Y              0          1          2
##   No  0.31261996 0.01391555 0.67346449
##   Yes 0.33525966 0.02991019 0.63483014
##
##      PED_ACC_R
## Y                0            1
##   No  0.9994401791 0.0005598209
##   Yes 0.9164388911 0.0835611089
##
##      RELJCT_I_R
## Y             0         1
##   No  0.4615323 0.5384677
##   Yes 0.4236626 0.5763374
##
##      REL_RWY_R
## Y             0         1
##   No  0.2477607 0.7522393
##   Yes 0.2200703 0.7799297
##
##      PROFIL_I_R
## Y             0         1
##   No  0.7515995 0.2484005
##   Yes 0.7650918 0.2349082
##
##      SPD_LIM
## Y                5           10           15           20           25
##   No  7.997441e-05 4.798464e-04 3.998720e-03 7.517594e-03 1.096449e-01
##   Yes 7.809449e-05 3.904725e-04 4.139008e-03 4.139008e-03 9.043342e-02
##      SPD_LIM
## Y               30           35           40           45           50
##   No  8.589251e-02 1.934581e-01 9.588932e-02 1.585893e-01 4.246641e-02
##   Yes 8.902772e-02 2.136665e-01 1.093323e-01 1.532995e-01 3.732917e-02
##      SPD_LIM
## Y               55           60           65           70           75
##   No  1.549904e-01 3.446897e-02 6.677863e-02 3.966731e-02 6.078055e-03
##   Yes 1.548614e-01 4.303007e-02 6.247560e-02 3.014447e-02 7.653260e-03
##
##      SUR_COND
## Y               1           2           3           4           9
##   No  0.777591171 0.175063980 0.015754958 0.027431222 0.004158669
##   Yes 0.811557985 0.156813745 0.010230379 0.015931277 0.005466615
##
##      TRAF_CON_R
## Y             0         1         2
##   No  0.6594690 0.1893794 0.1511516
##   Yes 0.6188208 0.2231941 0.1579852
##
##      TRAF_WAY
```

```
## Y              1          2          3
##   No  0.58125400 0.36668266 0.05206334
##   Yes 0.55642327 0.40117142 0.04240531
##
##      VEH_INVL
## Y                1            2            3            4            5
##   No  2.973448e-01 6.310781e-01 6.102047e-02 8.477287e-03 1.439539e-03
##   Yes 3.159703e-01 5.620461e-01 9.597813e-02 2.007029e-02 4.607575e-03
##      VEH_INVL
## Y                6            7            8            9           10
##   No  3.198976e-04 1.599488e-04 1.599488e-04 0.000000e+00 0.000000e+00
##   Yes 7.028504e-04 1.561890e-04 2.342835e-04 1.561890e-04 7.809449e-05
##      VEH_INVL
## Y               23
##   No  0.000000e+00
##   Yes 0.000000e+00
##
##      WEATHER_R
## Y             1         2
##   No  0.8427703 0.1572297
##   Yes 0.8714565 0.1285435
##
##      INJURY_CRASH
## Y              0          1
##   No  1.00000000 0.00000000
##   Yes 0.02186646 0.97813354
##
##      NO_INJ_I
## Y                0            1            2            3            4
##   No  1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   Yes 0.000000e+00 6.649746e-01 2.205389e-01 7.348692e-02 2.295978e-02
##      NO_INJ_I
## Y                5            6            7            8            9
##   No  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   Yes 1.038657e-02 3.748536e-03 2.108551e-03 7.809449e-04 3.904725e-04
##      NO_INJ_I
## Y               10           11           14           20           31
##   No  0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##   Yes 3.123780e-04 7.809449e-05 7.809449e-05 7.809449e-05 7.809449e-05
##
##      PRPTYDMG_CRASH
## Y      0 1
##   No  0 1
##   Yes 1 0
##
##      FATALITIES
## Y              0          1
##   No  1.00000000 0.00000000
##   Yes 0.97813354 0.02186646
```

```r
pred.class_train <- predict (train.nb, newdata = train.df)

# Confusion Matrix
confusionMatrix(pred.class_train, train.df$INJURY)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  12504     0
##        Yes     0 12805
##
##                Accuracy : 1
##                  95% CI : (0.9999, 1)
##     No Information Rate : 0.5059
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4941
##          Detection Rate : 0.4941
##    Detection Prevalence : 0.4941
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : No
##
```

(iii)

```
pred.prob_valid <- predict(train.nb, newdata = valid.df, type = "raw")
pred.class_valid <- predict(train.nb, newdata = valid.df)
confusionMatrix(pred.class_valid, valid.df$INJURY)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  8217    0
##        Yes    0 8657
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.513
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.487
```

```
##          Detection Rate : 0.487
##    Detection Prevalence : 0.487
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : No
##
```

The overall error rate for the validation set is 0.

(iv)

For both the training set and validation set, error rate is 0.

(v)

```r
table(valid.df$INJURY, valid.df$SPD_LIM, dnn = c("INJURY", "SPD_LIM"))
```

```
##       SPD_LIM
## INJURY    5   10   15   20   25   30   35   40   45   50   55   60   65
##    No     1    5   43   65  874  733 1575  779 1257  313 1368  296  536
##    Yes    3    6   37   39  802  768 1811  926 1384  343 1305  380  544
##       SPD_LIM
## INJURY   70   75
##    No   322   50
##    Yes  250   59
```

Among 25309 cases, there is only 1 case with INJURY = "No" and SPD_LIM = 5. Thus the probability is nearly zero.