Homework-3

Group IX (Rajeev Motwani)

Ashvin Khairnar

Dimple Bapna

Soumyajeet Patra

Balaji Kothandaraman Kalanidhi

206-941-8514(Ashvin Khairnar)

425-233-7801 (Dimple Bapna)

206-218-3144(Soumyajeet Patra)

206-915-5863 (Balaji Kothandaraman kalanidhi)

Percentage of Effort Contributed by Student 1: 25

Percentage of Effort Contributed by Student 2: 25

Percentage of Effort Contributed by Student 3: 25

Percentage of Effort Contributed by Student 4: 25

Signature of Student 1: Ashvin Khairnar

Signature of Student 2: Dimple Bapna

Signature of Student 3: Soumyajeet Patra

Signature of Student 4: Balaji Kothandaraman Kalanidhi

Submission Date: 04-01-2020

```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

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.
3.0 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v stringr 1.4.0
## v tidyr   1.0.2     v forcats 0.5.0
## v readr   1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflict
s() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

library(FNN)
library(class)

##
## Attaching package: 'class'

## The following objects are masked from 'package:FNN':
##
##     knn, knn.cv

library(e1071)
library(fastDummies)
```

```
library(caTools)
library(readr)
library(reshape2)

##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

Problem 7.1 [25 points]

Partition the data into training (60%) and validation (40%) sets.

```
dataset = read.csv("C:\\Users\\kkbal\\OneDrive\\Desktop\\Neu\\data mining\\As
signment-3\\UniversalBank.csv")

head(dataset)

##   ID Age Experience Income ZIP.Code Family CCAvg Education Mortgage
## 1  1  25          1     49    91107      4   1.6         1        0
## 2  2  45         19     34    90089      3   1.5         1        0
## 3  3  39         15     11    94720      1   1.0         1        0
## 4  4  35          9    100    94112      1   2.7         2        0
## 5  5  35          8     45    91330      4   1.0         2        0
## 6  6  37         13     29    92121      4   0.4         2      155
##   Personal.Loan Securities.Account CD.Account Online CreditCard
## 1             0                  1          0      0          0
## 2             0                  1          0      0          0
## 3             0                  0          0      0          0
## 4             0                  0          0      0          0
## 5             0                  0          0      0          1
## 6             0                  0          0      1          0

set.seed(100)

# Transforming the categorical variable "Education" into dummy variables

dataset <- dummy_cols(dataset, select_columns = 'Education', remove_selected_
columns = TRUE)
head(dataset)

##   ID Age Experience Income ZIP.Code Family CCAvg Mortgage Personal.Loan
## 1  1  25          1     49    91107      4   1.6        0             0
## 2  2  45         19     34    90089      3   1.5        0             0
## 3  3  39         15     11    94720      1   1.0        0             0
## 4  4  35          9    100    94112      1   2.7        0             0
## 5  5  35          8     45    91330      4   1.0        0             0
## 6  6  37         13     29    92121      4   0.4      155             0
##   Securities.Account CD.Account Online CreditCard Education_1 Education_2
## 1                  1          0      0          0           1           0
```

```
## 2                      1           0       0           0            1            0
## 3                      0           0       0           0            1            0
## 4                      0           0       0           0            0            1
## 5                      0           0       0           1            0            1
## 6                      0           0       1           0            0            1
##   Education_3
## 1           0
## 2           0
## 3           0
## 4           0
## 5           0
## 6           0
```

```
# Splitting the data into training(60%) and validation(40%) sets

split = sample.split(dataset$Personal.Loan, SplitRatio = 0.6)
training_set = subset(dataset, split == TRUE)
validation_set = subset(dataset, split == FALSE)
```

Problem 1.

a. 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. 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?

```
# Fitting K-NN to the training_set and predicting the test set result

test_set = data.frame(Age = as.integer(40), Experience = 10, Income = 84, Fam
ily = 2, CCAvg = 2,
                      Education_1 = 0, Education_2 = 1, Education_3 = 0,
                      Mortgage = 0, 'Securities Account' = 0, 'CD Account' =
0,
                      Online = 1, CreditCard = 1)
y_pred = knn(train = training_set[-c(1, 5, 9)],
             test = test_set,
             cl = training_set[,9],
             k = 1)
y_pred

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

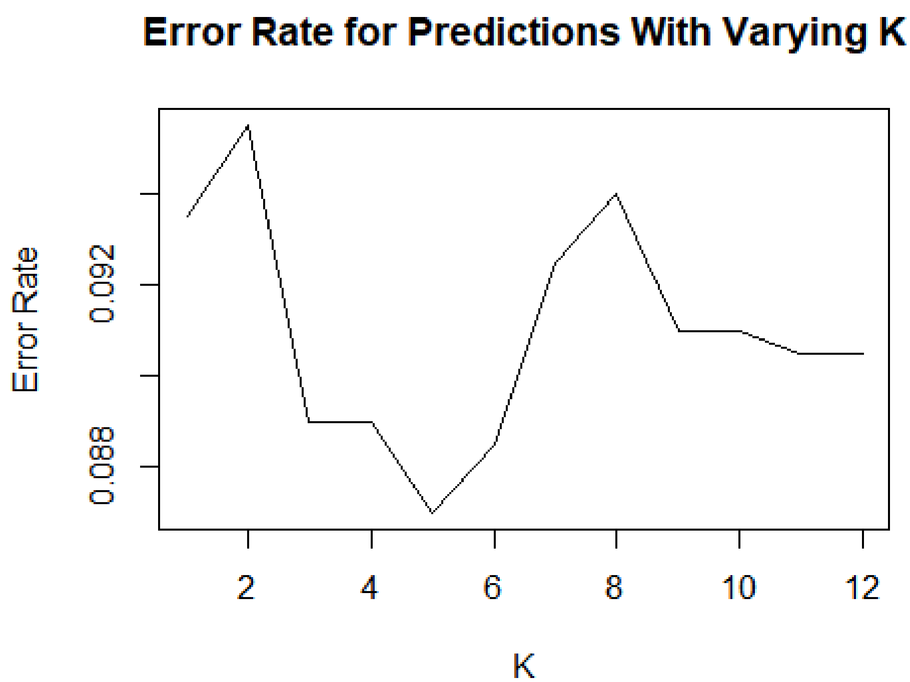
This customer did not accept the personal loan offered in the earlier campaign.

Problem 1.

b. What is a choice of k that balances between overfitting and ignoring the predictor information?

```
acc <- numeric()
for(i in 1:12) {
y_pred <- knn(train = training_set[-c(1, 5, 9)],
             test = validation_set[-c(1, 5, 9)],
             cl = training_set[,9],
             k = i)
acc <- c(acc, mean(y_pred == validation_set$Personal.Loan))
}

plot(1-acc,type="l",ylab="Error Rate",
 xlab="K",main="Error Rate for Predictions With Varying K")
```



As the error rate is lowest when k =5, it clearly balances between overfitting and ignoring the predictor information

Problem 1.

c) Show the confusion matrix for the validation data that results from using the best k.

```
y_pred = knn(train = training_set[-c(1, 5, 9)],
             test = validation_set[-c(1, 5, 9)],
             cl = training_set[,9],
             k = 5)
cm = table(validation_set[, 10], y_pred)
confusionMatrix(cm)
```

```
## Confusion Matrix and Statistics
##
##    y_pred
##        0    1
##   0 1676  125
##   1  186   13
##
##                Accuracy : 0.8445
##                  95% CI : (0.8279, 0.8601)
##     No Information Rate : 0.931
##     P-Value [Acc > NIR] : 1.0000000
##
##                   Kappa : -0.0047
##
##  Mcnemar's Test P-Value : 0.0006682
##
##             Sensitivity : 0.90011
##             Specificity : 0.09420
##          Pos Pred Value : 0.93059
##          Neg Pred Value : 0.06533
##              Prevalence : 0.93100
##          Detection Rate : 0.83800
##    Detection Prevalence : 0.90050
##       Balanced Accuracy : 0.49716
##
##        'Positive' Class : 0
##
```

Problem 1.

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.

```
y_pred = knn(train = training_set[-c(1, 5, 9)],
             test = test_set,
             cl = training_set[,9],
             k = 6)
y_pred
```

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

This customer did not accept the personal loan offered in the earlier campaign.

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 confusion matrix of the test set with that of the training and validation sets. Comment on the differences and their reason.

```
data <- sample(1:3, prob = c(0.5, 0.3, 0.2))
train_set <- dataset[data == 1, ]
validation_set <- dataset[data == 2, ]
test_set <- dataset[data == 3, ]

y_pred_validation = knn(train = training_set[-c(1, 5, 9)],
             test = validation_set[-c(1, 5, 9)],
             cl = training_set[,9],
             k = 6)

y_pred_test = knn(train = training_set[-c(1, 5, 9)],
             test = test_set[-c(1, 5, 9)],
             cl = training_set[,9],
             k = 6)

cm_validation = table(validation_set[, 10], y_pred_validation)
cm_test = table(test_set[, 10], y_pred_test)
```

```
confusionMatrix(cm_validation)
```

```
## Confusion Matrix and Statistics
##
##    y_pred_validation
##        0    1
##   0 1415   93
##   1  149   10
##
##                Accuracy : 0.8548
##                  95% CI : (0.837, 0.8714)
##     No Information Rate : 0.9382
##     P-Value [Acc > NIR] : 1.000000
##
##                   Kappa : 0.0015
##
##  Mcnemar's Test P-Value : 0.000407
##
##             Sensitivity : 0.90473
##             Specificity : 0.09709
##          Pos Pred Value : 0.93833
##          Neg Pred Value : 0.06289
##              Prevalence : 0.93821
##          Detection Rate : 0.84883
##    Detection Prevalence : 0.90462
##       Balanced Accuracy : 0.50091
##
```

```
##        'Positive' Class : 0
##
```

```r
confusionMatrix(cm_test)
```

```
## Confusion Matrix and Statistics
##
##    y_pred_test
##        0    1
##   0 1400   87
##   1  170    9
##
##                Accuracy : 0.8457
##                  95% CI : (0.8275, 0.8628)
##     No Information Rate : 0.9424
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0103
##
##  Mcnemar's Test P-Value : 3.137e-07
##
##             Sensitivity : 0.89172
##             Specificity : 0.09375
##          Pos Pred Value : 0.94149
##          Neg Pred Value : 0.05028
##              Prevalence : 0.94238
##          Detection Rate : 0.84034
##    Detection Prevalence : 0.89256
##       Balanced Accuracy : 0.49273
##
##        'Positive' Class : 0
##
```

Problem 7.2 [25 points] Predicting Housing Median Prices. The file BostonHousing.csv contains information on over 500 census tracts in Boston, where for each tract multiple 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 12 column

```r
housing<-read_csv('C:\\Users\\kkbal\\OneDrive\\Desktop\\Neu\\data mining\\Ass
ignment-3\\BostonHousing.csv')
```

```
## Parsed with column specification:
## cols(
##   CRIM = col_double(),
##   ZN = col_double(),
##   INDUS = col_double(),
##   CHAS = col_double(),
##   NOX = col_double(),
##   RM = col_double(),
##   AGE = col_double(),
```

```
##   DIS = col_double(),
##   RAD = col_double(),
##   TAX = col_double(),
##   PTRATIO = col_double(),
##   LSTAT = col_double(),
##   MEDV = col_double(),
##   `CAT. MEDV` = col_double()
## )
```

```
housing$`CAT. MEDV`<-as.factor(housing$`CAT. MEDV`)
head(housing)
```

```
## # A tibble: 6 x 14
##      CRIM    ZN INDUS  CHAS   NOX    RM   AGE   DIS   RAD   TAX PTRATIO
##     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 0.00632    18  2.31     0 0.538  6.58  65.2  4.09     1   296    15.3
## 2 0.0273      0  7.07     0 0.469  6.42  78.9  4.97     2   242    17.8
## 3 0.0273      0  7.07     0 0.469  7.18  61.1  4.97     2   242    17.8
## 4 0.0324      0  2.18     0 0.458  7.00  45.8  6.06     3   222    18.7
## 5 0.0690      0  2.18     0 0.458  7.15  54.2  6.06     3   222    18.7
## 6 0.0298      0  2.18     0 0.458  6.43  58.7  6.06     3   222    18.7
## # ... with 3 more variables: LSTAT <dbl>, MEDV <dbl>, `CAT. MEDV` <fct>
```

Partitioning into 60 and 40.

```
set.seed(111)

train<-sample(row.names(housing),0.6*dim(housing)[1])
validation<-setdiff(row.names(housing),train)

train.df<-housing[train,]
validation.df<-housing[validation,]
```

a. Perform a k-NN prediction with all 12 predictors (ignore the CAT.MEDV column),trying values of k from 1 to 5. Make sure to normalize the data, and choose functionknn() from package class rather than package FNN. To make sure R is using the classpackage (when both packages are loaded), use class::knn(). What is the best k? What does it mean?

normalization

```
train.norm.df<-train.df

validation.norm.df<-validation.df

new_housing<-housing

values.preprocess<-preProcess(train.df[,1:12],method=c('center','scale'))

train.norm.df[,1:12]<-predict(values.preprocess,train.df[,1:12])
```

```
new_housing<-predict(values.preprocess,housing)

validation.norm.df[,1:12]<-predict(values.preprocess,validation.df[,1:12])
```

Using class package to predict the outcome, since class package accounts only for classifications.

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

for (i in 1:5){

  knn<-class::knn(train.norm.df[,1:12],validation.norm.df[,1:12] ,cl=train.no
rm.df$MEDV,k=i)

  accuracy[i,2]<-sqrt(sum((validation.norm.df$MEDV- as.numeric(levels(knn))[k
nn])^2)/nrow(validation.norm.df))

}

accuracy

##   k     RMSE
## 1 1 4.474129
## 2 2 5.719317
## 3 3 6.351673
## 4 4 6.244083
## 5 5 7.233546
```

K=1 eventhough provides better accuracy rate but it can fit the noise. k=4 has lower Rmse and can help us to find local structures in the dataset.

We will now perform regression based knn using FNN package and interpret the results

Using different k values, Since MEDV is a continous variable we use R^2 as an accuracy metrics

```
### function to compute r^2

rsq<-function(x,y){
  cor(x,y)^2
}

accuracy<-data.frame(k=seq(1,5,1),'R-Square'=rep(0,5))

for (i in 1:5){

  knn<-FNN::knn.reg(train.norm.df[,1:12],validation.norm.df[,1:12],y=train.no
rm.df$MEDV,k=i)$pred

  accuracy[i,2]<- rsq(validation.norm.df$MEDV,knn)
```

```
}

accuracy

##   k  R.Square
## 1 1 0.7741707
## 2 2 0.7632306
## 3 3 0.7848677
## 4 4 0.7867138
## 5 5 0.7590216
```

For different values of k, k=4 gives better accuracy on validation set and it well help us to find local structure in our data, so we choose k=4.

b. Predict the MEDV for a tract with the following information, using the best k

```
tract<- 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)


tract.norm<-predict(values.preprocess,tract)

tract.pred<-FNN::knn.reg(new_housing[,1:12],tract.norm,y=new_housing$MEDV,k=4
)

tract.pred

## Prediction:
## [1] 19.3
```

The new predicted value is 19.3

c. If we used the above k-NN algorithm to score the training data, what would be the error of the training set?

```
knn.train<-FNN::knn.reg(train.norm.df[,1:12],train.norm.df[,1:12],y=train.nor
m.df$MEDV,k=1)$pred

rsq(train.norm.df$MEDV,knn.train)

## [1] 1
```

The Rsquare 1 indicates that the accuracy is 100%, the error rate is 0.

d. Why is the validation data error overly optimistic compared to the error rate when applying this k-NN predictor to new data?

Solution

The validation data closely matches the data from training set because the model is derived from the original dataset. Also the validation data is a sample from data set so the error is overly optimistic.

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.

solution

For the large tracts of data it need a long time to calculate K-NN. The algorithm used in K-NN has to calculate the distance between the cases in the dataset and thus the operation become little timetaking. Also one more problem is when there is large sets of data then there are large number ofpredictors and the time increases for the algorithm to run as it has to find even more number if distances in the calculations it run.

Problem 8.1 [25 points]

```r
bank_dataset <- read_csv("C:\\Users\\kkbal\\OneDrive\\Desktop\\Neu\\data mini
ng\\Assignment-3\\UniversalBank.csv")

## Parsed with column specification:
## cols(
##   ID = col_double(),
##   Age = col_double(),
##   Experience = col_double(),
##   Income = col_double(),
##   `ZIP Code` = col_double(),
##   Family = col_double(),
##   CCAvg = col_double(),
##   Education = col_double(),
##   Mortgage = col_double(),
##   `Personal Loan` = col_double(),
##   `Securities Account` = col_double(),
##   `CD Account` = col_double(),
##   Online = col_double(),
##   CreditCard = col_double()
## )

head(bank_dataset)

## # A tibble: 6 x 14
##      ID   Age Experience Income `ZIP Code` Family CCAvg Education Mortgage
##   <dbl> <dbl>      <dbl>  <dbl>      <dbl>  <dbl> <dbl>     <dbl>    <dbl>
## 1     1    25          1     49      91107      4   1.6         1        0
## 2     2    45         19     34      90089      3   1.5         1        0
## 3     3    39         15     11      94720      1   1           1        0
## 4     4    35          9    100      94112      1   2.7         2        0
## 5     5    35          8     45      91330      4   1           2        0
## 6     6    37         13     29      92121      4   0.4         2      155
## # ... with 5 more variables: `Personal Loan` <dbl>, `Securities
## #   Account` <dbl>, `CD Account` <dbl>, Online <dbl>, CreditCard <dbl>
```

```r
bank_dataset$Personal.Loan = as.factor(bank_dataset$`Personal Loan`)
bank_dataset$Online = as.factor(bank_dataset$Online)
bank_dataset$CreditCard = as.factor(bank_dataset$CreditCard)
set.seed(1)
train.index <-
  sample(row.names(bank_dataset), 0.6 * dim(bank_dataset)[1])
test.index <- setdiff(row.names(bank_dataset), train.index)
train.df <- bank_dataset[train.index,]
test.df <- bank_dataset[test.index,]
train <- bank_dataset[train.index,]
test = bank_dataset[train.index, ]
```

a. Create a pivot table for the training data with Online as a column variable, CC as a row variable, and Loan as a secondary row variable. The values inside the table should convey the count. In R use functions melt() and cast(), or function table().

```r
melted_bank <-
  melt(train,
       id = c("CreditCard", "Personal.Loan","Online"))
recast_bank <- dcast(melted_bank, CreditCard + Personal.Loan ~ Online)

## Aggregation function missing: defaulting to length

colnames(recast_bank) <- c("CreditCard","Personal.Loan","Online.0","Online.1"
)
recast_bank[, c("CreditCard", "Personal.Loan", "Online.0","Online.1")]

##   CreditCard Personal.Loan Online.0 Online.1
## 1          0             0     9660    13428
## 2          0             1      948     1428
## 3          1             0     3984     5628
## 4          1             1      360      564
```

b) Consider the task of classifying a customer who owns a bank credit card and is actively using online banking services. Looking at the pivot table, what is the probability that this customer will accept the loan offer? [This is the probability of loan acceptance (Loan= 1) conditional on having a bank credit card (CC = 1) and being an active user of online banking services (Online = 1)].

```r
online_if_cc_and_personal_loan <-
  subset(recast_bank, CreditCard == 1 & Personal.Loan == 1)
sum(online_if_cc_and_personal_loan$Online.1)/ sum(subset(recast_bank, CreditC
ard == 1)$Online.1)

## [1] 0.09108527
```

c) Create two separate pivot tables for the training data. One will have Loan (rows) as a function of Online (columns) and the other will have Loan (rows) as a function of CC.

```r
melted_bank_dataset1 <-
  melt(train, id = c("CreditCard"), variable = "Online")
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

```r
melted_bank_dataset2 <-
  melt(train, id = c("Personal.Loan"), variable = "Online")
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

```r
recast_bank_dataset1 <-
  dcast(melted_bank_dataset1, CreditCard ~ Online)
```

```
## Aggregation function missing: defaulting to length
```

```r
recast_bank_dataset2 <-
  dcast(melted_bank_dataset2, Personal.Loan ~ Online)
```

```
## Aggregation function missing: defaulting to length
```

```r
table_credit_card <-
  recast_bank_dataset1[, c("CreditCard", "Online")]
table_personal_loan <-
  recast_bank_dataset2[, c("Personal.Loan", "Online")]
```

d) Compute the following quantities [P(A | B) means "the probability of A given B"]:

```r
table(train[,c("CreditCard","Personal.Loan")])
```

```
##             Personal.Loan
## CreditCard    0    1
##           0 1924  198
##           1  801   77
```

```r
82/(82+209)
```

```
## [1] 0.2817869
```

```r
table(train[,c("Online","Personal.Loan")])
```

```
##         Personal.Loan
## Online    0    1
##       0 1137  109
##       1 1588  166
```

```r
180/(180+111)
```

```
## [1] 0.6185567
```

```r
table(train[,c("Personal.Loan")])
```

```
##
##    0    1
## 2725  275
```

```
(291)/2709

## [1] 0.1074197

(786)/(786+1923)

## [1] 0.290144

(1612)/(1612+1097)

## [1] 0.5950535

2709/(291+2709)

## [1] 0.903
```

e) Use the quantities computed above to compute the naive Bayes probability P(Loan = 1 | CC = 1, Online = 1).

```
(0.1074197 * 0.2817869 * 0.6185567)/((0.1074197 * 0.2817869 * 0.6185567)+(0.2
90144*0.903*0.5950535))

## [1] 0.107219
```

f) Compare this value with the one obtained from the pivot table in (b). Which is a more accurate estimate?

10.7% are very similar to the 9.9% the difference between the exact method and the naive-baise method is the exact method would need the the exact same independent variable classifications to predict, where the naive bayes method does not.

g) Which of the entries in this table are needed for computing P(Loan = 1 | CC = 1, Online =1)? In R, run naive Bayes on the data. Examine the model output on training data, and find the entry that corresponds to P(Loan = 1 | CC = 1, Online = 1). Compare this to the number you obtained in (e).

```
naive.train = train[,c("Online","Personal.Loan","CreditCard")]
naivebayes = naiveBayes(Personal.Loan~.,data=naive.train)
naivebayes

##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##          0          1
## 0.90833333 0.09166667
##
## Conditional probabilities:
##    Online
## Y           0         1
```

```
##   0 0.4172477 0.5827523
##   1 0.3963636 0.6036364
##
##    CreditCard
## Y          0        1
##   0 0.706055 0.293945
##   1 0.720000 0.280000
```

the naive bayes is the exact same output we recieved in the previous methods.

Question 8.2

```
Accidents <- read.csv("C:\\Users\\kkbal\\OneDrive\\Desktop\\Neu\\data mining\
\Assignment-3\\accidentsFull.csv")
Accidents$INJURY <- ifelse(Accidents$MAX_SEV_IR>0, "yes", "no")
```

a) 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?

```
Prob <- table(Accidents$INJURY)
Final =  scales::percent(Prob["yes"]/(Prob["yes"]+Prob["no"]),0.01)
Final

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

Since probability of Injury is higher~51% therefore we should predict injury in case of an accident

b) 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.

```
for (i in c(1:dim(Accidents)[2])){
  Accidents[,i] <- as.factor(Accidents[,i])
}

first12 <- Accidents[1:12, c(16,19,25)]
first12

##    TRAF_CON_R WEATHER_R INJURY
## 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
```

```r
table(first12$TRAF_CON_R, first12$WEATHER_R, first12$INJURY, dnn = c("TRAF_CON_R","WEATHER_R", "INJURY"))
```

```
## , , INJURY = no
## 
##           WEATHER_R
## TRAF_CON_R 1 2
##          0 1 5
##          1 1 1
##          2 1 0
## 
## , , INJURY = yes
## 
##           WEATHER_R
## TRAF_CON_R 1 2
##          0 2 1
##          1 0 0
##          2 0 0
```

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

```
## [1] 0.6666667
```

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

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

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

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

```r
# P(Injury=yes| WEATHER_R = 2, TRAF_CON_R =0):
numerator4 <- 1/3 * 3/12
denominator4 <- 6/12
prob4 <- numerator4/denominator4
prob4
```

```
## [1] 0.1666667
```

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

## [1] 0

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

## [1] NaN
```

iii) When the cutoff is 0.5, from the above calculations we see that only when WEATHER_R is 1 and TRAF_CON_R is 0 we will get an INJURY

```r
first12$predicted <- ifelse(first12$TRAF_CON_R == 0 & first12$WEATHER_R == 1,
"yes", "no")
first12

##    TRAF_CON_R WEATHER_R INJURY predicted
## 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

Probability <- 2/3 * 0/3 * 3/12
Probability

## [1] 0

Naive_1<- naiveBayes(INJURY ~ TRAF_CON_R + WEATHER_R, first12)
predicted_prob <- predict(Naive_1, newdata = first12, type = "raw")

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

##  cutoff = 0.5
predicted_class <- c("Yes", "No", "No", "No", "Yes", "No", "No", "Yes", "No",
"No", "No", "No")
df <- data.frame(actual = first12$INJURY, predicted = predicted_class, predic
ted_prob)
df
```

```
##    actual predicted        no          yes
## 1     yes       Yes 0.5000000 0.5000000000
## 2      no        No 0.8000000 0.2000000000
## 3      no        No 0.9992506 0.0007494379
## 4      no        No 0.9970090 0.0029910269
## 5      no       Yes 0.5000000 0.5000000000
## 6     yes        No 0.8000000 0.2000000000
## 7      no        No 0.8000000 0.2000000000
## 8     yes       Yes 0.5000000 0.5000000000
## 9      no        No 0.8000000 0.2000000000
## 10     no        No 0.8000000 0.2000000000
## 11     no        No 0.8000000 0.2000000000
## 12     no        No 0.9940358 0.0059642147
```

The errors that appear when running the naive Bayes on this sample set are nothing to really worry about, they just mean that these parameter do very poorly when classified. The classification is equivalent. The ranking (= ordering) of observations are also equivalent.

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

```
set.seed(571)
train.index <- sample(c(1:dim(Accidents)[1]), dim(Accidents)[1]*0.6)
train <- Accidents[train.index,]
valid <- Accidents[-train.index,]
```

i) We can use the predictors that describe the calendar time or road conditions: HOUR_I_R ALIGN_I WRK_ZONE WKDY_I_R INT_HWY LGTCON_I_R PROFIL_I_R SPD_LIM SUR_CON TRAF_CON_R TRAF_WAY WEATHER_R.

```
head(Accidents)
```

```
##   ï..HOUR_I_R ALCHL_I ALIGN_I STRATUM_R WRK_ZONE WKDY_I_R INT_HWY
## 1           0       2       2         1        0        1       0
## 2           1       2       1         0        0        1       1
## 3           1       2       1         0        0        1       0
## 4           1       2       1         1        0        0       0
## 5           1       1       1         0        0        1       0
## 6           1       2       1         1        0        1       0
##   LGTCON_I_R MANCOL_I_R PED_ACC_R RELJCT_I_R REL_RWY_R PROFIL_I_R SPD_LIM
## 1          3          0         0          1         0          1      40
## 2          3          2         0          1         1          1      70
## 3          3          2         0          1         1          1      35
## 4          3          2         0          1         1          1      35
## 5          3          2         0          0         1          1      25
## 6          3          0         0          1         0          1      70
##   SUR_COND TRAF_CON_R TRAF_WAY VEH_INVL WEATHER_R INJURY_CRASH NO_INJ_I
## 1        4          0        3        1         1            1        1
## 2        4          0        3        2         2            0        0
## 3        4          1        2        2         2            0        0
```

```
## 4        4          1        2        2         1            0        0
## 5        4          0        2        3         1            0        0
## 6        4          0        2        1         2            1        1
##   PRPTYDMG_CRASH FATALITIES MAX_SEV_IR INJURY
## 1              0          0          1    yes
## 2              1          0          0     no
## 3              1          0          0     no
## 4              1          0          0     no
## 5              1          0          0     no
## 6              0          0          1    yes
```

```
vars <- c("INJURY", "ï..HOUR_I_R",  "ALIGN_I" ,"WRK_ZONE",  "WKDY_I_R",
          "INT_HWY",  "LGTCON_I_R", "PROFIL_I_R", "SPD_LIM", "SUR_COND",
          "TRAF_CON_R",   "TRAF_WAY",   "WEATHER_R")
train_nb <- naiveBayes(INJURY ~ ., train[,vars])
train_nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        no       yes
## 0.4919989 0.5080011
##
## Conditional probabilities:
##      ï..HOUR_I_R
## Y             0         1
##   no  0.5663347 0.4336653
##   yes 0.5762620 0.4237380
##
##      ALIGN_I
## Y             1         2
##   no  0.8719884 0.1280116
##   yes 0.8650541 0.1349459
##
##      WRK_ZONE
## Y              0          1
##   no  0.97510440 0.02489560
##   yes 0.98016645 0.01983355
##
##      WKDY_I_R
## Y             0         1
##   no  0.2187600 0.7812400
##   yes 0.2415027 0.7584973
##
##      INT_HWY
```

```
## Y                0            1            9
##   no  0.8500642467 0.1492932862 0.0006424671
##   yes 0.8634207047 0.1358792875 0.0007000078
##
##      LGTCON_I_R
## Y             1         2         3
##   no  0.6903309 0.1249598 0.1847093
##   yes 0.6950299 0.1151902 0.1897799
##
##      PROFIL_I_R
## Y             0         1
##   no  0.7550594 0.2449406
##   yes 0.7632418 0.2367582
##
##      SPD_LIM
## Y                5           10           15           20           25
##   no  0.0001606168 0.0004818503 0.0048988114 0.0081111468 0.1082557019
##   yes 0.0001555573 0.0005444505 0.0038889321 0.0039667107 0.0914676830
##      SPD_LIM
## Y               30           35           40           45           50
##   no  0.0859299711 0.1926598137 0.0964503694 0.1570831995 0.0409572759
##   yes 0.0900676674 0.2156023956 0.1071789687 0.1554795053 0.0386559851
##      SPD_LIM
## Y               55           60           65           70           75
##   no  0.1611789271 0.0334885962 0.0656922583 0.0387086412 0.0059428204
##   yes 0.1531461461 0.0448782764 0.0607451194 0.0271447461 0.0070778564
##
##      SUR_COND
## Y               1           2           3           4           9
##   no  0.775778991 0.174349502 0.017266303 0.028188243 0.004416961
##   yes 0.815664618 0.154701719 0.010111223 0.014622385 0.004900054
##
##      TRAF_CON_R
## Y             0         1         2
##   no  0.6608577 0.1877610 0.1513813
##   yes 0.6164735 0.2231469 0.1603796
##
##      TRAF_WAY
## Y              1          2          3
##   no  0.57741728 0.37279152 0.04979120
##   yes 0.56646185 0.39122657 0.04231158
##
##      WEATHER_R
## Y             1         2
##   no  0.8409894 0.1590106
##   yes 0.8719764 0.1280236
```

```
confusionMatrix(train$INJURY, predict(train_nb, train[,vars]), positive = "ye
s")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  5460 6992
##        yes 4557 8300
##
##                Accuracy : 0.5437
##                  95% CI : (0.5375, 0.5498)
##     No Information Rate : 0.6042
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0843
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5428
##             Specificity : 0.5451
##          Pos Pred Value : 0.6456
##          Neg Pred Value : 0.4385
##              Prevalence : 0.6042
##          Detection Rate : 0.3279
##    Detection Prevalence : 0.5080
##       Balanced Accuracy : 0.5439
##
##        'Positive' Class : yes
##
```

```r
error=1-.544
percentage_error=scales::percent(error,0.01)
paste("Overall Error: ",percentage_error)
```

```
## [1] "Overall Error:  45.60%"
```

```r
# validation
confusionMatrix(valid$INJURY, predict(train_nb, valid[, vars]), positive = "y
es")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  3627 4642
##        yes 3138 5467
##
##                Accuracy : 0.5389
##                  95% CI : (0.5314, 0.5465)
##     No Information Rate : 0.5991
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0742
```

```
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5408
##             Specificity : 0.5361
##          Pos Pred Value : 0.6353
##          Neg Pred Value : 0.4386
##              Prevalence : 0.5991
##          Detection Rate : 0.3240
##    Detection Prevalence : 0.5100
##       Balanced Accuracy : 0.5385
##
##        'Positive' Class : yes
##
```

```
val_error=1-.5389
val_error_perc=scales::percent(val_error,0.01)
paste("Overall Error: ",val_error_perc)
```

```
## [1] "Overall Error:  46.11%"
```

```
improvement=val_error-error
paste("The percent improvement is ",scales::percent(improvement,0.01))
```

```
## [1] "The percent improvement is  0.51%"
```

```
options(digits = 2)
train_nb
```

```
##
## 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:
##      ï..HOUR_I_R
## Y        0    1
##   no  0.57 0.43
##   yes 0.58 0.42
##
##      ALIGN_I
## Y        1    2
##   no  0.87 0.13
##   yes 0.87 0.13
##
```

```
##      WRK_ZONE
## Y         0     1
##   no  0.975 0.025
##   yes 0.980 0.020
##
##      WKDY_I_R
## Y        0    1
##   no  0.22 0.78
##   yes 0.24 0.76
##
##      INT_HWY
## Y           0       1       9
##   no  0.85006 0.14929 0.00064
##   yes 0.86342 0.13588 0.00070
##
##      LGTCON_I_R
## Y        1    2    3
##   no  0.69 0.12 0.18
##   yes 0.70 0.12 0.19
##
##      PROFIL_I_R
## Y        0    1
##   no  0.76 0.24
##   yes 0.76 0.24
##
##      SPD_LIM
## Y           5      10      15      20      25      30      35      40
##   no  0.00016 0.00048 0.00490 0.00811 0.10826 0.08593 0.19266 0.09645
##   yes 0.00016 0.00054 0.00389 0.00397 0.09147 0.09007 0.21560 0.10718
##      SPD_LIM
## Y          45      50      55      60      65      70      75
##   no  0.15708 0.04096 0.16118 0.03349 0.06569 0.03871 0.00594
##   yes 0.15548 0.03866 0.15315 0.04488 0.06075 0.02714 0.00708
##
##      SUR_COND
## Y          1      2      3      4      9
##   no  0.7758 0.1743 0.0173 0.0282 0.0044
##   yes 0.8157 0.1547 0.0101 0.0146 0.0049
##
##      TRAF_CON_R
## Y        0    1    2
##   no  0.66 0.19 0.15
##   yes 0.62 0.22 0.16
##
##      TRAF_WAY
## Y         1     2     3
##   no  0.577 0.373 0.050
##   yes 0.566 0.391 0.042
##
##      WEATHER_R
```

```
## Y        1    2
##   no  0.84 0.16
##   yes 0.87 0.13
```

We do not actually get a probability of zero for no injury in accidents under the speed limit of 5 as there is a single accident out of all the records, it makes sense that the probability is quite close to 0.