Homework 5
Introduction to Data Analysis and Mining
Spring 2018
CSCI-B 365

Instructor: Hasan Kurban

April 13, 2018

Directions

Please follow the syllabus guidelines in turning in your homework. I am providing the LaTeX of this document too. This homework is due Sunday, April 8, 2018 10:00p.m. **OBSERVE THE TIME**. Absolutely no homework will be accepted after that time. All the work should be your own.
All the work herein is solely mine.

K-Nearest Neighbors (KNN) Algorithm in Theory

1: **ALGORITHM** `K-nearest neighbors`
2: **INPUT**

- training data Δ
- test data Δ'
- distance metric d, i.e., $d : \Delta^2 \rightarrow \mathbb{R}_{\geq 0}$
- integer k: nearest neighbors number

3: **OUTPUT**

- class label of each $z \in \Delta'$

4: **for** $z = (\mathbf{x}', y') \in \Delta'$ **do**
5: Compute $d(\mathbf{x}, \mathbf{x}')$, the distance between z and every example $(\mathbf{x}, y) \in \Delta$
6: Select $\Delta_z \subseteq \Delta$, the set of closest k training examples to z
7: Voting:

- majority voting: $y' = \text{argmax}_v \sum_{(\mathbf{x_i}, y_i) \in \Delta_z} I(v = y_i)$
- distance-weighted voting: $y' = \text{argmax}_v \sum_{(\mathbf{x_i}, y_i) \in \Delta_z} w_i \times I(v = y_i)$ where $w_i = \frac{1}{d(\mathbf{x}', \mathbf{x_i})^2}$

8: **end for**

K-Fold Cross Validation for Model Selection

1: **ALGORITHM** `k-fold cross validatiaon`
2: **INPUT**
- training data $\Delta = (\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)$
- set of parameter values Θ
- learning algorithm $\mathcal{H}$
- integer k

3: **OUTPUT**
- $\theta^* = \text{argmin}_\theta[error(\theta)]$
- $h_{\theta^*} = \mathcal{H}(\Delta; \theta^*)$

4: Randomly partition Δ into $\Delta_1, \ldots, \Delta_k$
5: $\star\star\star$ $\Delta_1 \cup \Delta_2 \ldots \cup \Delta_k = \Delta$ and $\Delta_i \cap \Delta_j = \varnothing$ for $i \neq j \in [1, 2, \ldots, k]$
6: **for** $\theta \in \Theta$ **do**
7: **for** $i = 1 \ldots k$ **do**
8: $\star\star\star$ Train a model for each training set
9: $h_{i,\theta} = \mathcal{H}(\Delta \setminus \Delta_i; \theta)$
10: **end for**
11: $\star\star\star$ Use the trained models over Δ_i (test data sets) to evaluate the models for each parameter
12: $\text{error}(\theta) = \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}_{\Delta_i}(h_{i,\theta})$
13: **end for**

In this homework, you are asked to train several classifiers using k-nearest neighbors (KNN) and naive bayes algorithms over car evaluation and credit approval data sets. The links for the data sets are provided below:

- Car Evaluation Data Set
- Credit Approval Data Set

README
Problem 1: I did my own implementation for K-fold. Also, everytime this file is sourced, it creates different 20 files. So this might affected the result in Problem 2 and 3.
Problem 2: I created a separate file called myknn.R and it is the actual implementation of knn. I feel it is easier for me to create only on file for knn and use this one file in both 2.1 and 2.2.
Problem 2.2: If you run on R, it might take a while for the program to run to get the final result.
Problem 3: There's 5 models for each data set and each model will have a average error rate.

Problem 1: K-Fold Cross Validation [20 points]

Create 5 training and 5 test data sets from each data set using 5-fold cross validation and save these 20 files. You will use these data sets to answer the rest of the questions. You are allowed to use R packages for k- fold cross validation. However, *students who implement it will receive* 15 *extra points from this question.*

Listing 1: K-Fold Cross Validation

```
library(data.table)

nFolds <- 5

datasetname = c("car", "credit")
datasets_link =
```

```
  c("https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.data",
    "https://archive.ics.uci.edu/ml/machine-learning-databases/credit-screening/crx.data")

for (i in 1 : length(datasetname)) {
  dataset = fread(datasets_link[i])

  # generate array containing fold-number for each sample (row)
  folds <- sample(1:nFolds, nrow(dataset), replace=T)

  for (k in 1:nFolds) {
   # actual split of the data
   fold <- which(folds == k)
   dataset.train <- dataset[-fold,]
   dataset.test <- dataset[fold,]
   write.table(dataset.train, file =
                 paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/",
                        datasetname[i], ".training.", k), quote = FALSE,
               row.names = F, col.names = F)
   write.table(dataset.test, file =
                 paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/",
                        datasetname[i], ".test.", k), quote = FALSE,
               row.names = F, col.names = F)
 }
}
```

Problem 2: K-Nearest Neighbors (KNN)[40 points]

2.1 Implement KNN algorithm with two different distance functions. You can either use existing distance functions, i.e., Euclidean or design your own.

Listing 2: KNN Implementation

```
###################################
# This file is the actual implementation of knn.
# I feel it is easier for me to create only on file for knn
# and use this file in both 2.1 and 2.2.
#####################################

library(data.table)


#first distance function: euclidean
dist.euclidean<- function(v1, v2) {
  sqrt(sum((v1-v2)^2))
}


#second distance function: manhattan
dist.manhattan<- function(v1, v2){
  sum(abs(v1-v2))
}

myknn <- function(train_data, test_data, k, dist_func=dist.euclidean){
  distances = rep(0, nrow(train_data))
  num_col = ncol(train_data)
  #the class of training data
```

```
  train_cl = unlist(train_data[,num_col:num_col], use.names=F)
  #the real class of test data
  test_cl = unlist(test_data[, num_col:num_col], use.names=F)
  # the predicted class of test data
  test_cl_predict = rep(0, nrow(test_data))

  print(paste("Total number of training data is", nrow(train_data)))
  print(paste("Total number of test data is", nrow(test_data)))

  for(i in 1: nrow(test_data)) {
     if (i %% 100 == 1) {
       #progress checking
       print(paste("progress checking: predicting for test_data", i))
     }
     test_data_point = unlist(test_data[i, 1:(num_col-1)], use.names=F)
    for (j in 1:nrow(train_data)) {
      train_data_point = unlist(train_data[j, 1:(num_col-1)], use.names=F)

      distances[j] = dist_func(test_data_point, train_data_point)
    }

    indices <- which(distances <= sort(distances, decreasing=F)[k], arr.ind=TRUE)

    #majority voting;
    labels = rep(0, length(indices))
    for (t in 1: length(indices)) {
      labels[t] = train_cl[indices[t]]
    }
    test_cl_predict[i] = names(which.max(table(labels)))
  }
  error_rate = sum(test_cl_predict != test_cl) / nrow(test_data)
  print(paste("error rate is", error_rate))
  return (error_rate)
}
```

Listing 3: KNN

```
library(data.table)

#import myknn.R file
#if the path changed, remember to point to the new path before running the script
if(!exists("myknn", mode="function"))
  source("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/myknn.R")


####
## test run over car dataset (partial)
####
car_train_data =
  fread("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/car.training.1")
car_test_data =
  fread("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/car.test.1")

car_train_data$V1 = as.numeric(factor(car_train_data$V1))
car_train_data$V2 = as.numeric(factor(car_train_data$V2))
car_train_data$V3 = as.numeric(factor(car_train_data$V3))
car_train_data$V4 = as.numeric(factor(car_train_data$V4))
```

```
car_train_data$V5 = as.numeric(factor(car_train_data$V5))
car_train_data$V6 = as.numeric(factor(car_train_data$V6))
car_train_data$V7 = as.numeric(factor(car_train_data$V7))

car_test_data$V1 = as.numeric(factor(car_test_data$V1))
car_test_data$V2 = as.numeric(factor(car_test_data$V2))
car_test_data$V3 = as.numeric(factor(car_test_data$V3))
car_test_data$V4 = as.numeric(factor(car_test_data$V4))
car_test_data$V5 = as.numeric(factor(car_test_data$V5))
car_test_data$V6 = as.numeric(factor(car_test_data$V6))
car_test_data$V7 = as.numeric(factor(car_test_data$V7))

error_rate = myknn(car_train_data, car_test_data, 5)
print(paste("The error rate is", error_rate,
            "for k=5","on car dataset (partial)"))



####
## test run over credit dataset (partial)
####

credit_train_data =
  fread("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/credit.training.1")
credit_test_data =
  fread("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/credit.test.1")

credit_train_data$V1 = as.numeric(factor(credit_train_data$V1))
credit_train_data$V2 = as.numeric(factor(credit_train_data$V2))
credit_train_data$V3 = as.numeric(factor(credit_train_data$V3))
credit_train_data$V4 = as.numeric(factor(credit_train_data$V4))
credit_train_data$V5 = as.numeric(factor(credit_train_data$V5))
credit_train_data$V6 = as.numeric(factor(credit_train_data$V6))
credit_train_data$V7 = as.numeric(factor(credit_train_data$V7))
credit_train_data$V8 = as.numeric(factor(credit_train_data$V8))
credit_train_data$V9 = as.numeric(factor(credit_train_data$V9))
credit_train_data$V10 = as.numeric(factor(credit_train_data$V10))
credit_train_data$V11 = as.numeric(factor(credit_train_data$V11))
credit_train_data$V12 = as.numeric(factor(credit_train_data$V12))
credit_train_data$V13 = as.numeric(factor(credit_train_data$V13))
credit_train_data$V14 = as.numeric(factor(credit_train_data$V14))
credit_train_data$V15 = as.numeric(factor(credit_train_data$V15))
credit_train_data$V16 = as.numeric(factor(credit_train_data$V16))

credit_test_data$V1 = as.numeric(factor(credit_test_data$V1))
credit_test_data$V2 = as.numeric(factor(credit_test_data$V2))
credit_test_data$V3 = as.numeric(factor(credit_test_data$V3))
credit_test_data$V4 = as.numeric(factor(credit_test_data$V4))
credit_test_data$V5 = as.numeric(factor(credit_test_data$V5))
credit_test_data$V6 = as.numeric(factor(credit_test_data$V6))
credit_test_data$V7 = as.numeric(factor(credit_test_data$V7))
credit_test_data$V8 = as.numeric(factor(credit_test_data$V8))
credit_test_data$V9 = as.numeric(factor(credit_test_data$V9))
credit_test_data$V10 = as.numeric(factor(credit_test_data$V10))
credit_test_data$V11 = as.numeric(factor(credit_test_data$V11))
credit_test_data$V12 = as.numeric(factor(credit_test_data$V12))
credit_test_data$V13 = as.numeric(factor(credit_test_data$V13))
credit_test_data$V14 = as.numeric(factor(credit_test_data$V14))
credit_test_data$V15 = as.numeric(factor(credit_test_data$V15))
```

```
credit_test_data$V16 = as.numeric(factor(credit_test_data$V16))

error_rate = myknn(credit_train_data, credit_test_data, 5)
print(paste("The error rate is", error_rate,
            "for k=5","on credit dataset (partial)"))
```

2.2 Use the data sets created in problem 1 to determine the optimal k over each data set for KNN algorithm. First, pick 5 different k values and then calculate average error rate of KNN classifiers over test data tests for each k to find the optimal k for each data set and distance function. Report the average error rate for each k, distance function and data set. What are the optimal k and distance function for each data set? PLEASE SEE myknn.R FOR THE IMPLEMENTATION OF KNN.

Listing 4: Car Evaluation Data Set

```
library(data.table)

#import myknn.R file
#if the path changed, remember to point to the new path before running the script
if(!exists("myknn", mode="function"))
  source("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/myknn.R")


k_sets = c(3,5,7,9,11)
avg_error_rate = rep(0,5)
dist_func_name = c("euclidean", "manhattan")
dist_funcs = c(dist.euclidean, dist.manhattan)
for(d in 1:length(dist_funcs)) {
  for (i in 1:5) {
    print(paste("Run knn for k=", k_sets[i]))
    error_rate_sum = 0

    for (j in 1:5) {

      train_data = fread(
        paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/car.training.", j))
      test_data = fread(
        paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/car.test.",j))

      train_data$V1 = as.numeric(factor(train_data$V1))
      train_data$V2 = as.numeric(factor(train_data$V2))
      train_data$V3 = as.numeric(factor(train_data$V3))
      train_data$V4 = as.numeric(factor(train_data$V4))
      train_data$V5 = as.numeric(factor(train_data$V5))
      train_data$V6 = as.numeric(factor(train_data$V6))
      train_data$V7 = as.numeric(factor(train_data$V7))

      test_data$V1 = as.numeric(factor(test_data$V1))
      test_data$V2 = as.numeric(factor(test_data$V2))
      test_data$V3 = as.numeric(factor(test_data$V3))
      test_data$V4 = as.numeric(factor(test_data$V4))
      test_data$V5 = as.numeric(factor(test_data$V5))
      test_data$V6 = as.numeric(factor(test_data$V6))
      test_data$V7 = as.numeric(factor(test_data$V7))

      error_rate_sum = error_rate_sum +
        myknn(train_data, test_data, k_sets[i], dist_funcs[d])
    }
```

```
    avg_error_rate[i] = error_rate_sum / 5
    print(paste("the average error rate is", avg_error_rate[i],
                " when k is", k_sets[i], " and distance function is", dist_func_name[d]))
  }

  optimal_k = k_sets[which.min(avg_error_rate)]
  print(paste("The optimal k is", optimal_k, "; the minimum avg error rate is",
              min(avg_error_rate), "when distance function is", dist_func_name[d]))
}



#The optimal k is 3 ; the minimum avg error rate is
#0.058512808942294 when distance function is manhattan
```

Listing 5: Credit Approval Data Set

```
library(data.table)

#import myknn.R file
#if the path changed, remember to point to the new path before running the script
if(!exists("myknn", mode="function"))
  source("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/myknn.R")

k_sets = c(3,5,7,9,11)
avg_error_rate = rep(0,5)
dist_func_name = c("euclidean", "manhattan")
dist_funcs = c(dist.euclidean, dist.manhattan)
for (d in 1:length(dist_funcs)){
  for (i in 1:5) {
    print(paste("Run knn for k=", k_sets[i]))
    error_rate_sum = 0

    for (j in 1:5) {
      credit_train_data = fread(
        paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/credit.training.",j))
      credit_test_data = fread(
        paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/credit.test.",j))

      credit_train_data$V1 = as.numeric(factor(credit_train_data$V1))
      credit_train_data$V2 = as.numeric(factor(credit_train_data$V2))
      credit_train_data$V3 = as.numeric(factor(credit_train_data$V3))
      credit_train_data$V4 = as.numeric(factor(credit_train_data$V4))
      credit_train_data$V5 = as.numeric(factor(credit_train_data$V5))
      credit_train_data$V6 = as.numeric(factor(credit_train_data$V6))
      credit_train_data$V7 = as.numeric(factor(credit_train_data$V7))
      credit_train_data$V8 = as.numeric(factor(credit_train_data$V8))
      credit_train_data$V9 = as.numeric(factor(credit_train_data$V9))
      credit_train_data$V10 = as.numeric(factor(credit_train_data$V10))
      credit_train_data$V11 = as.numeric(factor(credit_train_data$V11))
      credit_train_data$V12 = as.numeric(factor(credit_train_data$V12))
      credit_train_data$V13 = as.numeric(factor(credit_train_data$V13))
      credit_train_data$V14 = as.numeric(factor(credit_train_data$V14))
      credit_train_data$V15 = as.numeric(factor(credit_train_data$V15))
      credit_train_data$V16 = as.numeric(factor(credit_train_data$V16))

      credit_test_data$V1 = as.numeric(factor(credit_test_data$V1))
      credit_test_data$V2 = as.numeric(factor(credit_test_data$V2))
```

```
      credit_test_data$V3 = as.numeric(factor(credit_test_data$V3))
      credit_test_data$V4 = as.numeric(factor(credit_test_data$V4))
      credit_test_data$V5 = as.numeric(factor(credit_test_data$V5))
      credit_test_data$V6 = as.numeric(factor(credit_test_data$V6))
      credit_test_data$V7 = as.numeric(factor(credit_test_data$V7))
      credit_test_data$V8 = as.numeric(factor(credit_test_data$V8))
      credit_test_data$V9 = as.numeric(factor(credit_test_data$V9))
      credit_test_data$V10 = as.numeric(factor(credit_test_data$V10))
      credit_test_data$V11 = as.numeric(factor(credit_test_data$V11))
      credit_test_data$V12 = as.numeric(factor(credit_test_data$V12))
      credit_test_data$V13 = as.numeric(factor(credit_test_data$V13))
      credit_test_data$V14 = as.numeric(factor(credit_test_data$V14))
      credit_test_data$V15 = as.numeric(factor(credit_test_data$V15))
      credit_test_data$V16 = as.numeric(factor(credit_test_data$V16))

      error_rate_sum = error_rate_sum +
        myknn(credit_train_data, credit_test_data, k_sets[i], dist_funcs[d])
    }
    avg_error_rate[i] = error_rate_sum / 5
    print(paste("the average error rate is", avg_error_rate[i],
                " when k is", k_sets[i], " and distance function is", dist_func_name[d]))
  }
  optimal_k = k_sets[which.min(avg_error_rate)]
  print(paste("The optimal k is", optimal_k, "; the minimum avg error rate is",
              min(avg_error_rate), "when distance function is", dist_func_name[d]))
}



#The optimal k is 11 ; the minimum avg error rate is
#0.379497680076954 when distance function is manhattan.
```

2.3 Use the KNN package in R to validate your results from question 2.2.

Listing 6: Car Evaluation Data Set

```
library(class)

k_sets = c(3,5,7,9,11)
avg_error_rate = rep(0,5)
for (i in 1:5) {
  print(paste("Run knn for k=", k_sets[i]))
  error_rate_sum = 0

  for (j in 1:5) {

    train_data = fread(
      paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/car.training.", j))
    test_data = fread(
      paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/car.test.",j))

    train_data$V1 = as.numeric(factor(train_data$V1))
    train_data$V2 = as.numeric(factor(train_data$V2))
    train_data$V3 = as.numeric(factor(train_data$V3))
    train_data$V4 = as.numeric(factor(train_data$V4))
    train_data$V5 = as.numeric(factor(train_data$V5))
    train_data$V6 = as.numeric(factor(train_data$V6))
    train_data$V7 = as.numeric(factor(train_data$V7))
```

```
    test_data$V1 = as.numeric(factor(test_data$V1))
    test_data$V2 = as.numeric(factor(test_data$V2))
    test_data$V3 = as.numeric(factor(test_data$V3))
    test_data$V4 = as.numeric(factor(test_data$V4))
    test_data$V5 = as.numeric(factor(test_data$V5))
    test_data$V6 = as.numeric(factor(test_data$V6))
    test_data$V7 = as.numeric(factor(test_data$V7))

    num_col = ncol(train_data)
    train = train_data[ ,1:(num_col-1)]
    test = test_data[, 1:(num_col-1)]
    knn.res = knn(train, test, train_data$V7, k_sets[i])

    error_rate_sum = error_rate_sum +  sum(test_data$V7 == knn.res)/ nrow(test_data)
  }
  avg_error_rate[i] = error_rate_sum / 5
  print(paste("the average error rate is", avg_error_rate[i], " when k is", k_sets[i]))
}
optimal_k = k_sets[which.min(avg_error_rate)]
print(paste("The optimal k is", optimal_k,
            "; the minimum avg error rate is",min(avg_error_rate)))


#The optimal k is 11 ; the minimum avg error rate is 0.805088623609318
```

Listing 7: Credit Approval Data Set

```
library(class)

k_sets = c(3,5,7,9,11)
avg_error_rate = rep(0,5)
for (i in 1:5) {
  print(paste("Run knn for k=", k_sets[i]))
  error_rate_sum = 0

  for (j in 1:5) {

    credit_train_data = fread(
      paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/credit.training.",j))
    credit_test_data = fread(
      paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/credit.test.",j))

    credit_train_data$V1 = as.numeric(factor(credit_train_data$V1))
    credit_train_data$V2 = as.numeric(factor(credit_train_data$V2))
    credit_train_data$V3 = as.numeric(factor(credit_train_data$V3))
    credit_train_data$V4 = as.numeric(factor(credit_train_data$V4))
    credit_train_data$V5 = as.numeric(factor(credit_train_data$V5))
    credit_train_data$V6 = as.numeric(factor(credit_train_data$V6))
    credit_train_data$V7 = as.numeric(factor(credit_train_data$V7))
    credit_train_data$V8 = as.numeric(factor(credit_train_data$V8))
    credit_train_data$V9 = as.numeric(factor(credit_train_data$V9))
    credit_train_data$V10 = as.numeric(factor(credit_train_data$V10))
    credit_train_data$V11 = as.numeric(factor(credit_train_data$V11))
    credit_train_data$V12 = as.numeric(factor(credit_train_data$V12))
    credit_train_data$V13 = as.numeric(factor(credit_train_data$V13))
    credit_train_data$V14 = as.numeric(factor(credit_train_data$V14))
    credit_train_data$V15 = as.numeric(factor(credit_train_data$V15))
```

```
    credit_train_data$V16 = as.numeric(factor(credit_train_data$V16))

    credit_test_data$V1 = as.numeric(factor(credit_test_data$V1))
    credit_test_data$V2 = as.numeric(factor(credit_test_data$V2))
    credit_test_data$V3 = as.numeric(factor(credit_test_data$V3))
    credit_test_data$V4 = as.numeric(factor(credit_test_data$V4))
    credit_test_data$V5 = as.numeric(factor(credit_test_data$V5))
    credit_test_data$V6 = as.numeric(factor(credit_test_data$V6))
    credit_test_data$V7 = as.numeric(factor(credit_test_data$V7))
    credit_test_data$V8 = as.numeric(factor(credit_test_data$V8))
    credit_test_data$V9 = as.numeric(factor(credit_test_data$V9))
    credit_test_data$V10 = as.numeric(factor(credit_test_data$V10))
    credit_test_data$V11 = as.numeric(factor(credit_test_data$V11))
    credit_test_data$V12 = as.numeric(factor(credit_test_data$V12))
    credit_test_data$V13 = as.numeric(factor(credit_test_data$V13))
    credit_test_data$V14 = as.numeric(factor(credit_test_data$V14))
    credit_test_data$V15 = as.numeric(factor(credit_test_data$V15))
    credit_test_data$V16 = as.numeric(factor(credit_test_data$V16))

    num_col = ncol(train_data)
    train = train_data[ ,1:(num_col-1)]
    test = test_data[, 1:(num_col-1)]
    knn.res = knn(train, test, train_data$V7, k_sets[i])

    error_rate_sum = error_rate_sum +  sum(test_data$V7 == knn.res)/ nrow(test_data)
  }
  avg_error_rate[i] = error_rate_sum / 5
  print(paste("the average error rate is", avg_error_rate[i], " when k is", k_sets[i]))
}

optimal_k = k_sets[which.min(avg_error_rate)]
print(paste("The optimal k is", optimal_k,
            "; the minimum avg error rate is",min(avg_error_rate)))



#The optimal k is 11 ; the minimum avg error rate is 0.797050147492625.
```

The average error rates for the KNN package are a little different from the results in my KNN implementation in 2.2. My implementation has better results because the better method used to calculate the distance is Manhattan distance, whereas the build in function for KNN package is Euclidean distance.

Problem 3: Naive Bayes Classifier vs. K-Nearest Neighbors [20 points]

In this question, you are first asked to train Naive Bayes classifiers to find the optimal Naive Bayes model for car evaluation and credit approval data sets. Second, you will compare your optimal KNN and Naive Bayes models over car evaluation and credit approval data sets. Answer the following questions:

3.1 Train Naive Bayes classifiers over training data sets and test each classifier over corresponding test data. Report the error rates of the classifiers in a figure. Create one figure for car evaluation data set and another one for credit approval data set. You are allowed to use R packages for Naive Bayes algorithm.

Listing 8: Car Evaluation Data Set

```r
#install.packages("e1071")

library(e1071)
library(data.table)
library(class)

for(i in 1:5){
  train_data = fread(
    paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/car.training.", i))
  train_data$V1 = as.numeric(factor(train_data$V1))
  train_data$V2 = as.numeric(factor(train_data$V2))
  train_data$V3 = as.numeric(factor(train_data$V3))
  train_data$V4 = as.numeric(factor(train_data$V4))
  train_data$V5 = as.numeric(factor(train_data$V5))
  train_data$V6 = as.numeric(factor(train_data$V6))
  train_data$V7 = as.numeric(factor(train_data$V7))

  # naive bayes model
  nb_model <- naiveBayes(as.factor(V7)~., data=train_data)

  error_rate_sum = 0
  num_test_datasets = 5
  for (j in 1:num_test_datasets) {
    test_data = fread(
      paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/car.test.",j))

    test_data$V1 = as.numeric(factor(test_data$V1))
    test_data$V2 = as.numeric(factor(test_data$V2))
    test_data$V3 = as.numeric(factor(test_data$V3))
    test_data$V4 = as.numeric(factor(test_data$V4))
    test_data$V5 = as.numeric(factor(test_data$V5))
    test_data$V6 = as.numeric(factor(test_data$V6))
    test_data$V7 = as.numeric(factor(test_data$V7))

    # prediction
    pred <- predict(nb_model, test_data)

    res_table = table(pred, test_data$V7,dnn=c("Prediction","Actual"))

    error_rate = 1 - sum(diag(res_table)) / nrow(test_data)
    error_rate_sum = error_rate_sum + error_rate
    #print(paste0("the error rate for model ", i, " is ", error_rate, " on car.test.",j))
  }

  avg_error_rate = error_rate_sum / num_test_datasets
  print(paste0("the avg error rate for model " ,i, " is ", error_rate, " on 5 car.tests"))
}

#The best model is model 2 and it's average error rate is 0.309734513274336.
```

Listing 9: Credit Approval Data Set

```r
#install.packages("e1071")

library(e1071)
library(data.table)
library(class)
```

```
num_train_datasets = 5
for(i in 1:num_train_datasets){
  credit_train_data = fread(
    paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/credit.training.",i))
  credit_train_data$V1 = as.numeric(factor(credit_train_data$V1))
  credit_train_data$V2 = as.numeric(factor(credit_train_data$V2))
  credit_train_data$V3 = as.numeric(factor(credit_train_data$V3))
  credit_train_data$V4 = as.numeric(factor(credit_train_data$V4))
  credit_train_data$V5 = as.numeric(factor(credit_train_data$V5))
  credit_train_data$V6 = as.numeric(factor(credit_train_data$V6))
  credit_train_data$V7 = as.numeric(factor(credit_train_data$V7))
  credit_train_data$V8 = as.numeric(factor(credit_train_data$V8))
  credit_train_data$V9 = as.numeric(factor(credit_train_data$V9))
  credit_train_data$V10 = as.numeric(factor(credit_train_data$V10))
  credit_train_data$V11 = as.numeric(factor(credit_train_data$V11))
  credit_train_data$V12 = as.numeric(factor(credit_train_data$V12))
  credit_train_data$V13 = as.numeric(factor(credit_train_data$V13))
  credit_train_data$V14 = as.numeric(factor(credit_train_data$V14))
  credit_train_data$V15 = as.numeric(factor(credit_train_data$V15))
  credit_train_data$V16 = as.numeric(factor(credit_train_data$V16))

  # naive bayes model
  nb_model <- naiveBayes(as.factor(V16)~., data=credit_train_data)

  error_rate_sum = 0
  num_test_datasets = 5
  for (j in 1:num_test_datasets){
    credit_test_data = fread(
      paste0("/Users/yiningwang/Desktop/Academic/Junior2nd/B365/hw5/data/credit.test.",j))
    credit_test_data$V1 = as.numeric(factor(credit_test_data$V1))
    credit_test_data$V2 = as.numeric(factor(credit_test_data$V2))
    credit_test_data$V3 = as.numeric(factor(credit_test_data$V3))
    credit_test_data$V4 = as.numeric(factor(credit_test_data$V4))
    credit_test_data$V5 = as.numeric(factor(credit_test_data$V5))
    credit_test_data$V6 = as.numeric(factor(credit_test_data$V6))
    credit_test_data$V7 = as.numeric(factor(credit_test_data$V7))
    credit_test_data$V8 = as.numeric(factor(credit_test_data$V8))
    credit_test_data$V9 = as.numeric(factor(credit_test_data$V9))
    credit_test_data$V10 = as.numeric(factor(credit_test_data$V10))
    credit_test_data$V11 = as.numeric(factor(credit_test_data$V11))
    credit_test_data$V12 = as.numeric(factor(credit_test_data$V12))
    credit_test_data$V13 = as.numeric(factor(credit_test_data$V13))
    credit_test_data$V14 = as.numeric(factor(credit_test_data$V14))
    credit_test_data$V15 = as.numeric(factor(credit_test_data$V15))
    credit_test_data$V16 = as.numeric(factor(credit_test_data$V16))

    # prediction
    pred <- predict(nb_model, credit_test_data)
    res_table = table(pred, credit_test_data$V16, dnn=c("Prediction","Actual"))

    error_rate = 1 - sum(diag(res_table)) / nrow(credit_test_data)
    #print(paste0("the error rate for model ", i, " is ", error_rate, " on car.test.",j))

  }

  avg_error_rate = error_rate_sum / num_test_datasets
  print(paste0("the avg error rate for model " ,i, " is ", error_rate, " on 5 car.tests"))
}
```

```
65
# The best model is model 2 and it's average error rate is 0.138686131386861.
```

3.2 Pick the optimal Naive Bayes classifiers (one for car evaluation data set and another one for credit approval data set.) from question 3.1 and compare them with your best KNN models from question 2. Discuss the performances of the optimal classifiers over car evaluation and credit approval data sets, i.e, which one performed better?
The optimal Naive Bayes classifiers for car evaluation data set is model 2 and it's average error rate is 0.309734513274336. And the best KNN models from question 2 for car evaluation data set is k=3 and the minimum average error rate is 0.058512808942294 when the distance function is Manhattan distance. For the car evaluation data, the implementation of KNN performed better because it has a lower average error rate.
The optimal Naive Bayes classifiers for credit approval data set is model 2 and it's average error rate is 0.138686131386861. And the best KNN models from question 2 for credit approval data set is k=11 and the minimum average error rate is 0.379497680076954 when the distance function is Manhattan distance. For the credit approval data set, the Naive Bayes classifiers performed better because it's minimum average error rate is smaller.

Problem 4 [10 points]

From textbook, Chapter 5 exercise 7 (Page 318)

- (a).
$P(A = 1 \mid -) = 2/5 = 0.4$
$P(A = 0 \mid -) = 3/5 = 0.6$
$P(A = 1 \mid +) = 3/5 = 0.6$
$P(A = 0 \mid +) = 2/5 = 0.4$
$P(B = 1 \mid -) = 2/5 = 0.4$
$P(B = 0 \mid -) = 3/5 = 0.6$
$P(B = 1 \mid +) = 1/5 = 0.2$
$P(B = 0 \mid +) = 4/5 = 0.8$
$P(C = 1 \mid -) = 1$
$P(C = 0 \mid -) = 0$
$P(C = 1 \mid +) = 4/5 = 0.8$
$P(C = 0 \mid +) = 1/5 = 0.2$

- (b).
$K = P(A = 0, B = 1, C = 0)$
$P(+ \mid A = 0, B = 1, C = 0)$
$= \frac{P(A=0,B=1,C=0|+)\times P(+)}{P(A=0,B=1,C=0)}$
$= \frac{P(A=0|+)P(B=1|+)P(C=0|+)\times P(+)}{K}$
$= \frac{0.4\times 0.2\times 0.2\times 0.5}{K}$
$= \frac{0.024}{K}$

$P(- \mid A = 0, B = 1, C = 0)$
$= \frac{P(A=0,B=1,C=0|-)\times P(-)}{P(A=0,B=1,C=0)}$
$= \frac{P(A=0|-)P(B=1|-)P(C=0|-)\times P(-)}{K}$
$= \frac{0.6\times 0.4\times 0}{K}$
$= 0$

Therefore, the class label is "+".

- (c).
  $P(A = 0 \mid +) = \frac{2+2}{5+4} = \frac{4}{9}$
  $P(A = 0 \mid -) = \frac{3+2}{5+4} = \frac{5}{9}$
  $P(A = 1 \mid +) = \frac{3+2}{5+4} = \frac{5}{9}$
  $P(A = 1 \mid -) = \frac{2+2}{5+4} = \frac{4}{9}$
  $P(B = 1 \mid +) = \frac{1+2}{5+4} = \frac{3}{9}$
  $P(B = 1 \mid -) = \frac{2+2}{5+4} = \frac{4}{9}$
  $P(B = 0 \mid +) = \frac{4+2}{5+4} = \frac{6}{9}$
  $P(B = 0 \mid -) = \frac{3+2}{5+4} = \frac{5}{9}$
  $P(C = 0 \mid +) = \frac{1+2}{5+4} = \frac{3}{9}$
  $P(C = 0 \mid -) = \frac{0+2}{5+4} = \frac{2}{9}$
  $P(C = 1 \mid +) = \frac{4+2}{5+4} = \frac{6}{9}$
  $P(C = 1 \mid -) = \frac{5+2}{5+4} = \frac{7}{9}$

- (d).
  $K = P(A = 0, B = 1, C = 0)$
  $P(+ \mid A = 0, B = 1, C = 0)$
  $= \frac{P(A=0,B=1,C=0|+)\times P(+)}{P(A=0,B=1,C=0)}$
  $= \frac{P(A=0|+)P(B=1|+)P(C=0|+)\times P(+)}{K}$
  $= \frac{\frac{4}{9}\times\frac{3}{9}\times\frac{3}{9}\times 0.5}{K}$
  $= \frac{\frac{36}{729}}{K}$

  $P(- \mid A = 0, B = 1, C = 0)$
  $= \frac{P(A=0,B=1,C=0|-)\times P(-)}{P(A=0,B=1,C=0)}$
  $= \frac{P(A=0|-)P(B=1|-)P(C=0|-)\times P(-)}{K}$
  $= \frac{\frac{5}{9}\times\frac{4}{9}\times\frac{2}{9}\times 0.5}{K}$
  $= \frac{\frac{40}{729}}{K}$ Therefore, the class label is "-".

- (e). Under most conditions, both work well. However, the m-estimate one works better when there's a conditional probability that is equal to 0.

Extra Credit [30 points]

- From textbook, Chapter 5 exercises 8, 17 (Pages 318 & 322) [**15 points**]
- Problem 1: K-fold cross validation implementation [**15 points**]

1. 8.
   (a).
   $P(A = 1 \mid +) = 0.6$
   $P(A = 1 \mid -) = 0.4$
   $P(B = 1 \mid +) = 0.4$
   $P(B = 1 \mid -) = 0.6$
   $P(C = 1 \mid +) = 0.8$

$P(C = 1 \mid -) = 0.2$

(b).
$K = P(A = 1, B = 1, C = 1)$
$P(K \mid +) = P(A = 1 \mid +) \times P(B = 1 \mid +) \times P(C = 1 \mid +) = 0.192$
$P(K \mid -) = P(A = 1 \mid -) \times P(B = 1 \mid -) \times P(C = 1 \mid -) = 0.032$
$P(K \mid +) = 0.192 > P(K \mid -) = 0.032$
Therefore, the label is "+".

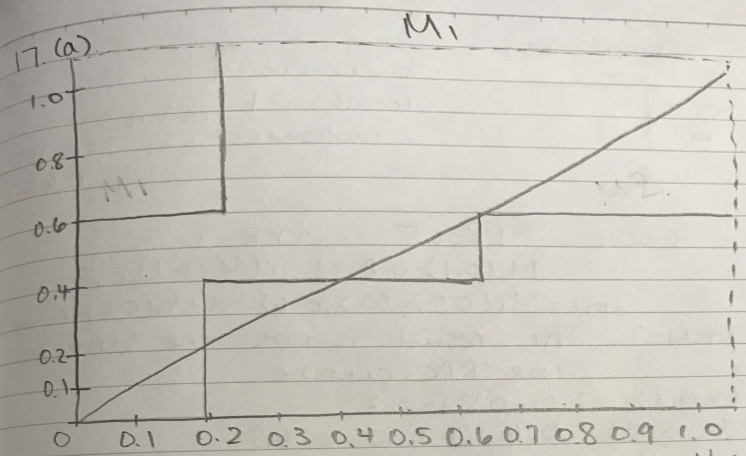
(c).
$P(A = 1) = 0.5$
$P(B = 1) = 0.4$
$P(A = 1, B = 1) = P(A) \times P(B) = 0.5 \times 0.4 = 0.2$
Therefore, A and B are independent.

(d).
$P(A = 1) = 0.5$
$P(B = 0) = 0.6$
$P(A = 1, B = 0) = P(A) \times P(B) = 0.5 \times 0.6 = 0.3$
Therefore, A and B are independent.

(e).
$P(A = 1, B = 1 \mid +) = 0.2$
$P(A = 1 \mid +) = 0.6$
$P(B = 1 \mid +) = 0.4$
$P(A = 1 \mid +) \times P(A = 1 \mid -) = 0.6 \times 0.4 = 0.24 \neq P(A = 1, B = 1 \mid +) = 0.2$
Therefore, A and B are not conditionally independent.

2. 17.

17. (a)



I think M1 is better because the area of M1 is larger than M2.

(b). t=0.5

M1

		Predicted class	
		+	−
Actual	+	3	2
	−	1	4

Precision: $\frac{3}{4}=0.75$ $\left(\frac{a}{a+c}\right)$

Recall: $\frac{3}{5}=0.6$ $\left(\frac{a}{a+b}\right)$

F-measure: $\frac{2\cdot 0.75\cdot 0.6}{0.75+0.6}=0.667$

(c). t=0.5

M2

		Predicted class	
		+	−
Actual	+	1	4
	−	1	4

Precision: $0.5=\frac{1}{2}$

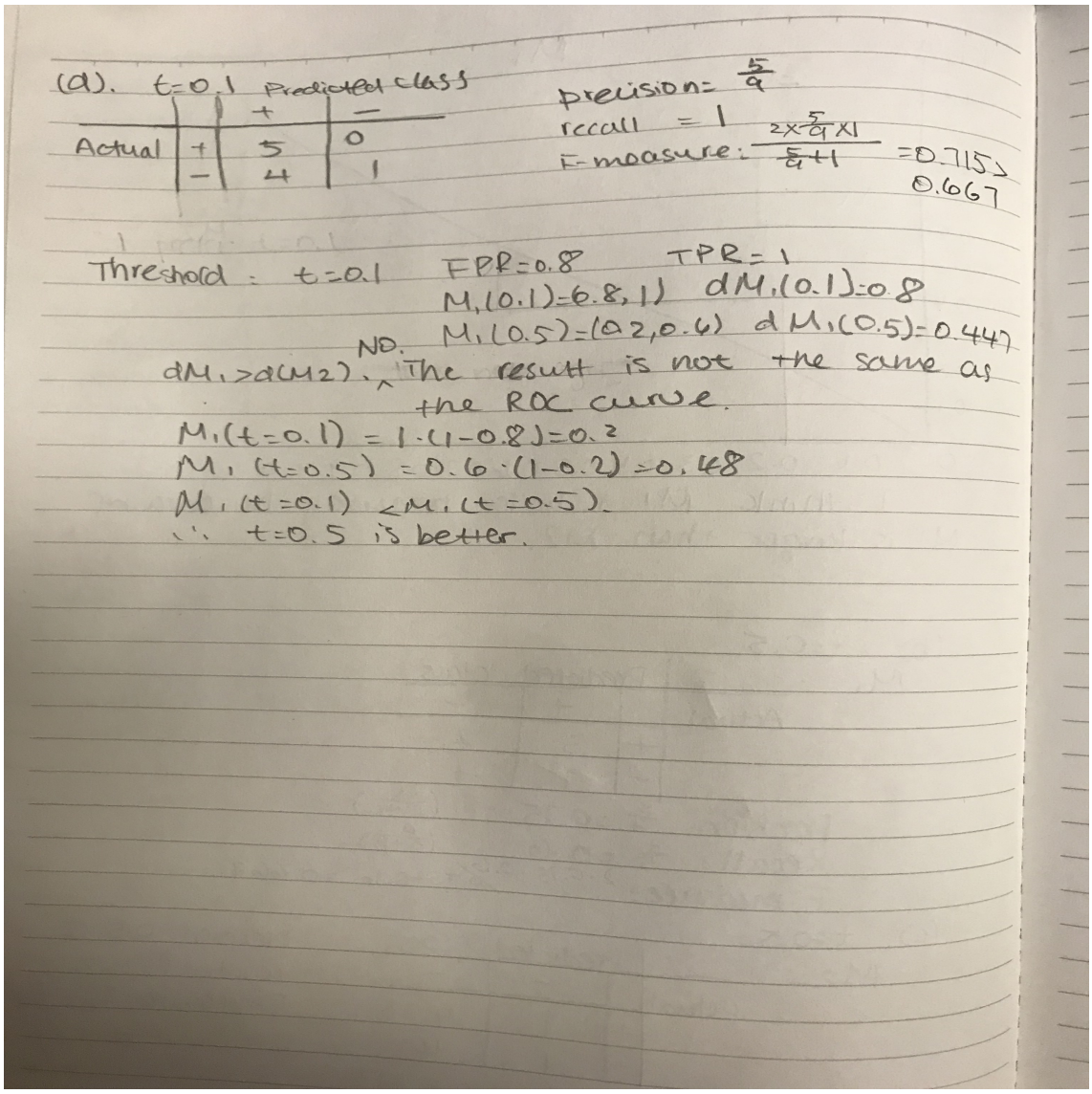
recall: $\frac{1}{5}=0.2$

F-measure: $\frac{2\times 0.5\times 0.2}{0.5+0.2}=\frac{0.2}{0.7}=\frac{2}{7}$

$=0.285$

<0.667

∴ M1 is better than M2



References

References

[1] k-Nearest Neighbour Classification *http://stat.ethz.ch/R-manual/R-devel/library/class/html/knn.html.*

[2] Nave Bayes classification in R *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4930525/.*

[3] The distance function effect on k-nearest neighbor classification for medical datasets `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4978658/`.

[4] How to split a data set to do 10-fold cross validation *https://stats.stackexchange.com/questions/61090/how-to-split-a-data-set-to-do-10-fold-cross-validation.*

[5] Understanding Nave Bayes Classifier Using R *https://www.r-bloggers.com/understanding-naive-bayes-classifier-using-r/.*

What to Turn-in

Submit a .zip file that includes the files below. Name the .zip file as "usename-section number", i.e., hakurban-B365.

- The *tex and *pdf of the written answers to this document.
- *Rfiles for:
  - Question 1: crossValidation.R, output of cross validation: training and test data sets
  - Question 2.1: knn.R, Question 2.3: knnValidation.R
  - Question 3: naiveBayes.R
- A README file that explains how to run your code and other files in the folder