Homework 2 - Professor Sokol, Nirmit Chetwani

Introduction to Analytical Modeling

```
# Loading and Reading the data
data = read.table("credit_card_data.txt")

library(kernlab)
library(kknn)
library(data.table)

head(data)
```

```
##   V1    V2    V3   V4 V5 V6 V7 V8  V9 V10 V11
## 1  1 30.83 0.000 1.25  1  0  1  1 202   0   1
## 2  0 58.67 4.460 3.04  1  0  6  1  43 560   1
## 3  0 24.50 0.500 1.50  1  1  0  1 280 824   1
## 4  1 27.83 1.540 3.75  1  0  5  0 100   3   1
## 5  1 20.17 5.625 1.71  1  1  0  1 120   0   1
## 6  1 32.08 4.000 2.50  1  1  0  0 360   0   1
```

Question 3.1 - Cross Validation (a)

```
# Using k nearest neighbor we perform cross validation with kcv as 20.
kcvarray = list(10, 20, 30, 40, 50, 60)
for (array in kcvarray) {

modelcv <- cv.kknn(V11 ~ ., data = data, kcv = 20)
head(modelcv[[1]])

# Exploring the results from the model
cv = data.table(modelcv[[1]])
cv$yhat <- ifelse(cv$yhat < 0.49, 0, 1)
# Printing model accuracy:
print('Cross Validation Accuracy:'); print(array)
print(table(cv$y == cv$yhat))
print(prop.table(table(cv$y == cv$yhat)))
print(mean(cv$y == cv$yhat))


}
```

```
## [1] "Cross Validation Accuracy:"
## [1] 10
##
## FALSE  TRUE
##   102   552
##
##     FALSE      TRUE
## 0.1559633 0.8440367
```

```
## [1] 0.8440367
## [1] "Cross Validation Accuracy:"
## [1] 20
##
## FALSE  TRUE
##   105   549
##
##     FALSE      TRUE
## 0.1605505 0.8394495
## [1] 0.8394495
## [1] "Cross Validation Accuracy:"
## [1] 30
##
## FALSE  TRUE
##   105   549
##
##     FALSE      TRUE
## 0.1605505 0.8394495
## [1] 0.8394495
## [1] "Cross Validation Accuracy:"
## [1] 40
##
## FALSE  TRUE
##   101   553
##
##     FALSE      TRUE
## 0.1544343 0.8455657
## [1] 0.8455657
## [1] "Cross Validation Accuracy:"
## [1] 50
##
## FALSE  TRUE
##    97   557
##
##    FALSE     TRUE
## 0.148318 0.851682
## [1] 0.851682
## [1] "Cross Validation Accuracy:"
## [1] 60
##
## FALSE  TRUE
##    96   558
##
##    FALSE     TRUE
## 0.146789 0.853211
## [1] 0.853211
```

kcv set to 10 gives us the best classifer. The model accuracy decreases if we increase the number of folds past 50.

Question 3.1 - Training, Validation, Test Datasets (b)

Dataset Creation

```
# Sampling the data to create a training set
Sample <- sample(1:654, (654*0.6))
data.train <- data[Sample, ]

# Creating a subset of the data for validation and test.
data.notrain <- data[-Sample, ]

# Sampling from the subset
Sample2 <- sample(1:197, (197*0.5))
# Validaton data set
data.validation <- (data.notrain[Sample2, ])

# Test data set
data.test <- data.notrain[-Sample2, ]
```

Creating a model w/ Training Dataset

```
# Trying different values of k to find the best classifier.
kmaxarray = list(11, 31, 71)

for(array in kmaxarray){

model <- train.kknn(V11 ~ ., data = data.train, kmax = array)
model

prediction <- predict(model, data.validation[, -11])
prediction
# Factoring the prediction
t_prediction <- ifelse(prediction < 0.49, 0, 1)

CM <- table(data.validation[, 11], t_prediction)
CM

# Printing accuracy of model
accuracy <- (sum(diag(CM)))/sum(CM)
print(array)
accuracy
print(accuracy)

}
```

```
## [1] 11
## [1] 0.8163265
## [1] 31
## [1] 0.8265306
## [1] 71
## [1] 0.8367347
```

Retraning the model on the combined dataset

```
combineddata = rbind(data.train, data.validation)
model <- train.kknn(V11 ~ ., data = combineddata, kmax = 61)
model
```

```
##
## Call:
## train.kknn(formula = V11 ~ ., data = combineddata, kmax = 61)
##
## Type of response variable: continuous
## minimal mean absolute error: 0.2131178
## Minimal mean squared error: 0.1081853
## Best kernel: optimal
## Best k: 52
```

```
# Testing the model
prediction <- predict(model, data.test[, -11])
prediction
```

```
##   [1] 0.917214787 0.842450654 0.047825932 0.913776436 0.759944176
##   [6] 0.983606256 0.988177531 0.801437043 0.966667147 0.813491961
##  [11] 0.936460631 0.504721499 0.546545312 0.856901697 0.562298813
##  [16] 0.483671549 0.046241305 0.497880699 0.842411382 0.471608562
##  [21] 0.583759679 0.579920570 0.446652813 0.506533210 0.851795872
##  [26] 0.901197261 0.891496915 0.796237979 0.800735969 0.813079645
##  [31] 0.845248356 0.546304285 0.518313882 0.530866062 0.884752888
##  [36] 0.891630797 0.991869148 0.583989669 0.840542462 0.904645534
##  [41] 0.484869829 0.471024992 0.816321775 0.037912013 0.095295127
##  [46] 0.097596952 0.065787182 0.260070629 0.071093880 0.145291606
##  [51] 0.324375731 0.101053400 0.081704274 0.033972261 0.101156584
##  [56] 0.065793684 0.069641752 0.112687877 0.040733378 0.067897417
##  [61] 0.121419424 0.057309382 0.099997046 0.087440993 0.097211909
##  [66] 0.098098374 0.069750846 0.057843443 0.119549335 0.065110628
##  [71] 0.113589706 0.076931103 0.101484902 0.147924967 0.080791646
##  [76] 0.039254300 0.201259817 0.144124768 0.052004210 0.116196592
##  [81] 0.049709986 0.124799177 0.049984403 0.323994615 0.065383309
##  [86] 0.126276223 0.057978602 0.044404206 0.051590376 0.876580821
##  [91] 0.707764779 0.940772607 0.884684999 0.791182526 0.536683226
##  [96] 0.512107475 0.721640461 0.611174252 0.484483227 0.457932679
## [101] 0.568621341 0.506545985 0.543765010 0.906327267 0.865874668
## [106] 0.897560163 0.918610644 0.894271533 0.998402832 0.494325144
## [111] 0.910555805 0.914764402 0.877301853 0.863890823 0.887574779
## [116] 0.956553348 0.989471582 0.871389644 0.922967870 0.559344387
## [121] 0.954510436 0.843266922 0.871418308 0.973095878 0.077638518
## [126] 0.430640910 0.871426814 0.042757545 0.368519832 0.088993861
## [131] 0.184311512 0.102190786 0.302095914 0.084507770 0.121599903
## [136] 0.072896876 0.223629338 0.101658529 0.111695686 0.122478762
## [141] 0.085242410 0.043959503 0.075802187 0.098075100 0.058372286
## [146] 0.134357789 0.060560782 0.089178101 0.223707308 0.055258183
## [151] 0.006732689 0.059465593 0.055749504 0.077957179 0.075667205
## [156] 0.068705004 0.080401057 0.090693422 0.537288509 0.102675671
## [161] 0.088084658 0.044543800 0.037877833 0.308995272
```

```
t_prediction <- ifelse(prediction < 0.49, 0, 1)

CM <- table(data.test[, 11], t_prediction)
CM
```

```
##    t_prediction
##      0  1
##   0 85 14
##   1  8 57
```

```
# Final Accuracy
accuracy <- (sum(diag(CM)))/sum(CM)
print(array)
```

```
## [1] 71
```

```
accuracy
```

```
## [1] 0.8658537
```

We get a high final accuracy ~ >85% from our model by splitting the data in training, validation and test data proving we have found a good classifier.

Question 4.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

One real-life situation for which a clustering model would be appropriate would be a retail business using k-means clustering to determine the optimal number of delivery routes/launch locations. Ventures such as Amazon with large numbers of warehouses often attempt to maximize their efficiency and profits by optimizing the locations of their distribution centers, making such a use of k-means clustering vital to the growth of their operations. Five predictors that could be used to create such an optimization via clustering include:

1. Population density (e.g. by district/metropolitan area)
2. Amount of businesses in the defined area (tied with economic status of average resident in area)
3. Most frequently ordered product categories in area
4. Distance from nearest existing distribution center(s)
5. Demographics of surrounding population

Using such predictors, among other fluctuating variables (e.g. weather in the area, etc.), Amazon is able to determine efficient routes that will minimize the amount of expenses/time spent in making frequent deliveries and thereby maximize their profits. As a result, we can see that k-means clustering has real-life applications with tangible impacts upon even entrepreneurial behemoths like Amazon.com.

Question 4.2

In this question we use a for loop to find the best combination of predictors.

```
iris = read.table("iris.txt")

library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:kernlab':
##
##     alpha
```

```
# Loading the data
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
# Looping through every single combination.
for(m in list(1, 2, 3, 4)){

  for(l in combn(list(1, 2, 3, 4), m, simplify = FALSE)){

    fit <- kmeans(iris[,unlist(l)], centers = 3, nstart=20)
    fit
    fit$cluster
```

```
    print(unlist(l))
    #Finding classification errors.
    print(table(fit$cluster, iris$Species))

  }


}
```

```
## [1] 1
##
##     setosa versicolor virginica
##   1     47         11         1
##   2      3         31        27
##   3      0          8        22
## [1] 2
##
##     setosa versicolor virginica
##   1      1         27        19
##   2     18         22        26
##   3     31          1         5
## [1] 3
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         48         6
##   3      0          2        44
## [1] 4
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0          2        46
##   3      0         48         4
## [1] 1 2
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         38        15
##   3      0         12        35
## [1] 1 3
##
##     setosa versicolor virginica
##   1      0         45        13
##   2     50          1         0
##   3      0          4        37
## [1] 1 4
##
##     setosa versicolor virginica
##   1      0         37        15
##   2      0          9        35
##   3     50          4         0
## [1] 2 3
##
##     setosa versicolor virginica
```

```
##   1      0          2        41
##   2      0         48         9
##   3     50          0         0
## [1] 2 4
##
##     setosa versicolor virginica
##   1      1         46         6
##   2     49          0         0
##   3      0          4        44
## [1] 3 4
##
##     setosa versicolor virginica
##   1      0         48         4
##   2      0          2        46
##   3     50          0         0
## [1] 1 2 3
##
##     setosa versicolor virginica
##   1      0         45        13
##   2     50          0         0
##   3      0          5        37
## [1] 1 2 4
##
##     setosa versicolor virginica
##   1      0         11        35
##   2     50          0         0
##   3      0         39        15
## [1] 1 3 4
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0          2        36
##   3      0         48        14
## [1] 2 3 4
##
##     setosa versicolor virginica
##   1      0          2        45
##   2      0         48         5
##   3     50          0         0
## [1] 1 2 3 4
##
##     setosa versicolor virginica
##   1      0         48        14
##   2     50          0         0
##   3      0          2        36
```

From the results we can see that `Petal.Width` and `Petal.Width, Sepal.Width` are the best predictors with both resulting in only 6 misclassifications.