ISYE6501 HW Wk1 Solution

Question 2.1

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

For my job, we need to do the target customer classification to recommend the right market products. Based on the customers profile or historical customer behaviers, we classify the customers based on factors such as age, gender, marriage status, kids, occupations to learn what kind of products each customer group is most likely to purchase.

Question 2.2

1.Using the support vector machine function ksvm contained in the R package kernlab, find a good classifier for this data. Show the equation of your classifier, and how well it classifies the data points in the full data set.

```
library(kernlab)
library(kknn)
```

```
# setwd("~/Alex/ISYE6501")
# Read data
data <-  read.table("credit_card_data.txt")

# explore the raw data
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
```

```
summary(data)
```

```
##        V1               V2              V3               V4
##  Min.   :0.0000   Min.   :13.75   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.:22.58   1st Qu.: 1.040   1st Qu.: 0.165
##  Median :1.0000   Median :28.46   Median : 2.855   Median : 1.000
##  Mean   :0.6896   Mean   :31.58   Mean   : 4.831   Mean   : 2.242
##  3rd Qu.:1.0000   3rd Qu.:38.25   3rd Qu.: 7.438   3rd Qu.: 2.615
##  Max.   :1.0000   Max.   :80.25   Max.   :28.000   Max.   :28.500
##        V5               V6              V7               V8
##  Min.   :0.0000   Min.   :0.0000   Min.   : 0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.:0.0000
##  Median :1.0000   Median :1.0000   Median : 0.000   Median :1.0000
##  Mean   :0.5352   Mean   :0.5612   Mean   : 2.498   Mean   :0.5382
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 3.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :67.000   Max.   :1.0000
```

```
##        V9               V10              V11
##  Min.   :   0.00   Min.   :     0   Min.   :0.0000
##  1st Qu.:  70.75   1st Qu.:     0   1st Qu.:0.0000
##  Median : 160.00   Median :     5   Median :0.0000
##  Mean   : 180.08   Mean   :  1013   Mean   :0.4526
##  3rd Qu.: 271.00   3rd Qu.:   399   3rd Qu.:1.0000
##  Max.   :2000.00   Max.   :100000   Max.   :1.0000
```

```
dim(data)
```

```
## [1] 654  11
```

```
str(data)
```

```
## 'data.frame':    654 obs. of  11 variables:
##  $ V1 : int  1 0 0 1 1 1 1 0 1 1 ...
##  $ V2 : num  30.8 58.7 24.5 27.8 20.2 ...
##  $ V3 : num  0 4.46 0.5 1.54 5.62 ...
##  $ V4 : num  1.25 3.04 1.5 3.75 1.71 ...
##  $ V5 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ V6 : int  0 0 1 0 1 1 1 1 1 1 ...
##  $ V7 : int  1 6 0 5 0 0 0 0 0 0 ...
##  $ V8 : int  1 1 1 0 1 0 0 1 1 0 ...
##  $ V9 : int  202 43 280 100 120 360 164 80 180 52 ...
##  $ V10: int  0 560 824 3 0 0 31285 1349 314 1442 ...
##  $ V11: int  1 1 1 1 1 1 1 1 1 1 ...
```

```
# Build Model
# use kvsm to build model
model <- ksvm(as.matrix(data[,1:10]),as.factor(data[,11]),
              type="C-svc",
              kernel="vanilladot",
              C=100,
              scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```
model
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 100
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 189
##
## Objective Function Value : -17887.92
## Training error : 0.136086
```

```
# calculate a1...am
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
a
```

```
##            V1            V2            V3            V4            V5
## -0.0010065348 -0.0011729048 -0.0016261967  0.0030064203  1.0049405641
##            V6            V7            V8            V9           V10
## -0.0028259432  0.0002600295 -0.0005349551 -0.0012283758  0.1063633995
```

```
# calculate a0
a0 <- -model@b
a0
```

```
## [1] 0.08158492
```

```
# see what the model predicts
pred <- predict(model, data[,1:10])

# see what fraction of the model's predictions match the actual classification
sum(pred == data[,11])/nrow(data)
```

```
## [1] 0.8639144
```

2. You are welcome, but not required, to try other (nonlinear) kernels as well; we're not covering them in this course, but they can sometimes be useful and might provide better predictions than vanilladot.

3. Using the k-nearest-neighbors classification function kknn contained in the R kknn package, suggest a good value of k, and show how well it classifies that data points in the full data set. Don't forget to scale the data (scale=TRUE in kknn).

```
correct = rep(0, 20)
for (x in seq(20)){
  predict_v = rep(0,nrow(data))
  predict_v
  for (i in 1:nrow(data)){
    model_knn <- kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,data[-i,],data[i,], k=x, scale=TRUE)
    predict_v[i] <- round(fitted(model_knn),0)
  }
  correct[x] = sum(predict_v==data[,11])/nrow(data)
}
# show the predict% for k from 1-20
correct
```

```
##  [1] 0.8149847 0.8149847 0.8149847 0.8149847 0.8516820 0.8455657 0.8470948
##  [8] 0.8486239 0.8470948 0.8501529 0.8516820 0.8532110 0.8516820 0.8516820
## [15] 0.8532110 0.8516820 0.8516820 0.8516820 0.8501529 0.8501529
```

```
# find the best k
which.max(correct)
```

```
## [1] 12
```

Question 3.1

Using the same data set (credit_card_data.txt or credit_card_data-headers.txt) as in Question 2.2, use the ksvm or kknn function to find a good classifier:

(a) using cross-validation (do this for the k-nearest-neighbors model; SVM is optional);

```
correct_3 = rep(0,20)
model_3 <- train.kknn(V11~.,data,kmax=20,scale=TRUE)

# use train.kknn to do the cross-validation
sum(round(fitted(model_3)[[2]][1:nrow(data)],0)==data[,11])
```

```
## [1] 533
```

```
# calculate the predict% of each k
for (k in 1:20){
correct_3[k] <- sum(round(fitted(model_3)[[k]][1:nrow(data)],0)==data[,11])/nrow(data)
}

correct_3
```

```
##  [1] 0.8149847 0.8149847 0.8149847 0.8149847 0.8516820 0.8455657 0.8470948
##  [8] 0.8486239 0.8470948 0.8516820 0.8516820 0.8532110 0.8516820 0.8516820
## [15] 0.8532110 0.8532110 0.8532110 0.8516820 0.8501529 0.8501529
```

```
# find the best k
which.max(correct_3)
```

```
## [1] 12
```

(b) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

```
# split data into training, validation, and test data as 60%, 20%, 20%
train_sample = sample(nrow(data), size=nrow(data)*0.6)
df_train = data[train_sample, ]
df_left = data[-train_sample,]

valid_sample = sample(nrow(df_left),size=nrow(df_left)*0.5)
df_valid = df_left[valid_sample,]
df_test = df_left[-valid_sample,]
```

```
# test on KNN model
correct_b = rep(0, 20)
for (x in seq(20)){
  model_knn <- kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,df_train,df_valid, k=x, scale=TRUE)
  predict_v <- round(fitted(model_knn),0)
  correct_b[x] = sum(predict_v==df_valid[,11])/nrow(df_valid)
}
# show the predict% for k from 1-20
which.max(correct_b)
```

```
## [1] 8
```

```
# use test data to test with the best k

model_knn_b <- kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10,df_train,df_test, k=which.max(correct_b), scale="
predict_b <- round(fitted(model_knn_b),0)
correct_best = sum(predict_b==df_test[,11])/nrow(df_test)
correct_best
```

```
## [1] 0.8778626
```