Consider the `Caravan` data set from the library `ISLR`. The dataset contains 5822 real customer records. Each record consists of 86 variables, containing sociodemographic data (variables 1-43) and product ownership (variables 44-86). The sociodemographic data is derived from zip codes. All customers living in areas with the same zip code have the same sociodemographic attributes. Variable 86 (Purchase) indicates whether the customer purchased a caravan insurance policy. Further information can be obtained from

`http://liacs.leidenuniv.nl/~puttenpwhvander/library/cc2000/data.html`

It is interest to predict if an individual with a given set of demographic characteristics would purchase the insurance.

a) Install library `class` to use function `knn()`

b) Scale predictor columns. Then split the observations into a test set, containing the first 1,000 observations, and a training set, containing the remaining observations.

c) Find the test error rates using K-Nearest Neighbors with $k = 1, 3, 5$

d) Fit a multiple logistic regression model, find the test error rate.

e) Find the test error rate with different cut-off values for classifying individuals as buyers and non-buyers

```
# knn.r

library(ISLR)    # Caravan Insurance Data
library(class)   # knn()

d0 = Caravan
dim(d0)          # 5822   86

table(d0$Purchase)
#   No  Yes
# 5474  348
prop.table(table(d0$Purchase))
#         No        Yes
# 0.94022673 0.05977327
# only 6% people purchased insurance

# scale required for distance based methods
d1y = d0$Purchase
d1x = scale(d0[,-86])        # exclude response in col 86
var(d0[,1])    # 165.0378
var(d1x[,1])   # 1

# test set is 1st 1000 obs
test=1:1000
test.y=d1y[test]
test.x=d1x[test,]
dim(test.x)         # [1] 1000   85

train.y=d1y[-test]
train.x=d1x[-test,]
dim(train.x)        # [1] 4822   85
```

```
# K Nearest neighbors
#=====================================================================
set.seed(1)
pred1=knn(train.x,test.x,train.y,k=1)   # arguments order

table(pred1,test.y)
#      test.y
# pred1     No Yes
#   No  873  50
#   Yes  68   9

prop.table(table(pred1,test.y))
#      test.y
# pred1       No   Yes
#   No  0.873 0.050
#   Yes 0.068 0.009

# error rate = 0.068 + 0.050 = 0.118
# error rate can be reduced to 0.06 by ALWAYS predicting NO

table(test.y)
# test.y
#  No Yes
# 941  59

# We want accurate predictions for those who buy insurance

table(pred1,test.y)
#      test.y
# pred1     No Yes
#   No  873  50
#   Yes  68   9

9/(68+9)        # [1] 0.1168831    accuracy rate better than random guessing
```

```
# increase k
#======================================================================
pred3=knn(train.x,test.x,train.y,k=3)
table(pred3,test.y)
#         test.y
# pred2     No Yes
#     No   920  54
#     Yes   21   5
5/26              # [1] 0.1923077

pred5=knn(train.x,test.x,train.y,k=5)
table(pred5,test.y)
#       test.y
# pred5  No Yes
#   No  930  55
#   Yes  11   4
4/11              # 0.27

# logistic regression
#======================================================================

glm1  = glm(Purchase~.,Caravan,family=binomial,subset=-test)
pi.hat= predict(glm1,Caravan[test,],type="response")
yhat=rep("No",1000)

# predict a purchase whenever pi.hat > cutoff
yhat[pi.hat > 0.50]="Yes"
table(yhat,test.y)
#       test.y
# yhat   No Yes
#   No  934  59
#   Yes   7   0

# error rate is 100% for predicting individuals who will buy insurance

yhat[pi.hat > 0.3]="Yes"
table(yhat,test.y)
#      test.y
# yhat   No Yes
#   No  928  51
#   Yes  13   8
8/21           # [1] 0.3809524

yhat[pi.hat > 0.25]="Yes"
table(yhat,test.y)
#      test.y
# yhat   No Yes
#   No  919  48
#   Yes  22  11

11/(22+11)    # 0.33     accuracy rate
```