Garg_Bheeni_Stat6620_Homework2

Bheeni Garg

April 25, 2016

Question 1:

Chapter 3: Classification using Nearest Neighbors ————————

Example: Classifying Cancer Samples —-

Step 1: Collecting Data

The "Breast Cancer Wisconsin Diagnostic" dataset from the UCI MAchine Learning Repository has been utilized.

The breast cancer data includes 569 examples of cancer biopsies, each with 32 features. One feature is an identication number, another is the cancer diagnosis, and 30 are numeric-valued laboratory measurements. The diagnosis is coded as M to indicate malignant or B to indicate benign.

The 30 numeric measurements comprise the mean, standard error, and worst (that is, largest) value for 10 different characteristics of the digitized cell nuclei. These include:

• Radius • Texture • Perimeter • Area • Smoothness • Compactness • Concavity • Concave points • Symmetry • Fractal dimension

```
## Step 2: Exploring and preparing the data ----

# import the CSV file
wbcd <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)

# examine the structure of the wbcd data frame
str(wbcd)
```

```
## 'data.frame':    569 obs. of  32 variables:
##  $ id               : int  87139402 8910251 905520 868871 9012568 906539 925291 87880 862989 89827 .
##  $ diagnosis        : chr  "B" "B" "B" "B" ...
##  $ radius_mean      : num  12.3 10.6 11 11.3 15.2 ...
##  $ texture_mean     : num  12.4 18.9 16.8 13.4 13.2 ...
##  $ perimeter_mean   : num  78.8 69.3 70.9 73 97.7 ...
##  $ area_mean        : num  464 346 373 385 712 ...
##  $ smoothness_mean  : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
##  $ compactness_mean : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
##  $ concavity_mean   : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
##  $ points_mean      : num  0.037 0.0264 0.0248 0.048 0.0266 ...
##  $ symmetry_mean    : num  0.196 0.192 0.171 0.177 0.172 ...
##  $ dimension_mean   : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
##  $ radius_se        : num  0.236 0.451 0.197 0.338 0.178 ...
##  $ texture_se       : num  0.666 1.197 1.387 1.343 0.412 ...
##  $ perimeter_se     : num  1.67 3.43 1.34 1.85 1.34 ...
##  $ area_se          : num  17.4 27.1 13.5 26.3 17.7 ...
```

```
##  $ smoothness_se    : num  0.00805 0.00747 0.00516 0.01127 0.00501 ...
##  $ compactness_se   : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
##  $ concavity_se     : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
##  $ points_se        : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
##  $ symmetry_se      : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
##  $ dimension_se     : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
##  $ radius_worst     : num  13.5 11.9 12.4 11.9 16.2 ...
##  $ texture_worst    : num  15.6 22.9 26.4 15.8 15.7 ...
##  $ perimeter_worst  : num  87 78.3 79.9 76.5 104.5 ...
##  $ area_worst       : num  549 425 471 434 819 ...
##  $ smoothness_worst : num  0.139 0.121 0.137 0.137 0.113 ...
##  $ compactness_worst: num  0.127 0.252 0.148 0.182 0.174 ...
##  $ concavity_worst  : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
##  $ points_worst     : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
##  $ symmetry_worst   : num  0.283 0.294 0.3 0.21 0.249 ...
##  $ dimension_worst  : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...
```

```
# drop the id feature
wbcd <- wbcd[-1]

# table of diagnosis
table(wbcd$diagnosis)
```

```
##
##   B   M
## 357 212
```

```
# recode diagnosis as a factor
wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"), labels = c("Benign",
    "Malignant"))

# table or proportions with more informative labels
round(prop.table(table(wbcd$diagnosis)) * 100, digits = 1)
```

```
##
##    Benign Malignant
##      62.7      37.3
```

```
# summarize three numeric features
summary(wbcd[c("radius_mean", "area_mean", "smoothness_mean")])
```

```
##   radius_mean       area_mean      smoothness_mean
##  Min.   : 6.981   Min.   : 143.5   Min.   :0.05263
##  1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
##  Median :13.370   Median : 551.1   Median :0.09587
##  Mean   :14.127   Mean   : 654.9   Mean   :0.09636
##  3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
##  Max.   :28.110   Max.   :2501.0   Max.   :0.16340
```

```
# create normalization function to bring all measurements on equal scale
normalize <- function(x) {
    return((x - min(x))/(max(x) - min(x)))
```

```
}

# normalize the wbcd data
wbcd_n <- as.data.frame(lapply(wbcd[2:31], normalize))

# confirm that normalization worked
summary(wbcd_n$area_mean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1174  0.1729  0.2169  0.2711  1.0000
```

```
# create training and test data
wbcd_train <- wbcd_n[1:469, ]
wbcd_test <- wbcd_n[470:569, ]

# create labels for training and test data

wbcd_train_labels <- wbcd[1:469, 1]
wbcd_test_labels <- wbcd[470:569, 1]

## Step 3: Training a model on the data ----

# load the 'class' library
library(class)

wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels,
    k = 21)  # k= 21, sqrt(total number of    training examples) = sqrt(469) = 21.66

# Predictions produced
wbcd_test_pred
```

```
##   [1] Benign    Benign    Benign    Benign    Malignant Benign    Malignant
##   [8] Benign    Malignant Benign    Malignant Benign    Malignant Malignant
##  [15] Benign    Benign    Malignant Benign    Malignant Benign    Malignant
##  [22] Malignant Malignant Malignant Benign    Benign    Benign    Benign
##  [29] Malignant Malignant Malignant Benign    Malignant Malignant Benign
##  [36] Benign    Benign    Benign    Benign    Malignant Malignant Benign
##  [43] Malignant Malignant Benign    Malignant Malignant Malignant Malignant
##  [50] Malignant Benign    Benign    Benign    Benign    Benign    Benign
##  [57] Benign    Benign    Malignant Benign    Benign    Benign    Benign
##  [64] Benign    Malignant Malignant Benign    Benign    Benign    Benign
##  [71] Benign    Malignant Benign    Benign    Malignant Malignant Benign
##  [78] Benign    Benign    Benign    Benign    Benign    Benign    Malignant
##  [85] Benign    Benign    Malignant Benign    Benign    Benign    Benign
##  [92] Malignant Benign    Benign    Benign    Benign    Benign    Malignant
##  [99] Benign    Malignant
## Levels: Benign Malignant
```

```
## Step 4: Evaluating model performance ----

# load the 'gmodels' library
library(gmodels)
```

```r
# Create the cross tabulation of predicted vs. actual
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | wbcd_test_pred
## wbcd_test_labels |    Benign | Malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           Benign |        61 |         0 |        61 |
##                  |     1.000 |     0.000 |     0.610 |
##                  |     0.968 |     0.000 |           |
##                  |     0.610 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##        Malignant |         2 |        37 |        39 |
##                  |     0.051 |     0.949 |     0.390 |
##                  |     0.032 |     1.000 |           |
##                  |     0.020 |     0.370 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        63 |        37 |       100 |
##                  |     0.630 |     0.370 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```r
# Accuracy = (TP+TN)/total
(37 + 61)/100
```

```
## [1] 0.98
```

```r
# Misclassification rate/ Error Rate/ 1 - Accuracy = (FP + FN)/total
(0 + 2)/100
```

```
## [1] 0.02
```

```r
# False Positive Rate = FP/ actual no
0/61
```

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

```r
# Sensitivity/Recall/True positive Rate= TP/actual yes
37/39
```

```
## [1] 0.9487179
```

```r
# Specificity/1-FP rate= TN/actual no
61/61
```

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

```r
# Precision = TP/predicted yes
37/37
```

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

```r
# Prevalence = Actual yes/total
39/100
```

```
## [1] 0.39
```

```r
## Step 5: Improving model performance ----

# use the scale() function to z-score standardize a data frame
wbcd_z <- as.data.frame(scale(wbcd[-1]))

# confirm that the transformation was applied correctly
summary(wbcd_z$area_mean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.4530 -0.6666 -0.2949  0.0000  0.3632  5.2460
```

```r
# create training and test datasets
wbcd_train <- wbcd_z[1:469, ]
wbcd_test <- wbcd_z[470:569, ]

# re-classify test cases
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels,
    k = 21)

# Create the cross tabulation of predicted vs. actual
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
```

```
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | wbcd_test_pred
## wbcd_test_labels |    Benign | Malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           Benign |        61 |         0 |        61 |
##                  |     1.000 |     0.000 |     0.610 |
##                  |     0.924 |     0.000 |           |
##                  |     0.610 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##        Malignant |         5 |        34 |        39 |
##                  |     0.128 |     0.872 |     0.390 |
##                  |     0.076 |     1.000 |           |
##                  |     0.050 |     0.340 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        66 |        34 |       100 |
##                  |     0.660 |     0.340 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```
# Accuracy = (TP+TN)/total
(34 + 61)/100
```

```
## [1] 0.95
```

It can be noted that the accuracy of the model is greater using min-max normalization (0.98) than z-score standardization. Hence the z-score standardization offers no improvement in the model

```
# Misclassification rate/ Error Rate/ 1 - Accuracy = (FP + FN)/total
(0 + 5)/100
```

```
## [1] 0.05
```

```
# Sensitivity/Recall/True positive Rate= TP/actual yes
34/39
```

```
## [1] 0.8717949
```

```
# Specificity/1-FP rate= TN/actual no
61/61
```

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

```r
# False Positive Rate = FP/ actual no
0/61
```

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

```r
# Precision = TP/predicted yes
34/34
```

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

```r
# Prevalence = Actual yes/total
39/100
```

```
## [1] 0.39
```

```r
# Achieving an optimum k for highest accuracy (min-max normalization)
wbcd_train <- wbcd_n[1:469, ]
wbcd_test <- wbcd_n[470:569, ]

# k=1
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels,
    k = 1)
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | wbcd_test_pred
## wbcd_test_labels |    Benign | Malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           Benign |        58 |         3 |        61 |
##                  |     0.951 |     0.049 |     0.610 |
##                  |     0.983 |     0.073 |           |
##                  |     0.580 |     0.030 |           |
## -----------------|-----------|-----------|-----------|
##        Malignant |         1 |        38 |        39 |
##                  |     0.026 |     0.974 |     0.390 |
##                  |     0.017 |     0.927 |           |
##                  |     0.010 |     0.380 |           |
## -----------------|-----------|-----------|-----------|
```

```
##     Column Total |        59 |        41 |       100 |
##                  |     0.590 |     0.410 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```
# Accuracy
(58 + 38)/100
```

```
## [1] 0.96
```

```
# Error Rate
(3 + 1)/100
```

```
## [1] 0.04
```

```
# k= 5
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels,
    k = 5)
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | wbcd_test_pred
## wbcd_test_labels |    Benign | Malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           Benign |        61 |         0 |        61 |
##                  |     1.000 |     0.000 |     0.610 |
##                  |     0.968 |     0.000 |           |
##                  |     0.610 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##        Malignant |         2 |        37 |        39 |
##                  |     0.051 |     0.949 |     0.390 |
##                  |     0.032 |     1.000 |           |
##                  |     0.020 |     0.370 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        63 |        37 |       100 |
##                  |     0.630 |     0.370 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```r
# Accuracy
(61 + 37)/100
```

```
## [1] 0.98
```

```r
# Error Rate
(0 + 2)/100
```

```
## [1] 0.02
```

```r
# k=11
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels,
    k = 11)
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | wbcd_test_pred
## wbcd_test_labels |    Benign | Malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           Benign |        61 |         0 |        61 |
##                  |     1.000 |     0.000 |     0.610 |
##                  |     0.953 |     0.000 |           |
##                  |     0.610 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##        Malignant |         3 |        36 |        39 |
##                  |     0.077 |     0.923 |     0.390 |
##                  |     0.047 |     1.000 |           |
##                  |     0.030 |     0.360 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        64 |        36 |       100 |
##                  |     0.640 |     0.360 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```r
# Accuracy
(61 + 36)/100
```

```
## [1] 0.97
```

```r
# Error Rate
(0 + 3)/100
```

```
## [1] 0.03
```

```r
# k = 15
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels,
    k = 15)
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | wbcd_test_pred
## wbcd_test_labels |    Benign | Malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           Benign |        61 |         0 |        61 |
##                  |     1.000 |     0.000 |     0.610 |
##                  |     0.953 |     0.000 |           |
##                  |     0.610 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##        Malignant |         3 |        36 |        39 |
##                  |     0.077 |     0.923 |     0.390 |
##                  |     0.047 |     1.000 |           |
##                  |     0.030 |     0.360 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        64 |        36 |       100 |
##                  |     0.640 |     0.360 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```r
# Accuracy
(61 + 36)/100
```

```
## [1] 0.97
```

```r
# Error Rate
(0 + 3)/100
```

```
## [1] 0.03
```

```r
# k = 21
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels,
    k = 21)
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | wbcd_test_pred
## wbcd_test_labels |    Benign | Malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           Benign |        61 |         0 |        61 |
##                  |     1.000 |     0.000 |     0.610 |
##                  |     0.968 |     0.000 |           |
##                  |     0.610 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##        Malignant |         2 |        37 |        39 |
##                  |     0.051 |     0.949 |     0.390 |
##                  |     0.032 |     1.000 |           |
##                  |     0.020 |     0.370 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        63 |        37 |       100 |
##                  |     0.630 |     0.370 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```r
# Accuracy
(61 + 37)/100
```

```
## [1] 0.98
```

```r
# Error Rate
(0 + 2)/100
```

```
## [1] 0.02
```

```r
# k = 27
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels,
    k = 27)
CrossTable(x = wbcd_test_labels, y = wbcd_test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                  | wbcd_test_pred
## wbcd_test_labels |    Benign | Malignant | Row Total |
## -----------------|-----------|-----------|-----------|
##           Benign |        61 |         0 |        61 |
##                  |     1.000 |     0.000 |     0.610 |
##                  |     0.938 |     0.000 |           |
##                  |     0.610 |     0.000 |           |
## -----------------|-----------|-----------|-----------|
##        Malignant |         4 |        35 |        39 |
##                  |     0.103 |     0.897 |     0.390 |
##                  |     0.062 |     1.000 |           |
##                  |     0.040 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |        65 |        35 |       100 |
##                  |     0.650 |     0.350 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```
# Accuracy
(61 + 35)/100
```

```
## [1] 0.96
```

```
# Error Rate
(0 + 4)/100
```

```
## [1] 0.04
```

It can be noted that the accuracy is highest (= 0.98) when k=5 and k=21. The value of k is chosen which gives the best bias-variance trade-off

Question 2:

Step 1: Collecting Data

I have used k-Nearest Neighbors algorithm (kNN) to predict whether price of Apple stock will increase or decrease. The data is obtained from Yahoo Finance.

Step 2: Exploring and preparing the data

```
library(class)
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(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```
set.seed(100)

stocks <- read.csv('stocks.csv')
str(stocks)
```

```
## 'data.frame':    1461 obs. of  5 variables:
##  $ Date    : Factor w/ 1461 levels "2010-01-04","2010-01-05",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Apple   : num  214 214 211 211 212 ...
##  $ Google  : num  627 624 608 594 602 ...
##  $ MSFT    : num  31 31 30.8 30.5 30.7 ...
##  $ Increase: logi  TRUE TRUE FALSE FALSE TRUE FALSE ...
```

```
table(stocks$Increase)
```

```
##
## FALSE  TRUE
##   697   764
```

The 3 columns represent the closing price of the stock of their respective companies on the given date. The Increase column represents whether the price of Apple rose or fell as compared to the previous day.

Splitting the dataset into training and test sets- data before 2014 is put in the training set and after 2014 is put in test set

```
stocks$Date <- ymd(stocks$Date)
stocksTrain <- year(stocks$Date) < 2014
```

Step 3: Training a model on the data

It will consist of the prices of stocks of Apple, Google, and Microsoft on the previous day. For this, we can use the lag function in dplyr.

```
predictors <- cbind(lag(stocks$Apple, default = 210.73), lag(stocks$Google,
    default = 619.98), lag(stocks$MSFT, default = 30.48))
```

Since for the very first value (corresponding to January 4, 2010), the lag function has nothing to compare it to, it will default to NA. To avoid this, I set the default values for each stock to be its value on the previous business day (December 31, 2009).

Step 4: Evaluating model performance

```
test_pred <- knn(predictors[stocksTrain, ], predictors[!stocksTrain, ], cl = stocks$Increase[stocksTrai

CrossTable(x = stocks$Increase[!stocksTrain], y = test_pred, prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  455
##
##
##                              | test_pred
## stocks$Increase[!stocksTrain] |     FALSE |      TRUE | Row Total |
## -----------------------------|-----------|-----------|-----------|
##                        FALSE |        69 |       152 |       221 |
##                              |     0.312 |     0.688 |     0.486 |
##                              |     0.476 |     0.490 |           |
##                              |     0.152 |     0.334 |           |
## -----------------------------|-----------|-----------|-----------|
##                         TRUE |        76 |       158 |       234 |
##                              |     0.325 |     0.675 |     0.514 |
##                              |     0.524 |     0.510 |           |
##                              |     0.167 |     0.347 |           |
## -----------------------------|-----------|-----------|-----------|
##                 Column Total |       145 |       310 |       455 |
##                              |     0.319 |     0.681 |           |
```

```
## -----------------------------|-----------|-----------|-----------|
##
##
```

```
## Accuracy
mean(test_pred == stocks$Increase[!stocksTrain])
```

```
## [1] 0.4989011
```

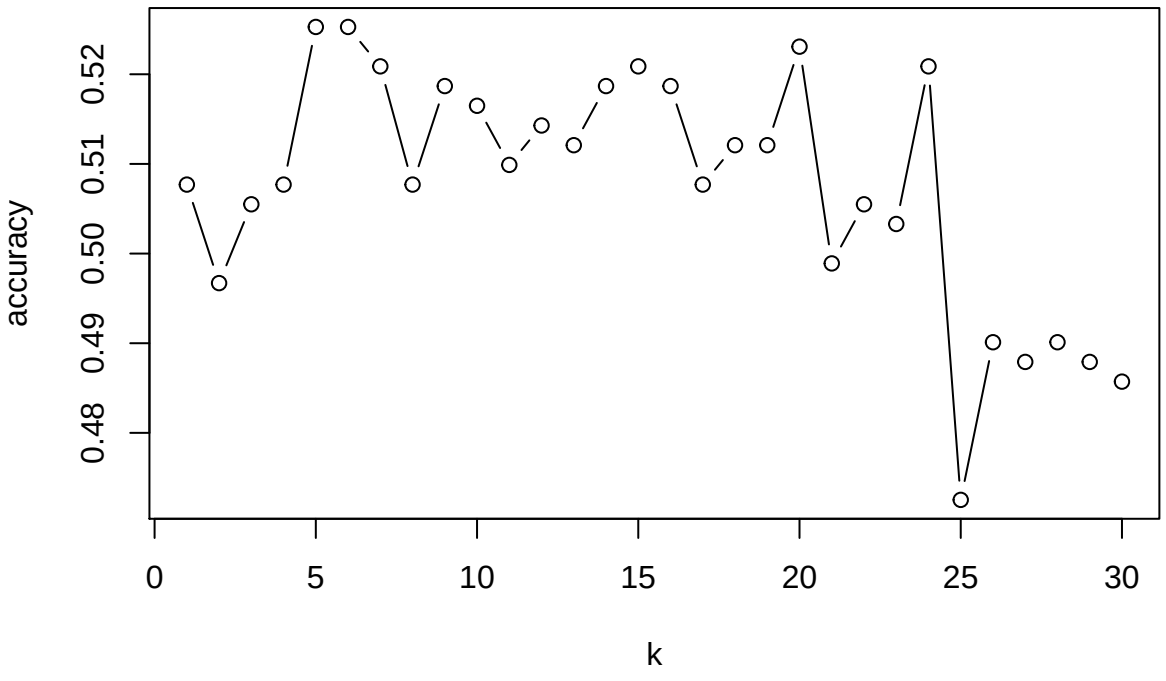
This is only marginally better than random guessing (50%).

Step 5: Improving model performance (improving accuracy by choosing optimum k)

We can use a for loop to see how the algorithm performs for different values of k.

```
accuracy <- rep(0, 10)
k <- 1:30
for (x in k) {
    prediction <- knn(predictors[stocksTrain, ], predictors[!stocksTrain, ],
        stocks$Increase[stocksTrain], k = x)
    accuracy[x] <- mean(prediction == stocks$Increase[!stocksTrain])
}

plot(k, accuracy, type = "b")
```




```
test_pred <- knn(predictors[stocksTrain, ], predictors[!stocksTrain, ], cl = stocks$Increase[stocksTrain],
    k = 5)

CrossTable(x = stocks$Increase[!stocksTrain], y = test_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  455
##
##
##                                | test_pred
## stocks$Increase[!stocksTrain] |     FALSE |      TRUE | Row Total |
## ------------------------------|-----------|-----------|-----------|
##                         FALSE |        70 |       151 |       221 |
##                               |     0.317 |     0.683 |     0.486 |
##                               |     0.519 |     0.472 |           |
##                               |     0.154 |     0.332 |           |
## ------------------------------|-----------|-----------|-----------|
##                          TRUE |        65 |       169 |       234 |
##                               |     0.278 |     0.722 |     0.514 |
##                               |     0.481 |     0.528 |           |
##                               |     0.143 |     0.371 |           |
## ------------------------------|-----------|-----------|-----------|
##                  Column Total |       135 |       320 |       455 |
##                               |     0.297 |     0.703 |           |
## ------------------------------|-----------|-----------|-----------|
##
##
```

```
## Accuracy
mean(test_pred == stocks$Increase[!stocksTrain])
```

```
## [1] 0.5252747
```

As we can see, the model has the highest accuracy of ~52.5% when k = 5. While this may not seem any good, it is often extremely hard to predict the price of stocks. Even the 2.5% improvement over random guessing can make a difference given the amount of money at stake

Question 3

a)

Obs.	X_1	X_2	X_3	Y	Distance from test point $X_1 = X_2 = X_3 = 0$
1	0	3	0	Red	$\sqrt{(0-0)^2) + (0-3)^2 + (0-0)^2)}$ = 3
2	2	0	0	Red	$\sqrt{(0-2)^2) + (0-0)^2 + (0-0)^2)}$ = 2
3	0	1	3	Red	$\sqrt{(0-0)^2) + (0-1)^2 + (0-3)^2)}$ = $\sqrt{10}$ = 3.2
4	0	1	2	Green	$\sqrt{(0-0)^2) + (0-1)^2 + (0-2)^2)}$ = $\sqrt{5}$ = 2.2
5	-1	0	1	Green	$\sqrt{(0+1)^2) + (0-0)^2 + (0-1)^2)}$ = $\sqrt{2}$ = 1.4
6	1	1	1	Red	$\sqrt{(0-1)^2) + (0-1)^2 + (0-1)^2)}$ = $\sqrt{3}$ = 1.7

b) When K = 1, the closest point is observation 5 with Euclidean distance = 1.4. Thus the prediction for Y is Green.

c) When K = 3, the three closest points are observations 5, 6 and 2. Hence the prediction for Y is Red.

d) Small K value is best for non-linear decision boundary because as the value of K increases the boundary becomes more linear.