Nearest Neighbour Machine Learning

Rahul Singh

1/11/2018

R Markdown for Using Nearest Neighbour (Basics)

Nearest Neighbor Classification

(Lazy Algorithm)

Example: k-nearest neighbours k-NN algorithm; the KNN classifier is also a non parametric (so the algorithm makes no explicit assumptions and therefore you need not worry if you end up applying Gaussian model to a non Gaussian dataset; note that this has its own pitfalls) and instance-based (so the algorithm chooses to memorize the training instances which are subsequently used as "knowledge" for the prediction phase).

Use: Classify unlabeled examples by putting them in a class of similarly labeled examples; one of the first choices for a classification study when there is little or no prior knowledge about the distribution of the data OR when relationships among the features and the target classes are complex

When not to use it: When the data is too noisy and thus no clear distinction exists among groups

Pros: simple, effective, does't make any assumptions about data distribution Cons: doesn't produce a model, you need to select the right k, classification phase is slow

Other points: k-NN treats features as coordinates in a multidimensional space

Traditionally k-NN uses Euclidean distance (imagine using a ruler to connect two points/ shortest distance routes)

Trade-off for choosing appropriate k: balance between overfitting and underfitting; large k reduces the impact or variance of noisy data BUT biases the results to ignore small but important patterns. On the contrary, small k value will mean that you may end up letting the noise influence your results. One common practice is to begin with k equal to the square root of the number of training examples. An alternative approach is to test several k values on a variety of test datasets and choose the one that delivers the best classi cation performance.

It is important to make every range comparable; for this we can use min-max normalization (but the limitation of this is that we may not know the minimum and maximum values of future cases, and it may be outside the initially defined max-min range); or another way to transform data is to do z-score standardization with the resulting value for every feature as a Z Score (with the assumption that the future examples will have similar mean and standard deviation as the training examples). For nominal variables (such as male or female) we need to put dummy variables. Also, if you want to fin out if the weather is cold then it is okay to classify cold as 1 and 0 otherwise (hot or medium temp).

Remember: k-means is unsupervised and is used for clustering and that k-NN is supervised

Reference dataset W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993. (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

```
#load libraries
library(ggplot2)
library(base) #to run lapply in this code
```

```
#import file
input_data<-read.csv("~/Desktop/Nearest Neighbour R/Breast Cancer Wisconsin Diagnostic Data Set/wisc_bc
```

```r
#see structure of the input_data
str(input_data)
```

```
## 'data.frame':    569 obs. of  32 variables:
##  $ id                      : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844
##  $ diagnosis               : chr  "M" "M" "M" "M" ...
##  $ radius_mean             : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean            : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean          : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean               : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean         : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean        : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean          : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean     : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean           : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se               : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se              : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se            : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                 : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se           : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se          : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se            : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se       : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se             : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se    : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst            : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst           : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst         : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst              : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst        : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst       : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst         : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst    : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst          : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

```r
View(input_data)
dim(input_data)
```

```
## [1] 569  32
```

```r
head(input_data)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
```

```
## 4         0.14250          0.28390         0.2414             0.10520
## 5         0.10030          0.13280         0.1980             0.10430
## 6         0.12780          0.17000         0.1578             0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave.points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst
## 1     0.03003             0.006193        25.38         17.33
## 2     0.01389             0.003532        24.99         23.41
## 3     0.02250             0.004571        23.57         25.53
## 4     0.05963             0.009208        14.91         26.50
## 5     0.01756             0.005115        22.54         16.67
## 6     0.02165             0.005082        15.47         23.75
##   perimeter_worst area_worst smoothness_worst compactness_worst
## 1          184.60     2019.0           0.1622            0.6656
## 2          158.80     1956.0           0.1238            0.1866
## 3          152.50     1709.0           0.1444            0.4245
## 4           98.87      567.7           0.2098            0.8663
## 5          152.20     1575.0           0.1374            0.2050
## 6          103.40      741.6           0.1791            0.5249
##   concavity_worst concave.points_worst symmetry_worst
## 1          0.7119               0.2654         0.4601
## 2          0.2416               0.1860         0.2750
## 3          0.4504               0.2430         0.3613
## 4          0.6869               0.2575         0.6638
## 5          0.4000               0.1625         0.2364
## 6          0.5355               0.1741         0.3985
##   fractal_dimension_worst
## 1                 0.11890
## 2                 0.08902
## 3                 0.08758
## 4                 0.17300
## 5                 0.07678
## 6                 0.12440
```

```
#summary(input_data)
```

```
#dropping id as it is a unique identifier and has no role to play in the analysis
input_data<- input_data[,-1]
```

```
#to see how many patients have been diagnosis with Benign Cancer and Malignant Cancer
table(input_data$diagnosis)
```

```
##
```

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

```
#to code diagnosis as factors
input_data$diagnosis<- factor(input_data$diagnosis, levels = c("B", "M"), labels = c("Benign", "Malignan
```

```
#to see the percentage of benign and malignant wrt to entire dataset
prop.table(table(input_data$diagnosis))*100
```

```
##
##    Benign Malignant
##  62.74165  37.25835
```

```
#creating normalizing function
normalize <- function (x) {
  return ((x-min(x))/(max(x)-min(x)))
}
```

```
#applying the normalize function
input_data_normalized<- as.data.frame(lapply(input_data[,2:31], normalize)) #remember in normalized dat
#to check if we have normalized out dataset
View(input_data_normalized)
#summary(input_data_normalized)
```

```
#Data preparation into training and test datasets (roughly as 80/20 split of data to train/test)
testing_data<-input_data_normalized[1:100,]
training_data<-input_data_normalized[101:569,]

#we now use the excluded diagnosis column and call them label_train and label_test
label_test<-input_data[1:100,1]
label_train<-input_data[101:569,1]
```

```
#install library(class) for classification
library(class)
#to train the model we use the knn function
predicted_data<-knn(train=training_data,test=testing_data,cl=label_train, k=21) #cl refers to the class
#we chose k as 21 because it is roughlythe square root of 429 (our no. of rows in the training dataset)
```

```
#Evaluating the model performance
#Load gmodels library that provides for CrossTable function which compares two vectors
library(gmodels)
confusion_matrix<-CrossTable(x=label_test, y=predicted_data, prop.chisq = FALSE) #prop.chisq is false s
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
```

```
##              | predicted_data
##   label_test |    Benign | Malignant | Row Total |
## -------------|-----------|-----------|-----------|
##       Benign |        34 |         1 |        35 |
##              |     0.971 |     0.029 |     0.350 |
##              |     0.756 |     0.018 |           |
##              |     0.340 |     0.010 |           |
## -------------|-----------|-----------|-----------|
##    Malignant |        11 |        54 |        65 |
##              |     0.169 |     0.831 |     0.650 |
##              |     0.244 |     0.982 |           |
##              |     0.110 |     0.540 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        45 |        55 |       100 |
##              |     0.450 |     0.550 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
print(confusion_matrix)
```

```
## $t
##            y
## x           Benign Malignant
##   Benign        34         1
##   Malignant     11        54
##
## $prop.row
##            y
## x                Benign  Malignant
##   Benign    0.97142857 0.02857143
##   Malignant 0.16923077 0.83076923
##
## $prop.col
##            y
## x                Benign  Malignant
##   Benign    0.75555556 0.01818182
##   Malignant 0.24444444 0.98181818
##
## $prop.tbl
##            y
## x           Benign Malignant
##   Benign      0.34      0.01
##   Malignant   0.11      0.54
```

```
#In the output you will observe:
#True Negative-> top left cell
#True Positive-> bottom right cell
#False Negative-> lower left cell; errors in this direction are bad!
#False Positive-> top right cell; less dangerous than false negative
cat("We can observe that 12% of 100 test data were incorrectly classified.")
```

```
## We can observe that 12% of 100 test data were incorrectly classified.
```

Variation 1 to improve the model: transformation using Z score standardization

```
input_data_zScore<-as.data.frame(scale(input_data[-1]))
summary(input_data_zScore) #the mean z score should always be zero
```

```
##   radius_mean       texture_mean     perimeter_mean      area_mean
##  Min.   :-2.0279   Min.   :-2.2273   Min.   :-1.9828   Min.   :-1.4532
##  1st Qu.:-0.6888   1st Qu.:-0.7253   1st Qu.:-0.6913   1st Qu.:-0.6666
##  Median :-0.2149   Median :-0.1045   Median :-0.2358   Median :-0.2949
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4690   3rd Qu.: 0.5837   3rd Qu.: 0.4992   3rd Qu.: 0.3632
##  Max.   : 3.9678   Max.   : 4.6478   Max.   : 3.9726   Max.   : 5.2459
##  smoothness_mean    compactness_mean  concavity_mean
##  Min.   :-3.10935   Min.   :-1.6087   Min.   :-1.1139
##  1st Qu.:-0.71034   1st Qu.:-0.7464   1st Qu.:-0.7431
##  Median :-0.03486   Median :-0.2217   Median :-0.3419
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.63564   3rd Qu.: 0.4934   3rd Qu.: 0.5256
##  Max.   : 4.76672   Max.   : 4.5644   Max.   : 4.2399
##  concave.points_mean symmetry_mean      fractal_dimension_mean
##  Min.   :-1.2607     Min.   :-2.74171   Min.   :-1.8183
##  1st Qu.:-0.7373     1st Qu.:-0.70262   1st Qu.:-0.7220
##  Median :-0.3974     Median :-0.07156   Median :-0.1781
##  Mean   : 0.0000     Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.6464     3rd Qu.: 0.53031   3rd Qu.: 0.4706
##  Max.   : 3.9245     Max.   : 4.48081   Max.   : 4.9066
##    radius_se         texture_se       perimeter_se        area_se
##  Min.   :-1.0590   Min.   :-1.5529   Min.   :-1.0431   Min.   :-0.7372
##  1st Qu.:-0.6230   1st Qu.:-0.6942   1st Qu.:-0.6232   1st Qu.:-0.4943
##  Median :-0.2920   Median :-0.1973   Median :-0.2864   Median :-0.3475
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.2659   3rd Qu.: 0.4661   3rd Qu.: 0.2428   3rd Qu.: 0.1067
##  Max.   : 8.8991   Max.   : 6.6494   Max.   : 9.4537   Max.   :11.0321
##  smoothness_se     compactness_se     concavity_se     concave.points_se
##  Min.   :-1.7745   Min.   :-1.2970   Min.   :-1.0566   Min.   :-1.9118
##  1st Qu.:-0.6235   1st Qu.:-0.6923   1st Qu.:-0.5567   1st Qu.:-0.6739
##  Median :-0.2201   Median :-0.2808   Median :-0.1989   Median :-0.1404
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.3680   3rd Qu.: 0.3893   3rd Qu.: 0.3365   3rd Qu.: 0.4722
##  Max.   : 8.0229   Max.   : 6.1381   Max.   :12.0621   Max.   : 6.6438
##   symmetry_se      fractal_dimension_se  radius_worst
##  Min.   :-1.5315   Min.   :-1.0960      Min.   :-1.7254
##  1st Qu.:-0.6511   1st Qu.:-0.5846      1st Qu.:-0.6743
##  Median :-0.2192   Median :-0.2297      Median :-0.2688
##  Mean   : 0.0000   Mean   : 0.0000      Mean   : 0.0000
##  3rd Qu.: 0.3554   3rd Qu.: 0.2884      3rd Qu.: 0.5216
##  Max.   : 7.0657   Max.   : 9.8429      Max.   : 4.0906
##  texture_worst      perimeter_worst     area_worst      smoothness_worst
##  Min.   :-2.22204   Min.   :-1.6919   Min.   :-1.2213   Min.   :-2.6803
##  1st Qu.:-0.74797   1st Qu.:-0.6890   1st Qu.:-0.6416   1st Qu.:-0.6906
##  Median :-0.04348   Median :-0.2857   Median :-0.3409   Median :-0.0468
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
```

```
##  3rd Qu.: 0.65776   3rd Qu.: 0.5398   3rd Qu.: 0.3573   3rd Qu.: 0.5970
##  Max.   : 3.88249   Max.   : 4.2836   Max.   : 5.9250   Max.   : 3.9519
##  compactness_worst concavity_worst   concave.points_worst
##  Min.   :-1.4426   Min.   :-1.3047   Min.   :-1.7435
##  1st Qu.:-0.6805   1st Qu.:-0.7558   1st Qu.:-0.7557
##  Median :-0.2693   Median :-0.2180   Median :-0.2233
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.5392   3rd Qu.: 0.5307   3rd Qu.: 0.7119
##  Max.   : 5.1084   Max.   : 4.6965   Max.   : 2.6835
##  symmetry_worst    fractal_dimension_worst
##  Min.   :-2.1591   Min.   :-1.6004
##  1st Qu.:-0.6413   1st Qu.:-0.6913
##  Median :-0.1273   Median :-0.2163
##  Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4497   3rd Qu.: 0.4504
##  Max.   : 6.0407   Max.   : 6.8408
```

```r
#Data preparation into training and test datasets (roughly as 80/20 split of data to train/test)
testing_data_z<-input_data_zScore[1:100,]
training_data_z<-input_data_zScore[101:569,]

#we now use the excluded diagnosis column and call them label_train and label_test (this is the same as
label_test<-input_data[1:100,1]
label_train<-input_data[101:569,1]
```

```r
#install library(class) for classification
library(class)
#to train the model we use the knn function
predicted_data_z<-knn(train=training_data_z,test=testing_data_z,cl=label_train, k=21)
CrossTable(x=label_test, y=predicted_data_z, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##              | predicted_data_z
##   label_test |    Benign | Malignant | Row Total |
## -------------|-----------|-----------|-----------|
##       Benign |        34 |         1 |        35 |
##              |     0.971 |     0.029 |     0.350 |
##              |     0.756 |     0.018 |           |
##              |     0.340 |     0.010 |           |
## -------------|-----------|-----------|-----------|
##    Malignant |        11 |        54 |        65 |
##              |     0.169 |     0.831 |     0.650 |
```

```
##              |     0.244 |     0.982 |           |
##              |     0.110 |     0.540 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        45 |        55 |       100 |
##              |     0.450 |     0.550 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
cat("The new results are not improving the model as they are similar.")
```

```
## The new results are not improving the model as they are similar.
```

Variation 2 to improve the model: transformation using different k values

(With k=1)

```
#install library(class) for classification
library(class)
#to train the model we use the knn function
predicted_data<-knn(train=training_data,test=testing_data,cl=label_train, k=1) #cl refers to the class
#we chose k as 21 because it is roughlythe square root of 429 (our no. of rows in the training dataset)
```

```
#Evaluating the model performance
#Load gmodels library that provides for CrossTable function which compares two vectors
library(gmodels)
confusion_matrix<-CrossTable(x=label_test, y=predicted_data, prop.chisq = FALSE) #prop.chisq is false s
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##              | predicted_data
##   label_test |    Benign | Malignant | Row Total |
## -------------|-----------|-----------|-----------|
##       Benign |        33 |         2 |        35 |
##              |     0.943 |     0.057 |     0.350 |
##              |     0.917 |     0.031 |           |
##              |     0.330 |     0.020 |           |
## -------------|-----------|-----------|-----------|
##    Malignant |         3 |        62 |        65 |
##              |     0.046 |     0.954 |     0.650 |
##              |     0.083 |     0.969 |           |
```

```
##              |     0.030 |     0.620 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        36 |        64 |       100 |
##              |     0.360 |     0.640 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```r
print(confusion_matrix)
```

```
## $t
##            y
## x           Benign Malignant
##   Benign        33         2
##   Malignant      3        62
##
## $prop.row
##            y
## x               Benign  Malignant
##   Benign    0.94285714 0.05714286
##   Malignant 0.04615385 0.95384615
##
## $prop.col
##            y
## x               Benign  Malignant
##   Benign    0.91666667 0.03125000
##   Malignant 0.08333333 0.96875000
##
## $prop.tbl
##            y
## x           Benign Malignant
##   Benign      0.33      0.02
##   Malignant   0.03      0.62
```

```r
#In the output you will observe:
#True Negative-> top left cell
#True Positive-> bottom right cell
#False Negative-> lower left cell; errors in this direction are bad!
#False Positive-> top right cell; less dangerous than false negative
cat("We can observe that 5% of 100 test data were incorrectly classified.")
```

```
## We can observe that 5% of 100 test data were incorrectly classified.
```

(With k=5)

```r
#install library(class) for classification
library(class)
#to train the model we use the knn function
predicted_data<-knn(train=training_data,test=testing_data,cl=label_train, k=5) #cl refers to the class
#we chose k as 21 because it is roughlythe square root of 429 (our no. of rows in the training dataset)
```

```r
#Evaluating the model performance
#Load gmodels library that provides for CrossTable function which compares two vectors
library(gmodels)
confusion_matrix<-CrossTable(x=label_test, y=predicted_data, prop.chisq = FALSE) #prop.chisq is false s
```

```
##
```

```
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##              | predicted_data
##   label_test |    Benign | Malignant | Row Total |
## -------------|-----------|-----------|-----------|
##       Benign |        33 |         2 |        35 |
##              |     0.943 |     0.057 |     0.350 |
##              |     0.825 |     0.033 |           |
##              |     0.330 |     0.020 |           |
## -------------|-----------|-----------|-----------|
##    Malignant |         7 |        58 |        65 |
##              |     0.108 |     0.892 |     0.650 |
##              |     0.175 |     0.967 |           |
##              |     0.070 |     0.580 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        40 |        60 |       100 |
##              |     0.400 |     0.600 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
print(confusion_matrix)
```

```
## $t
##            y
## x           Benign Malignant
##   Benign        33         2
##   Malignant      7        58
##
## $prop.row
##            y
## x               Benign  Malignant
##   Benign    0.94285714 0.05714286
##   Malignant 0.10769231 0.89230769
##
## $prop.col
##            y
## x               Benign  Malignant
##   Benign    0.82500000 0.03333333
##   Malignant 0.17500000 0.96666667
##
## $prop.tbl
##            y
## x           Benign Malignant
```

```
##   Benign      0.33      0.02
##   Malignant   0.07      0.58
```

```
#In the output you will observe:
#True Negative-> top left cell
#True Positive-> bottom right cell
#False Negative-> lower left cell; errors in this direction are bad!
#False Positive-> top right cell; less dangerous than false negative
cat("We can observe that 9% of 100 test data were incorrectly classified.")
```

```
## We can observe that 9% of 100 test data were incorrectly classified.
```

(With k=11)

```
#install library(class) for classification
library(class)
#to train the model we use the knn function
predicted_data<-knn(train=training_data,test=testing_data,cl=label_train, k=11) #cl refers to the class
#we chose k as 21 because it is roughlythe square root of 429 (our no. of rows in the training dataset)
```

```
#Evaluating the model performance
#Load gmodels library that provides for CrossTable function which compares two vectors
library(gmodels)
confusion_matrix<-CrossTable(x=label_test, y=predicted_data, prop.chisq = FALSE) #prop.chisq is false s
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##              | predicted_data
##   label_test |    Benign | Malignant | Row Total |
## -------------|-----------|-----------|-----------|
##       Benign |        34 |         1 |        35 |
##              |     0.971 |     0.029 |     0.350 |
##              |     0.829 |     0.017 |           |
##              |     0.340 |     0.010 |           |
## -------------|-----------|-----------|-----------|
##    Malignant |         7 |        58 |        65 |
##              |     0.108 |     0.892 |     0.650 |
##              |     0.171 |     0.983 |           |
##              |     0.070 |     0.580 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        41 |        59 |       100 |
##              |     0.410 |     0.590 |           |
## -------------|-----------|-----------|-----------|
##
```

```
##
```

```
print(confusion_matrix)
```

```
## $t
##            y
## x           Benign Malignant
##   Benign        34         1
##   Malignant      7        58
##
## $prop.row
##            y
## x               Benign  Malignant
##   Benign    0.97142857 0.02857143
##   Malignant 0.10769231 0.89230769
##
## $prop.col
##            y
## x               Benign  Malignant
##   Benign    0.82926829 0.01694915
##   Malignant 0.17073171 0.98305085
##
## $prop.tbl
##            y
## x           Benign Malignant
##   Benign      0.34      0.01
##   Malignant   0.07      0.58
```

```
#In the output you will observe:
#True Negative-> top left cell
#True Positive-> bottom right cell
#False Negative-> lower left cell; errors in this direction are bad!
#False Positive-> top right cell; less dangerous than false negative
cat("We can observe that 8% of 100 test data were incorrectly classified.")
```

```
## We can observe that 8% of 100 test data were incorrectly classified.
```

(With k=27)

```
#install library(class) for classification
library(class)
#to train the model we use the knn function
predicted_data<-knn(train=training_data,test=testing_data,cl=label_train, k=27) #cl refers to the class
#we chose k as 21 because it is roughlythe square root of 429 (our no. of rows in the training dataset)
```

```
#Evaluating the model performance
#Load gmodels library that provides for CrossTable function which compares two vectors
library(gmodels)
confusion_matrix<-CrossTable(x=label_test, y=predicted_data, prop.chisq = FALSE) #prop.chisq is false s
```

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

```
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##              | predicted_data
##   label_test |    Benign | Malignant | Row Total |
## -------------|-----------|-----------|-----------|
##       Benign |        34 |         1 |        35 |
##              |     0.971 |     0.029 |     0.350 |
##              |     0.723 |     0.019 |           |
##              |     0.340 |     0.010 |           |
## -------------|-----------|-----------|-----------|
##    Malignant |        13 |        52 |        65 |
##              |     0.200 |     0.800 |     0.650 |
##              |     0.277 |     0.981 |           |
##              |     0.130 |     0.520 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        47 |        53 |       100 |
##              |     0.470 |     0.530 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
print(confusion_matrix)
```

```
## $t
##            y
## x           Benign Malignant
##   Benign        34         1
##   Malignant     13        52
##
## $prop.row
##            y
## x               Benign  Malignant
##   Benign    0.97142857 0.02857143
##   Malignant 0.20000000 0.80000000
##
## $prop.col
##            y
## x               Benign  Malignant
##   Benign    0.72340426 0.01886792
##   Malignant 0.27659574 0.98113208
##
## $prop.tbl
##            y
## x           Benign Malignant
##   Benign      0.34      0.01
##   Malignant   0.13      0.52
```

```
#In the output you will observe:
#True Negative-> top left cell
#True Positive-> bottom right cell
```

```r
#False Negative-> lower left cell; errors in this direction are bad!
#False Positive-> top right cell; less dangerous than false negative
cat("We can observe that 14% of 100 test data were incorrectly classified.")
```

```
## We can observe that 14% of 100 test data were incorrectly classified.
```

Conclusion: We can observe that the kNN algorithm with k=1 yields the best results in terms of predicting whether a cancer is beningn or malignant.