Lab2_Answers

Diala Ezzeddine

October 15, 2018

Contents

Objective: Learn how to apply the K-NN algorithm.

Routine breast cancer screening allows the disease to be diagnosed and treated prior to it causing noticeable symptoms. The process of early detection involves examining the breast tissue for abnormal lumps or masses. If a lump is found, a fine-needle aspiration biopsy is performed, which uses a hollow needle to extract a small sample of cells from the mass. A clinician then examines the cells under a microscope to determine whether the mass is likely to be malignant or benign.

If machine learning could automate the identification of cancerous cells, it would provide considerable benefit to the health system. Automated processes are likely to improve the efficiency of the detection process, allowing physicians to spend less time diagnosing and more time treating the disease. An automated screening system might also provide greater detection accuracy by removing the inherently subjective human component from the process.

We will investigate the utility of machine learning for detecting cancer by applying the k-NN algorithm to measurements of biopsied cells from women with abnormal breast masses.

1 Collecting data

We will utilize the Wisconsin Breast Cancer Diagnostic dataset from the UCI Machine Learning Repository. This data was donated by researchers of the University of Wisconsin and includes the measurements from digitized images of fine-needle aspirate of a breast mass. The values represent the characteristics of the cell nuclei present in the digital image.

2 Exploring and preparing the data

2.1 Exploring the data

Let's explore the data and see whether we can shine some light on the relationships.

In doing so, we will prepare the data for use with the k-NN learning method.

1. Import and read the csv data file.

```
# import the CSV file
wbcd <- read.csv("C:/Users/diala/OneDrive/Documents/R_labs/Data/wisc_bc_data.csv",
                     stringsAsFactors = FALSE);
```

2. 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 ...
```

Using the str(wbcd) command, we can see that the data is structured with 569 examples and 32 features.

3. How to drop the id feature from the analysis? Why to remove it?

The first variable is an integer variable named **id**. As this is simply a unique identifier (ID) for each patient in the data, it does not provide useful information, and we will need to exclude it from the model.

```
wbcd <- wbcd[-1]
```

4. Find the target variable. How many levels this variable have?

The next variable in the str() results, **diagnosis**, is of particular interest as it is the outcome we hope to predict. This feature indicates whether the example is from a benign or malignant mass.

```
table(wbcd$diagnosis)
```

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

The table() output indicates that 357 masses are benign "B" while 212 are malignant "M".

5. Transform this nominal variable to factor and give its levels a better names. What are their correspondent's percentage with 1 decimal place?

```
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
```

When we look at the prop.table() output, we notice that the values have been labeled Benign and Malignant with 62.7 percent and 37.3 percent of the masses, respectively.

The remaining 30 features are all numeric, and as expected, they consist of three different measurements of ten characteristics.

For illustrative purposes, we will only take a closer look at three of these features:

6. Find the summary of the following 3 features: radius_mean, area_mean, and the smoothness_mean.

```
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
```

7. Looking at the features side-by-side, do you notice anything problematic about the values?

Since smoothness ranges from 0.05 to 0.16 and area ranges from 143.5 to 2501.0, the impact of area is going to be much larger than the smoothness in the distance calculation. This could potentially cause problems for our classifier, so we need to apply normalization to rescale the features to a standard range of values.

2.2 Transformation - normalizing numeric data

Let's normalize just the numeric variable.

To normalize these features, we need to create a **normalize()** function in R. This function takes a vector x of numeric values, and for each value in x, subtracts the minimum value in x and divides by the range of values in x. Finally, the resulting vector is returned.

1. Create the **normalize()** function and check if it work correctly using a vector of numbers before applying this function to the whole data.

```
# create normalization function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)));
}

# test normalization function - result should be identical
normalize(c(1, 2, 3, 4, 5))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
normalize(c(10, 20, 30, 40, 50))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

2. To apply the **normalize()** function to the whole data, we can use the **lapply()** function that takes a list and applies a specified function to each list element. As a data frame is a list of equal-length vectors, we can use lapply() to apply normalize() to each feature in the data frame. The final step is to convert the list returned by lapply() to a data frame, using the **as.data.frame()** function. Write the code of this process and rename the new data **wbcd_n**.

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

3. How to check if the normalization is working?

```
# 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
```

2.3 Data preparation - creating training and test datasets

In the absence of new unlabeled data, we can simulate the scenario of testing new unlabeled data by dividing our data into two portions: a training dataset that will be used to build the k-NN model and a test dataset that will be used to estimate the predictive accuracy of the model. We will use the first 469 records for the training dataset and the remaining 100 to simulate new patients.

1. Create those training and test sets named **wbcd_train** and **wbcd_test**.

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

2. Can you store its correspondents target variable in 2 separate vectors? Name them**wbcd_train_labels** and **wbcd_test_labels**.

```
# create labels for training and test data

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

3 Training a model on the data

Equipped with our training data and labels vector, we are now ready to classify our unknown records.

For the k-NN algorithm, the training phase actually involves no model building; the process of training a lazy learner like k-NN simply involves storing the input data in a structured format.

To classify our test instances, we will use a k-NN implementation from the **class** package, which provides a set of basic R functions for classification.

1. How to install the class package in R?

```
#install.packages("class");
```

To load the package during any session in which you wish to use the functions, simply enter the **library(class);** command.

```
library(class);
```

The knn() function in the class package provides a standard, classic implementation of the k-NN algorithm. For each instance in the test data, the function will identify the k-Nearest Neighbors, using Euclidean distance, where k is a user-specified number. The test instance is classified by taking a "vote" among the k-Nearest Neighbors-specifically, this involves assigning the class of the majority of the k neighbors.

Training and classification using the knn() function is performed in a single function call, using four parameters, as shown:

p <- knn(train, test, class, k);

where

- **train** is a data frame containing numeric training data
- **test** is a data frame containing numeric test data
- **class** "cl" is a factor vector with the class for each row in the training data
- **k** is an integer indicating the number of nearest neighbors

The function returns a factor vector of predicted classes for each row in the test data frame.

2. Find k.

As our training data includes 469 instances, we will try k = 21, an odd number roughly equal to the square root of 469. With a two-category outcome, using an odd number eliminates the chance of ending with a tie vote.

3. Use the knn() function from the class package to classify the test data. Name the result vector **wbcd_test_pred**.

```
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test,
                      cl = wbcd_train_labels, k=21);
```

4 Evaluating model performance

The next step of the process is to evaluate how well the predicted classes in the **wbcd_ test_pred** vector matches up with the known values in the **wbcd_test_labels** vector.

1. How to compare **wbcd_ test_pred** vector to **wbcd_test_labels**? **Tip**: Those are 2 categorical variables, how to study the relationship of two categorical variables?

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

```
## Warning: package 'gmodels' was built under R version 3.5.1
```

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

2. What is accuracy of this model?

A total of 2 out of 100, or 2 percent of masses were incorrectly classified by the k-NN approach that means an accuracy of 98 percent.

3. Discuss and explain the results of using this model by looking into the false negative and the false positive numbers.

The cell percentages in the table indicate the proportion of values that fall into four categories: the true negative, the true positive, the false negative and the false positive.

- The top-left cell indicates the **true negative** results. These 61 of 100 values are cases where the mass was benign and the k-NN algorithm correctly identified it as such.
- The bottom-right cell indicates the **true positive** results, where the classifier and the clinically determined label agree that the mass is malignant. A total of 37 of 100 predictions were true positives.

The cells falling on the other diagonal contain counts of examples where the k-NN approach disagreed with the true label.

- The two examples in the lower-left cell are **false negative** results; in this case, the predicted value was benign, but the tumor was actually malignant. Errors in this direction could be extremely costly as they might lead a patient to believe that she is cancer-free, but in reality, the disease may continue to spread.
- The top-right cell would contain the **false positive** results, if there were any. These values occur when the model classifies a mass as malignant, but in reality, it was benign. Although such errors are less dangerous than a false negative result, they should also be avoided as they could lead to additional financial burden on the health care system or additional stress for the patient as additional tests or treatment may have to be provided.

4. Is it possible to improve the performance of this model?

While 98 percent accuracy seems impressive for a few lines of R code, we might try another iteration of the model to see whether we can improve the performance and reduce the number of values that have been incorrectly classified, particularly because the errors were dangerous false negatives.

5 Improving model performance

1. How can you improve the performance of your knn model? list two ways.

We will attempt two simple variations on our previous classifier.

First, we will employ an alternative method for rescaling our numeric features using z-score.

Second, we will try several different values for k.

2. Apply those 2 different ways of improvements. Discuss and explain your results.

- Transformation - z-score standardization

```
# 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.
## -1.4532 -0.6666 -0.2949  0.0000  0.3632  5.2459
```

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

Unfortunately, the results of our new transformation show a slight decline in accuracy. The instances where we had correctly classified 98 percent of examples previously, we classified only 95 percent correctly this time. Making matters worse, we did no better at classifying the dangerous false negatives.

- Testing alternative values of k

```
# try several different values of k
wbcd_train <- wbcd_n[1:469, ]
wbcd_test <- wbcd_n[470:569, ]

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)$t;
```

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

```
##            y
## x           Benign Malignant
##   Benign        58         3
##   Malignant      1        38
```

```
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)$t
```

```
##
##
##    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 |           |
```

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

```
##            y
## x           Benign Malignant
##   Benign        61         0
##   Malignant      2        37
```

```
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)$t
```

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

```
##            y
## x           Benign Malignant
##   Benign        61         0
##   Malignant      3        36
```

```
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)$t
```

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


##            y
## x           Benign Malignant
##   Benign        61         0
##   Malignant      3        36
```

```
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)$t
```

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

```
##            y
## x           Benign Malignant
##   Benign        61         0
##   Malignant      2        37
```

```
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)$t
```

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

```
##            y
## x           Benign Malignant
##   Benign        61         0
##   Malignant      4        35
```

Although the classifier was never perfect, the 1-NN approach was able to avoid some of the false negatives at the expense of adding false positives.

It is important to keep in mind, however, that it would be unwise to tailor our approach too closely to our test data; after all, a different set of 100 patient records is likely to be somewhat different from those used to measure our performance.