USING KNN CLASSIFICATION TO PREDICT BRAST CANCER IN R (WEEK 2 ASSIGNMENT)

Course: ALY 6020

Name: Yuanying Li

Table of content

1.Introduction

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.

In this project, 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.

2. DATA AND PREPROCESSING

2.1 Define the decision variables

We will utilize the Wisconsin Breast Cancer Diagnostic dataset from the UCI Machine Learning Repository at http://archive.ics.uci.edu/ml. The values represent the characteristics of the cell nuclei present in the digital image.

Here is the explanation of features' name in dataset.

- ID number
- Diagnosis (M = malignant, B = benign)
- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter

- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

Based on these names, all the features seem to relate to the shape and size of the cell nuclei. Unless you are an oncologist, you are unlikely to know how each relates to benign or malignant. These patterns will be revealed as we continue in the machine learning process.

2.2 Exploratory Data Analysis

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. We'll begin by importing the CSV data file, as we have done, saving the Wisconsin breast cancer data to the wbcd data frame:

Code snippets:

```
#Load the "wbcd" data and assign it to variable house
wbcd <- read.csv("bresastcancer_data.csv")
View(wbcd)

#Check Null values Number
sum(is.na(wbcd))

#Prodide information about the structure of housing dataset
#This data set contains 569 observation and 32 variables
str(wbcd)
```

Results:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
1	842302	M	17.990	10.38	122.80	1001.0	0.11840
2	842517	M	20.570	17.77	132.90	1326.0	0.08474
3	84300903	M	19.690	21.25	130.00	1203.0	0.10960
4	84348301	M	11.420	20.38	77.58	386.1	0.14250

```
> sum(is.na(wbcd))
[1] 0
> #Prodide information about the structure of housing dataset
> #This data set contains 569 observation and 32 variables
> str(wbcd)
'data.frame':   569 obs. of  32 variables:
 $ id                     : int  842302 842517 84300903 84348301 84358402 843786
 84458202 844981 84501001 ...
 $ diagnosis              : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
 $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
```

Observations:

1. Sample size is not that large (within 569), with 32 variables, the dependent variable is called diagnosis. There is not missing value in this dataset.

2. 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.

2.3 Data preprocessing

In this study we firstly focus on 30 factors that possibly influence Breast Cancer, as we mentioned before, ID is not necessary to keep for predicting the final diagnosis, so that we could just drop it.

What's more, we could also rename the diagnosis's result, replacing B to Benign and M to Malignant, so it is readable in the model. Here is the new dataset.

Code snippets:

```
#Drop some features that is not related to model
Drop <- names(wbcd) %in% c("id")
wbcd <- wbcd[!Drop]
View(wbcd)

#Rename the diagnosis's values
wbcd$diagnosis<- factor(wbcd$diagnosis, levels = c("B", "M"),labels = c("Be
```

Results:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactnes
1	Malignant	17.990	10.38	122.80	1001.0	0.11840	0.27760
2	Malignant	20.570	17.77	132.90	1326.0	0.08474	0.07864

2.4 Date visualization

2.4.1 Diagnosis analysis

Diagnosis is of particular interest as it is the outcome we hope to predict. We could calculate the proportion of diagnosis result to see distribution of benign or malignant mass.

Code snippets:

```
#Compute the proportion of Benign and Maliganant in diagnosis
table(wbcd$diagnosis)
round(prop.table(table(wbcd$diagnosis)) * 100, digits = 1)
```

Results:

```
   Benign Malignant
      357       212
> round(prop.table(table(wbcd$diagnosis)) * 100, digits = 1)

   Benign Malignant
     62.7      37.3
```

Observation:

1. The table() output indicates that 357 masses are benign while 212 are malignant. Now, 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. it shows in real life, Breast Cancer is big issue among human being we couldn't ignore this disease and should take it seriously.

2.4.2 Summary Data

Use the summary command to calculate each feature's mathematics detail.

Code snippets:

```
#Data summary
summary(wbcd)
```

Results:

```
> #Data summary
> summary(wbcd)
     diagnosis    radius_mean      texture_mean   perimeter_mean     area_mean
 Benign   :357   Min.   : 6.981   Min.   : 9.71   Min.   : 43.79   Min.   : 143.5
 Malignant:212   1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17   1st Qu.: 420.3
                 Median :13.370   Median :18.84   Median : 86.24   Median : 551.1
                 Mean   :14.127   Mean   :19.29   Mean   : 91.97   Mean   : 654.9
                 3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10   3rd Qu.: 782.7
                 Max.   :28.110   Max.   :39.28   Max.   :188.50   Max.   :2501.0
 smoothness_mean   compactness_mean   concavity_mean    concave points_mean
```

Observation:

1.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 four of these features. It is a little hard to see what the relationship between breast cancer and those measurements, let's use KNN classification to predict the final result.

2. In addition, Looking at the features side-by-side, we could notice anything problematic about the values. Recall that the distance calculation for k-NN is heavily dependent upon the measurement scale of the input features. Since the range of radius_mean is from 6.981 to 28.110 and the range of area_mean is from 143.5 to 2501.0, the impact of area is going to be much larger than the radius in the distance calculation. This could potentially cause problems for our classifier, so let's apply normalization to rescale the features to a standard range of values.

3.BUILD THE MODEL

3.1 Data Transformation – normalizing numeric data

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.

After executing the preceding code, the normalize() function is available for use in R. . The code for this function and test is as follows:

Code snippets:

```
#Apply normalization to rescale the features to standard range of values.
#Create a normaliza() function in R
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

#Test the function
normalize(c(10, 20, 30, 40, 50))
```

Results:

```
> normalize(c(10, 20, 30, 40, 50))
[1] 0.00 0.25 0.50 0.75 1.00
```

```
#Apply the normalize() function to the numeric features in our data frame
wbcd_n <- as.data.frame(lapply(wbcd[2:31], normalize))
summary(wbcd_n)
```

```
> summary(wbcd_n)
  radius_mean      texture_mean    perimeter_mean     area_mean
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.2233   1st Qu.:0.2185   1st Qu.:0.2168   1st Qu.:0.1174
 Median :0.3024   Median :0.3088   Median :0.2933   Median :0.1729
 Mean   :0.3382   Mean   :0.3240   Mean   :0.3329   Mean   :0.2169
 3rd Qu.:0.4164   3rd Qu.:0.4089   3rd Qu.:0.4168   3rd Qu.:0.2711
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

The function appears to be working correctly. We can now apply the normalize() function to the numeric features in our data frame. Rather than normalizing each of the 30 numeric variables individually, we will use one of R's functions to automate the process.

The lapply() function 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. The full process looks like this:

Code snippets:

```
#Apply the normalize() function to the numeric features in our data frame
wbcd_n <- as.data.frame(lapply(wbcd[2:31], normalize))
summary(wbcd_n)
```

Results:

```
> summary(wbcd_n)
  radius_mean      texture_mean    perimeter_mean     area_mean
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.2233   1st Qu.:0.2185   1st Qu.:0.2168   1st Qu.:0.1174
 Median :0.3024   Median :0.3088   Median :0.2933   Median :0.1729
 Mean   :0.3382   Mean   :0.3240   Mean   :0.3329   Mean   :0.2169
 3rd Qu.:0.4164   3rd Qu.:0.4089   3rd Qu.:0.4168   3rd Qu.:0.2711
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Observation:

1. After using min-max normalization to rescale the features and this process transforms the feature such that all of its values fall in a range between 0 and 1.

3.2 Use KNN Classification

3.2.1 Split train dataset and test train

In this section, we can simulate this scenario 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.

Using the data extraction methods, Managing and Understanding Data, we will split the wbcd_n data frame into wbcd_train and wbcd_test:

Code snippets:

```
#TRAIN, TEST & SPLIT
#Data splicing basically involves splitting the data set into trair
wbcd_train <- wbcd_n[1:469, ]
wbcd_test <- wbcd_n[470:569, ]


#After deriving the training and testing data set,
#the below code snippet is going to create a separate data frame
#for the 'diagnosis' variable so that our
#final outcome can be compared with the actual value.
wbcd_train_labels <- wbcd[1:469, 1]
wbcd_test_labels <- wbcd[470:569, 1]
```

3.2.2 Train the model

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

We now have nearly everything that we need to apply the k-NN algorithm to this data. We've split our data into training and test datasets, each with exactly the same numeric features. The labels for the training data are stored in a separate factor vector. The only remaining parameter is k, which specifies the number of neighbors to include in the vote.

As our training data includes 469 instances, we might 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.

Now we can use the knn() function to classify the test data:

Code snippets:

```
#Apply KNN classfication
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test,
                      cl = wbcd_train_labels, k = 21)
```

3.3 Evaluation Model

After using a classification, evaluation is one of the important parts to do, so we predict the diagnosis from test model and check the accuracy comparing with the real price and visualize the residuals through plot the difference

Code snippets:

```
#Compare the true values with prediciton value using crosstable function
validation_table <- CrossTable(x = wbcd_test_labels,
                               y = wbcd_test_pred, prop.chisq=FALSE)
validation_table
```

Result:

```
Total Observations in Table:  100
```

True negeative

wbcd_test_labels	wbcd_test_pred: Benign	Malignant	Row Total
Benign	77	0	77
	1.000	0.000	0.770
	0.975	0.000	
	0.770	0.000	
Malignant	2	21	23
	0.087	0.913	0.230
	0.025	1.000	
	0.020	0.210	
Column Total	79	21	100
	0.790	0.210	

True positive

```
> #Evaluate the classfication
> accuracy=mean(wbcd_test_pred==wbcd_test_labels)
> print("Accuracy:")
[1] "Accuracy:"
> print(accuracy)
[1] 0.98
```

```r
#Evaluate the classfication
accuracy=mean(wbcd_test_pred==wbcd_test_labels)
print("Accuracy:")
print(accuracy)
```

Observation:

1. The cell percentages in the table indicate the proportion of values that fall into four categories. The top-left cell indicates the true negative results. These 77 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 21 of 100 predictions were true positives.

2. 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.

3. The accuracy of this classification is 98%.

4.CONCLUSION

In our project, We used KNN above to predict the diagnosis of breast cancer, A total of 2 out of 100, or 2 percent of masses were incorrectly classified by the k-NN approach. 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.REFERENCE

1. Breast Cancer Diagnosis Using KNN with R

https://www.kaggle.com/diwash1/breast-cancer-diagnosis-using-knn-with-r

2. Credit Card Fraud Detection: KNN & Naive Bayes

https://www.kaggle.com/yuridias/credit-card-fraud-detection-knn-naive-bayes