Applying classification method in Machine Learning to determine the disorder based on test reports of patients in predicting Thyroid disease infections.

Nguyen Tran

August 2020

This project is coded in ***R language*** to build a model that can classify whether a patient has infection of Thyroid disease based on their test report. We obtain the dataset "Thyroid_data" from a test of Thyroid disease infections on selected patients. Thyroid is a major disorder that occurs due to the lack of the Thyroid hormone among women. It is not easy to determine the disorder manually based on the patients' test reports. However, using a machine learning technique would make this task simpler by predicting the disorder.

Our task is to detect whether a given patient is healthy (1) or suffers from hyperthyroidism (2) or suffers from hypothyroidism (3) by developing a model using classification. This model should in the future help to automatize the decisions on the patients who have a certain type of Thyroid disease (2)/ (3) or are healthy. The information of the features in the dataset are as follows:

− CLASS: class attribute (healthy (1) or hyperthyroidism (2) or hypothyroidism (3))

− T3: T3-resin uptake test results

− TST: Total Serum thyroxin as measured by the isotopic displacement method

− TSTR: Total serum triiodothyronine as measured by radioimmunoassay

− TSH: Basal thyroid-stimulating hormone (TSH) as measured by radioimmunoassay

− MAD-TSH: Maximal absolute difference of TSH value after injection of 200 micro grams of thyrotropin-releasing hormone as compared to the basal value

Descriptive Statistics

The dataset is loaded with its structure as the following:

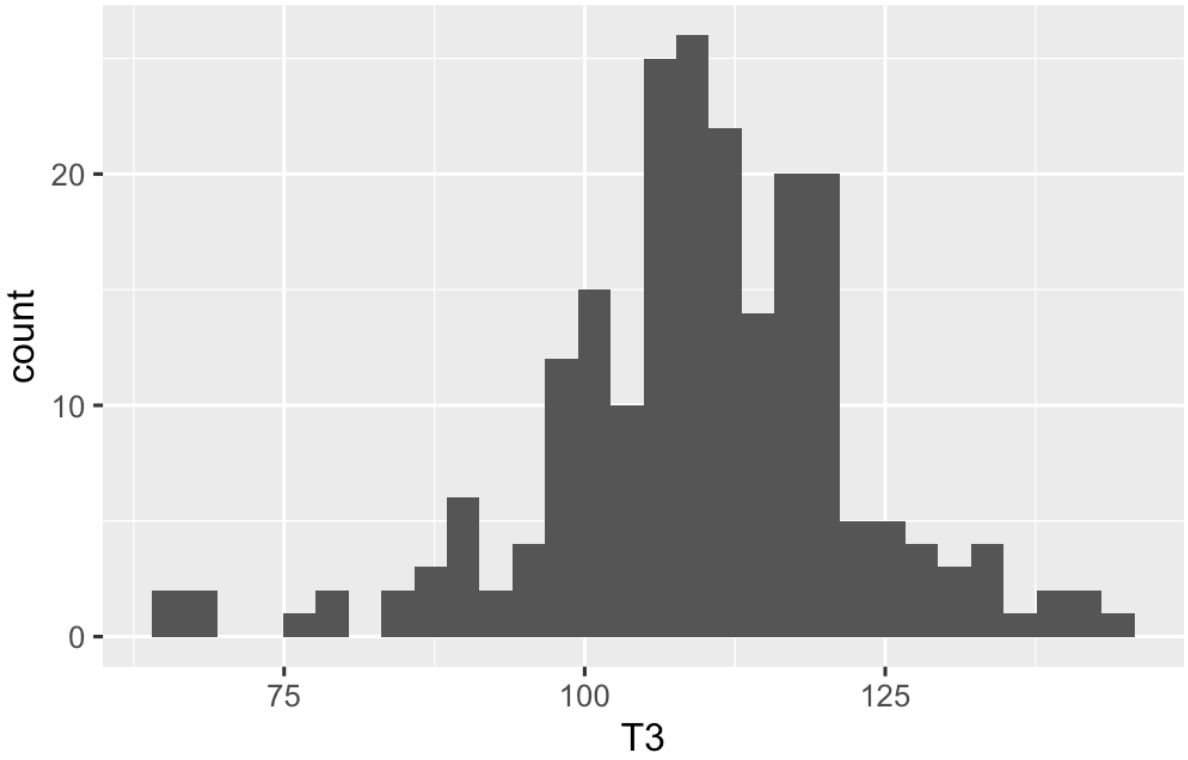
```
> mydata<-read.csv('Thyroid_data.csv',header=TRUE,sep=',')
> str(mydata)
'data.frame':   215 obs. of  6 variables:
 $ CLASS  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ T3     : int  107 113 127 109 105 105 110 114 106 107 ...
 $ TST    : num  10.1 9.9 12.9 5.3 7.3 6.1 10.4 9.9 9.4 13 ...
 $ TSTR   : num  2.2 3.1 2.4 1.6 1.5 2.1 1.6 2.4 2.2 1.1 ...
 $ TSH    : num  0.9 2 1.4 1.4 1.5 1.4 1.6 1.5 1.5 0.9 ...
 $ MAD.TSH: num  2.7 5.9 0.6 1.5 -0.1 7 2.7 5.7 0 3.1 ...
```

This data set has 6 variables and 215 observations. All of the variables are quantitative type. While values of the first two variables (CLASS and T3) are discrete numbers, value of the other variables (TST, TSTR, TSH, MAD.TSH) are continuous numbers.

All variables are plotted in a histogram for easy study.

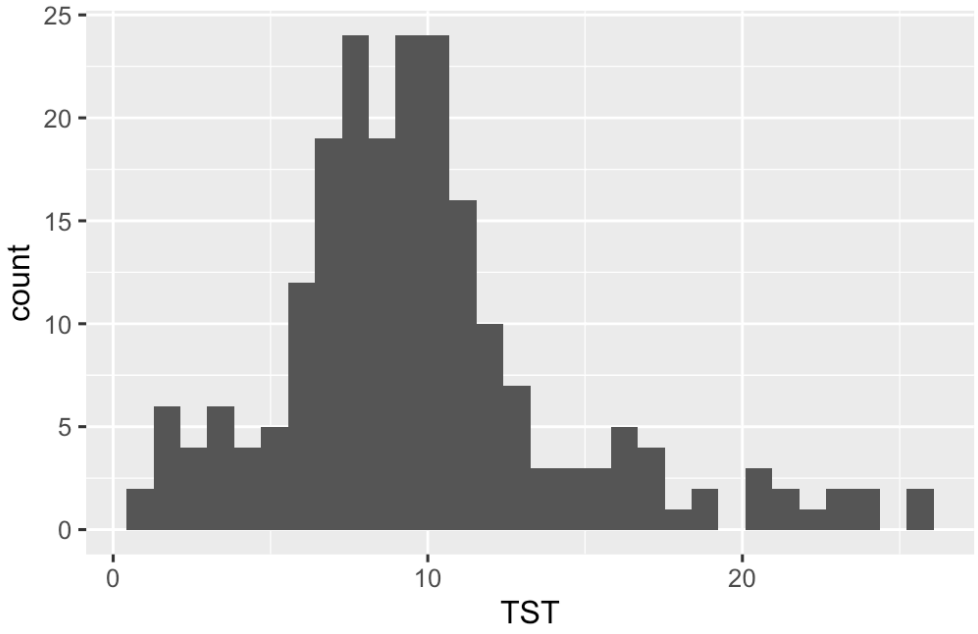
1. T3 variable

As we can see, the graph seems to be symmetric.



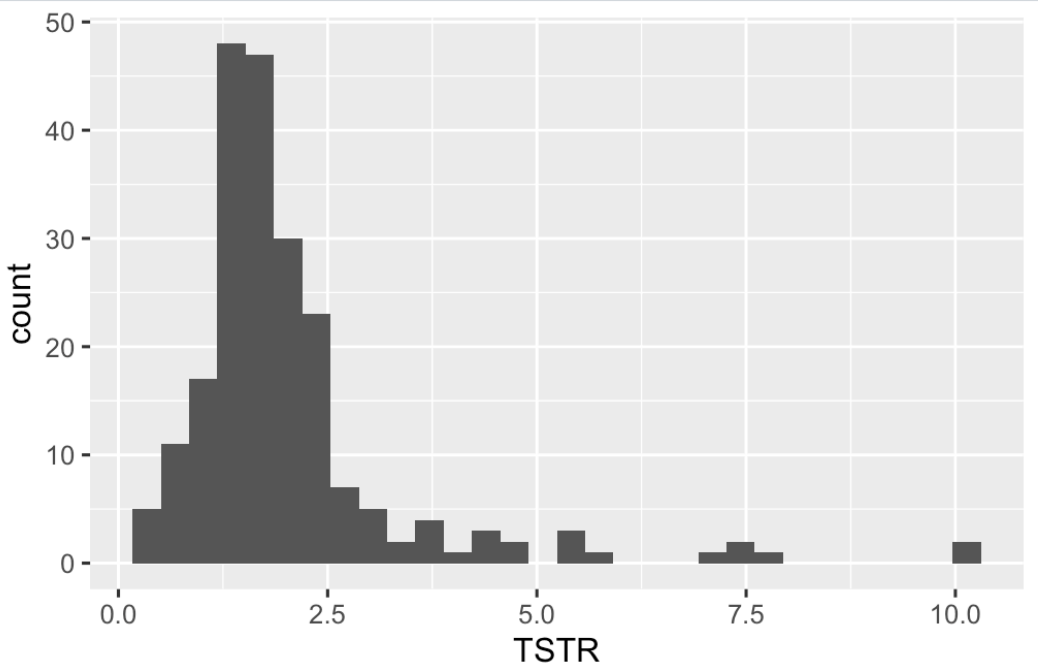
2. TST variable

The graph seems to be asymmetric, right-skewed.



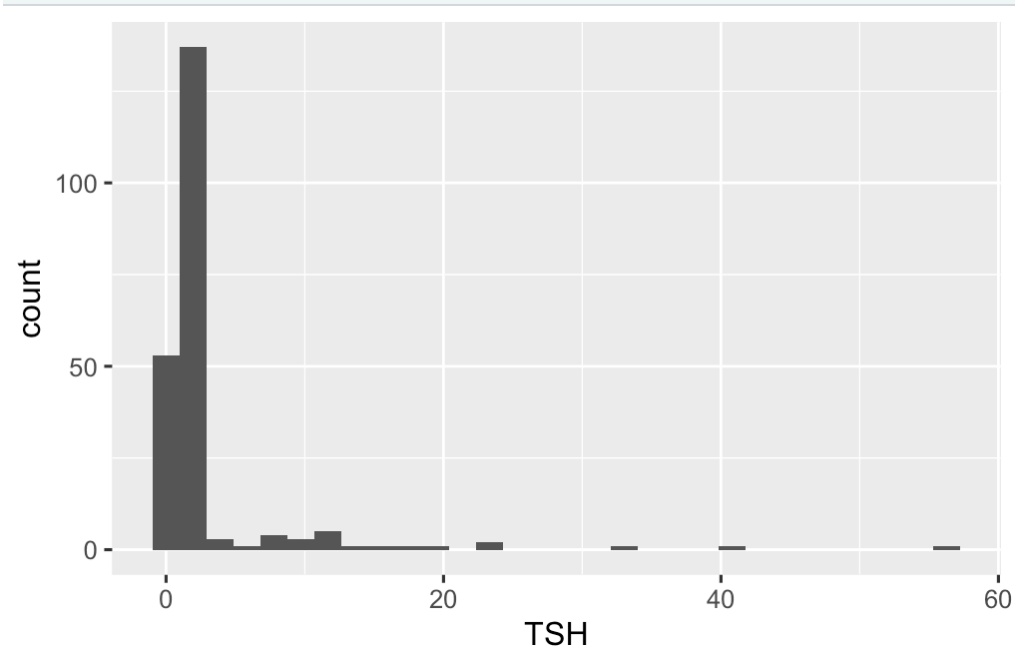
3. TSTR variable

This graph is also asymmetric, right-skewed. The value around 1.25 has the highest count number.



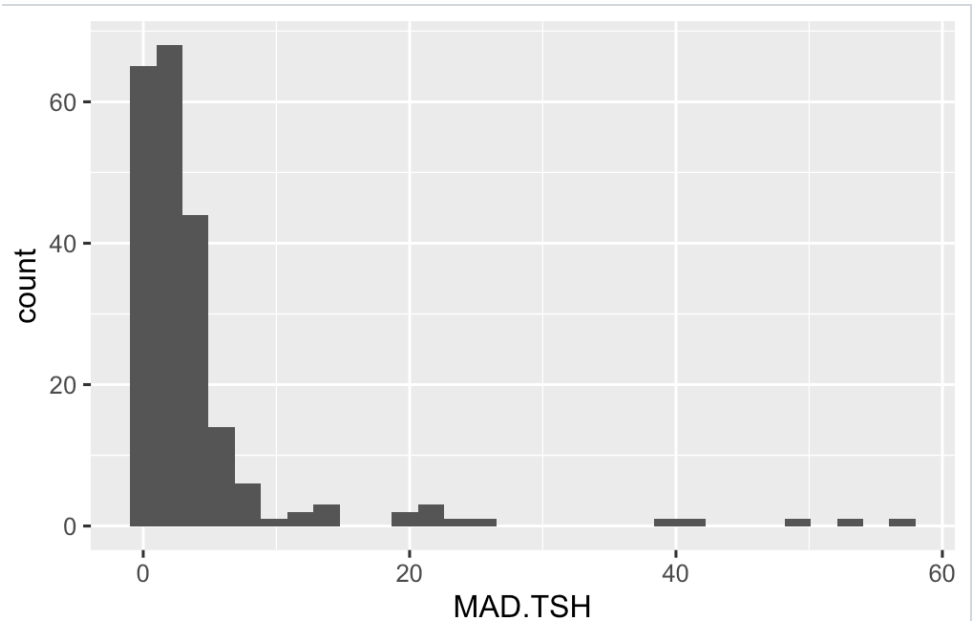
4. TSH variable

The graph below is super asymmetric.



5. MAD.TSH variable

It seems like that the graphs of most variables are asymmetric and right-skewed.



Split train and test data

We use **table** ( ) function to classify Class and its proportion. The table separate the data set in to 3 classes: 1,2 and 3, in which class 1 is indicated as healthy, class 2 and 3 are categorised as suffered from hyperthyroidism and hypothyroidism respectively. Their proportions are 0.70, 0.16 and 0.14.

```
> table(mydata$CLASS)

  1   2   3
150  35  30
> prop.table(table(mydata$CLASS))

        1         2         3
0.6976744 0.1627907 0.1395349
```

We normalise the data using the min-max method.

```
mydata1<-select(mydata,T3,TST,TSTR,TSH,MAD.TSH)
mydata2<-select(mydata,CLASS)
mydata1=apply(mydata1,2,rescale,to=c(0,1))
mydata3<-data.frame(mydata2,mydata1)
```

Now, the stratified random sampling will be applied, divide data for training and testing as 70% of the data is training and 30% of the data is testing. Then we use **subset** ( ) **function** have smaller dataset.

```
sample<-sample.split(mydata$CLASS,SplitRatio=0.7)
train<-subset(mydata3,sample==TRUE)
test<-subset(mydata3,sample==FALSE)
```

We check the proportion of classes in each training data set and testing data set. As we can see, the proportion of classes in training data set and testing data set are the same, it's even equal to the proportion of the original data set.

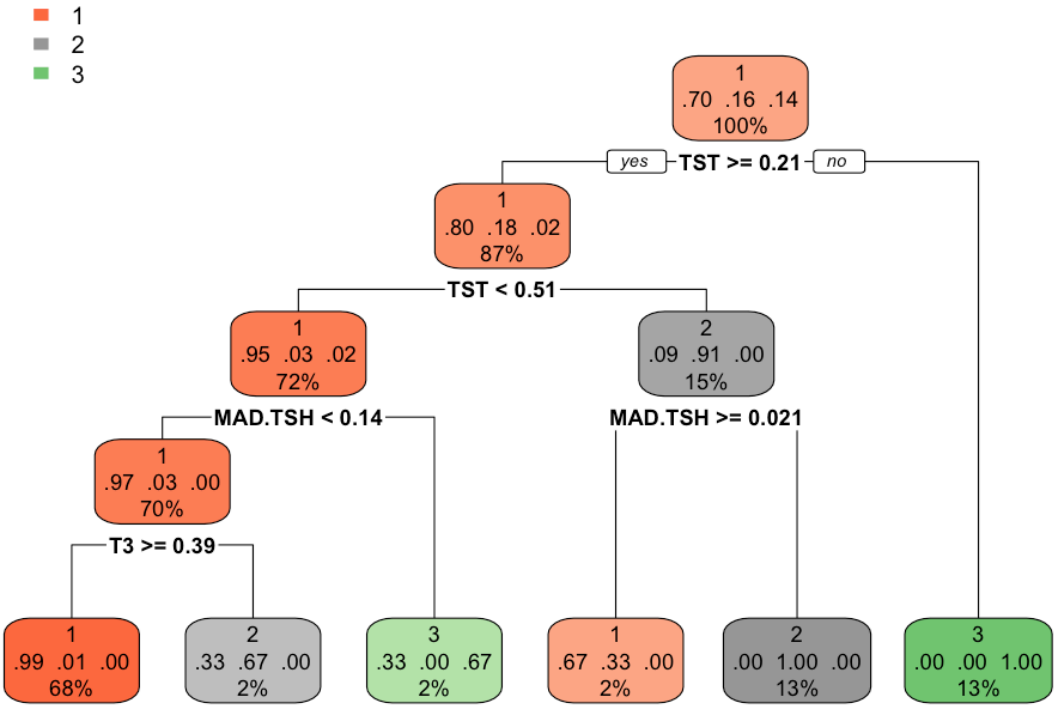
```
> prop.table(table(train$CLASS))

   1    2    3
0.70 0.16 0.14
> prop.table(table(test$CLASS))

        1         2         3
0.6923077 0.1692308 0.1384615
```

Now we conduct the two classification models (decision tree and k-nearest neighbour)

Decision tree method.

```
treemdel1<-rpart(CLASS~.,data=train,method='class')
rpart.plot(treemdel1)
```

The plot we have is:



Now we run the **predict ( )** function. The result is

```
> treeTest
  2   4   5  11  13  15  18  34  39  40  41  43  44  45  46  47  48  50  53
  1   3   1   3   1                                   1   1   1   1   3   1
 54  57  60  64  65  7                                00 108 109 115 118 123
  1   2   1   1   1                                   1   1   1   1   1   1
126 136 137 138 140 145 150 153 159 161 163 165 166 167 169 172 173 184 187
  1   1   1   1   1   2   1   2   2   2   2   2   1   2   1   2   2   2   3
188 190 194 195 204 206 209 215
  3   3   3   3   3   3   3   3
Levels: 1 2 3
```

treemdel1

rpart.rules(treemdel1)

According to the plot we have above, we can use **rpart.rules** ( ) function to see the rules of the decision tree.

```
> treemdel1
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 150 45 1 (0.700000000 0.160000000 0.140000000)
   2) TST>=0.2076613 131 26 1 (0.801526718 0.183206107 0.015267176)
     4) TST< 0.5120968 108  5 1 (0.953703704 0.027777778 0.018518519)
       8) MAD.TSH< 0.1412281 105  3 1 (0.971428571 0.028571429 0.000000000)
        16) T3>=0.3924051 102  1 1 (0.990196078 0.009803922 0.000000000) *
        17) T3< 0.3924051 3  1 2 (0.333333333 0.666666667 0.000000000) *
       9) MAD.TSH>=0.1412281 3  1 3 (0.333333333 0.000000000 0.666666667) *
     5) TST>=0.5120968 23  2 2 (0.086956522 0.913043478 0.000000000)
      10) MAD.TSH>=0.02105263 3  1 1 (0.666666667 0.333333333 0.000000000) *
      11) MAD.TSH< 0.02105263 20  0 2 (0.000000000 1.000000000 0.000000000) *
   3) TST< 0.2076613 19  0 3 (0.000000000 0.000000000 1.000000000) *
```

The rule of classification is:

```
> rpart.rules(treemdel1)
 CLASS     1    2    3
     1 [ .67  .33  .00] when TST >=         0.51 & MAD.TSH >= 0.021
     1 [ .99  .01  .00] when TST is 0.21 to 0.51 & MAD.TSH <  0.141 & T3 >= 0.39
     2 [ .33  .67  .00] when TST is 0.21 to 0.51 & MAD.TSH <  0.141 & T3 <  0.39
     2 [ .00 1.00  .00] when TST >=         0.51 & MAD.TSH <  0.021
     3 [ .33  .00  .67] when TST is 0.21 to 0.51 & MAD.TSH >= 0.141
     3 [ .00  .00 1.00] when TST <  0.21
```

Next, we go to the k-Nearest neighbour (KNN) method.

KNN method

```
knnTest=knn(train[,2:6],test[,2:6],as.factor(train$CLASS),20)
```

The result of knn is:

```
> knnTest
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 2 1 2 2 2 3 1 3 3 3 3 3 3 1
Levels: 1 2 3
```

Comparison 2 performances.

The result of decision tree is

```
> treeTest
  2   4   5  11  13  15  18  34  39  40  41  43  44  45  46  47  48  50  53 
  1   3   1   3   1   1   1   2   1   1   1   1   1   1   1   1   1   3   1 
 54  57  60  64  65  71  74  80  83  89  91  96  99 100 108 109 115 118 123 
  1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
126 136 137 138 140 145 150 153 159 161 163 165 166 167 169 172 173 184 187 
  1   1   1   1   1   2   1   2   2   2   2   2   1   2   1   2   2   2   3 
188 190 194 195 204 206 209 215 
  3   3   3   3   3   3   3   3 
Levels: 1 2 3
```

and the result of knn is


```
> knnTest
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 2 1 2 2 2 3 1 3 3 3 3 3 3 1
Levels: 1 2 3
```


Now we run the **table ( )** function of those 2 results. We can see clearly that with decision tree, the data set after classified distributed better than the one with knn test.

```
> table(treeTest)
treeTest
 1  2  3 
41 12 12 
> table(knnTest)
knnTest
 1  2  3 
51  7  7 
```

The performance of the decision method with the best performing parameter is not a good indicator of the new performance on new observations that are not included in any of the two data sets. Because, if the performing parameter is best fitted with all the variations, even noise, it will be overfitted, which will not be able to generalise in new observations. The performing parameter needs to be just right to be fitted. It is not too simple to be under fitted nor too complex to be overfitted. Just right is enough.

The optimal value for k based on KNN classification, for me, can be identified through different methods, such as Elbow method, Silhouette method, GAP statistics, Calinski-Harabasz Index.

However, based on the way the method executed, the k number should be not too small nor too big. It just needs to be in the middle and neutral enough to classify new observations.

Now, we redo the KNN classification for the original data (without scaling)

```
sample0<-sample.split(mydata$CLASS,SplitRatio=0.7)
train0<-subset(mydata,sample==TRUE)
test0<-subset(mydata,sample==FALSE)
knnTest0=knn(train0[,2:6],test0[,2:6],as.factor(train0$CLASS),20)
```

With the original data, the result is not well distributed as the one normalised

```
> knnTest0
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[37] 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 2 2 2 3 1 1 3 3 3 3 3 1
Levels: 1 2 3
```

In conclusion, the methods applied is to help people in the medical field to have better judgement and make better decision according to data and analysis and the system.

To sum up, the findings are:

- Data analysis of all variables given in histogram.
- The original dataset is split into train and test data.
- Based in train and test data, we built models to classify patients with or without thyroid disease.
- We compared the two models.
- Also built model based on original dataset without scaling in KNN method and compare.
- Methods for choosing the optimal value of k neighbours.
- The models need to be rechecked to make sure the they are correctly fitted, not overrated nor underrated.