KNN

Faith

9/7/2020

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
set.seed(1234)
```

Randomizing the rows, creates a uniform distribution of 150

```
random <- runif(150)
iris_random <- iris[order(random),]
```

```
head(iris_random)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 7            4.6         3.4          1.4         0.3     setosa
## 64           6.1         2.9          4.7         1.4 versicolor
## 73           6.3         2.5          4.9         1.5 versicolor
## 98           6.2         2.9          4.3         1.3 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
## 110          7.2         3.6          6.1         2.5  virginica
```

Normalizing numerical variables of the dataset. We define a normal function which will normalize the set of values according to its minimum value and maximum value.

```
normal <- function(x) (
  return( ((x - min(x)) /(max(x)-min(x))) ))
```

```
normal(1:5)
```

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

```r
iris_new <- as.data.frame(lapply(iris_random[,-5],normal))
iris_new
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     0.08333333  0.58333333   0.06779661  0.08333333
## 2     0.50000000  0.37500000   0.62711864  0.54166667
## 3     0.55555556  0.20833333   0.66101695  0.58333333
## 4     0.52777778  0.37500000   0.55932203  0.50000000
## 5     0.55555556  0.54166667   0.84745763  1.00000000
## 6     0.80555556  0.66666667   0.86440678  1.00000000
## 7     0.22222222  0.54166667   0.11864407  0.16666667
## 8     0.44444444  0.41666667   0.54237288  0.58333333
## 9     0.11111111  0.50000000   0.10169492  0.04166667
## 10    0.52777778  0.08333333   0.59322034  0.58333333
## 11    0.66666667  0.41666667   0.67796610  0.66666667
## 12    0.75000000  0.50000000   0.62711864  0.54166667
## 13    0.58333333  0.37500000   0.55932203  0.50000000
## 14    0.58333333  0.29166667   0.72881356  0.75000000
## 15    0.36111111  0.29166667   0.54237288  0.50000000
## 16    0.22222222  0.62500000   0.06779661  0.04166667
## 17    0.44444444  0.50000000   0.64406780  0.70833333
## 18    0.38888889  0.20833333   0.67796610  0.79166667
## 19    0.94444444  0.25000000   1.00000000  0.91666667
## 20    0.58333333  0.33333333   0.77966102  0.87500000
## 21    0.83333333  0.37500000   0.89830508  0.70833333
## 22    0.19444444  0.12500000   0.38983051  0.37500000
## 23    0.61111111  0.41666667   0.81355932  0.87500000
## 24    0.41666667  0.25000000   0.50847458  0.45833333
## 25    0.41666667  0.29166667   0.49152542  0.45833333
## 26    0.61111111  0.33333333   0.61016949  0.58333333
## 27    0.16666667  0.20833333   0.59322034  0.66666667
## 28    0.08333333  0.66666667   0.00000000  0.04166667
## 29    0.36111111  0.41666667   0.52542373  0.50000000
## 30    0.33333333  0.25000000   0.57627119  0.45833333
## 31    0.63888889  0.37500000   0.61016949  0.50000000
## 32    0.16666667  0.45833333   0.08474576  0.04166667
## 33    0.38888889  0.75000000   0.11864407  0.08333333
## 34    0.30555556  0.41666667   0.59322034  0.58333333
## 35    0.33333333  0.62500000   0.05084746  0.04166667
## 36    0.55555556  0.37500000   0.77966102  0.70833333
## 37    0.13888889  0.58333333   0.15254237  0.04166667
## 38    0.41666667  0.33333333   0.69491525  0.95833333
## 39    0.22222222  0.75000000   0.08474576  0.08333333
## 40    0.19444444  0.58333333   0.08474576  0.04166667
## 41    0.36111111  0.37500000   0.44067797  0.50000000
## 42    0.27777778  0.70833333   0.08474576  0.04166667
## 43    0.52777778  0.33333333   0.64406780  0.70833333
## 44    0.16666667  0.66666667   0.06779661  0.00000000
## 45    0.30555556  0.58333333   0.08474576  0.12500000
## 46    0.22222222  0.62500000   0.06779661  0.08333333
## 47    0.94444444  0.75000000   0.96610169  0.87500000
## 48    0.77777778  0.41666667   0.83050847  0.83333333
## 49    0.13888889  0.41666667   0.06779661  0.00000000
```

```
## 50    0.80555556  0.50000000   0.84745763  0.70833333
## 51    0.30555556  0.79166667   0.05084746  0.12500000
## 52    0.41666667  0.83333333   0.03389831  0.04166667
## 53    0.38888889  0.37500000   0.54237288  0.50000000
## 54    0.22222222  0.70833333   0.08474576  0.12500000
## 55    0.25000000  0.87500000   0.08474576  0.00000000
## 56    0.36111111  0.41666667   0.59322034  0.58333333
## 57    0.22222222  0.20833333   0.33898305  0.41666667
## 58    0.58333333  0.50000000   0.59322034  0.58333333
## 59    0.55555556  0.12500000   0.57627119  0.50000000
## 60    0.02777778  0.50000000   0.05084746  0.04166667
## 61    0.30555556  0.58333333   0.11864407  0.04166667
## 62    0.47222222  0.08333333   0.50847458  0.37500000
## 63    0.80555556  0.41666667   0.81355932  0.62500000
## 64    0.47222222  0.37500000   0.59322034  0.58333333
## 65    0.41666667  0.29166667   0.69491525  0.75000000
## 66    0.91666667  0.41666667   0.94915254  0.83333333
## 67    0.22222222  0.75000000   0.15254237  0.12500000
## 68    0.66666667  0.54166667   0.79661017  1.00000000
## 69    0.69444444  0.33333333   0.64406780  0.54166667
## 70    0.66666667  0.45833333   0.62711864  0.58333333
## 71    0.55555556  0.33333333   0.69491525  0.58333333
## 72    0.66666667  0.20833333   0.81355932  0.70833333
## 73    0.13888889  0.45833333   0.10169492  0.04166667
## 74    0.94444444  0.41666667   0.86440678  0.91666667
## 75    0.33333333  0.16666667   0.45762712  0.37500000
## 76    0.69444444  0.50000000   0.83050847  0.91666667
## 77    0.08333333  0.50000000   0.06779661  0.04166667
## 78    0.66666667  0.54166667   0.79661017  0.83333333
## 79    0.55555556  0.54166667   0.62711864  0.62500000
## 80    0.58333333  0.33333333   0.77966102  0.83333333
## 81    0.13888889  0.41666667   0.06779661  0.08333333
## 82    0.66666667  0.45833333   0.77966102  0.95833333
## 83    0.50000000  0.41666667   0.66101695  0.70833333
## 84    0.38888889  0.33333333   0.59322034  0.50000000
## 85    0.33333333  0.12500000   0.50847458  0.50000000
## 86    0.33333333  0.91666667   0.06779661  0.04166667
## 87    0.41666667  0.29166667   0.52542373  0.37500000
## 88    0.38888889  0.41666667   0.54237288  0.45833333
## 89    0.16666667  0.45833333   0.08474576  0.00000000
## 90    0.63888889  0.41666667   0.57627119  0.54166667
## 91    0.19444444  0.58333333   0.10169492  0.12500000
## 92    0.47222222  0.29166667   0.69491525  0.62500000
## 93    0.13888889  0.58333333   0.10169492  0.04166667
## 94    0.19444444  0.62500000   0.05084746  0.08333333
## 95    0.36111111  0.20833333   0.49152542  0.41666667
## 96    0.41666667  0.29166667   0.69491525  0.75000000
## 97    0.61111111  0.41666667   0.71186441  0.79166667
## 98    0.44444444  0.41666667   0.69491525  0.70833333
## 99    0.58333333  0.45833333   0.76271186  0.70833333
## 100   0.11111111  0.50000000   0.05084746  0.04166667
## 101   0.19444444  0.62500000   0.10169492  0.20833333
## 102   0.16666667  0.41666667   0.06779661  0.04166667
## 103   0.08333333  0.45833333   0.08474576  0.04166667
```

```
## 104   0.66666667  0.41666667   0.71186441  0.91666667
## 105   0.47222222  0.41666667   0.64406780  0.70833333
## 106   1.00000000  0.75000000   0.91525424  0.79166667
## 107   0.30555556  0.79166667   0.11864407  0.12500000
## 108   0.05555556  0.12500000   0.05084746  0.08333333
## 109   0.02777778  0.37500000   0.06779661  0.04166667
## 110   0.38888889  0.25000000   0.42372881  0.37500000
## 111   0.22222222  0.75000000   0.10169492  0.04166667
## 112   0.30555556  0.70833333   0.08474576  0.04166667
## 113   0.66666667  0.45833333   0.57627119  0.54166667
## 114   0.61111111  0.50000000   0.69491525  0.79166667
## 115   0.72222222  0.45833333   0.66101695  0.58333333
## 116   0.55555556  0.20833333   0.67796610  0.75000000
## 117   0.38888889  0.33333333   0.52542373  0.50000000
## 118   0.72222222  0.50000000   0.79661017  0.91666667
## 119   0.16666667  0.16666667   0.38983051  0.37500000
## 120   0.19444444  0.50000000   0.03389831  0.04166667
## 121   0.19444444  0.54166667   0.06779661  0.04166667
## 122   0.50000000  0.33333333   0.62711864  0.45833333
## 123   0.47222222  0.08333333   0.67796610  0.58333333
## 124   0.22222222  0.58333333   0.08474576  0.04166667
## 125   0.19444444  0.41666667   0.10169492  0.04166667
## 126   0.25000000  0.58333333   0.06779661  0.04166667
## 127   0.38888889  1.00000000   0.08474576  0.12500000
## 128   0.25000000  0.29166667   0.49152542  0.54166667
## 129   0.19444444  0.66666667   0.06779661  0.04166667
## 130   0.19444444  0.00000000   0.42372881  0.37500000
## 131   0.72222222  0.45833333   0.74576271  0.83333333
## 132   0.50000000  0.33333333   0.50847458  0.50000000
## 133   0.33333333  0.20833333   0.50847458  0.50000000
## 134   0.47222222  0.58333333   0.59322034  0.62500000
## 135   0.50000000  0.41666667   0.61016949  0.54166667
## 136   0.55555556  0.58333333   0.77966102  0.95833333
## 137   0.58333333  0.50000000   0.72881356  0.91666667
## 138   0.50000000  0.25000000   0.77966102  0.54166667
## 139   0.25000000  0.62500000   0.08474576  0.04166667
## 140   0.36111111  0.33333333   0.66101695  0.79166667
## 141   0.00000000  0.41666667   0.01694915  0.00000000
## 142   0.33333333  0.16666667   0.47457627  0.41666667
## 143   0.55555556  0.29166667   0.66101695  0.70833333
## 144   0.61111111  0.41666667   0.76271186  0.70833333
## 145   0.69444444  0.41666667   0.76271186  0.83333333
## 146   0.86111111  0.33333333   0.86440678  0.75000000
## 147   0.52777778  0.58333333   0.74576271  0.91666667
## 148   0.72222222  0.45833333   0.69491525  0.91666667
## 149   0.02777778  0.41666667   0.05084746  0.04166667
## 150   0.94444444  0.33333333   0.96610169  0.79166667
```

```
summary(iris_new)
```

```
##   Sepal.Length     Sepal.Width      Petal.Length     Petal.Width
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.2222   1st Qu.:0.3333   1st Qu.:0.1017   1st Qu.:0.08333
##  Median :0.4167   Median :0.4167   Median :0.5678   Median :0.50000
```

```
##  Mean   :0.4287   Mean   :0.4406   Mean   :0.4675   Mean   :0.45806
##  3rd Qu.:0.5833   3rd Qu.:0.5417   3rd Qu.:0.6949   3rd Qu.:0.70833
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
```

Test and Train sets

```
train <- iris_new[1:130,]
test <- iris_new[131:150,]
train_sp <- iris_random[1:130,5]
test_sp <- iris_random[131:150,5]
```

class package that contains knn algorithm

```
library(class)
require(class)
model <- knn(train= train, test=test, ,cl= train_sp,k=13)
table(factor(model))
```

```
##
##     setosa versicolor  virginica
##          3          6         11
```

```
table(test_sp,model)
```

```
##             model
## test_sp      setosa versicolor virginica
##   setosa          3          0         0
##   versicolor      0          5         0
##   virginica       0          1        11
```

Example 2

```
library(ggplot2)
```

Loading diamond dataset

```
head(diamonds)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

storing it as a dataframe

```
dia <- data.frame(diamonds)
head(dia)
```

```
##   carat       cut color clarity depth table price    x    y    z
## 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
```

Creating a random number equal 90% of total number of rows

```
ran <- sample(1:nrow(dia),0.9 * nrow(dia))
```

The normalization function is created

```
nor <-function(x) { (x -min(x))/(max(x)-min(x))   }
```

Normalization function is applied to the dataframe

```
dia_nor <- as.data.frame(lapply(dia[,c(1,5,6,7,8,9,10)], nor))
```

The training dataset extracted

```
dia_train <- dia_nor[ran,]
```

The test dataset extracted

```
dia_test <- dia_nor[-ran,]
```

The 2nd column of training dataset because that is what we need to predict about testing dataset. also convert ordered factor to normal factor

```
dia_target <- as.factor(dia[ran,2])
```

The actual values of 2nd couln of testing dataset to compaire it with values that will be predicted also convert ordered factor to normal factor

```
test_target <- as.factor(dia[-ran,2])
```

Running the knn function

```
library(class)
pr <- knn(dia_train,dia_test,cl=dia_target,k=20)
```

Creating the confucion matrix

```
tb <- table(pr,test_target)
```

Checking the accuracy

```
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tb)
```

```
## [1] 71.00482
```

```
getwd()
```

```
## [1] "C:/Users/FGakori/Documents/supervised and unsupervised"
```

```
setwd('C:/Users/FGakori/Documents/supervised and unsupervised')
getwd()
```

```
## [1] "C:/Users/FGakori/Documents/supervised and unsupervised"
```