K Nearest Neighbors Classification

Alex Weirth

2023-04-09

Exploring the Data

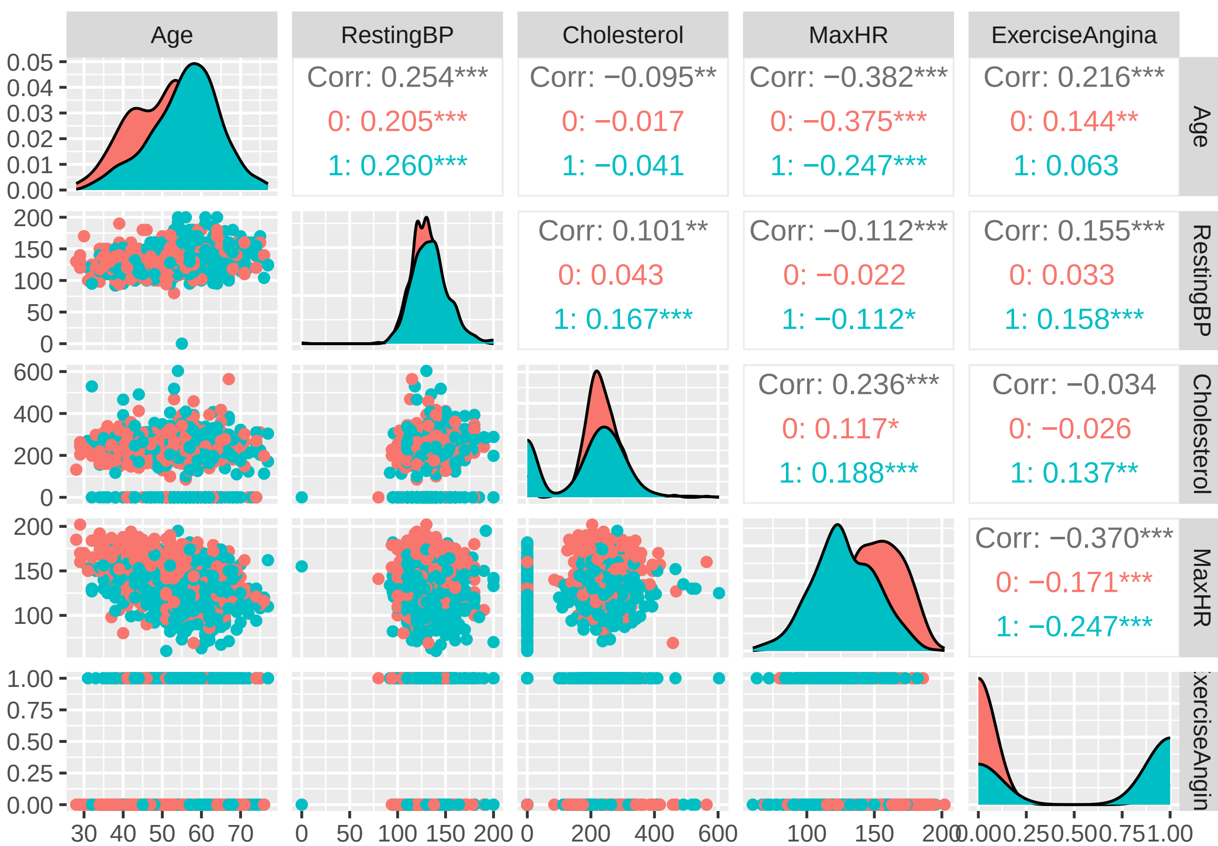
I am working with a heart disease classification dataset that I previously cleaned

Pairs Plot:

```
heart <- read.csv("heart_clean.csv")
```

```
library(GGally)

ggpairs(heart, columns=c(1,4,5,8,9),
        ggplot2::aes(colour = as.factor(HeartDisease)))
```



Identifying features:

Outcome: HeartDisease - This is the outcome variable of either "0" for a person diagnosed with no heart disease or a "1" indicating that person was diagnosed by a doctor with a form of heart disease.

Numeric Feature: MaxHR - Numeric feature of the maximum heart rate (bpm) that a person achieved at the time of examination.

Categorical Feature: ExcerciseAngina - This is a categorical feature that was determined at the time of examination with either a "1" indicating the patient had Angina from exercise or "0" indicating the patient did not have it. Exercise Angina is a condition where a patient experiences chest neck or jaw pain after or during a workout indicating a person's blood flow cannot be maintained at high enough levels. This is usually due to cholesterol-clogged coronary arteries.

Scaling the Numeric Feature

For KNN I need to add a MaxHRNorm that is scaled:

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

heart$MaxHRNorm <- normalize(heart$MaxHR)
min(heart$MaxHRNorm)
```

```
## [1] 0
```

```
max(heart$MaxHRNorm)
```

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

My explanatory feature MaxHR is now normalized. Now, we are not using all the features in our KNN model only two. So I am going to create a new data frame that contains only my outcome "HeartDisease", numeric explanatory "MaxHRNorm", and my categorical feature "ExcerciseAngina" which needs to be encoded as a factor.

```
library(dplyr)
heartKnn <- select(heart, MaxHRNorm, ExerciseAngina, HeartDisease)
heartKnn$ExerciseAngina <- as.factor(heartKnn$ExerciseAngina)
str(heartKnn)
```

```
## 'data.frame':    918 obs. of  3 variables:
##  $ MaxHRNorm     : num  0.789 0.676 0.268 0.338 0.437 ...
##  $ ExerciseAngina: Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 2 1 ...
##  $ HeartDisease  : int  0 1 0 1 0 0 0 0 1 0 ...
```

Stratified splitting for training and testing sets:

```
## PARTITION DATA
heart0<-heartKnn%>%
  filter(HeartDisease==0)
dim(heart0)
```

```
## [1] 410   3
```

```
heart1<-heartKnn%>%
  filter(HeartDisease==1)
dim(heart1)
```

```
## [1] 508   3
```

```
## SAMPLE INDECES
set.seed(100)
heart_sample0<-sample(1:410, 287)
heart_sample1<-sample(1:508, 356)

## TRAINING AND TESTING SETS
trainStrat<-rbind(heart0[heart_sample0, ],
                  heart1[heart_sample1, ])

testStrat<-rbind(heart0[-heart_sample0, ],
                 heart1[-heart_sample1, ])

## PROPORITON OF OUTCOME
mean(trainStrat$HeartDisease)
```

```
## [1] 0.5536547
```

```
mean(testStrat$HeartDisease)
```

```
## [1] 0.5527273
```

Now I have training and testing sets that are stratified and a numeric explanatory feature and can implement the KNN algorithm. The means for the testStrat$HeartDisease$ and trainStrat$HeartDisease$ are 0.001 different which is acceptable.

K Nearest Neighbors

K Nearest Neighbors (k=3):

```
library(class)
```

```
### Specify Arguments
trainFea<-trainStrat%>%
  select(-HeartDisease)

testFea<-testStrat%>%
  select(-HeartDisease)

trainOut<-trainStrat$HeartDisease
testOut<-testStrat$HeartDisease

set.seed(1234)
knn.heartPred=knn(train = trainFea,
             test = testFea,
             cl = trainOut,
             k=3)
```

Confusion Matrix:

```r
cmHeart<-table(knn.heartPred,testOut)
cmHeart
```

```
##              testOut
## knn.heartPred   0   1
##             0  86  45
##             1  37 107
```

Correct Rate

```r
mean(knn.heartPred==testOut)
```

```
## [1] 0.7018182
```

70.2% Correct Rate

Error Rate

```r
1-mean(knn.heartPred==testOut)
```

```
## [1] 0.2981818
```

29.8% Error Rate

False Positives: 37

False Negatives: 45

Sensitivity

```r
### Sensitivity = True Positive / (True Positive + False Negative)
### cm: 1=TN 2=FP 3=FN 4=TP

cmHeart[4]/(cmHeart[4] + cmHeart[3])
```

```
## [1] 0.7039474
```

Specificity

```r
### Specificity = True Negative / (False Positive + True Negative)

cmHeart[1]/(cmHeart[2] + cmHeart[1])
```

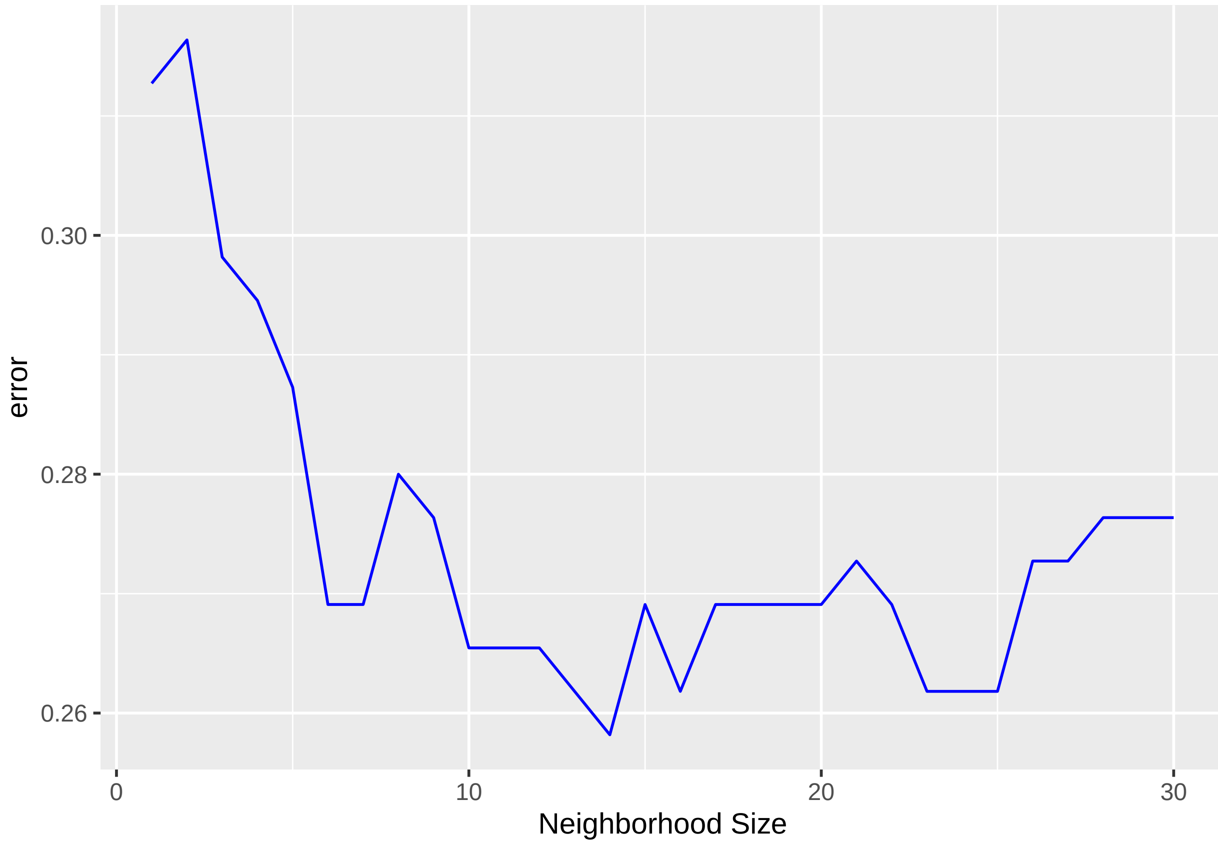
```
## [1] 0.699187
```

Grid Search for best "k" to optimize model performance:

```r
set.seed(123)
error <- c()
for (i in 1:30){
  knnHeart<- knn(train = trainFea,
                test = testFea,
                cl = trainOut,
                k = i)
```

```
  error[i] = 1- mean(knnHeart==testOut)
}

ggplot(data = data.frame(error), aes(x = 1:30, y = error)) +
  geom_line(color = "Blue")+
  xlab("Neighborhood Size")
```




```
which.min(error)
```

```
## [1] 14
```

The best model will contain 14 neighbors.

Fit the best Model:

```
set.seed(1234)
knn.heartPredFinal=knn(train = trainFea,
             test = testFea,
             cl = trainOut,
             k=14)
```

Confusion Matrix for best model:

```
cmHeartBest<-table(knn.heartPredFinal,testOut)
cmHeartBest
```

```
##                     testOut
## knn.heartPredFinal   0   1
##                   0  98  44
##                   1  25 108
```

Correct Rate

```
mean(knn.heartPredFinal==testOut)
```

```
## [1] 0.7490909
```

75% Correct Rate - Increased from 70.2%

Error Rate

```
1-mean(knn.heartPredFinal==testOut)
```

```
## [1] 0.2509091
```

25% Error Rate - Decreased from 29.8%

False Positives: 25

False Negatives: 44

Sensitivity

```
### Sensitivity = True Positive / (True Positive + False Negative)
### cm: 1=TN 2=FP 3=FN 4=TP

cmHeartBest[4]/(cmHeartBest[4] + cmHeartBest[3])
```

```
## [1] 0.7105263
```

Increased from 0.703

Specificity

```
### Specificity = True Negative / (False Positive + True Negative)

cmHeartBest[1]/(cmHeartBest[2] + cmHeartBest[1])
```

```
## [1] 0.796748
```

Increased even more dramatically from 0.699