K-NN

Mikołaj Malec

April 20, 2020

Czym jest algorytm K Nearest Neighbor?

Jest to algorytm który klasyfikuje dany rekord(w naszym przypadku dane o kliencie banku) za pomocą obliczenia k najbliższych sąsiadów. Z tych k sąsiadów przyporządkowuje go do najlczniejszego typu wśród tej k-licznej grupy.

Wczytanie danych i bibliotek

```
library( class)
```

```
## Warning: package 'class' was built under R version 3.6.3
```

```
x_test <- read.csv( "x_test.csv")[-1]
x_train <- read.csv( "x_train.csv")[-1]
y_test <- read.csv( "y_test.csv")[-1]
y_train <- read.csv( "y_train.csv")[-1]
```

Normalizacja

Do wyznaczenia sąsiadów będziemy używać zwykłej metryki eukidysowej. Jednakżę dane trzeba znormalizować ponieważ w innym wypadku odległości pomiędzy danymi o wypłacie będą o wiele bardziej wpływowe niż np liczba dzieci. Taka różnica w wielkośći zmiennych nieunormowanych spowodowała nieistitność zmiennej liczby dzieci co źle wpłynełoby na nasz model

```
#normaliacion to ~(0,1)
for( coli in 1:dim(x_train)[2]){
  #x and y have o be noemalized in the same way
  c_min <- min( c(x_train[,coli], x_test[,coli]))
  c_max <- max( c(x_train[,coli], x_test[,coli]))

  x_train[,coli] <- (x_train[,coli] - c_min) / (c_max - c_min)
  x_test[,coli] <- (x_test[,coli] - c_min) / (c_max - c_min)
}
```

Metryka dokładności

Do pomocy w określeniu jakośći naszego modelu napiszemy funkcje która wyliczy nam konkretne metryki.

```
confusion_matrix_values <- function(confusion_matrix){
  TP <- confusion_matrix[2,2]
  TN <- confusion_matrix[1,1]
  FP <- confusion_matrix[1,2]
  FN <- confusion_matrix[2,1]
  return (c(TP, TN, FP, FN))
}

accuracy <- function(confusion_matrix){
  conf_matrix <- confusion_matrix_values(confusion_matrix)
  return((conf_matrix[1] + conf_matrix[2]) / (conf_matrix[1] + conf_matrix[2] + conf_matrix[3] + conf_ma
}

precision <- function(confusion_matrix){
  conf_matrix <- confusion_matrix_values(confusion_matrix)
  return(conf_matrix[1]/ (conf_matrix[1] + conf_matrix[3]))
}

recall <- function(confusion_matrix){
  conf_matrix <- confusion_matrix_values(confusion_matrix)
  return(conf_matrix[1] / (conf_matrix[1] + conf_matrix[4]))
}

f1 <- function(confusion_matrix){
  conf_matrix <- confusion_matrix_values(confusion_matrix)
  rec <- recall(confusion_matrix)
  prec <- precision(confusion_matrix)
  return(2 * (rec * prec) / (rec + prec))
}
```

Testowanie i Trenowanie naszego modelu dla różnej wartośći parametru k

Spróbujmy teraz sprawdzić jak celny jest nasz model w zależnośći od parametru **k.** Do obliczenia jakośći naszego modelu sprawdzimy jego działanie w z parametrem w przedziale od 1 do pierwiastka z wielkości danych trenningowych.

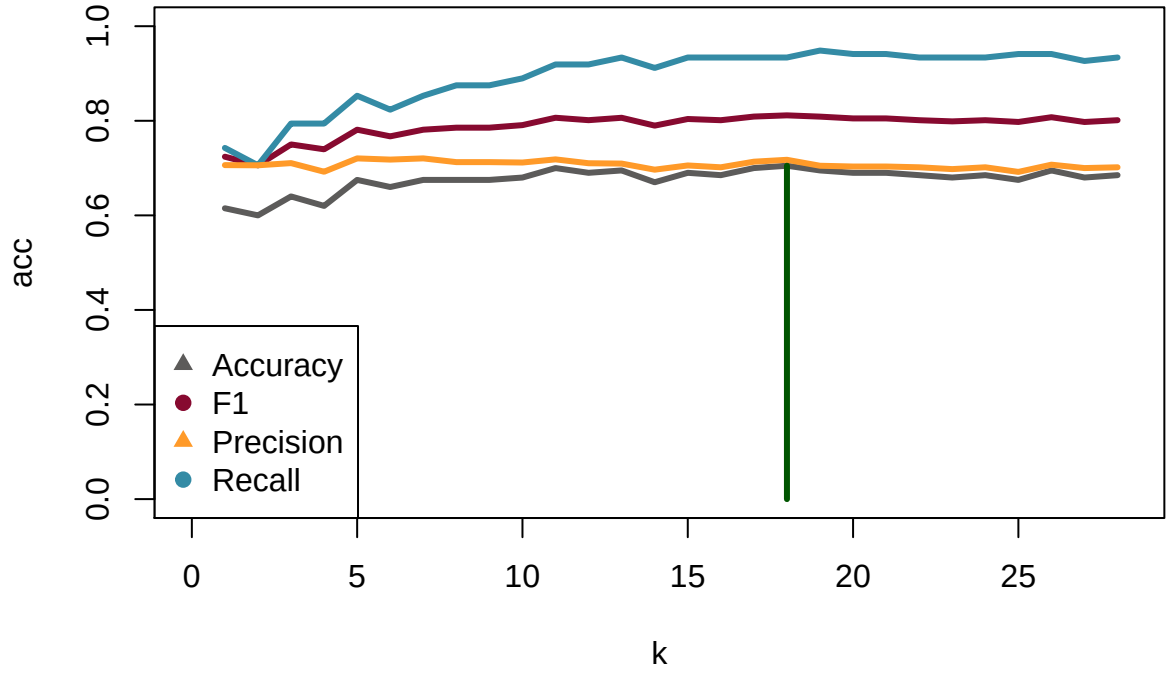
```
n <- sqrt( dim(x_train)[1])
acc <- rep(0,n)
ftest<- rep(0,n)
rec <- rep(0,n)
pre <- rep(0,n)
for(k in 1:n){
  test_pred <- knn( train = x_train[1:800,], test = x_test, cl = y_train[1:800,], k=k)

  tab <-  table( y_test[,1], test_pred)
  acc[k] <- accuracy(tab)
  ftest[k] <- f1(tab)
  rec[k] <- recall(tab)
  pre[k] <- precision(tab)
```

```
}
k=1:n
best_k <- which.max( acc)
```

Po przeiterowaniu przez wszytskie parametry zmiennej k można pokazać dane na wykresie.

```
plot(k, acc, type = "l", lwd = 3, xlim = c(0,n),ylim = c(0,1), col = "#5C5B5A")
lines(k,ftest,type = "l", lwd = 3,col = "#870A30")
lines(k,rec,type = "l", lwd = 3,col = "#348BA5")
lines(k,pre,type = "l", lwd = 3,col = "#FF9A29")
lines(c(best_k,best_k),c(0,acc[best_k]),col = "#005500",lwd = 3)
legend("bottomleft", legend = c("Accuracy","F1","Precision","Recall"),
  col = c("#5C5B5A","#870A30","#FF9A29","#348BA5"),
  pch = c(17,19),
  text.col = "black",
  horiz = F )
```



Jak widać na wykresie najwyższą wartość celności ,zaznaczoną na zielono, nasz model uzyskuje dla parametru k = 18. Sprawdzmy dokładniej jak nasz model przewiduje dla najlepszego parametru **k.**

Test modelu dla najlepszego parametru

```
test_pred <- knn( train = x_train[1:800,], test = x_test, cl = y_train[1:800,], k=best_k)
tab <-  table( y_test[,1], test_pred)
knitr::kable(tab)
```

	0	1
0	12	52
1	9	127

Tak wygląda nasza tabela. Finalnie model uzyskuje 69.5% celności.