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 najliczniejszego typu wśród tej k-licznej grupy.

Wczytanie danych i bibliotek

```
library( class)
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 euklidesowej. Jednakże dane trzeba znormalizować, ponieważ w innym wypadku odległości pomiędzy danymi o wieku będą o wiele bardziej wpływowe niż np. czy posiada telefon. Taka różnica w wielkości zmiennych nieunormowanych spowodowała nieistotność zmiennej posiadania telefonu, co źle wpłynęłoby 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 jakości 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_mat
}

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))
}
```

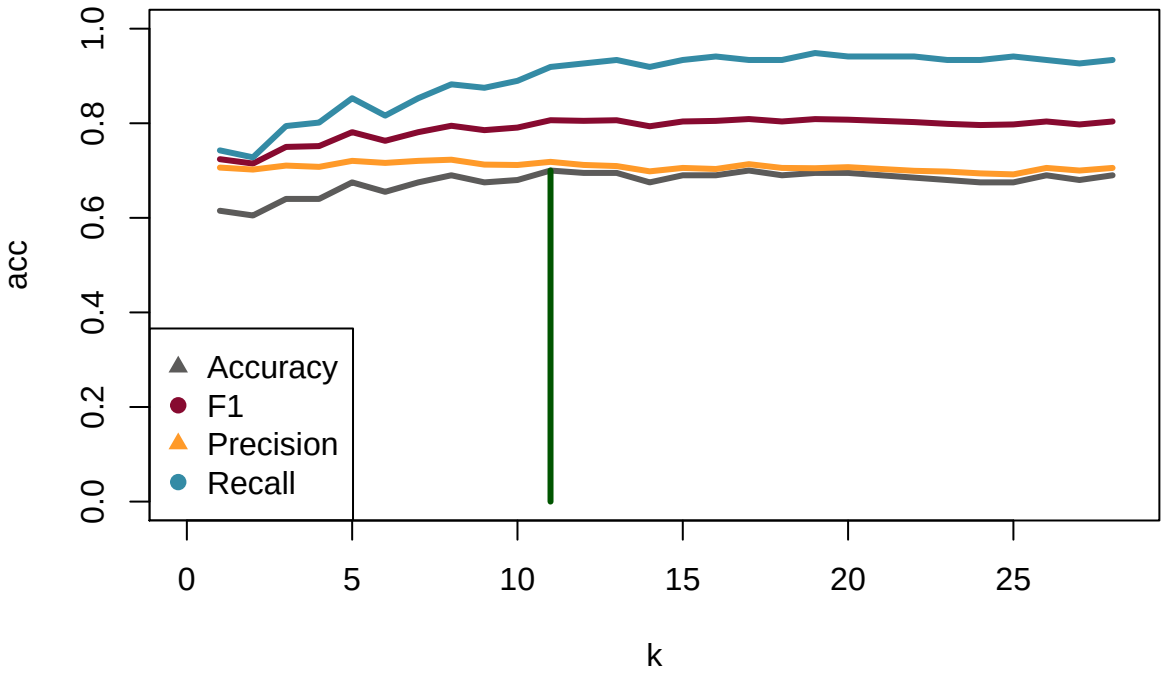
Testowanie i Trenowanie naszego modelu dla różnej wartości parametru k

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

```
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){
#tworzenie modelu dla danego k
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)
}
```

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



Jak widać na wykresie najwyższą wartość celności, zaznaczoną na zielono, nasz model uzyskuje dla parametru k = 11. Sprawdźmy, jak dokładnie nasz model przewiduje dla najlepszego parametru **k.**

Test modelu dla najlepszego parametru

	0	1
0	15	49
1	11	125

Tak wygląda nasza tabela. Finalnie model uzyskuje 70% celności.