

```

1  ##K近傍法(K-NN)#####
2  library(kernlab);data(spam)
3  anyNA(spam)
4  str(spam)
5  #class(spam$type)#classの確認
6  colnames(spam)
7  library(caret)
8  set.seed(123)
9  dat.s <- createDataPartition( spam$type , p = 0.75 , list = F )
10 train.spam <- spam[ dat.s , ]
11 test.spam <- spam[-dat.s , ]
12 #knn:k近傍法は距離の近いk個のラベルを用いて分類する.距離自体は分類に
13 #用いない,近いデータを探すのに用いるだけ.
14 #e1071()によるチューニング
15 library(e1071)
16 ##knn
17 set.seed(123)
18 knnfit <- tune.knn(x = train.spam[,-58], y = train.spam[,58], k = 1:20,
19                  tunecontrol=tune.control(sampling="cross"),
20                  cross=5)#boot,fix:回数
21 summary(knnfit)
22 plot(knnfit)
23 library(class)
24 set.seed(123)
25 knn.fit <- knn(train = train.spam[,-58] , test = test.spam[,-58] ,
26              cl = train.spam[,58] , prob = T , k = 1 )
27 tt <- table(test.spam$type, knn.fit)
28 round((tt[1,1]+tt[2,2])/length(test.spam$type),4)*100
29 #confusion matrix
30 library(caret)
31 confusionMatrix(knn.fit, test.spam$type)
32 ##重みづけk近傍法kknn:距離が近い方が類似性が高いと考える
33 #ミンコフスキー距離を用いて重みづけ
34 ##kknn:チューニング
35 library(kknn)
36 train1 <- train.kknn(type ~ ., data = train.spam,kmax = 20,
37                    kernel = c("rectangular", "triangular",
38                               "epanechnikov", "gaussian",
39                               "rank", "optimal"),
40                    distance = 1 , scale = T )#ミンコフスキー定数
41 plot(train1)
42 train1
43 train2 <- train.kknn(type ~ ., data = train.spam,kmax = 20,
44                    kernel = c("rectangular", "triangular",
45                               "epanechnikov", "gaussian",
46                               "rank", "optimal"),
47                    distance = 2 , scale = T )#ミンコフスキー定数
48 plot(train2)
49 train2
50
51 ##kknn
52 library(kknn)
53 set.seed(123)
54 kknn.out <- kknn(type ~ ., train.spam, test.spam,
55                 k = 15 ,kernel="triangular", distance = 2 , scale = T)
56 t <- table(test.spam$type, kknn.out$fit)
57 t
58 round((t[1,1]+t[2,2])/length(test.spam$type),4)*100
59 library(gmodels)
60 CrossTable(x = test.spam$type , y = kknn.out$fit ,prop.chisq = F )
61 #confusion matrix
62 library(caret)
63 confusionMatrix(kknn.out$fit, test.spam$type)
64 ##多クラス分類
65 ##手書き数 data
66 test.su <- ("http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits//optdigits.tes")
67 test.su <- read.table(test.su , sep = ",")
68 train.su <- ("http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits//optdigits.tra")
69 train.su <- read.table(train.su , sep = ",")
70 str(train.su)
71 str(test.su)
72 #factor変換
73 train.su$V65 <- factor(train.su$V65)
74 str(train.su)
75 test.su$V65 <- factor(test.su$V65)
76 str(test.su)
77 colnames(train.su)
78 colnames(test.su)
79 table(test.su$V65)
80 table(train.su$V65)
81 #z変換
82 train.su.z <- scale(train.su[,c(1:64)])
83 train.su.z <- cbind(data.frame(train.su.z) , train.su$V65)
84 names(train.su.z)[65] <- c("su")
85 str(train.su.z)
86 test.su.z <- scale(test.su[,c(1:64)])
87 test.su.z <- cbind(data.frame(test.su.z) , test.su$V65)
88 names(test.su.z)[65] <- c("su")
89 str(test.su.z)
90 #normalize変換

```

```

91 norm <- function(x){
92   return((x-min(x))/(max(x)-min(x)))
93 }
94 train.su.norm <- data.frame(lapply(train.su[1:64] , norm))
95 train.su.norm <- cbind(data.frame(train.su.norm) , train.su$V65)
96 names(train.su.norm)[65] <- c("su")
97 str(train.su.norm)
98 anyNA(train.su.norm[,c(2:30,41:64)])
99 test.su.norm <- data.frame(lapply(test.su[1:64] , norm))
100 test.su.norm <- cbind(data.frame(test.su.norm) , test.su$V65)
101 names(test.su.norm)[65] <- c("su")
102 str(test.su.norm)
103 anyNA(test.su.norm[,c(2:30,41:64)])
104
105 ##knn
106 set.seed(123)
107 knnfit.m <- tune.knn(x = train.su[,-65], y = train.su[,65]
108   , tunecontrol = tune.control(sampling = "boot"),
109   k = 1:11)
110 summary(knnfit.m)
111 plot(knnfit.m)
112 library(class)
113 set.seed(123)
114 knn.fitm <- knn(train = train.su[,-65] , test = test.su[,-65] ,
115   cl = train.su[,65] , prob = T , k = 1 )
116 tt.m <- table(test.su$V65, knn.fitm)
117 round(sum(diag(tt.m))/length(test.su$V65),4)*100
118 #confusion matrix
119 library(caret)
120 confusionMatrix(knn.fitm, test.su$V65)
121 ##knn.z
122 set.seed(123)
123 knnfit.z <- tune.knn(x = train.su.z[c(2:30,41:64)], y = train.su.z[,65]
124   , tunecontrol = tune.control(sampling = "boot"),
125   k = 1:11)
126 summary(knnfit.z)
127 plot(knnfit.z)
128 library(class)
129 set.seed(123)
130 knn.fitm.z <- knn(train = train.su.z[c(2:30,41:64)] ,
131   test = test.su.z[c(2:30,41:64)] ,
132   cl = train.su.z[,65] , prob = T , k = 1 )
133 tt.m.z <- table(test.su.z$su, knn.fitm.z)
134 round(sum(diag(tt.m.z))/length(test.su.z$su),4)*100
135 #confusion matrix
136 library(caret)
137 confusionMatrix(knn.fitm.z, test.su.z$su)
138 ###kkn
139 library(kknn)
140 train.m <- train.kknn(V65 ~ ., data = train.su, kmax = 25,
141   kernel = c("rectangular", "triangular",
142     "epanechnikov", "gaussian",
143     "rank", "optimal"),
144   distance = 2 , scale = T )#ミンコフスキー定数
145 plot(train.m)
146 train.m
147 ##kkn
148 library(kknn)
149 set.seed(123)
150 kknn.outm <- kknn(V65 ~ ., train.su, test.su,
151   k = 4 ,kernel="triangular", distance = 2 , scale = T)
152 tm <- table(test.su$V65, kknn.outm$fit)
153 round(sum(diag(tm)) /sum(tm),4)*100
154 #library(gmodels)
155 #CrossTable(x = test.su$class , y = kknn.out$fit ,prop.chisq = F )
156 #confusion matrix
157 library(caret)
158 confusionMatrix(kknn.outm$fit, test.su$V65)
159

```