Winequality\_White

Charlotte Bahr

6/27/2020

library(kerasR) #DNN  
library(keras)  
library(tensorflow)#DNN  
library(data.table)  
library(dataPreparation)  
library(e1071) #Naive Bayes  
library(randomForest) #random Forest  
library(class) #k-Nearest Neighbor   
library(kernlab) #Support Vector Machine  
library(mlbench)#contains Glass data  
library(keras) #contains MNIST data  
library(caret)  
library(mltest)  
library(dplyr)  
use\_session\_with\_seed(5748)

## Set session seed to 5748 (disabled GPU, CPU parallelism)

## DATA PRE-PROCESSING  
wq <- read.csv("~/Desktop/Bachelorarbeit/R/R\_Files/winequality-white.csv", sep=";")  
wq <- as.data.frame(wq)  
  
wq$quality[wq$quality > 7] <- "7"  
wq$quality[wq$quality < 4] <- "4"  
  
wq$quality <- as.factor(wq$quality)  
wq$quality <- droplevels(wq$quality)  
  
standard.features <- scale(wq[,1:11])  
wq <- cbind(standard.features, wq[12])  
  
set.seed(1234)  
train\_index\_SC <- sample(1:nrow(wq), 0.7\*nrow(wq))  
test\_index\_SC <- setdiff (1:nrow(wq), train\_index\_SC)  
  
Train <- wq[train\_index\_SC,]  
Test <- wq[test\_index\_SC,]  
  
True\_Label <- Test$quality  
  
#prep for DNN  
  
X\_train <- Train %>%   
 select(-quality) %>%   
 as.matrix()  
  
Y\_train <- to\_categorical(Train$quality)  
  
X\_test <- Test %>%   
 select(-quality) %>%   
 as.matrix()  
  
Y\_test <- to\_categorical(Test$quality)  
  
##MODELS   
  
#k-Nearest Neighbor  
  
pc <- proc.time()  
model\_KNN <- knn(Train[1:11], Test[1:11], as.factor(Train$quality), k=58)  
print(proc.time() - pc)

## user system elapsed   
## 0.107 0.001 0.108

#Naive Bayes  
  
pc <- proc.time()   
model\_NB <- naiveBayes(as.factor(Train$quality) ~. , Train[1:11])  
print(proc.time() - pc)

## user system elapsed   
## 0.010 0.001 0.011

#Random Forest  
  
pc <- proc.time()   
model\_RF <- randomForest(as.factor(Train$quality) ~. , Train[1:11])  
print(proc.time() - pc)

## user system elapsed   
## 3.218 0.158 3.416

#Support Vector Machine  
   
pc <- proc.time()   
model\_SVM <- ksvm(Train$quality ~. , Train[1:11], type = "C-svc", C = 1, kernel = "rbfdot" )  
print(proc.time() - pc)

## user system elapsed   
## 1.128 0.162 1.294

#Deep Neural Network  
  
  
pc <- proc.time()   
  
model\_DNN <- Sequential()  
  
model\_DNN$add(Dense(units=50, input\_shape = dim(X\_train)[2]))  
model\_DNN$add(LeakyReLU())  
model\_DNN$add(Dropout(0.4))  
  
model\_DNN$add(Dense(units=50))  
model\_DNN$add(LeakyReLU())  
model\_DNN$add(Dropout(0.3))  
  
model\_DNN$add(Dense(units=50))  
model\_DNN$add(LeakyReLU())  
model\_DNN$add(Dropout(0.2))  
  
  
model\_DNN$add(Dense(units=50))  
model\_DNN$add(LeakyReLU())  
model\_DNN$add(Dropout(0.1))  
  
  
model\_DNN$add(Dense(8))  
model\_DNN$add(Activation("softmax"))  
  
# compile  
keras\_compile(model\_DNN, loss ="categorical\_crossentropy", optimizer = RMSprop(), metrics = "accuracy")  
keras\_fit(model\_DNN, X\_train, Y\_train, batch\_size = 128, epochs = 32, verbose= 1, validation\_split = 0.2)  
  
print(proc.time() - pc)

## user system elapsed   
## 3.159 0.070 3.207

##EVALUATION METRICS  
  
#predictions  
  
pred\_KNN <- model\_DNN  
  
pred\_NB <- as.factor(predict(model\_NB, Test))  
  
pred\_RF <- as.factor(predict(model\_RF, Test))  
  
pred\_SVM <- as.factor(predict(model\_SVM, Test))  
  
pred\_DNN <- as.factor(keras\_predict\_classes(model\_DNN, X\_test))  
  
  
CF\_KNN <- table(model\_KNN, True\_Label)  
   
CF\_NB <- table(pred\_NB, True\_Label)  
   
CF\_RF <- table(pred\_RF, True\_Label)  
   
CF\_SVM <- table(pred\_SVM, True\_Label)  
  
CF\_DNN <- table(pred\_DNN, True\_Label)  
  
  
print(CF\_KNN)

## True\_Label  
## model\_KNN 4 5 6 7  
## 4 0 0 0 0  
## 5 45 223 136 8  
## 6 24 180 436 167  
## 7 4 11 97 139

print(CF\_NB)

## True\_Label  
## pred\_NB 4 5 6 7  
## 4 14 15 10 3  
## 5 27 228 179 26  
## 6 15 122 214 60  
## 7 17 49 266 225

print(CF\_RF)

## True\_Label  
## pred\_RF 4 5 6 7  
## 4 13 7 0 0  
## 5 34 292 98 8  
## 6 23 107 514 106  
## 7 3 8 57 200

print(CF\_SVM)

## True\_Label  
## pred\_SVM 4 5 6 7  
## 4 3 4 0 0  
## 5 43 247 127 13  
## 6 26 160 478 171  
## 7 1 3 64 130

print(CF\_DNN)

## True\_Label  
## pred\_DNN 4 5 6 7  
## 4 2 0 1 0  
## 5 45 276 177 13  
## 6 24 133 421 180  
## 7 2 5 70 121

#metrics  
set.seed(2445)  
  
ml\_test\_KNN <- ml\_test(model\_KNN, True\_Label, output.as.table = FALSE)  
  
ml\_test\_NB <- ml\_test(pred\_NB, True\_Label, output.as.table = FALSE)  
  
ml\_test\_RF <- ml\_test(pred\_RF, True\_Label, output.as.table = FALSE)  
  
ml\_test\_SVM <- ml\_test(pred\_SVM, True\_Label, output.as.table = FALSE)  
  
ml\_test\_DNN <- ml\_test(pred\_DNN, True\_Label, output.as.table = FALSE)  
  
#Macro Average Accuracy  
  
MAvA\_KNN <- print((sum(ml\_test\_KNN$balanced.accuracy, na.rm = TRUE))/4)

## [1] 0.5917818

MAvA\_NB <- print((sum(ml\_test\_NB$balanced.accuracy, na.rm = TRUE))/4)

## [1] 0.5852457

MAvA\_RF <- print((sum(ml\_test\_RF$balanced.accuracy, na.rm = TRUE))/4)

## [1] 0.715639

MAvA\_SVM <- print((sum(ml\_test\_SVM$balanced.accuracy, na.rm = TRUE))/4)

## [1] 0.6201585

MAvA\_DNN <- print((sum(ml\_test\_DNN$balanced.accuracy, na.rm = TRUE))/4)

## [1] 0.6060893

#Macro Average F1   
  
MAvF1\_KNN <- print((sum(ml\_test\_KNN$F1, na.rm = TRUE))/4)

## [1] 0.4056932

MAvF1\_NB <- print((sum(ml\_test\_NB$F1, na.rm = TRUE))/4)

## [1] 0.4195403

MAvF1\_RF <- print((sum(ml\_test\_RF$F1, na.rm = TRUE))/4)

## [1] 0.5954041

MAvF1\_SVM <- print((sum(ml\_test\_SVM$F1, na.rm = TRUE))/4)

## [1] 0.4509397

MAvF1\_DNN <- print((sum(ml\_test\_DNN$F1, na.rm = TRUE))/4)

## [1] 0.4280234

#MAvMCC  
  
MAvMCC\_KNN <- print((sum(ml\_test\_KNN$MCC, na.rm = TRUE))/4)

## [1] 0.1896363

MAvMCC\_NB <- print((sum(ml\_test\_NB$MCC, na.rm = TRUE))/4)

## [1] 0.1738367

MAvMCC\_RF <- print((sum(ml\_test\_RF$MCC, na.rm = TRUE))/4)

## [1] 0.4745666

MAvMCC\_SVM <- print((sum(ml\_test\_SVM$MCC, na.rm = TRUE))/4)

## [1] 0.2742284

MAvMCC\_DNN <- print((sum(ml\_test\_DNN$MCC, na.rm = TRUE))/4)

## [1] 0.2457224

#MAvGeometricMean  
  
MAvGM\_KNN <- print((sum(ml\_test\_KNN$geometric.mean, na.rm = TRUE))/4)

## [1] 0.4547875

MAvGM\_NB <- print((sum(ml\_test\_NB$geometric.mean, na.rm = TRUE))/4)

## [1] 0.537691

MAvGM\_RF <- print((sum(ml\_test\_RF$geometric.mean, na.rm = TRUE))/4)

## [1] 0.6700183

MAvGM\_SVM <- print((sum(ml\_test\_SVM$geometric.mean, na.rm = TRUE))/4)

## [1] 0.5255313

MAvGM\_DNN <- print((sum(ml\_test\_DNN$geometric.mean, na.rm = TRUE))/4)

## [1] 0.5052457