Glass\_assigned\_classes

Charlotte Bahr

7/21/2020

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) #wg. select  
use\_session\_with\_seed(9876)

## Set session seed to 9876 (disabled GPU, CPU parallelism)

# data pre-processing   
  
data("Glass")  
Glass <- Glass  
  
standard.features <- scale(Glass[,1:9])  
Glass <- cbind(standard.features, Glass[10])  
  
Glass$Type[Glass$Type == 3] <- "5"  
Glass$Type[Glass$Type == 4] <- "5"  
Glass$Type[Glass$Type == 6] <- "5"  
  
Glass$Type <- droplevels(Glass$Type)  
  
  
train\_index\_SC <- sample(1:nrow(Glass), 0.7\*nrow(Glass))  
test\_index\_SC <- setdiff(1:nrow(Glass), train\_index\_SC)  
  
Train <- Glass[train\_index\_SC,]  
Test <- Glass[test\_index\_SC,]  
  
True\_Label <- Test$Type  
  
#prep for DNN  
  
X\_train <- Train %>%   
 select(-Type)%>%  
 as.matrix()  
  
Y\_train <- to\_categorical(Train$Type) # one hot encoding  
  
X\_test <- Test %>%   
 select(-Type) %>%   
 as.matrix()  
  
Y\_test <- to\_categorical(Test$Type)  
##MODELS   
  
#k-Nearest Neighbor  
  
pc <- proc.time()  
model\_KNN <- knn(Train, Test, as.factor(Train$Type), k=15) ##sqrt214 =14,63  
print(proc.time() - pc)

## user system elapsed   
## 0.006 0.000 0.006

#Naive Bayes  
  
pc <- proc.time()   
model\_NB <- naiveBayes(as.factor(Train$Type) ~. , Train)  
print(proc.time() - pc)

## user system elapsed   
## 0.006 0.001 0.007

#Random Forest  
  
pc <- proc.time()   
model\_RF <- randomForest(as.factor(Train$Type) ~. , Train)  
print(proc.time() - pc)

## user system elapsed   
## 0.074 0.005 0.078

#Support Vector Machine  
   
pc <- proc.time()   
model\_SVM <- ksvm(Train$Type ~. , Train, type = "C-svc", C = 1, kernel = "vanilladot" )

## Setting default kernel parameters

print(proc.time() - pc)

## user system elapsed   
## 0.015 0.002 0.017

#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   
## 1.359 0.033 1.394

##EVALUATION METRICS  
  
#predictions  
  
pred\_KNN <- #has no prediction value  
  
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 1 2 5 7  
## 1 23 1 0 0  
## 2 2 21 0 0  
## 5 0 0 6 0  
## 7 0 0 0 12

print(CF\_NB)

## True\_Label  
## pred\_NB 1 2 5 7  
## 1 20 13 5 0  
## 2 5 8 1 1  
## 5 0 1 0 2  
## 7 0 0 0 9

print(CF\_RF)

## True\_Label  
## pred\_RF 1 2 5 7  
## 1 18 1 1 0  
## 2 5 18 3 1  
## 5 2 3 2 1  
## 7 0 0 0 10

print(CF\_SVM)

## True\_Label  
## pred\_SVM 1 2 5 7  
## 1 20 4 2 1  
## 2 5 17 3 0  
## 5 0 0 1 1  
## 7 0 1 0 10

print(CF\_DNN)

## True\_Label  
## pred\_DNN 1 2 5 7  
## 1 7 2 2 0  
## 2 18 20 3 1  
## 5 0 0 1 0  
## 7 0 0 0 11

#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] 0.9753792

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

## [1] 0.6412383

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

## [1] 0.7870243

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

## [1] 0.7651406

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

## [1] 0.703091

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

## [1] 0.9680272

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

## [1] 0.481124

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

## [1] 0.6823748

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

## [1] 0.6530501

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

## [1] 0.5640312

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

## [1] 0.950208

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

## [1] 0.3001354

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

## [1] 0.5834428

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

## [1] 0.5561155

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

## [1] 0.4793258

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

## [1] 0.9752794

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

## [1] 0.5076525

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

## [1] 0.7677011

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

## [1] 0.7221965

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

## [1] 0.6284071