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(9123)

## Set session seed to 9123 (disabled GPU, CPU parallelism)

## DATA PRE-PROCESSING
wq <- read.csv("~/Desktop/Bachelorarbeit/R/R_Files/winequality-white.csv",
sep=";")

wq$quality[wq$quality > 8] <- "8"
wq$quality[wq$quality < 4] <- "4"

wq <- as.data.frame(wq)

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) %>%
  scale()

Y_train <- to_categorical(Train$quality)
```

```r
X_test <- Test %>%
  select(-quality) %>%
  scale()

Y_test <- to_categorical(Test$quality)

##MODELS

#k-Nearest Neighbor

pc <- proc.time()
model_KNN <- knn(Train, Test, as.factor(Train$quality), k=58)
print(proc.time() - pc)
```

```
##    user  system elapsed
##   0.198   0.008   0.213
```

```r
#Naive Bayes

pc <- proc.time()
model_NB <- naiveBayes(as.factor(Train$quality) ~. , Train)
print(proc.time() - pc)
```

```
##    user  system elapsed
##   0.012   0.006   0.017
```

```r
#Random Forest

pc <- proc.time()
model_RF <- randomForest(as.factor(Train$quality) ~. , Train)
print(proc.time() - pc)
```

```
##    user  system elapsed
##   2.997   0.125   3.125
```

```r
#Support Vector Machine

pc <- proc.time()
model_SVM <- ksvm(Train$quality ~. , Train, type = "C-svc", C = 1, kernel =
"rbfdot" )
print(proc.time() - pc)
```

```
##    user  system elapsed
##   1.640   0.281   1.926
```

```r
#Deep Neural Network


pc <- proc.time()
set.seed(42)
```

```
model_DNN <- Sequential()

model_DNN$add(Dense(units=250, input_shape = dim(X_train)[2]))
model_DNN$add(LeakyReLU())
model_DNN$add(Dropout(0.4))

model_DNN$add(Dense(units=250))
model_DNN$add(LeakyReLU())
model_DNN$add(Dropout(0.3))

model_DNN$add(Dense(units=250))
model_DNN$add(LeakyReLU())
model_DNN$add(Dropout(0.2))

model_DNN$add(Dense(9))
model_DNN$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
##   6.056   0.457   6.492

##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)
```

```r
CF_DNN <- table(pred_DNN, True_Label)


print(CF_KNN)
```

```
##           True_Label
## model_KNN   4   5   6   7   8
##         4   0   0   0   0   0
##         5  38 161 141  27   6
##         6  35 253 516 230  40
##         7   0   0  12   6   5
##         8   0   0   0   0   0
```

```r
print(CF_NB)
```

```
##         True_Label
## pred_NB   4   5   6   7   8
##       4  14  14  10   1   0
##       5  27 228 179  23   3
##       6  17 127 226  58   9
##       7  14  44 250 175  35
##       8   1   1   4   6   4
```

```r
print(CF_RF)
```

```
##         True_Label
## pred_RF   4   5   6   7   8
##       4  12   7   1   0   0
##       5  34 286  93   8   0
##       6  26 116 529 103  23
##       7   1   5  46 151  12
##       8   0   0   0   1  16
```

```r
print(CF_SVM)
```

```
##          True_Label
## pred_SVM   4   5   6   7   8
##        4   3   4   0   0   0
##        5  46 247 126  13   1
##        6  23 163 500 177  35
##        7   1   0  43  73  15
##        8   0   0   0   0   0
```

```r
print(CF_DNN)
```

```
##          True_Label
## pred_DNN   4   5   6   7   8
##        4   6   4   1   1   0
##        5  45 273 160  17   1
##        6  21 134 462 162  35
##        7   1   2  45  83  14
##        8   0   1   1   0   1
```

```r
#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))/5)

## [1] 0.5100024

MAvA_NB <- print((sum(ml_test_NB$balanced.accuracy, na.rm = TRUE))/5)

## [1] 0.5656745

MAvA_RF <- print((sum(ml_test_RF$balanced.accuracy, na.rm = TRUE))/5)

## [1] 0.692838

MAvA_SVM <- print((sum(ml_test_SVM$balanced.accuracy, na.rm = TRUE))/5)

## [1] 0.5788775

MAvA_DNN <- print((sum(ml_test_DNN$balanced.accuracy, na.rm = TRUE))/5)

## [1] 0.5900534

#Macro Average F1

MAvF1_KNN <- print((sum(ml_test_KNN$F1, na.rm = TRUE))/5)

## [1] 0.2086379

MAvF1_NB <- print((sum(ml_test_NB$F1, na.rm = TRUE))/5)

## [1] 0.3495931

MAvF1_RF <- print((sum(ml_test_RF$F1, na.rm = TRUE))/5)

## [1] 0.5534347

MAvF1_SVM <- print((sum(ml_test_SVM$F1, na.rm = TRUE))/5)

## [1] 0.3332035

MAvF1_DNN <- print((sum(ml_test_DNN$F1, na.rm = TRUE))/5)
```

```
## [1] 0.3616275
#MAvMCC
MAvMCC_KNN <- print((sum(ml_test_KNN$MCC, na.rm = TRUE))/5)
## [1] 0.01992839
MAvMCC_NB <- print((sum(ml_test_NB$MCC, na.rm = TRUE))/5)
## [1] 0.1430763
MAvMCC_RF <- print((sum(ml_test_RF$MCC, na.rm = TRUE))/5)
## [1] 0.4646652
MAvMCC_SVM <- print((sum(ml_test_SVM$MCC, na.rm = TRUE))/5)
## [1] 0.1867606
MAvMCC_DNN <- print((sum(ml_test_DNN$MCC, na.rm = TRUE))/5)
## [1] 0.2215057
#MAvGeometricMean
MAvGM_KNN <- print((sum(ml_test_KNN$geometric.mean, na.rm = TRUE))/5)
## [1] 0.219316
MAvGM_NB <- print((sum(ml_test_NB$geometric.mean, na.rm = TRUE))/5)
## [1] 0.4795046
MAvGM_RF <- print((sum(ml_test_RF$geometric.mean, na.rm = TRUE))/5)
## [1] 0.632817
MAvGM_SVM <- print((sum(ml_test_SVM$geometric.mean, na.rm = TRUE))/5)
## [1] 0.3919383
MAvGM_DNN <- print((sum(ml_test_DNN$geometric.mean, na.rm = TRUE))/5)
## [1] 0.4485263
```