R Notebook

Code

Hide

```
# Find how many cores are on your machine
detectCores() # Result = Typically 4 to 6
```

```
[1] 4
```

Hide

```
# Create Cluster with desired number of cores. Don't use them all! Your computer is running other processes.
cl <- makeCluster(2)

# Register Cluster
registerDoParallel(cl)

# Confirm how many cores are now "assigned" to R and RStudio
getDoParWorkers() # Result 2
```

```
[1] 2
```

Hide

```
# create reproducable results from random sampling
set.seed(234)
```

Hide

```
# Stop Cluster. After performing your tasks, stop your cluster.
stopCluster(cl)
```

Hide

```
install.packages("caret")
install.packages("plotly")
install.packages("dplyr")
install.packages("kknn")
install.packages("corrplot")
install.packages("corrr")
```

Hide

```
library(caret)
library(plotly)
library(corrplot)
library(corrr)
library(dplyr)
```

Hide

```
options(max.print=1000000)
options(scipen=999)
```

Hide

```
samsungDF <- read.csv("galaxy_smallmatrix_labeled_9d.csv")
```

Hide

```
# variables
names(samsungDF)
```

Hide

```
#summary
str(samsungDF)
```

Hide

```
# no missing values
sum(is.na(samsungDF))
```

Hide

```
# There are no empty columns
colSums(samsungDF)
```

Hide

```
# visualize distribution of iphone and samsung sentiments
plot_ly(samsungDF, x= ~samsungDF$galaxysentiment, type='histogram')
```

Notes: 0: very negative 1: negative 2: somewhat negative 3: somewhat positive 4: positive 5: very positive

Hide

```
# correlation matrix
corr_matrix <- cor(samsungDF)
corr_plot <- corrplot(as.matrix(corr_matrix))
corr_plot
```

Hide

```
# returns correlation  greater than .9
corr_df <- correlate(samsungDF, diagonal = NA) %>% stretch()
# Examine variables with correlation above .9
corr_df_filtered <- corr_df %>% filter(r > .9)
```

Which variables will I keep? I copied corr_df_big_filtered into a spreadsheet and removed all duplicates.

"iphone","htcphone","ios","nokiacampos","nokiacamneg","nokiacamunc","samsungdispos","sonydispos","nokiadispos","htcdispos","samsungdisneg","sonydisr

Hide

```
#columns to remove

corr_to_remove <- c("iphone","htcphone","ios","nokiacampos","nokiacamneg","nokiacamunc","samsungdispos","sonydisp
os","nokiadispos","htcdispos","samsungdisneg","sonydisneg","nokiadisneg","samsungdisunc","nokiadisunc","nokiaperp
os","samsungperneg","nokiaperneg","samsungperunc","nokiaperunc","iosperpos","googleperpos","iosperneg")
```

Hide

```
samsungCOR <- samsungDF[ , -which(names(samsungDF) %in% corr_to_remove)]
```

Hide

```
names(samsungCOR)
```

nzv

Recursive Feature Elimination

Hide

```
# Let's sample the data before using RFE
set.seed(123)
iphoneSample <- samsungDF[sample(1:nrow(samsungDF), 1000, replace=FALSE),]

# Set up rfeControl with randomforest, repeated cross validation and no updates
ctrl <- rfeControl(functions = rfFuncs,
                   method = "repeatedcv",
                   repeats = 5,
                   verbose = FALSE)

# Use rfe and omit the response variable (attribute 59 galaxysentiment)
rfeResultsBIG <- rfe(iphoneSample[,1:58],
                  iphoneSample$galaxysentiment,
                  sizes=(1:58),
                  rfeControl=ctrl)

# Get results
rfeResultsBIG

# Plot results
plot(rfeResultsBIG, type=c("g", "o"))
```

BIG RFE

Hide

```
# create new data set with rfe recommended features
samsungDFRFE <- samsungDF[,predictors(rfeResultsBIG)]

# add the dependent variable to iphoneRFE
samsungDFRFE$galaxysentiment <- samsungDF$galaxysentiment

# review outcome
str(samsungDFRFE)
```

My_Data_Sets

iPhoneBig - samsungDF <- samsungDF - samsungDFCOR <- - samsungDFNZV <- - samsungDFRFE <- samsungDFRFE - samsungDFRecoded <- optional tasks - samsungDFPCA <- optional tasks

Model Building

Hide

```
# create 10-fold cross validation fitcontrol
fitControl <- trainControl(method = "cv", number = 10)
```

samsungDF

Hide

```
# convert variable types, categorical
samsungDF$galaxysentiment <- as.factor(samsungDF$galaxysentiment)
```

Train and Test Set:

Hide

```
# Create Train and Test Set for samsungDF
# create 75% sample of row indices
in_training <-createDataPartition(samsungDF$galaxysentiment, p = .7, list = FALSE)
# create 75% sample of data and save it to trainData
trainData_samsungDF <- samsungDF[in_training, ]
 # create 25% sample of data and save it to test_data
testData_samsungDF <- samsungDF[-in_training, ]
# verify split percentages
nrow(trainData_samsungDF) / nrow(samsungDF)
```

```
[1] 0.7001781
```

Hide

```
#c5
c5_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "C5.0",
                trControl = fitControl)
```

Hide

```
# randomforest
rf_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "rf",
                trControl = fitControl)
```

Hide

```
# svm   (kernlab)
svm_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "svmLinear",
                trControl = fitControl)
```

Hide

```
# kknn
kknn_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "kknn",
                trControl = fitControl)
```

Hide

```
# gbm
#gbm_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "gbm",
#                trControl = fitControl)
```

Compare Accuracy on Prediction Results:

Hide

```
#c5
prediction_c5_samsungDF <- predict(c5_samsungDF, testData_samsungDF)
postResample(prediction_c5_samsungDF, testData_samsungDF$galaxysentiment)
```

```
 Accuracy     Kappa 
0.7672436 0.5328534 
```

Hide

```
#randomforest
prediction_rf_samsungDF  <- predict(rf_samsungDF, testData_samsungDF)
postResample(prediction_rf_samsungDF, testData_samsungDF$galaxysentiment)
```

```
 Accuracy     Kappa 
0.7638853 0.5314217 
```

Hide

```
#svm
prediction_svm_samsungDF  <- predict(svm_samsungDF, testData_samsungDF)
postResample(prediction_svm_samsungDF, testData_samsungDF$galaxysentiment)
```

```
 Accuracy     Kappa 
0.7083441 0.3906673 
```

Hide

```
# kknn
prediction_kknn_samsungDF  <- predict(kknn_samsungDF, testData_samsungDF)
postResample(prediction_kknn_samsungDF, testData_samsungDF$galaxysentiment)
```

```
 Accuracy     Kappa 
0.7414105 0.4968576 
```

Hide

```
# gbm
#prediction_gbm_samsungDF  <- predict(gbm_samsungDF, testData_samsungDF)
#Model summary for comparisons
```

Hide

```
modelData_samsungDF <- resamples(list(C50 = c5_samsungDF, randomForest = rf_samsungDF, svMLinear = svm_samsungDF,
kknn = kknn_samsungDF))
```

Hide

```
summary(modelData_samsungDF)
```

```


Call:
summary.resamples(object = modelData_samsungDF)

Models: C50, randomForest, svMLinear, kknn 
Number of resamples: 10 

Accuracy 
                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
C50          0.7483444 0.7607781 0.7664613 0.7654906 0.7724004 0.7743363    0
randomForest 0.7430786 0.7573919 0.7653575 0.7637116 0.7732301 0.7798673    0
svMLinear    0.6902655 0.6985619 0.7033757 0.7047587 0.7090353 0.7223451    0
kknn         0.7226519 0.7348629 0.7421133 0.7443624 0.7551864 0.7721239    0

Kappa 
                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
C50          0.4943726 0.5158095 0.5336840 0.5297828 0.5451081 0.5498118    0
randomForest 0.4912960 0.5178298 0.5354207 0.5296130 0.5476841 0.5558303    0
svMLinear    0.3375591 0.3669960 0.3759651 0.3809882 0.3954813 0.4221965    0
kknn         0.4671263 0.4813138 0.4892321 0.5016563 0.5211410 0.5600537    0
```

Train and Test Set:

Hide

```
# Create Train and Test Set for samsungDF
# create 75% sample of row indices
in_training <-createDataPartition(samsungDF$galaxysentiment, p = .7, list = FALSE)
# create 75% sample of data and save it to trainData
trainData_samsungDF <- samsungDF[in_training, ]
 # create 25% sample of data and save it to test_data
testData_samsungDF <- samsungDF[-in_training, ]
# verify split percentages
nrow(trainData_samsungDF) / nrow(samsungDF)
```

Hide

```
#c5
c5_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "C5.0",
                trControl = fitControl)
```

Hide

```
# randomforest
rf_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "rf",
                trControl = fitControl)
```

Hide

```
# svm   (kernlab)
svm_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "svmLinear",
                trControl = fitControl)
```

Hide

```
# kknn
kknn_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "kknn",
                trControl = fitControl)
```

Hide

```
# gbm
#gbm_samsungDF <- train(galaxysentiment ~., data = trainData_samsungDF, method = "gbm",
#                 trControl = fitControl)
```

Compare Accuracy on Prediction Results:

Hide

```
#c5
prediction_c5_samsungDF <- predict(c5_samsungDF, testData_samsungDF)
postResample(prediction_c5_samsungDF, testData_samsungDF$galaxysentiment)
#randomforest
prediction_rf_samsungDF  <- predict(rf_samsungDF, testData_samsungDF)
postResample(prediction_rf_samsungDF, testData_samsungDF$galaxysentiment)
#svm
prediction_svm_samsungDF  <- predict(svm_samsungDF, testData_samsungDF)
postResample(prediction_svm_samsungDF, testData_samsungDF$galaxysentiment)
# kknn
prediction_kknn_samsungDF  <- predict(kknn_samsungDF, testData_samsungDF)
postResample(prediction_kknn_samsungDF, testData_samsungDF$galaxysentiment)
```

Hide

```
modelData_samsungDF <- resamples(list(C50 = c5_samsungDF, randomForest = rf_samsungDF, svMLinear = svm_samsungDF,
kknn = kknn_samsungDF))
```

Hide

```
summary(modelData_samsungDF)
```

Choose final model: Evaluating model efficiency

Hide

```
# Create a confusion matrix from random forest predictions
cmC5 <- confusionMatrix(prediction_c5_samsungDF, testData_samsungDF$galaxysentiment)
cmC5
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  345    1    1    3    7   31
         1    0    0    0    0    0    0
         2    0    0   17    0    1    0
         3    2    1    2  209    3   26
         4    7    1    0    0  133   14
         5  154  111  115  140  281 2266

Overall Statistics

               Accuracy : 0.7672
                 95% CI : (0.7536, 0.7805)
    No Information Rate : 0.6037
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5329

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity           0.67913  0.00000 0.125926  0.59375  0.31294   0.9696
Specificity           0.98721  1.00000 0.999732  0.99034  0.99362   0.4778
Pos Pred Value        0.88918      NaN 0.944444  0.86008  0.85806   0.7388
Neg Pred Value        0.95320  0.97055 0.969375  0.96058  0.92142   0.9117
Prevalence            0.13123  0.02945 0.034875  0.09093  0.10979   0.6037
Detection Rate        0.08912  0.00000 0.004392  0.05399  0.03436   0.5854
Detection Prevalence  0.10023  0.00000 0.004650  0.06277  0.04004   0.7923
Balanced Accuracy     0.83317  0.50000 0.562829  0.79204  0.65328   0.7237
```

Hide

```
cmRM <- confusionMatrix(prediction_rf_samsungDF, testData_samsungDF$galaxysentiment)
cmRM
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5
         0  351    1    0    3    8   32
         1    0    0    0    0    1    1
         2    0    1   17    0    1    4
         3    1    1    2  214    6   31
         4    7    1    0    2  134   28
         5  149  110  116  133  275 2241

Overall Statistics

               Accuracy : 0.7639
                 95% CI : (0.7502, 0.7772)
    No Information Rate : 0.6037
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5314

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0  Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
Sensitivity           0.69094 0.0000000 0.125926  0.60795  0.31529   0.9589
Specificity           0.98692 0.9994677 0.998394  0.98835  0.98897   0.4896
Pos Pred Value        0.88861 0.0000000 0.739130  0.83922  0.77907   0.7411
Neg Pred Value        0.95483 0.9705350 0.969335  0.96184  0.92133   0.8867
Prevalence            0.13123 0.0294498 0.034875  0.09093  0.10979   0.6037
Detection Rate        0.09067 0.0000000 0.004392  0.05528  0.03462   0.5789
Detection Prevalence  0.10204 0.0005167 0.005942  0.06587  0.04443   0.7812
Balanced Accuracy     0.83893 0.4997338 0.562160  0.79815  0.65213   0.7242
```