Annex 1 - R comparison code

```
require(knitr)
require(stats)
require(caret)
require(e1071)
require(ica)
require(psych)
require(MASS)
require(ggplot2)
require(ggfortify)
require(DMwR)
```

```
# Data metavariables
data_class_factors <- 7
data_class_numeric <- 2867
data_response_index <- 2
```

```
# Classes of variable by column
colClasses <- append(c(rep("factor", data_class_factors)),
    c(rep("numeric", data_class_numeric)))

# Load CSV data
data_df <- read.csv2(file = "data/data.csv", sep = ",",
    dec = ".", colClasses = colClasses, stringsAsFactors = FALSE)

# Model formula
data_formula <- as.formula(paste(colnames(data_df[data_response_index]),
    "~", paste(colnames(data_df)[-(1:data_class_factors)],
        collapse = " + ")))

# Balance out data with SMOTE, as ROSE only works
# on binary classifications.
data_df <- SMOTE(data_formula, perc.over = 200, k = 5)

# Seed for controlled randomization
set.seed(123)

# Response and predictor subsets
data_df_predictors <- data_df[, 8:2874]
data_df_predictors <- scale(data_df_predictors)
data_df_responses <- data_df[, data_response_index]

# Training indices
data_train_index <- sample(1:nrow(data_df), ceiling(nrow(data_df) *
    0.66))
write.table(as.vector(data_train_index), file = "data/data_train_index.csv",
    row.names = FALSE, col.names = FALSE, sep = ",")

# Training responses subset
data_df_train_responses <- data_df_responses[data_train_index]

# Test responses subset
data_df_test_responses <- data_df_responses[-data_train_index]
```

```r
# Fit PCA
data_pca <- prcomp(data_df_predictors)

# Summarize PCA
data_pca_summary <- summary(data_pca)

# Get the number of variables explaining at the
# very least 85% of variance
for (i in seq(from = 10, to = 250, by = 10)) {
    data_varExp <- sum(data_pca_summary$importance[2,
        1:i])
    if (data_varExp >= 0.95) {
        varImpMessage <- paste("For ", i, " components, the percentage of variance explained is ",
            data_varExp, ".", sep = "")
        data_optimal_exfeat <- i
        return(print(varImpMessage))
    }
}
```

```r
cat(varImpMessage)
```

```r
# Choose the selected number of PCs
data_pca_exfeat <- data_pca$x[, 1:data_optimal_exfeat]

# Generate training and test sets
data_pca_train <- subset(data_pca_exfeat[data_train_index,
    ])
data_pca_test <- subset(data_pca_exfeat[-data_train_index,
    ])

# Fit SVM with training sets
data_pca_svm <- svm(data_pca_train, y = data_df_train_responses,
    type = "C-classification", kernel = "radial")

# Feed SVM with test data for prediction
data_pca_predict <- stats::predict(data_pca_svm, data_pca_test)

# Tabulate and solve Cohen's Kappa
data_pca_table <- table(data_pca_predict, data_df_test_responses)
data_pca_perc_table <- prop.table(data_pca_table) *
    100
data_pca_perc_hit <- sum(diag(data_pca_perc_table))
data_pca_cohen <- cohen.kappa(data_pca_table, n.obs = length(data_df_test_responses))
```

```r
kable(data_pca_table, caption = "PCA observed versus predicted results",
    digits = 2, format = "latex")
```

```r
kable(data_pca_perc_table, caption = "PCA observed versus predicted results - percentages",
    digits = 2, format = "latex")
```

```r
# Choose the selected number of features while
# fitting ICA
data_ica <- icafast(data_df_predictors, nc = data_optimal_exfeat)
data_ica_exfeat <- data_ica$S
```

```r
# Generate training and test sets
data_ica_train <- subset(data_ica_exfeat[data_train_index,
    ])
data_ica_test <- subset(data_ica_exfeat[-data_train_index,
    ])

# Fit SVM with training sets
data_ica_svm <- svm(data_ica_train, y = data_df_train_responses,
    type = "C-classification", kernel = "radial")

# Feed SVM with test data for prediction
data_ica_predict <- stats::predict(data_ica_svm, data_ica_test)

# Tabulate and solve Cohen's Kappa
data_ica_table <- table(data_ica_predict, data_df_test_responses)
data_ica_perc_table <- prop.table(data_ica_table) *
    100
data_ica_perc_hit <- sum(diag(data_ica_perc_table))
data_ica_cohen <- cohen.kappa(data_ica_table, n.obs = length(data_df_test_responses))
```

```r
kable(data_ica_table, caption = "ICA observed versus predicted results",
    digits = 2, format = "latex")
```

```r
kable(data_ica_perc_table, caption = "ICA observed versus predicted results - percentages",
    digits = 2, format = "latex")
```

```r
# plot(data_ica)
```

```r
# Choose the selected number of features with a
# lower tolerance boundary extracted on trial and
# error to be the best in convergence
data_factanal <- factanal(data_df_predictors, factors = data_optimal_exfeat,
    scores = "Bartlett", lower = 0.07)

# Use loadings to transform preditor values
data_factanal_exfeat <- data_df_predictors %*% data_factanal$loadings

# Generate training and test sets
data_factanal_train <- subset(data_factanal_exfeat[data_train_index,
    ])
data_factanal_test <- subset(data_factanal_exfeat[-data_train_index,
    ])

# Fit SVM with those factors
data_factanal_svm <- svm(data_factanal_train, y = data_df_train_responses,
    type = "C-classification", kernel = "radial")

# Use fitted SVM to predict test responses
data_factanal_predict <- stats::predict(data_factanal_svm,
    data_factanal_test)

# Tabulate and solve Coehn's Kappa
data_factanal_table <- table(data_factanal_predict,
    data_df_test_responses)
```

```
data_factanal_perc_table <- prop.table(data_factanal_table) *
    100
data_factanal_perc_hit <- sum(diag(data_factanal_perc_table))
data_factanal_cohen <- cohen.kappa(data_factanal_table)
```

After fitting and predicting, the hit and accuracy values are extracted and represented in **Table 5** and **Table 6**, in absolute and percentage values, respectively.

```
kable(data_factanal_table, caption = "Factor Analysis observed versus predicted results",
    digits = 2, format = "latex")
```

```
kable(data_factanal_perc_table, caption = "Factor Analysis observed versus predicted results - percentag
    digits = 2, format = "latex")
```

```
# Fit LDA with predictors
data_lda <- lda(data_df_predictors, grouping = data_df_responses)

# Transform predictors with linear discriminants
data_lda_predictors_trans <- data_df_predictors %*%
    data_lda$scaling

# Generate training and test sets
data_lda_train <- subset(data_lda_predictors_trans[data_train_index,
    ])
data_lda_test <- subset(data_lda_predictors_trans[-data_train_index,
    ])

# Use training values to fit SVM
data_lda_svm <- svm(data_lda_train, y = data_df_train_responses,
    type = "C-classification", kernel = "radial")

# Feed test data to SVM and predict
data_lda_predict <- stats::predict(data_lda_svm, data_lda_test)

# Tabulate and solve Cohen's Kappa
data_lda_table <- table(data_lda_predict, data_df_test_responses)
data_lda_perc_table <- prop.table(data_lda_table) *
    100
data_lda_perc_hit <- sum(diag(data_lda_perc_table))
data_lda_cohen <- cohen.kappa(data_lda_table)
```

```
kable(data_lda_table, caption = "LDA observed versus predicted results",
    digits = 2, format = "latex")
```

```
kable(data_lda_perc_table, caption = "LDA observed versus predicted results - percentages",
    digits = 2, format = "latex")
```