Annex 2 - Python comparison code

```
#Import all necessary modules
import scipy
import numpy
import pandas as pd
from sklearn.decomposition import PCA
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix

#Define data metavariables and load data
data_pydf_predictor_ids = range(7,2874)
data_pydf_predictors = pd.read_csv('data/data.csv', usecols=data_pydf_predictor_ids)
data_pydf_predictors = data_pydf_predictors.values
data_pydf_responses = pd.read_csv('data/data.csv', usecols=['tipoCelula'])
data_pydf_responses = data_pydf_responses.values

#Configure PCA object
data_pca_py = PCA(n_components=210,copy=True)

#Fit PCA with predictors
data_pca_py_fit = data_pca_py.fit(data_pydf_predictors, data_pydf_responses.ravel())

#Transform predictors on selected PCs
data_pca_py_transf = data_pca_py_fit.transform(data_pydf_predictors)

#Subset transformed data into test and training subsets
data_pca_predictors_train, data_pca_predictors_test, data_pca_responses_train, data_pca_responses_test =

#Convert response array to 1D for prediction
data_pca_responses_test_1D = data_pca_responses_test[:,0]

#Initialize SVM
data_pca_py_svm = svm.SVC(kernel='rbf')

#Fit SVM
data_pca_py_svm_fit = data_pca_py_svm.fit(data_pca_predictors_train, data_pca_responses_train.ravel()).

#Put classification results in temp file
numpy.savetxt("data_pca_responses_test_1D.csv", data_pca_responses_test_1D, delimiter=",", fmt="%s")

numpy.savetxt("data_pca_py_svm_fit.csv", data_pca_py_svm_fit, delimiter=",", fmt="%s")
```

```
require(knitr)
require(psych)
# Load classification data as R objects
data_pca_responses_test_1D <- read.csv(file = "data_pca_responses_test_1D.csv",
    sep = ",", header = FALSE)
data_pca_responses_test_1D <- data_pca_responses_test_1D[,
    1]

data_pca_py_svm_fit <- read.csv(file = "data_pca_py_svm_fit.csv",
```

```
    sep = ",", header = FALSE)
data_pca_py_svm_fit <- data_pca_py_svm_fit[, 1]

# Tabulate and solve Cohen's Kappa
data_pca_py_table <- table(data_pca_py_svm_fit, data_pca_responses_test_1D)
data_pca_py_perc_table <- prop.table(data_pca_py_table) *
    100
data_pca_py_perc_hit <- sum(diag(data_pca_py_perc_table))
data_pca_py_cohen <- cohen.kappa(data_pca_py_table,
    n.obs = length(data_pca_responses_test_1D))
```

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

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

```
#Import all necessary modules
import scipy
import numpy
import pandas as pd
from sklearn.decomposition import FastICA
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix

#Define data metavariables and load data
data_pydf_predictor_ids = range(7,2874)
data_pydf_predictors = pd.read_csv('data/data.csv', usecols=data_pydf_predictor_ids)
data_pydf_predictors = data_pydf_predictors.values
data_pydf_responses = pd.read_csv('data/data.csv', usecols=['tipoCelula'])
data_pydf_responses = data_pydf_responses.values

#Configure ICA object
data_ica_py = FastICA(n_components=210)

#Fit ICA with predictors
data_ica_py_fit = data_ica_py.fit(data_pydf_predictors, data_pydf_responses.ravel())

#Transform predictors on selected factors
data_ica_py_transf = data_ica_py_fit.transform(data_pydf_predictors)

#Subset transformed data into test and training subsets
data_ica_predictors_train, data_ica_predictors_test, data_ica_responses_train, data_ica_responses_test =

#Convert response array to 1D for prediction
data_ica_responses_test_1D = data_ica_responses_test[:,0]

#Initialize SVM
data_ica_py_svm = svm.SVC(kernel='rbf')

#Fit SVM
data_ica_py_svm_fit = data_ica_py_svm.fit(data_ica_predictors_train, data_ica_responses_train.ravel()).
```

```
#Put classification results in temp file
numpy.savetxt("data_ica_responses_test_1D.csv", data_ica_responses_test_1D, delimiter=",", fmt="%s")

numpy.savetxt("data_ica_py_svm_fit.csv", data_ica_py_svm_fit, delimiter=",", fmt="%s")
```

```
# Load classification data as R objects
data_ica_responses_test_1D <- read.csv(file = "data_ica_responses_test_1D.csv",
    sep = ",", header = FALSE)
data_ica_responses_test_1D <- data_ica_responses_test_1D[,
    1]

data_ica_py_svm_fit <- read.csv(file = "data_ica_py_svm_fit.csv",
    sep = ",", header = FALSE)
data_ica_py_svm_fit <- data_ica_py_svm_fit[, 1]

# Tabulate and solve Cohen's Kappa
data_ica_py_table <- table(data_ica_py_svm_fit, data_ica_responses_test_1D)
data_ica_py_perc_table <- prop.table(data_ica_py_table) *
    100
data_ica_py_perc_hit <- sum(diag(data_ica_py_perc_table))
data_ica_py_cohen <- cohen.kappa(data_ica_py_table,
    n.obs = length(data_ica_responses_test_1D))
```

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

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

```
#Import all necessary modules
import scipy
import numpy
import pandas as pd
from sklearn.decomposition import FactorAnalysis
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix

#Define data metavariables and load data
data_pydf_predictor_ids = range(7,2874)
data_pydf_predictors = pd.read_csv('data/data.csv', usecols=data_pydf_predictor_ids)
data_pydf_predictors = data_pydf_predictors.values
data_pydf_responses = pd.read_csv('data/data.csv', usecols=['tipoCelula'])
data_pydf_responses = data_pydf_responses.values

#Configure Factor Analysis object
data_factanal_py = FactorAnalysis(n_components=210)

#Fit FA with predictors
data_factanal_py_fit = data_factanal_py.fit(data_pydf_predictors, data_pydf_responses.ravel())

#Transform predictors on selected factors
data_factanal_py_transf = data_factanal_py_fit.transform(data_pydf_predictors)
```

```
#Subset transformed data into test and training subsets
data_factanal_predictors_train, data_factanal_predictors_test, data_factanal_responses_train, data_facta

#Convert response array to 1D for prediction
data_factanal_responses_test_1D = data_factanal_responses_test[:,0]

#Initialize SVM
data_factanal_py_svm = svm.SVC(kernel='rbf')

#Fit SVM
data_factanal_py_svm_fit = data_factanal_py_svm.fit(data_factanal_predictors_train, data_factanal_respor

#Put classification results in temp file
numpy.savetxt("data_factanal_responses_test_1D.csv", data_factanal_responses_test_1D, delimiter=",", fm

numpy.savetxt("data_factanal_py_svm_fit.csv", data_factanal_py_svm_fit, delimiter=",", fmt="%s")
```

```
# Load classification data as R objects
data_factanal_responses_test_1D <- read.csv(file = "data_factanal_responses_test_1D.csv",
    sep = ",", header = FALSE)
data_factanal_responses_test_1D <- data_factanal_responses_test_1D[,
    1]

data_factanal_py_svm_fit <- read.csv(file = "data_factanal_py_svm_fit.csv",
    sep = ",", header = FALSE)
data_factanal_py_svm_fit <- data_factanal_py_svm_fit[,
    1]

# Tabulate and solve Cohen's Kappa
data_factanal_py_table <- table(data_factanal_py_svm_fit,
    data_factanal_responses_test_1D)
data_factanal_py_perc_table <- prop.table(data_factanal_py_table) *
    100
data_factanal_py_perc_hit <- sum(diag(data_factanal_py_perc_table))
data_factanal_py_cohen <- cohen.kappa(data_factanal_py_table,
    n.obs = length(data_factanal_responses_test_1D))
```

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

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

```
#Import all necessary modules
import scipy
import numpy
import pandas as pd
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix

#Define data metavariables and load data
```

```
data_pydf_predictor_ids = range(7,2874)
data_pydf_predictors = pd.read_csv('data/data.csv', usecols=data_pydf_predictor_ids)
data_pydf_predictors = data_pydf_predictors.values
data_pydf_responses = pd.read_csv('data/data.csv', usecols=['tipoCelula'])
data_pydf_responses = data_pydf_responses.values

#Configure LDA object
data_lda_py = LinearDiscriminantAnalysis(n_components=210)

#Fit LDA with predictors
data_lda_py_fit = data_lda_py.fit(data_pydf_predictors, data_pydf_responses.ravel())

#Transform predictors on selected factors
data_lda_py_transf = data_lda_py_fit.transform(data_pydf_predictors)

#Subset transformed data into test and training subsets
data_lda_predictors_train, data_lda_predictors_test, data_lda_responses_train, data_lda_responses_test =

#Convert response array to 1D for prediction
data_lda_responses_test_1D = data_lda_responses_test[:,0]

#Initialize SVM
data_lda_py_svm = svm.SVC(kernel='rbf')

#Fit SVM
data_lda_py_svm_fit = data_lda_py_svm.fit(data_lda_predictors_train, data_lda_responses_train.ravel()).p

#Put classification results in temp file
numpy.savetxt("data_lda_responses_test_1D.csv", data_lda_responses_test_1D, delimiter=",", fmt="%s")

numpy.savetxt("data_lda_py_svm_fit.csv", data_lda_py_svm_fit, delimiter=",", fmt="%s")
```

```
# Load classification data as R objects
data_lda_responses_test_1D <- read.csv(file = "data_lda_responses_test_1D.csv",
    sep = ",", header = FALSE)
data_lda_responses_test_1D <- data_lda_responses_test_1D[,
    1]

data_lda_py_svm_fit <- read.csv(file = "data_lda_py_svm_fit.csv",
    sep = ",", header = FALSE)
data_lda_py_svm_fit <- data_lda_py_svm_fit[, 1]

# Tabulate and solve Cohen's Kappa
data_lda_py_table <- table(data_lda_py_svm_fit, data_lda_responses_test_1D)
data_lda_py_perc_table <- prop.table(data_lda_py_table) *
    100
data_lda_py_perc_hit <- sum(diag(data_lda_py_perc_table))
data_lda_py_cohen <- cohen.kappa(data_lda_py_table,
    n.obs = length(data_lda_responses_test_1D))
```

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

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