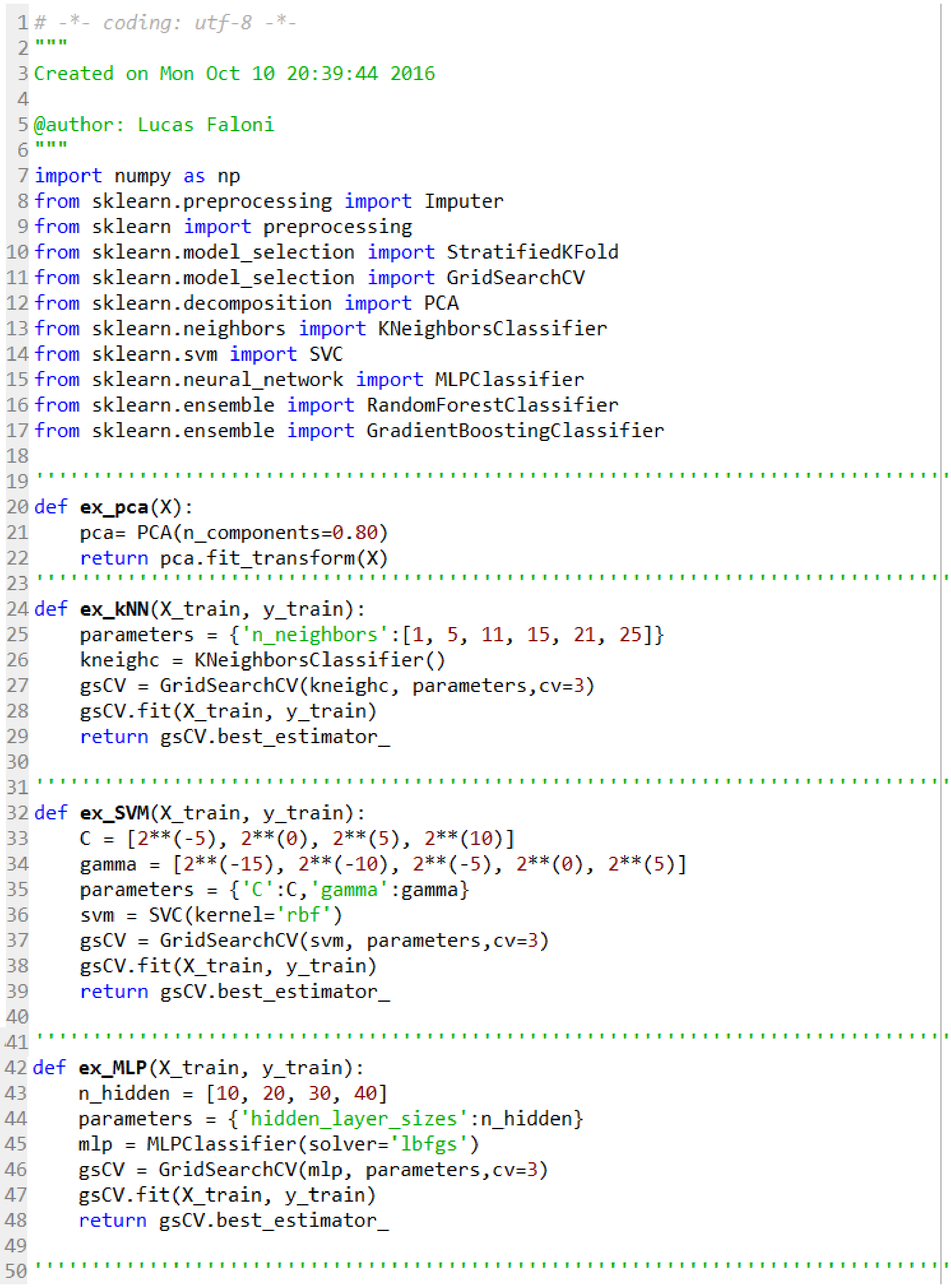
MO444/MC886 - Atividade 3

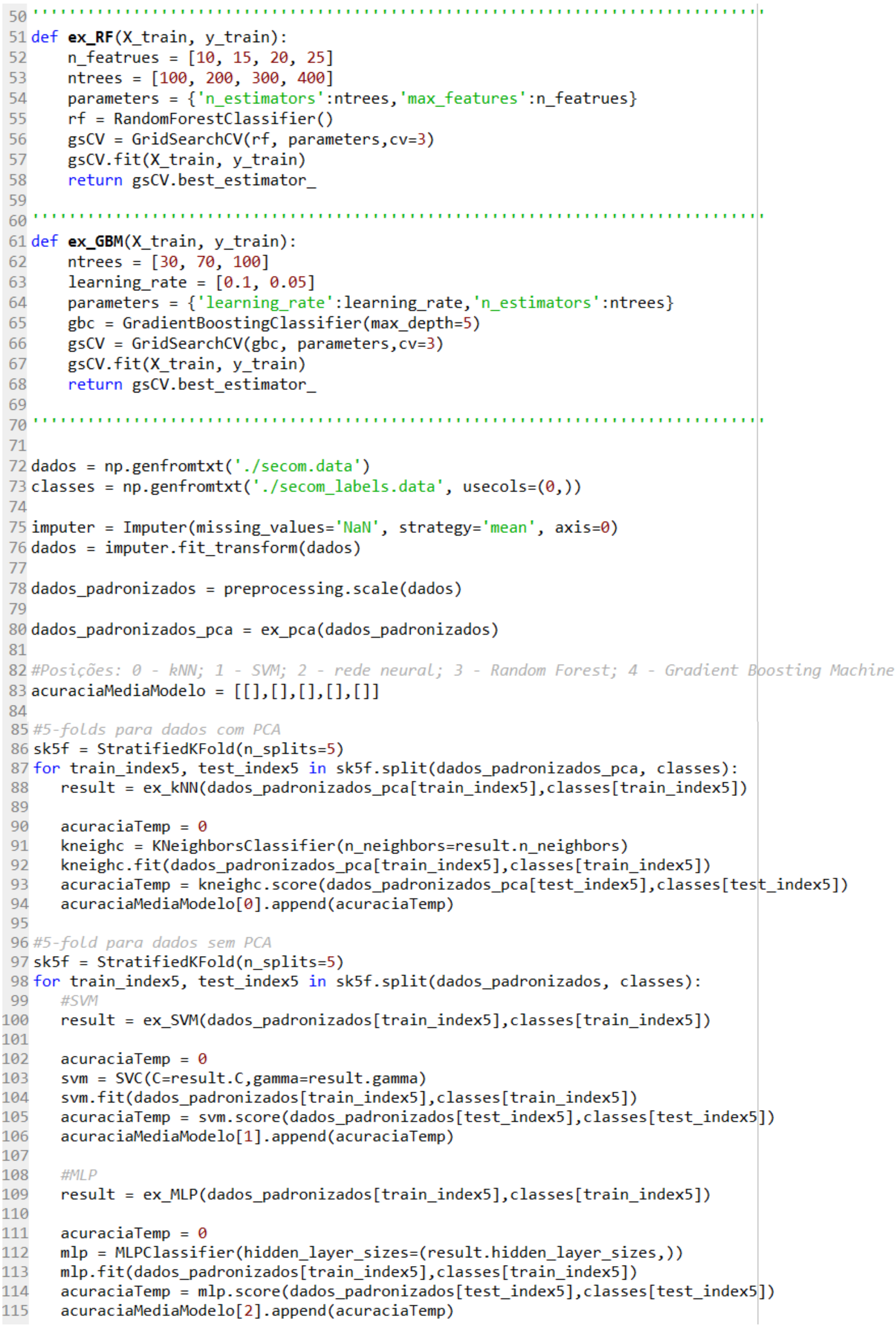
Prof. Dr. Jacques Wainer

Lucas Faloni Ferreira RA: 188943

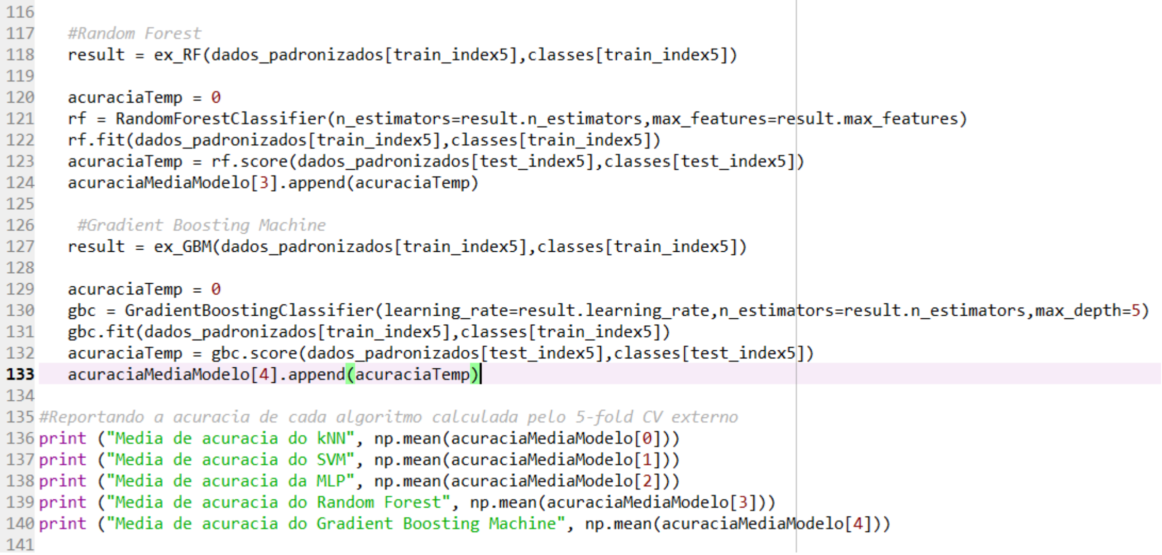
Código:



```python
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 10 20:39:44 2016

@author: Lucas Faloni
"""
import numpy as np
from sklearn.preprocessing import Imputer
from sklearn import preprocessing
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def ex_pca(X):
    pca= PCA(n_components=0.80)
    return pca.fit_transform(X)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def ex_kNN(X_train, y_train):
    parameters = {'n_neighbors':[1, 5, 11, 15, 21, 25]}
    kneighc = KNeighborsClassifier()
    gsCV = GridSearchCV(kneighc, parameters,cv=3)
    gsCV.fit(X_train, y_train)
    return gsCV.best_estimator_

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def ex_SVM(X_train, y_train):
    C = [2**(-5), 2**(0), 2**(5), 2**(10)]
    gamma = [2**(-15), 2**(-10), 2**(-5), 2**(0), 2**(5)]
    parameters = {'C':C,'gamma':gamma}
    svm = SVC(kernel='rbf')
    gsCV = GridSearchCV(svm, parameters,cv=3)
    gsCV.fit(X_train, y_train)
    return gsCV.best_estimator_

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def ex_MLP(X_train, y_train):
    n_hidden = [10, 20, 30, 40]
    parameters = {'hidden_layer_sizes':n_hidden}
    mlp = MLPClassifier(solver='lbfgs')
    gsCV = GridSearchCV(mlp, parameters,cv=3)
    gsCV.fit(X_train, y_train)
    return gsCV.best_estimator_

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```



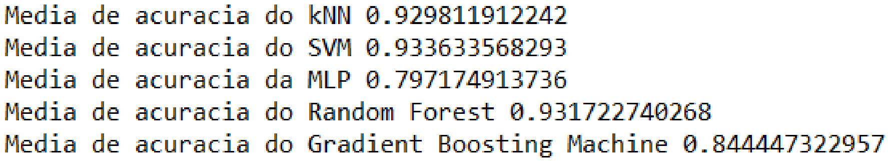
```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def ex_RF(X_train, y_train):
    n_featrues = [10, 15, 20, 25]
    ntrees = [100, 200, 300, 400]
    parameters = {'n_estimators':ntrees,'max_features':n_featrues}
    rf = RandomForestClassifier()
    gsCV = GridSearchCV(rf, parameters,cv=3)
    gsCV.fit(X_train, y_train)
    return gsCV.best_estimator_

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def ex_GBM(X_train, y_train):
    ntrees = [30, 70, 100]
    learning_rate = [0.1, 0.05]
    parameters = {'learning_rate':learning_rate,'n_estimators':ntrees}
    gbc = GradientBoostingClassifier(max_depth=5)
    gsCV = GridSearchCV(gbc, parameters,cv=3)
    gsCV.fit(X_train, y_train)
    return gsCV.best_estimator_

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

dados = np.genfromtxt('./secom.data')
classes = np.genfromtxt('./secom_labels.data', usecols=(0,))

imputer = Imputer(missing_values='NaN', strategy='mean', axis=0)
dados = imputer.fit_transform(dados)

dados_padronizados = preprocessing.scale(dados)

dados_padronizados_pca = ex_pca(dados_padronizados)

#Posições: 0 - kNN; 1 - SVM; 2 - rede neural; 3 - Random Forest; 4 - Gradient Boosting Machine
acuraciaMediaModelo = [[],[],[],[],[]]

#5-folds para dados com PCA
sk5f = StratifiedKFold(n_splits=5)
for train_index5, test_index5 in sk5f.split(dados_padronizados_pca, classes):
   result = ex_kNN(dados_padronizados_pca[train_index5],classes[train_index5])

   acuraciaTemp = 0
   kneighc = KNeighborsClassifier(n_neighbors=result.n_neighbors)
   kneighc.fit(dados_padronizados_pca[train_index5],classes[train_index5])
   acuraciaTemp = kneighc.score(dados_padronizados_pca[test_index5],classes[test_index5])
   acuraciaMediaModelo[0].append(acuraciaTemp)

#5-fold para dados sem PCA
sk5f = StratifiedKFold(n_splits=5)
for train_index5, test_index5 in sk5f.split(dados_padronizados, classes):
   #SVM
   result = ex_SVM(dados_padronizados[train_index5],classes[train_index5])

   acuraciaTemp = 0
   svm = SVC(C=result.C,gamma=result.gamma)
   svm.fit(dados_padronizados[train_index5],classes[train_index5])
   acuraciaTemp = svm.score(dados_padronizados[test_index5],classes[test_index5])
   acuraciaMediaModelo[1].append(acuraciaTemp)

   #MLP
   result = ex_MLP(dados_padronizados[train_index5],classes[train_index5])

   acuraciaTemp = 0
   mlp = MLPClassifier(hidden_layer_sizes=(result.hidden_layer_sizes,))
   mlp.fit(dados_padronizados[train_index5],classes[train_index5])
   acuraciaTemp = mlp.score(dados_padronizados[test_index5],classes[test_index5])
   acuraciaMediaModelo[2].append(acuraciaTemp)
```



```python

    #Random Forest
    result = ex_RF(dados_padronizados[train_index5],classes[train_index5])

    acuraciaTemp = 0
    rf = RandomForestClassifier(n_estimators=result.n_estimators,max_features=result.max_features)
    rf.fit(dados_padronizados[train_index5],classes[train_index5])
    acuraciaTemp = rf.score(dados_padronizados[test_index5],classes[test_index5])
    acuraciaMediaModelo[3].append(acuraciaTemp)

     #Gradient Boosting Machine
    result = ex_GBM(dados_padronizados[train_index5],classes[train_index5])

    acuraciaTemp = 0
    gbc = GradientBoostingClassifier(learning_rate=result.learning_rate,n_estimators=result.n_estimators,max_depth=5)
    gbc.fit(dados_padronizados[train_index5],classes[train_index5])
    acuraciaTemp = gbc.score(dados_padronizados[test_index5],classes[test_index5])
    acuraciaMediaModelo[4].append(acuraciaTemp)

#Reportando a acuracia de cada algoritmo calculada pelo 5-fold CV externo
print ("Media de acuracia do kNN", np.mean(acuraciaMediaModelo[0]))
print ("Media de acuracia do SVM", np.mean(acuraciaMediaModelo[1]))
print ("Media de acuracia da MLP", np.mean(acuraciaMediaModelo[2]))
print ("Media de acuracia do Random Forest", np.mean(acuraciaMediaModelo[3]))
print ("Media de acuracia do Gradient Boosting Machine", np.mean(acuraciaMediaModelo[4]))

```

Perguntas e Respostas (saída do código):

1) Reporte a acurácia de cada algoritmo calculada pelo 5-fold CV externo.



```
Media de acuracia do kNN 0.929811912242
Media de acuracia do SVM 0.933633568293
Media de acuracia da MLP 0.797174913736
Media de acuracia do Random Forest 0.931722740268
Media de acuracia do Gradient Boosting Machine 0.844447322957
```