# -\*- coding: utf-8 -\*-

"""

validacao\_cruzada\_grid\_iris.py: Avaliação da performance de classificação do dataset Iris da UCI.

Avaliar a performance da classificação da base de dados Iris.

@author: Prof. Hugo de Paula

@contact: hugo@pucminas.br

@date 22 Outubro 2017

@version 1.0

"""

from sklearn import svm, datasets

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import confusion\_matrix, classification\_report

import matplotlib.pyplot as plt

import numpy as np

import itertools

# Métodos de avaliação do scikit-learn (http://scikit-learn.org/stable/modules/model\_evaluation.html)

def plot\_confusion\_matrix(cm, classes,

normalize=False,

title="Matriz de confusão",

cmap=plt.cm.Blues):

"""

Esta função plota a matriz de confusão.

Normalização pode ser feita usando o parâmetro 'normalize=True'.

O mapa de cores pode ser alterado pelo parâmetro cmap. Default: 'cmap=plt.cm.Blues'.

O parâmetro 'title' altera o título.

"""

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=30)

plt.yticks(tick\_marks, classes)

if normalize:

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

print("Matriz de confusão normalizada")

else:

print("Matriz de confusão sem normalização")

print(cm)

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, cm[i, j],

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('Classe real')

plt.xlabel('Classe prevista')

# importa a base de dados iris

iris = datasets.load\_iris()

X, y = iris.data, iris.target

class\_names = iris.target\_names

# Particiona a base de dados

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

# Instancia um classificador SVM (Support Vector Machine)

classificador = svm.SVC(probability=True, random\_state=0)

# Avalia o modelo usando validação cruzada e a mediga de Log-loss.

# A validação cruzada divide a base em K=3 partes e desta cada uma dessas

# parte frente ao restante usado no treinamento.

print("cross\_val\_score(clf, X, y, scoring='neg\_log\_loss'): ",

cross\_val\_score(classificador, X, y, scoring='neg\_log\_loss'))

print("\n---------------------------------------------------------------\n")

# Executa um classificador SVM. Propositalmente escolheu-se C bem baixo

# para analisar o impacto nos resultados. (mmodelo exgeradamente regulariado)

classificador = svm.SVC(kernel='linear', C=0.01)

y\_pred = classificador.fit(X\_train, y\_train).predict(X\_test)

print("\nClassificado usando SVM com C=0.01 (muito baixo)")

print(classification\_report(y\_test, y\_pred, target\_names=class\_names))

# Calcula a matriz de confusão

cnf\_matrix = confusion\_matrix(y\_test, y\_pred)

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names,

title='Matriz de confusão sem normalização')

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Matriz de confusão normalizada')

plt.show()

# Usa o método de busca em grade para variar o parâmetro C de 1 a 10 na

# tentativa de enocntrar o melhor classificador. baseado na minha medida de

# desempenho.

svr = svm.SVC()

params={'kernel':('linear', 'rbf'), 'C':[1, 10]}

classificadorGrid = GridSearchCV(svr, param\_grid=params)

y\_pred = classificadorGrid.fit(X\_train, y\_train).predict(X\_test)

print("\nClassificado usando SVM com GridSearch")

print("Best parameters set found on development set:")

print(classificadorGrid.best\_params\_,'\n')

print(classification\_report(y\_test, y\_pred, target\_names=class\_names))

# Calcula a matriz de confusão

cnf\_matrix = confusion\_matrix(y\_test, y\_pred)

np.set\_printoptions(precision=2)

# Plot normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize=True,

title='Matriz de confusão normalizada')

plt.show()