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

"""

visualizacao\_iris.py: Problema de classificação do dataset Iris da UCI.

@author: Prof. Hugo de Paula

@contact: hugo@pucminas.br

Baseado no livro: Andreas C. Müller, Sarah Guido (2016)

Introduction to Machine Learning with Python: A Guide for Data Scientists 1st Edition

"""

from sklearn.datasets import load\_iris

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

from sklearn.neighbors import KNeighborsClassifier

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import seaborn as sns

# Scikit-learn já possui um conjunto de datasets disponível,

# o que inclui os datasets da UCI.

iris = load\_iris()

X = iris['data']

y = iris['target']

# Um dataset é um objeto do tipo Bunch (datasets.base.Bunch), que é

# semelhante a um dicionário, com chaves ("keys") e valores ("values").

print("\nCampos to dataset Iris ({0}):\n{1}\n"

.format("iris.keys()", iris.keys()))

print("\nParte da descrição do dataset Iris ({0}):\n{1}\n"

.format("iris['DESCR'][:471]", iris['DESCR'][:471]))

print("\nNomes das classes ({0}):\n{1}\n"

.format("iris['target\_names']", iris['target\_names']))

print("\nNomes das features ({0}):\n{1}\n"

.format("iris['feature\_names']", iris['feature\_names']))

# Os dados propriamente ditos estão armazenados nos campos "data" e "target"

print("\nTipo do campo data ({0}):\n{1}\n"

.format("type(iris['data'])", type(iris['data'])))

print("\nEstrutura do campo 'data' ({0}):\n{1}\n"

.format("iris['data'].shape", iris['data'].shape))

print("\nAs primeiras 5 amostras de 'data' ({0}):\n{1}\n"

.format("iris['data'][:5]", iris['data'][:5]))

# As classes são codificadas na forma numérica, com os valores 0, 1 e 2.

# Baseado no target\_names: 0 -> Setosa, 1 -> Versicolor e 2 -> Virginica.

print("\nTipo do campo 'target' ({0}):\n{1}\n"

.format("type(iris['target'])", type(iris['target'])))

print("\nEstrutura do campo 'target' ({0}):\n{1}\n"

.format("iris['target'].shape", iris['target'].shape))

print("\nDados de 'target' ({0}):\n{1}\n"

.format("iris['target']", iris['target']))

# Particiona a base de dados Iris em base de treinamento e teste.

# O Scikit-learn possui a função que embaralha os dados e

# particiona a base:

# No Scikit-learn, normalmente os dados são representados por X (maiúsculo),

# e as classes são representadas por y (minúsculo).

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

iris['data'], iris['target'], random\_state=0)

print("\nBase de treinamento ({0}):\n{1}\n"

.format("X\_train.shape", X\_train.shape))

print("\nBase de teste ({0}):\n{1}\n"

.format("X\_test.shape", X\_test.shape))

#iris\_dataframe = pd.DataFrame(X, columns=iris.feature\_names)

iris\_dataframe = pd.read\_csv("../datasets/iris.csv", sep=';')

# Cria um gráfico de dispersao do dataframe

plt.figure()

ax2 = pd.scatter\_matrix(iris\_dataframe.iloc[:,:5], c=y, figsize=(15, 15), marker='o',

hist\_kwds={'bins': 20}, s=60, alpha=.8)

plt.figure()

ax3 = pd.plotting.parallel\_coordinates(iris\_dataframe, "Class")

# Utiliza o modelo do KNN com número de vizinhos = 1

# Todo modelo no Scikit-learn é implementado em uma classe própria, derivada

# de Estimator. A classe do KNN é chamada KNeighborsClassifier

# Instancia um objeto da classe KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=1)

# Para construir o modelo a partir da base de treinamento usa-se

# o método fit()

knn.fit(X\_train, y\_train)

# As propriedades do modelo podem ser vistas pela função print

# KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

# metric\_params=None, n\_jobs=1, n\_neighbors=1, p=2, weights='uniform')

print("\nModelo K-NN criado a partir do treinamento da Iris ({0}):\n{1}\n"

.format("knn", knn))

# Fazendo previsão com o modelo cosntruído

X\_novo = np.array([[5, 2.9, 1, 0.2]])

previsao = knn.predict(X\_novo)

print("\nEntrada: {0} -> Classe prevista: {1}\n"

.format(X\_novo, iris['target\_names'][previsao]))

# Avaliando a qualidade do modelo

# 1a forma: calculando a previsão do dado de teste e comparando com as

# respostas conhecidas.

y\_prev = knn.predict(X\_test)

acuracia = np.mean(y\_prev == y\_test)

print("\nAcurácia do modelo: {0:.2f}%\n".format(100\*acuracia))

acuracia2 = knn.score(X\_test, y\_test)

print("\nAcurácia do modelo ({0}): {1:.2f}%\n"

.format("knn.score(X\_test, y\_test)", 100\*acuracia2))

# Biblioteca seaborn. seaborn jointplot exibe gráficos de dispersão

# bivariados e histogramas univariados na mesma figura.

sns.jointplot(x="sepal length", y="sepal width", data=iris\_dataframe, size=5)

# Para adicionar as espécias de cada planta usa-se o seaborn FacetGrid

# para colorir os pontos por espécies.

sns.FacetGrid(iris\_dataframe, hue="Class", size=5) \

.map(plt.scatter, "sepal length", "sepal width") \

.add\_legend()

sns.boxplot(x="Class", y="petal length", data=iris\_dataframe)

sns.pairplot(iris\_dataframe, hue="Class", size=3)

sns.pairplot(iris\_dataframe, hue="Class", size=3, diag\_kind="kde")

iris\_dataframe.boxplot(by="Class", figsize=(12, 6))