LAB 4 – Inteligência Artificial e Robótica (CC7711)

**RNA Classificador Iris PCA e sem PCA**

**Integrantes:**

Camylla Lima Dias

Evelyn Santana de Brito

Patrícia Helena Medeiros

**SEM PCA**

Código:

from sklearn.datasets import load\_iris

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import plot\_confusion\_matrix

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

data = load\_iris()

features =data.data

target = data.target

plt.figure(figsize=(16,8))

plt.subplot(2,2,1)

plt.scatter(features[:,0], features[:,1], c=target,marker='o',cmap='viridis')

Classificador = MLPClassifier(hidden\_layer\_sizes = (10), alpha=1, max\_iter=1000)

Classificador.fit(features,target)

predicao = Classificador.predict(features)

plt.subplot(2,2,3)

plt.scatter(features[:,0], features[:,1], c=predicao,marker='d',cmap='viridis',s=150)

plt.scatter(features[:,0], features[:,1], c=target,marker='o',cmap='viridis',s=15)

#pca = PCA(n\_components= 4, whiten=True, svd\_solver='randomized')

#pca = pca.fit(features)

#pca\_features = pca.transform(features)

#print('Mantida %5.2f%% da informação do conjunto inicial de dados'%(sum(pca.explained\_variance\_ratio\_)\*100))

#plt.subplot(2,2,2)

#plt.scatter(pca\_features[:,0], pca\_features[:,1], c=target,marker='o',cmap='viridis')

#ClassificadorPCA = MLPClassifier(hidden\_layer\_sizes = (10), alpha=1, max\_iter=1000)

#ClassificadorPCA.fit(pca\_features,target)

#predicao = ClassificadorPCA.predict(pca\_features)

#plt.subplot(2,2,4)

#plt.scatter(pca\_features[:,0], pca\_features[:,1], c=predicao,marker='d',cmap='viridis',s=150)

#plt.scatter(pca\_features[:,0], pca\_features[:,1], c=target,marker='o',cmap='viridis',s=15)

#plt.show()

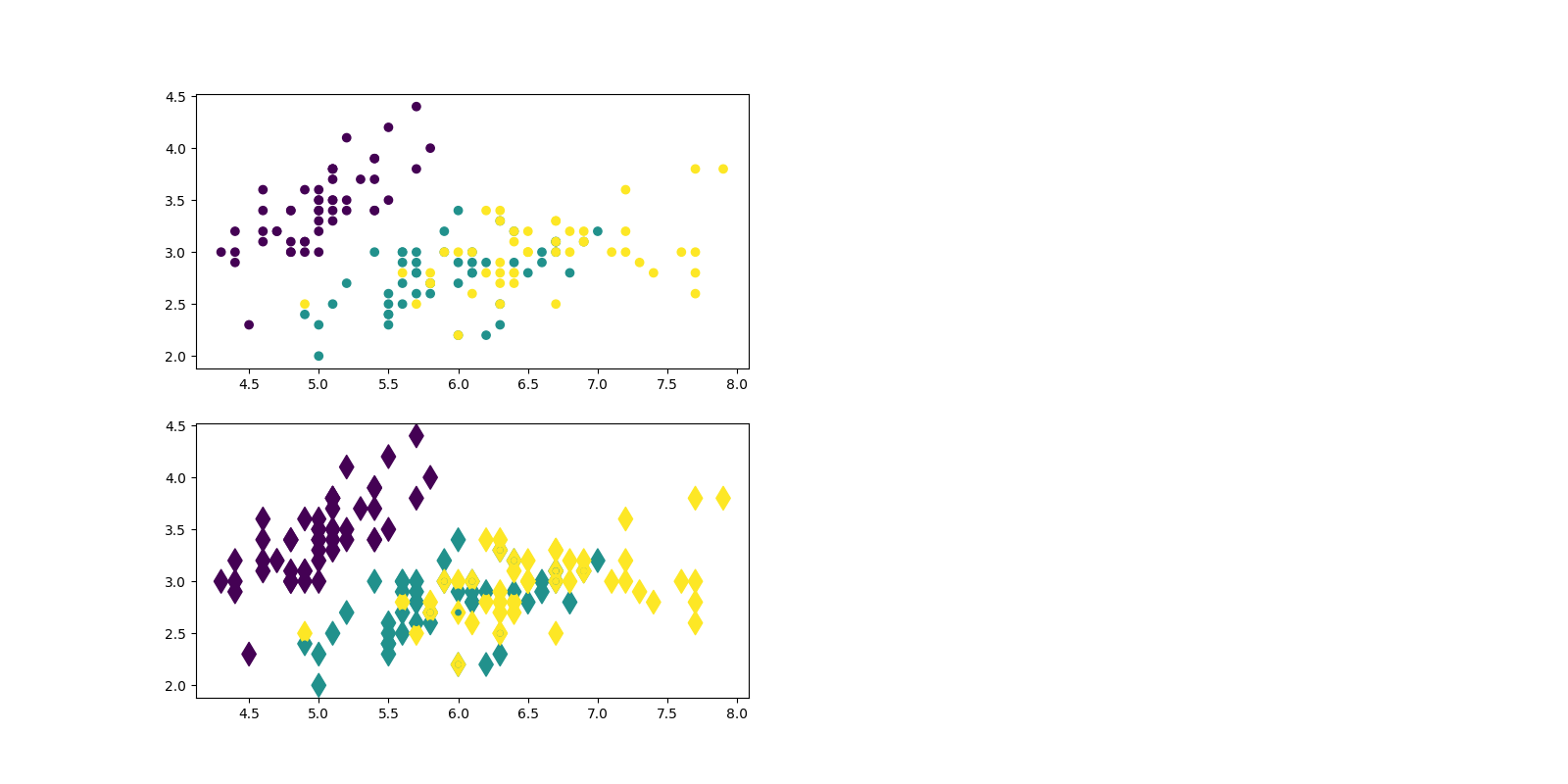
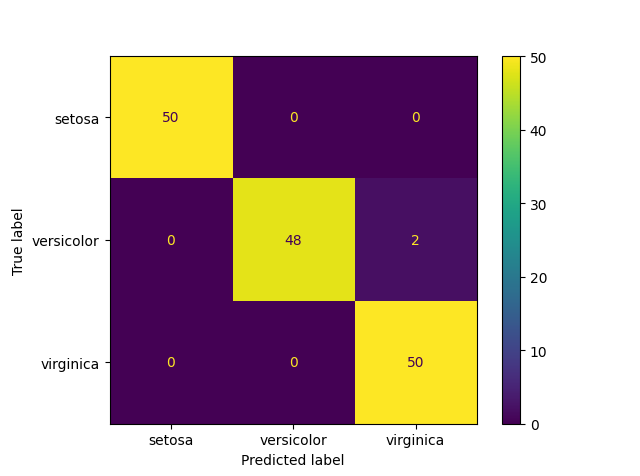
plot\_confusion\_matrix(Classificador, features, target,include\_values=True,display\_labels=data.target\_names)

plt.show()

#plot\_confusion\_matrix(ClassificadorPCA, pca\_features, target,include\_values=True,display\_labels=data.target\_names)

#plt.show()

Resultados:



COM PCA

Código:

from sklearn.datasets import load\_iris

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import plot\_confusion\_matrix

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

data = load\_iris()

features =data.data

target = data.target

plt.figure(figsize=(16,8))

plt.subplot(2,2,1)

plt.scatter(features[:,0], features[:,1], c=target,marker='o',cmap='viridis')

Classificador = MLPClassifier(hidden\_layer\_sizes = (10), alpha=1, max\_iter=1000)

Classificador.fit(features,target)

predicao = Classificador.predict(features)

plt.subplot(2,2,3)

plt.scatter(features[:,0], features[:,1], c=predicao,marker='d',cmap='viridis',s=150)

plt.scatter(features[:,0], features[:,1], c=target,marker='o',cmap='viridis',s=15)

pca = PCA(n\_components= 4, whiten=True, svd\_solver='randomized')

pca = pca.fit(features)

pca\_features = pca.transform(features)

print('Mantida %5.2f%% da informação do conjunto inicial de dados'%(sum(pca.explained\_variance\_ratio\_)\*100))

plt.subplot(2,2,2)

plt.scatter(pca\_features[:,0], pca\_features[:,1], c=target,marker='o',cmap='viridis')

ClassificadorPCA = MLPClassifier(hidden\_layer\_sizes = (10), alpha=1, max\_iter=1000)

ClassificadorPCA.fit(pca\_features,target)

predicao = ClassificadorPCA.predict(pca\_features)

plt.subplot(2,2,4)

plt.scatter(pca\_features[:,0], pca\_features[:,1], c=predicao,marker='d',cmap='viridis',s=150)

plt.scatter(pca\_features[:,0], pca\_features[:,1], c=target,marker='o',cmap='viridis',s=15)

plt.show()

plot\_confusion\_matrix(Classificador, features, target,include\_values=True,display\_labels=data.target\_names)

plt.show()

plot\_confusion\_matrix(ClassificadorPCA, pca\_features, target,include\_values=True,display\_labels=data.target\_names)

plt.show()

Resultados

