Atividade laboratório - Inteligência Artificial Prof. Destro

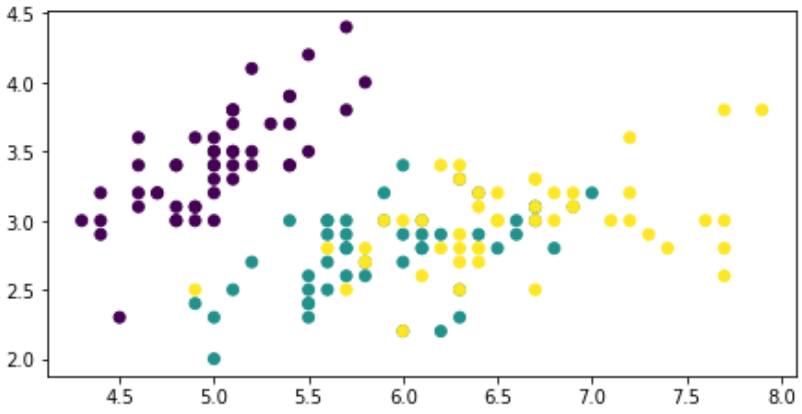
Antonio Muniz RA: 22.119.001-0 Henrique vital Carvalho RA:22.119.078-8

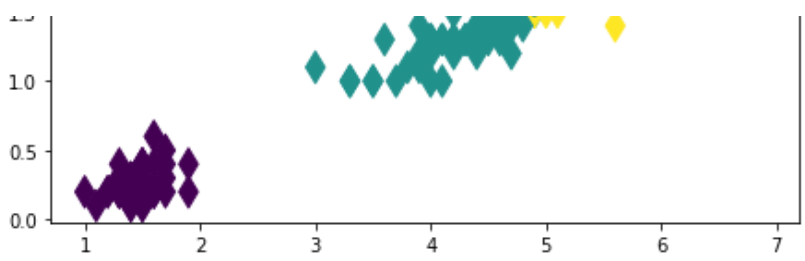
O código abaixo utilizamos o modelo de MLP sem o PCA e conseguimos com duas camadas de 20 e 10 neurônios, respectivamente, e 1000 iterações atingimos um numero de erro 2. Conforme exibe o gráfico.

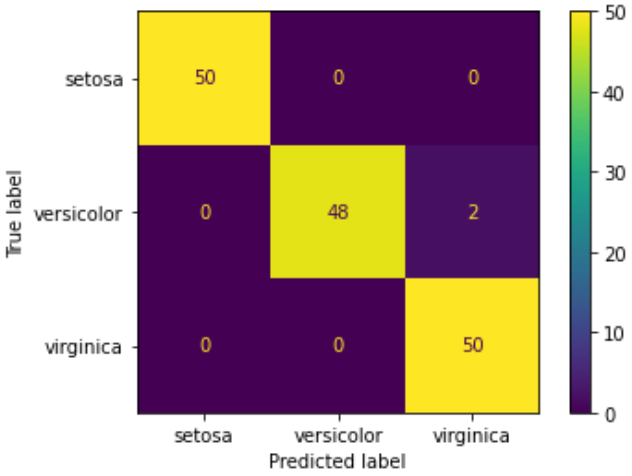
In [75]:

```
import pandas as pd
import numpy as np
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 = (20, 10), alpha=1, max_iter=1000)
Classificador.fit(features,target)
predicao = Classificador.predict(features)

plt.subplot(2,2,3)
plt.scatter(features[:,2], features[:,3], c=predicao,marker='d',cmap='viridis',s=150)
plt.scatter(features[:,2], features[:,3], c=target,marker='o',cmap='viridis',s=15)


plot_confusion_matrix(Classificador, features, target,include_values=True,display_labels=
data.target_names)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Fu
nction plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecate
d in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay
.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```





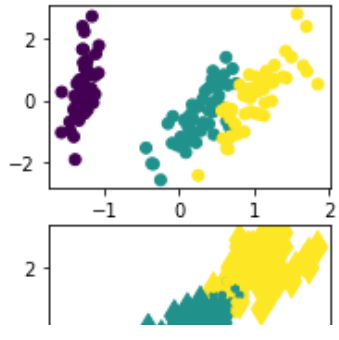


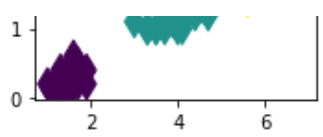
No código abaixo utilizamos o modelo de MLP com o PCA e conseguimos com duas camadas de 15 e 10 neurônios, respectivamente, e 1000 iterações atingimos um numero de erro 2. Conforme exibe o gráfico.

In [ ]:

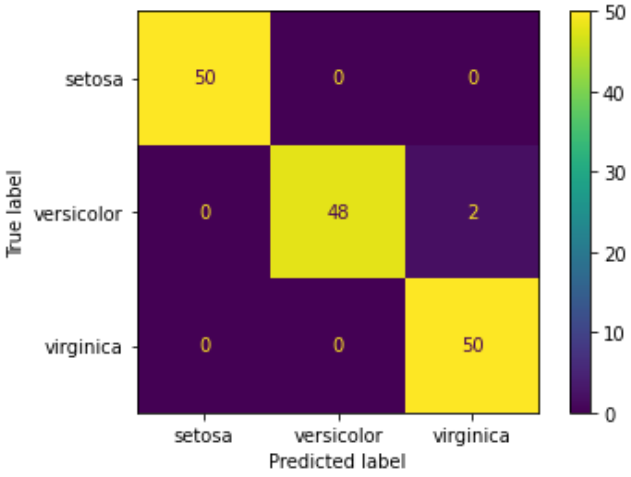
```
pca = PCA(n_components=2, whiten=True, svd_solver='randomized')
pca = pca.fit(features)
pca_features = pca.transform(features)
print('Mantida %5.2f%% da informação do conjunto inicial de dados'%(sum(pca.explained_var
iance_ratio_)*100))

plt.subplot(2,2,2)
plt.scatter(pca_features[:,0], pca_features[:,1], c=target,marker='o',cmap='viridis')



Classificador = MLPClassifier(hidden_layer_sizes = (15, 10), alpha=1, max_iter=1000)
Classificador.fit(features,target)


predicao = Classificador.predict(features)

plt.subplot(2,2,4)
plt.scatter(features[:,2], features[:,3], c=predicao,marker='d',cmap='viridis',s=150)
plt.scatter(features[:,2], features[:,3], 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()
```

```
Mantida 97.77% da informação do conjunto inicial de dados
```






```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Fu
nction plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecate
d in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay
.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



Vistos os resultados acima, é possivel observar que o modelo utilizando PCA teve um aprendizado superior ao sem PCA, pois com menos camadas e a mesma quantidade de iterações, alcançamos resultados semelhantes, deste modo o modelo utilizando PCA é melhor para este tipo de problema.

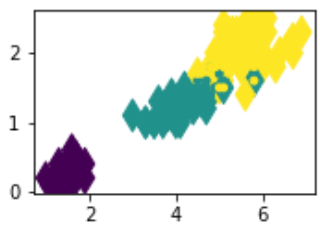
In [76]:

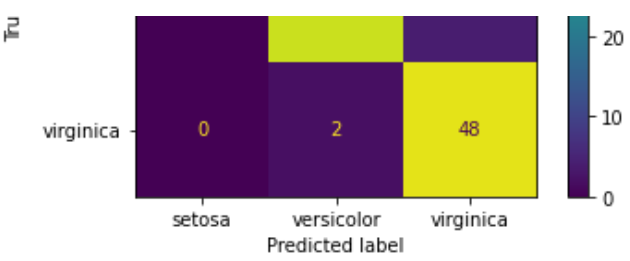
```
Classificador = MLPClassifier(hidden_layer_sizes = (15, 10), alpha=1, max_iter=1000)
Classificador.fit(features,target)
predicao = Classificador.predict(features)

plt.subplot(2,2,3)
plt.scatter(features[:,2], features[:,3], c=predicao,marker='d',cmap='viridis',s=150)
plt.scatter(features[:,2], features[:,3], c=target,marker='o',cmap='viridis',s=15)


plot_confusion_matrix(Classificador, features, target,include_values=True,display_labels=
data.target_names)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Fu
nction plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecate
d in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay
.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[76]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fe7bb87fa50>
```





Conforme o print acima exibe, quando utilizamos a mesma configuração de rede neural (número de neurônios, camadas, iterações) que o modelo com PCA, temos um aumento consideravel na quantidade de erros.

Link do repositorio: https://github.com/henriquevital00/RNA-classifier