Machine Learning -Classificação:

Construindo o modelo com k-Nearest Neighbors

Prof. Ismar Vicente

A ideia aqui é criar um modelo de Machine Learnig que pode aprender, por meio de medidas dispostas no dataset iris (medidas de pétalas e de sépalas de flores iris) e então fazer a predição da espécie da iris, a partir de medidas que não foram informadas.

O conjunto de dados Iris foi usado em R.A. O artigo clássico de 1936 de Fisher, "O Uso de Múltiplas Medições em Problemas Taxonômicos", e também pode ser encontrado no Repositório de Aprendizado de Máquina da UCI.

Inclui três espécies de íris com 50 amostras cada, bem como algumas propriedades sobre cada flor.

```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Carregando o Dataset de uma url

```
iris_dataset = pd.read_csv('https://raw.githubusercontent.com/Ismar-Vicente/Datascience/master/iris2.txt')
```

Observando o Dataset

```
print('Shape of data: \n{}'.format(iris_dataset.shape))
```

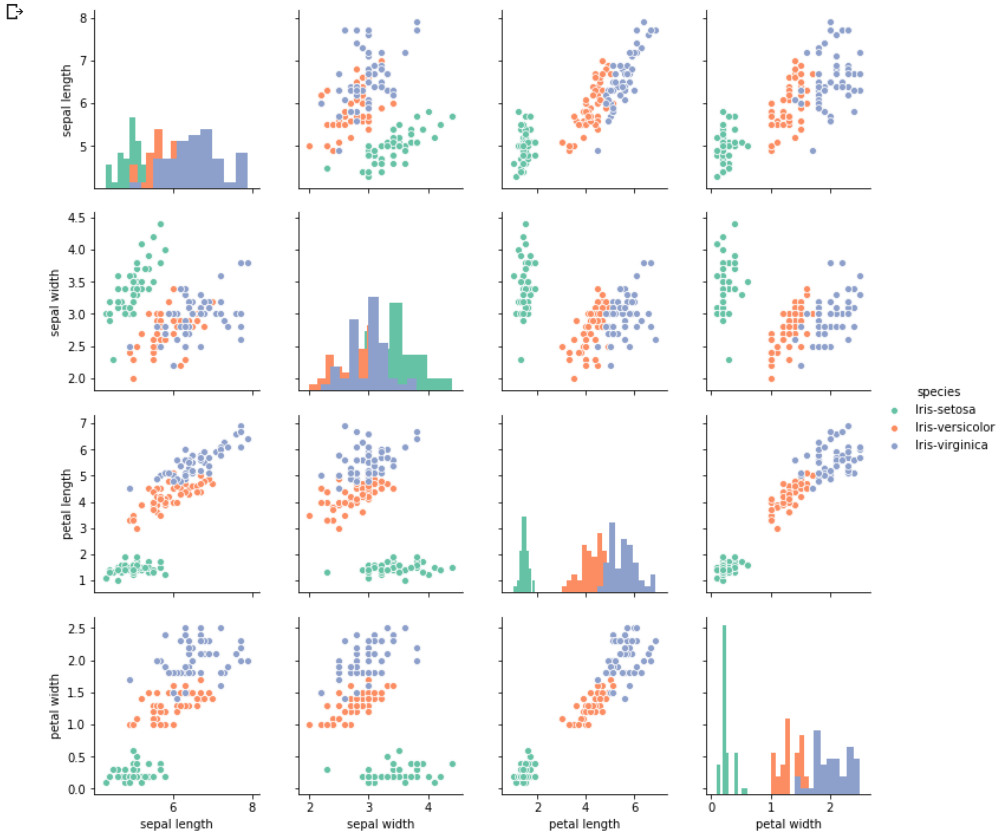
```
Shape of data:
(150, 5)
```

```
print('Primeiras 5 linhas: \n{}'.format(iris_dataset.head()))
```

```
Primeiras 5 linhas:
   sepal length  sepal width  petal length  petal width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

```
# Tire o comentário para testar outros tipos de gráficos

#g = sns.pairplot(iris_dataset)
#g = sns.pairplot(iris_dataset,  hue="species", palette="Set2")
g = sns.pairplot(iris_dataset, diag_kind="hist", hue="species", palette="Set2")
#g = sns.pairplot(iris_dataset, kind="reg")
```



Separando o dataset em train e test

Agora estaremos dividindo o dataset em dados de treinamento (train) e dados de teste (test). O conjunto de treinamento contém uma saída conhecida e o modelo aprende com esses dados para ser generalizado para outros dados posteriormente. O conjunto de dados de teste serve para testar a previsão do nosso modelo.

```
from sklearn.model_selection import train_test_split
```

```
iris_dataset.head(1)
```

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	Iris-setosa

```
x = iris_dataset[['sepal length', 'sepal width',  'petal length', 'petal width']]
y = iris_dataset['species']
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=0)
```

```
print('X_train shape:{}'.format(X_train.shape))
print('y_train shape:{}'.format(y_train.shape))
```

```
X_train shape:(112, 4)
y_train shape:(112,)
```

```
print('X_test shape:{}'.format(X_test.shape))
print('y_test shape:{}'.format(y_test.shape))
```

```
X_test shape:(38, 4)
y_test shape:(38,)
```

Construindo o modelo k-Nearest Neighbors

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```

```
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

Fazendo Predições

```
X_new = np.array([[5, 2.9, 1, 0.2]])
print('X_new.shape:{}'.format(X_new.shape))
```

```
X_new.shape:(1, 4)
```

```
prediction = knn.predict(X_new)
print('Prediction:{}'.format(prediction))
```

```
Prediction:['Iris-setosa']
```

Avaliando o modelo

```
y_pred = knn.predict(X_test)
print('Test set predictions:\n{}'.format(y_pred))
```

```
Test set predictions:
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica']
```

```
print('Test set score: {:.2f}'.format(knn.score(X_test, y_test)))
```

```
Test set score: 0.97
```

Isto significa que o modelo está certo em 97% de suas predições.