Estudo Random Forest

Estudo de algoritmos de Decision Tree e Random Forest

Decision Tree

In [1]:
```
#Importa bibliotecas
import numpy as np
import pandas as pd
import sklearn as sk
from matplotlib import pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [2]:
```
#Carrega o dataset
irisData = pd.read_csv("iris_data.csv")
```

EDA

In [3]:
```
#Verifica dados básicos do dataset
irisData.describe()
```

Out[3]:

	SepalLength	SepalWidth	PetalLength	PetalWidth
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [4]:
```
#Verifica os dados das primeiras linhas
print(irisData.head())
```

```
   SepalLength  SepalWidth  PetalLength  PetalWidth        Class
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
```

In [5]:
```
#Verifica correlação dos dados
print(irisData.corr())
```

	SepalLength	SepalWidth	PetalLength	PetalWidth
SepalLength	1.000000	-0.109369	0.871754	0.817954
SepalWidth	-0.109369	1.000000	-0.420516	-0.356544
PetalLength	0.871754	-0.420516	1.000000	0.962757
PetalWidth	0.817954	-0.356544	0.962757	1.000000

Preparação dos dados para criar modelo

In [6]:
```
#Separa os atributos do target
features = irisData[["SepalLength","SepalWidth","PetalLength","PetalWidth"]]
targetVariables = irisData.Class
```

In [7]:
```
#Separa os dados em treino e teste
featureTrain, featureTest, targetTrain, targetTest = train_test_split(features,
                                                                       targetVariable
                                                                       test_size = .2
```

In [8]:
```
#Cria o modelo de Decision Tree com o critério Gini de classificação
clf = DecisionTreeClassifier(criterion='gini')
```

Quando usado o critério Gini, o modelo irá medir a probabilidade de dois itens pertencerem à mesma classe, somando os quadrados das proporções das classes.

O Índice de Gini diz: se selecionarmos dois itens de uma população aleatoriamente, então eles devem ser da mesma classe e a probabilidade para isto é 1 se a população é pura.

In [9]:
```
#Treina o modelo
modelo = clf.fit(featureTrain, targetTrain)
```

In [10]:
```
#Executa predições com os dados de teste
previsoes = modelo.predict(featureTest)
```

In [11]:
```
#Verifica a acurácia do modelo
print (accuracy_score(targetTest, previsoes))
```

```
1.0
```

In [12]:
```
#Cria o modelo de Decision Tree com o critério Entropy de classificação
clf2 = DecisionTreeClassifier(criterion='entropy')
```

O critério "Entropy" é aplicado para verificar o ganho de redução da incerteza dos dados entre a sua escolha. Entropia = medida de incerteza; Ganho = redução da Entropia

Quando usado, o modelo irá verificar qual atributo tem o ganho mais alto de informação em cada nó, escolhendo sempre o nó com o ganho mais alto.

In [13]:
```
#Treina o modelo
modelo2 = clf2.fit(featureTrain, targetTrain)
```

In [14]:
```
#Executa predições com os dados de teste
previsoes2 = modelo2.predict(featureTest)
```

In [15]:
```
#Verifica a acurácia do modelo
print (accuracy_score(targetTest, previsoes2))
```

```
1.0
```

Testando com outro dataset

In [16]:
```
#Carrega o dataset de digitos
from sklearn.preprocessing import scale
from sklearn.datasets import load_digits
digitos = load_digits()
data = scale(digitos.data)
```

In [17]:
```
#Obtem a "resposta" das observações
n_digits = len(np.unique(digitos.target))
labels = digitos.target
```

In [18]:
```
#Cria o modelo com o critério Gini de classificação
from sklearn.ensemble import RandomForestClassifier
clfGini = RandomForestClassifier(criterion='gini')
```

In [19]:
```
#Treina o modelo
clfGini = clfGini.fit(data, labels)
```

In [20]:
```
#Verifica o score do modelo
print(clfGini.score(data,labels))
```

```
1.0
```

In [21]:
```
#Altera o parâmetro do modelo de número de árvores
#Default = 100, alterado para 10
clfGini2 = RandomForestClassifier(n_estimators  = 10, criterion='gini')
```

In [22]:
```
#Treina o modelo novamente
clfGini2 = clfGini2.fit(data, labels)
```

In [23]:
```
#Verifica o score do modelo com novos parâmetros
print(clfGini2.score(data,labels))
```

```
1.0
```

In [24]:
```
#Cria o modelo com o critério Entropy de classificação
clfEntr = RandomForestClassifier(criterion='entropy')
```

In [25]:
```
#Treina o modelo
clfEntr = clfEntr.fit(data, labels)
```

In [26]:
```
#Verifica o score do modelo
print(clfEntr.score(data,labels))
```

```
1.0
```

In [27]:
```
#Altera o parâmetro do modelo de número de árvores
#Default = 100, alterado para 10
clfEntr2 = RandomForestClassifier(n_estimators  = 10, criterion='entropy')
```

In [28]:
```
#Treina o modelo novamente
clfEntr2 = clfEntr2.fit(data, labels)
```

In [29]:
```
#Verifica o score do modelo com novos parâmetros
print(clfEntr2.score(data,labels))
```

```
0.9988870339454646
```

Compara o resultado dos modelos

In [30]:
```
print("Modelo com critério Gini e número de árvores default (1000): ", clfGini.score
print("Modelo com critério Gini e número de árvores em 10: ", clfGini2.score(data,la
print("Modelo com critério Entropy e número de árvores default (1000): ", clfEntr.sc
print("Modelo com critério Entropy e número de árvores em 10: ", clfEntr2.score(data
```

```
Modelo com critério Gini e número de árvores default (1000):  1.0
Modelo com critério Gini e número de árvores em 10:  1.0
Modelo com critério Entropy e número de árvores default (1000):  1.0
Modelo com critério Entropy e número de árvores em 10:  0.9988870339454646
```

Comparando os resultados, podemos perceber a diferença de resultado entre os critérios de avaliação (Gini e Entropy), e a diferença quando alterado os parâmetros de número de árvores dentro de cada modelo.