Inteligência Artificial

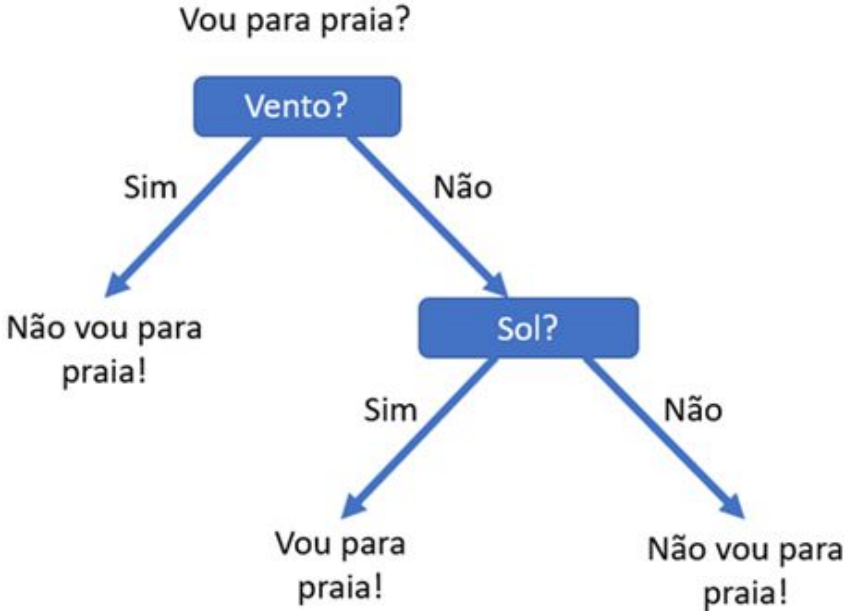
Prof. MSc. Bruno Santos
email: bruno.santos@saojudas.br

OBJETIVOS da AULA

Assimilar os fundamentos da tecnologia de Inteligência Artificial com Árvores de decisão (teoria) por meio da execução de programas e análises.

Árvore de Decisão

Árvores de decisão são métodos de aprendizado de máquinas supervisionado, muito utilizados em tarefas de classificação e regressão.

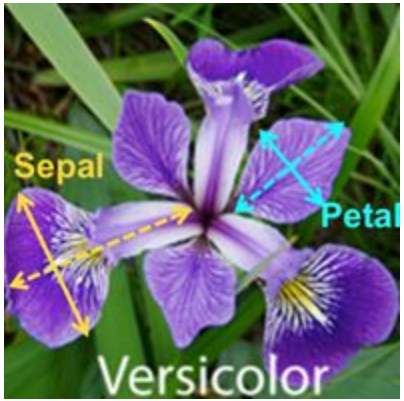


Árvore de Decisão

Para testar o nosso algoritmo vamos utilizar a base de dados Iris. Iris é uma flor que pode ser dividida em 3 espécies: versicolor, virginica e setosa.

Árvore de Decisão

Apesar de parecidas, cada espécie apresenta algumas particularidades relacionadas ao tamanho das pétalas e sépalas. Ou seja, podemos CLASSIFICAR a espécie de uma flor íris observando apenas as dimensões da pétala e sépala.



Árvore de Decisão

Para isso, biólogos analisaram centenas de milhares de exemplares destas espécies e, após catalogar tudo, conseguiram descobrir um “padrão”.

Não é uma tarefa objetiva e com limiares bem definidos, existem padrões ocultos dentro do padrão descoberto.

Árvore de Decisão

Esse conhecimento pode facilmente ser repassado entre seres humanos, mas a nossa intenção é repassar este conhecimento para uma máquina.

Em outras palavras, ensinar uma máquina a CLASSIFICAR a espécie de uma flor íris, sem falar explicitamente quais são os limiares que diferenciam as espécies.

Biblioteca scikit-learn

A scikit-learn é uma biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python.

Ela inclui vários algoritmos de classificação, regressão e agrupamento e é projetada para interagir com as bibliotecas Python numéricas e científicas NumPy e SciPy.

Esta biblioteca possui a base de dados íris.

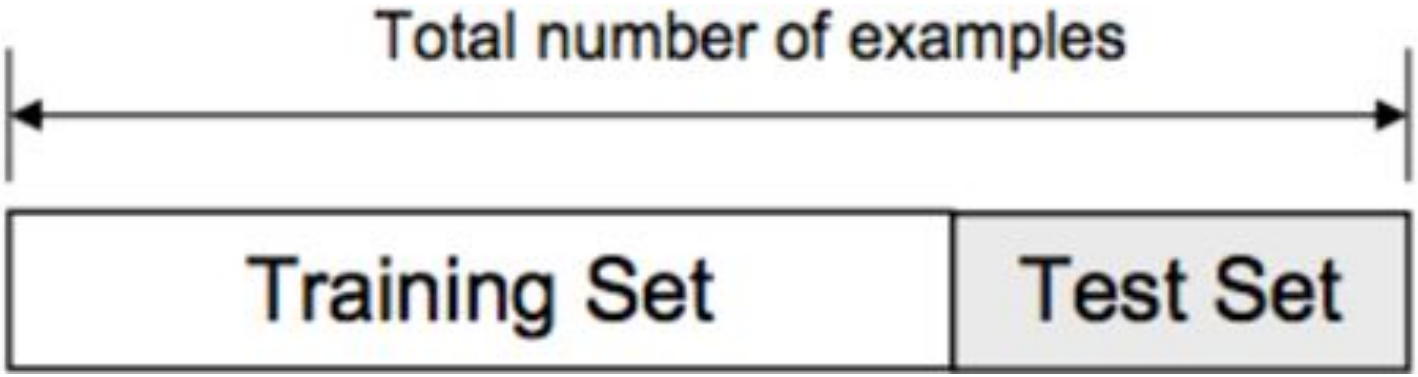
Biblioteca scikit-learn

```
1  from sklearn.datasets import load_iris
2  from sklearn import tree

4  iris = load_iris()

6  print(iris['DESCR'])
```

Biblioteca scikit-learn - Procedimento



Biblioteca scikit-learn - Procedimento

trecho t(X, y, test_size=0.25):

- 0.25 (é o que está sendo usado, que significa que 75% dos dados são usados para treinamento e 25% para avaliação);
- 0.10 (90% dos dados são usados para treinamento e 10% para avaliação);
- 0.50 (50% dos dados são usados para treinamento e 50% para avaliação);
- 0.75 (25% dos dados são usados para treinamento e 75% para avaliação); e
- 0.90 (10% dos dados são usados para treinamento e 90% para avaliação).

Biblioteca scikit-learn - Procedimento

```
1  from sklearn.datasets import load_iris
2  from sklearn import tree

4  from sklearn.model_selection import train_test_split

6  iris = load_iris()

8  X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.25)
```

Biblioteca scikit-learn - Predição

```
1  from sklearn.datasets import load_iris
2  from sklearn import tree

4  from sklearn.model_selection import train_test_split

6  iris = load_iris()

8  X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.25)

10  clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3)
11  clf = clf.fit(X_train, y_train)

13  predictions = clf.predict(X_test)
```

Biblioteca scikit-learn - Predição

clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3):

- 3 (é o que está sendo usado, ou seja, 3 níveis na árvore),
- 1 (1 nível na árvore),
- 2 (2 níveis na árvore),
- 4 (4 níveis na árvore),
- ilimitado; retire este parâmetro.

Biblioteca scikit-learn - Relatório

Medidas usadas em avaliação:

Precisão ou precision (a precisão é intuitivamente a habilidade de não classificar como positiva uma amostra que é negativa) = VP/(VP+FP);

Acurácia ou accuracy = (VP + VN) / (VP+VN+FP+FN);

Cobertura ou recall (acerto com os verdadeiros positivos ou acurácia positiva; intuitivamente a habilidade do classificador de encontrar todas as amostras positivas) = VP/(VP+FN);

Medida-F ou F1 score (média harmônica entre precisão e cobertura; é apropriada para situações de desbalanceamento entre classes)

Biblioteca scikit-learn - Relatório

```
1  from sklearn.datasets import load_iris
2  from sklearn import tree
3  from sklearn.model_selection import train_test_split

5  import pandas as pd

7  iris = load_iris()

8  X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.25)

12 clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3)
13 clf = clf.fit(X_train, y_train)
14 predictions = clf.predict(X_test)

16 print("\nMatriz de confusão detalhada:\n",
17       pd.crosstab(y_test, predictions, rownames=['Real'], colnames=['Predito'],
18       margins=True, margins_name='Todos'))
```

Biblioteca scikit-learn - Relatório

```
Matriz de confusão detalhada:
 Predito   0   1   2  Todos
Real
0         15   0   0     15
1          0   9   0      9
2          0   1  13     14
Todos     15  10  13     38
```

Biblioteca scikit-learn - Relatório

```
1  from sklearn.datasets import load_iris
2  from sklearn import tree
3  from sklearn.model_selection import train_test_split
4  import pandas as pd
```

```
6   import sklearn.metrics as metrics
```

```
7   iris = load_iris()
```

```
8  X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.25)
```

```
12   clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3)
13   clf = clf.fit(X_train, y_train)
14   predictions = clf.predict(X_test)
```

```
16  print("Relatório sobre a qualidade:\n")
17  print(metrics.classification_report(y_test, predictions, target_names=['Setosa', 'Versicolor', 'Virgínica']))
```

Biblioteca scikit-learn - Relatório

Relatório sobre a qualidade:

	precision	recall	f1-score	support
Setosa	1.00	1.00	1.00	10
Versicolor	0.92	1.00	0.96	12
Virgínica	1.00	0.94	0.97	16
accuracy			0.97	38
macro avg	0.97	0.98	0.98	38
weighted avg	0.98	0.97	0.97	38

Biblioteca scikit-learn - Exportação

```
1  from sklearn.datasets import load_iris
2  from sklearn import tree
3  from sklearn.model_selection import train_test_split

5  import graphviz

7  iris = load_iris()

8  X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.25)

12 clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3)
13 clf = clf.fit(X_train, y_train)
14 predictions = clf.predict(X_test)

15 dot_data = tree.export_graphviz(clf, out_file=None)
16 graph = graphviz.Source(dot_data)
17 graph.render("iris")
```

...

Biblioteca scikit-learn - Exportação

```
19  dot_data = tree.export_graphviz(clf, out_file=None,
20                                  feature_names=iris.feature_names,
21                                  class_names=iris.target_names,
22                                  filled=True, rounded=True,
23                                  special_characters=True)
24  graph = graphviz.Source(dot_data, format="png")
25  graph
```

Biblioteca scikit-learn - Exportação