PYTHON para Inteligência Artificial – Lab05

1.Conceitos Importantes

1.1 Algoritmo de Classificação

Em inteligência artificial (IA), um algoritmo de classificação é um método utilizado para categorizar dados em diferentes classes ou categorias. Esse tipo de algoritmo é comumente aplicado em problemas de aprendizado supervisionado, onde o algoritmo é treinado com um conjunto de dados rotulados, ou seja, dados onde as classes ou categorias já estão previamente definidas.

O objetivo principal de um algoritmo de classificação é aprender a relação entre as características (ou atributos) dos dados e suas respectivas classes, de modo que, quando apresentado com novos dados não rotulados, seja capaz de predizer a classe correta para esses dados.

Um algoritmo de classificação tem como objetivo classificar uma característica, como definir a espécie de uma flor, por exemplo, como veremos a seguir.

1.2 Variável Alvo (ou Target)

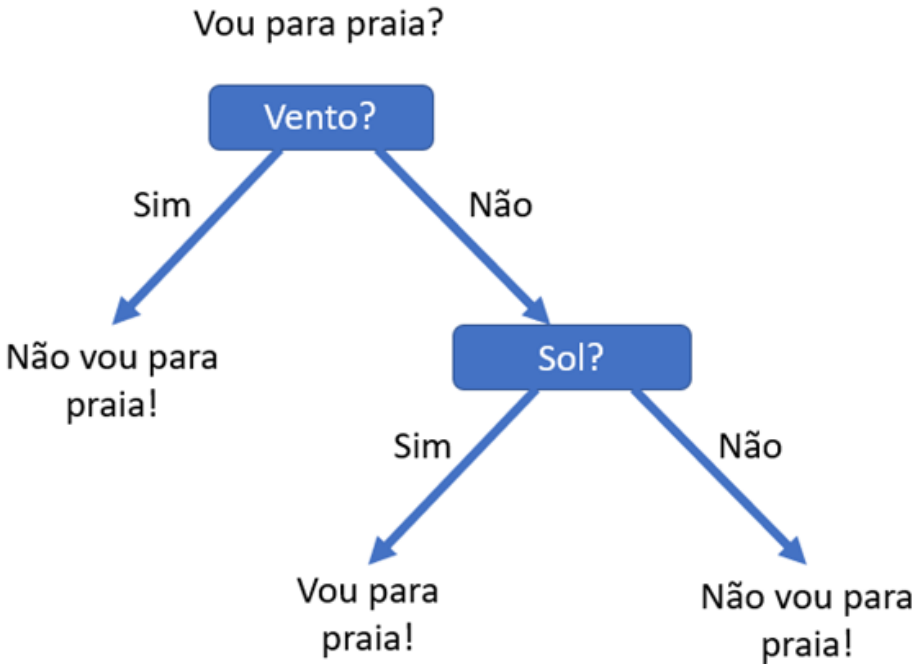
Uma variável alvo em IA, também conhecida como variável de resposta ou variável dependente (em inglês, target variable), é a variável que um modelo de aprendizado de máquina ou inteligência artificial tenta prever ou explicar. Em outras palavras, é a variável que estamos interessados em prever com base em outras variáveis, chamadas de variáveis independentes ou características (em inglês, features).

Por exemplo, se estivermos construindo um modelo para prever o preço de uma casa com base em várias características, como tamanho, número de quartos, localização, etc., então o preço da casa seria a variável alvo. O modelo tentaria aprender uma relação entre as características fornecidas e o preço da casa, de modo que, dadas as características de uma nova casa, ele possa fazer uma previsão precisa do preço.

Identificar corretamente a variável alvo é crucial na construção de modelos de IA eficazes, pois ela determina o que o modelo está tentando aprender e prever.

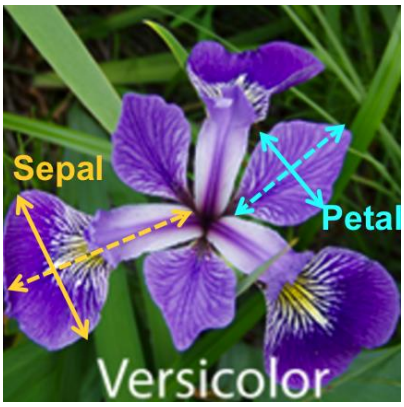
2. Exemplo de Algoritmo de Classificação: Árvores de Decisão

Árvores de decisão são métodos de aprendizado de máquina supervisionados, muito utilizados em tarefas de classificação. No exemplo abaixo temos uma árvore de decisão para classificar uma situação em “Não vou para praia” e “Vou para praia”, considerando as variáveis “Vento” e “Sol”.



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.

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.



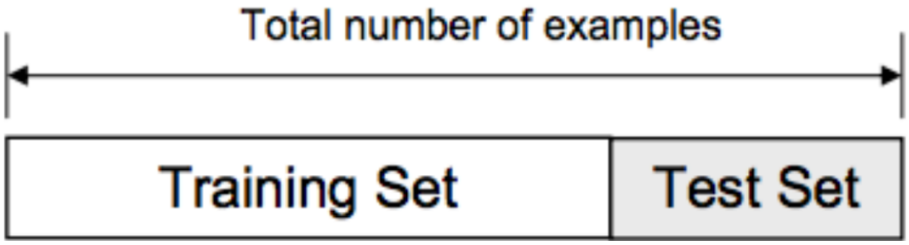
Para isso, biólogos analisaram centenas de milhares de exemplares destas espécies e, após catalogar tudo, conseguiram descobrir um “padrão”. Não é uma tarefa objetiva e com limiares bem definidos, existem padrões ocultos dentro do padrão descoberto. Esse conhecimento pode facilmente ser repassado entre 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.

O primeiro passo é importar as bibliotecas necessárias. 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.

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

iris = load_iris()
print(iris['DESCR'])
```

Após carregar e observar a base de dados, vamos dividir nosso conjunto de dados em duas partes: treino e teste.



Isso é feito facilmente com a função train_test_split do model_selection. O parâmetro 'test_size' define a proporção de dados usados para teste.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris
.target, test_size=0.25)
```

Agora podemos ajustar os dados de treinamento ao nosso modelo de Árvore de Decisão (com profundidade máxima de 3). Após treinado, podemos usá-lo para fazer previsões, usando o método predict() do nosso modelo:

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

Agora podemos criar um relatório sobre a qualidade da classificação e uma "matriz de confusão" para avaliar nosso modelo:

```
import pandas as pd

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

Apresentação da avaliação:

```
import sklearn.metrics as metrics
print("Relatório sobre a qualidade:\n")
print(metrics.classification_report(y_test, predictions, target_nam
es=['Setosa', 'Versicolor', 'Virgínica']))
```