Árvores de Decisão

Classificação

Slides extraídos do livro "*Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*" de Aurélien Géron

Sumário

1. Árvores de Decisão
2. Treinamento e Visualização de uma Árvore de Decisão

Sumário

1 Árvores de Decisão

2 Treinamento e Visualização de uma Árvore de Decisão

- **Árvores de Decisão** são algoritmos versáteis de Aprendizado de Máquina que podem realizar tarefas de classificação e regressão, e até mesmo tarefas multi-saída.

- São algoritmos poderosos, capazes de ajustar datasets complexos.

- As Árvores de Decisão também são os componentes fundamentais das *Random Forests* (Florestas Aleatórias), que estão entre os algoritmos de AM mais poderosos disponíveis atualmente.

Sumário

1 Árvores de Decisão

2 Treinamento e Visualização de uma Árvore de Decisão

Treinando uma Árvore de Decisão

Vamos construir uma árvore e ver como ela faz previsões.
O código a seguir treina uma `DecisionTreeClassifier` no dataset iris.

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

iris = load_iris(as_frame=True)
X_iris = iris.data[["petal length (cm)", "petal width (cm)"]].values
y_iris = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X_iris, y_iris)
```

Treinando uma Árvore de Decisão

Podemos visualizar a Árvore de Decisão treinada usando o método `export_graphviz()` para gerar um arquivo de definição de gráfico chamado `iris_tree.dot`.

```
from sklearn.tree import export_graphviz

export_graphviz(
    tree_clf,
    out_file="iris_tree.dot",
    feature_names=["petal length (cm)", "petal width (cm)"],
    class_names=iris.target_names,
    rounded=True,
    filled=True
)
```

Visualização da Árvore de Decisão

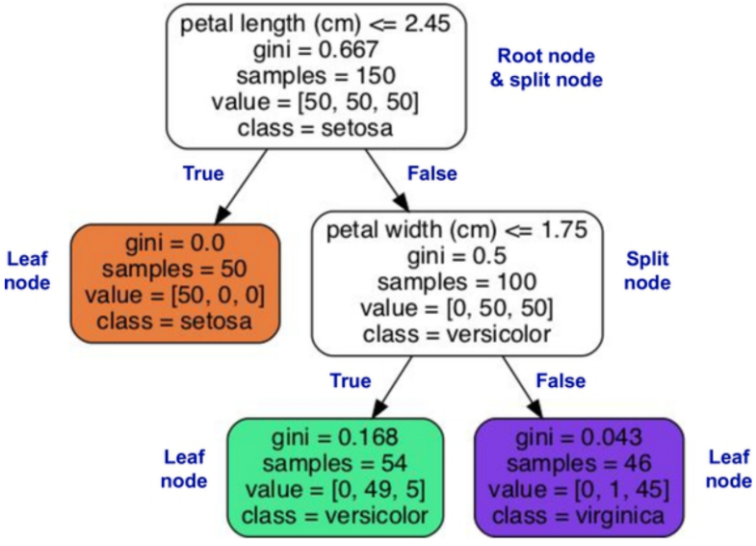

Figure: Árvore de Decisão para o dataset Iris.

Fazendo Previsões

Vamos ver como a árvore da figura anterior faz previsões.

- Você começa no **nó raiz** (profundidade 0, no topo). Este nó pergunta se o comprimento da pétala da flor é menor que 2.45 cm.
- Se for, você desce para o nó filho esquerdo da raiz (profundidade 1, esquerda). Neste caso, é um **nó folha** (não tem filhos), então ele não faz mais perguntas. A árvore prevê que sua flor é uma *Iris setosa*.
- Se o comprimento da pétala for maior que 2.45 cm, você desce para o nó filho direito (profundidade 1, direita). Este é um **nó de divisão**, então ele faz outra pergunta: a largura da pétala é menor que 1.75 cm?
- Se sim, sua flor é provavelmente uma *Iris versicolor* (profundidade 2, esquerda). Se não, é uma *Iris virginica* (profundidade 2, direita).

Atributos dos Nós e Impureza de Gini

- O atributo `samples` de um nó conta quantas instâncias de treinamento ele se aplica.
- O atributo `value` de um nó informa quantas instâncias de treinamento de cada classe este nó se aplica.
- O atributo `gini` de um nó mede sua **impureza de Gini**.
  Um nó é "puro" (`gini=0`) se todas as instâncias de treinamento a que se aplica pertencem à mesma classe.

Equação 6-1. Impureza de Gini

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

- $p_{i,k}$ é a proporção de instâncias da classe k entre as instâncias de treinamento no i-ésimo nó.
- **Exemplo**: O nó esquerdo da profundidade 2 tem uma impureza de Gini igual a $1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168$.

Limites de Decisão da Árvore

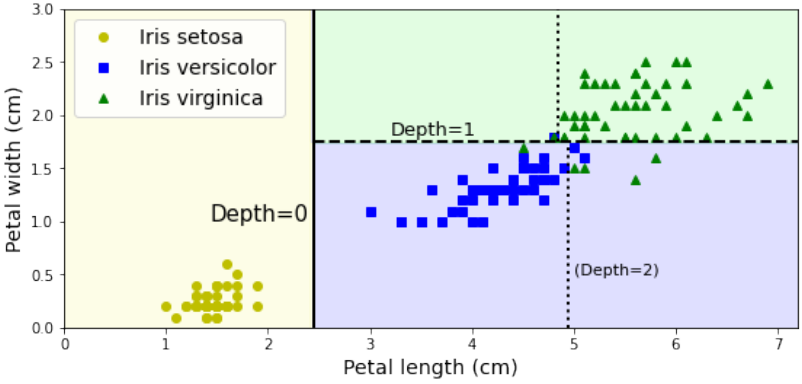

Figure: Limites de decisão da Árvore de Decisão.

- A linha vertical espessa representa o limite de decisão do nó raiz (prof. 0): comprimento pétala = 2.45 cm.
- A área à esquerda é pura (apenas *Iris setosa*), então não pode ser dividida mais.
- Area à direita é impura, então o nó da direita na prof. 1 a divide em largura da pétala = 1.75 cm (tracejada).
- Como `max_depth` foi definido como 2, a árvore para aqui.

Interpretação do Modelo: Caixa Branca vs. Caixa Preta

Modelos Caixa Branca (White Box)

As Árvores de Decisão são intuitivas, e suas decisões são fáceis de interpretar.
Elas fornecem regras de classificação simples que podem até ser aplicadas manualmente.

Modelos Caixa Preta (Black Box)

Em contraste, as Random Forests ou redes neurais são geralmente consideradas modelos de caixa preta. Eles fazem ótimas previsões, mas geralmente é difícil explicar em termos simples por que as previsões foram feitas.

O campo de **ML Interpretável** visa criar sistemas de AM que possam explicar suas decisões de uma maneira que os humanos possam entender.

Estimando Probabilidades de Classe

Uma Árvore de Decisão pode estimar a probabilidade de uma instância pertencer a uma classe particular k.

- Ela percorre a árvore para encontrar o nó folha para esta instância.
- Em seguida, retorna a proporção de instâncias de treinamento da classe k neste nó.

Exemplo

Uma flor com pétalas de 5 cm de comprimento e 1.5 cm de largura.
O nó folha correspondente é o nó esquerdo da profundidade 2.
As probabilidades são: 0% para Setosa (0/54), 90.7% para Versicolor (49/54) e 9.3% para Virginica (5/54).

```
>>> tree_clf.predict_proba([[5, 1.5]]).round(3)
array([[0.   , 0.907, 0.093]])
>>> tree_clf.predict([[5, 1.5]])
array([1])
```

O Algoritmo de Treinamento CART

O Scikit-Learn usa o algoritmo **CART (Classification and Regression Tree)** para treinar Árvores de Decisão.

- O algoritmo funciona dividindo primeiro o conjunto de treinamento em dois subconjuntos usando uma única feature k e um limiar t_k (ex: "comprimento da pétala ≤ 2.45 cm").
- Como ele escolhe k e t_k?
  Ele procura o par (k, t_k) que produz os subconjuntos mais puros, ponderados pelo seu tamanho.

Equação 6-2. Função de custo CART para classificação

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

- O algoritmo para a recursão quando atinge a profundidade máxima (`max_depth`) ou se não consegue encontrar uma divisão que reduza a impureza ($G_i \leq J(k, t_k)$ para todo k e t_k).
- CART é um **algoritmo guloso (greedy)**: ele procura uma divisão ótima no nível superior e repete o processo em cada nível subsequente. Ou seja, não verifica se a divisão produzirá impureza baixa nos níveis abaixo. Encontrar a árvore que minimiza um métrica global (ex.: Gini total) é um problema NP-Completo.

Previsões

Fazer previsões requer atravessar a Árvore de Decisão da raiz até uma folha. Como as Árvores de Decisão são geralmente aproximadamente balanceadas, isso requer percorrer cerca de $\mathcal{O}(\log_2(m))$ nós. Como cada nó requer apenas a verificação do valor de uma feature, a complexidade geral da previsão é $\mathcal{O}(\log_2(m))$, independente do número de features. As previsões são muito rápidas.

Treinamento

O algoritmo de treinamento compara todas as features (ou menos, se `max_features` for definido) em cada nó. Para cada feature, gasta $\mathcal{O}(m \log_2(m))$ para ordenar as instâncias de acordo com a feature, e $\mathcal{O}(m)$ para determinar o gini da divisão em cada instância.
Isso resulta em uma complexidade de treinamento de $\mathcal{O}(n \times m \log_2(m))$.

Impureza de Gini ou Entropia?

Por padrão, a classe `DecisionTreeClassifier` usa a medida de impureza de Gini, mas você pode selecionar a medida de impureza de **entropia** definindo o hiperparâmetro `criterion="entropy"`.

- Em Machine Learning, a entropia é frequentemente usada como uma medida de impureza: a entropia de um conjunto é zero quando ele contém instâncias de apenas uma classe.

Equação 6-3. Entropia

$$H_i = -\sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log_2(p_{i,k})$$

Qual usar?

Na maioria das vezes, não faz muita diferença: eles levam a árvores semelhantes.
A impureza de Gini é um pouco mais rápida de calcular, então é um bom padrão.
No entanto, quando diferem, a impureza de Gini tende a isolar a classe mais frequente em seu próprio ramo da árvore (desbalanceada), enquanto a entropia tende a produzir árvores um pouco mais balanceadas.

Fim