ANA RAQUEL

POS TECH

IA PARA DEVS

MACHINE LEARNING AVANÇADO

AULA 04

SUMÁRIO

O QUE VEM POR AÍ?

Você já ouviu falar em modelos baseados em árvores? Árvores de Decisão são algoritmos muito poderosos e versáteis de aprendizado de máquina que podem executar tarefas de classificação e regressão sobre conjunto de dados complexos. Nessa aula, você vai aprender os principais tipos de modelos baseados em árvores e quais são suas características.

HANDS ON

Você aprendeu sobre um dos modelos mais simples e poderosos do mundo de Machine Learning, incluindo o modelo tradicional de Árvore de Decisão, capaz de executar classificação, regressão e modelo de ensemble, como o Random Forest. Agora vamos ver na prática como os modelos de árvores podem ajudar a resolver problemas complexos de forma simples no Python!

Para essa aula, temos alguns notebooks para você. Acesse abaixo:

- Notebook 1
- Notebook 2

Além disso, também disponibilizamos as bases de dados, para te ajudar com os estudos e exercícios.

- Base de dados 1
- Base de dados 2

SAIBA MAIS

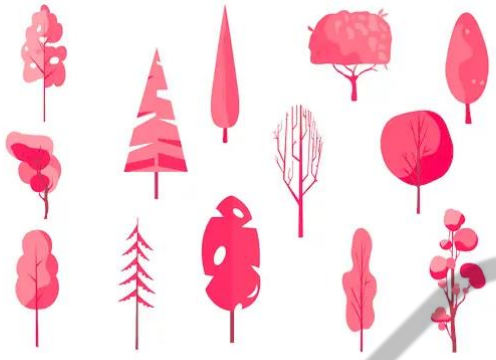

Figura 1 - Modelos de Árvores
Fonte: Elaborado pela autora (2024)

Um modelo de Árvore de Decisão é uma estrutura recursiva que representa uma **lista de regras de decisões** (tais como IF e ELSE quando associamos à programação) em forma de uma árvore. A analogia com árvores é usada porque esse modelo supervisionado de Machine Learning basicamente constrói uma estrutura de lógica baseada em nós de folhas e raízes. Entenderemos mais com um exemplo prático!

Vamos aplicar, de forma bem simples, um modelo de árvore em uma base de dados que contém informações sobre transações de cartão de crédito, para classificarmos quando pode se tratar ou não de uma transação fraudulenta.

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.tree import plot_tree

from sklearn import tree
```

```
#subindo a base de dados

dados = pd.read_csv("card_transdata.csv", sep=",")
```

```
#Separando os dados em treino e teste:

x = dados.drop(columns=['fraud'])

y = dados['fraud'] #O que eu quero prever. (Target)
```

```
#Criando o modelo de árvore de decisão:

dt = DecisionTreeClassifier(random_state=7, criterion='entropy', max_depth = 2)

dt.fit(x_train, y_train)

dt.predict(x_test)
```

```
#Plotando a árvore:

tree.plot_tree(dt)

class_names = ['Fraude', 'Não Fraude']

label_names = ['distance_from_home', 'distance_from_last_transaction',
        'ratio_to_median_purchase_price',        'repeat_retailer',      'used_chip',
         'used_pin_number', 'online_order']

fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (10,10), dpi=300)
```

```
tree.plot_tree(dt,
               feature_names = label_names,
               class_names=class_names,
               filled = True)


fig.savefig('imagename.png')
```

Aplicamos, de forma bem básica, um modelo de árvores para você compreender como funciona a lógica de modelos de árvores. Observe que, após treinarmos o modelo com algumas variáveis características de transações de cartão de crédito, geramos a seguinte árvore, dadas as configurações de hiperparâmetros:

- criterion='entropy'
- max_depth = 2

Em breve explicaremos como funciona a configuração dos hiperparâmetros; mas primeiro, vamos entender a formação da árvore. Observe a figura 2 – “Árvore de Decisão” gerada pelo DecisionTreeClassifier, que foi gerado pelo algoritmo de árvore de decisão aplicado anteriormente.

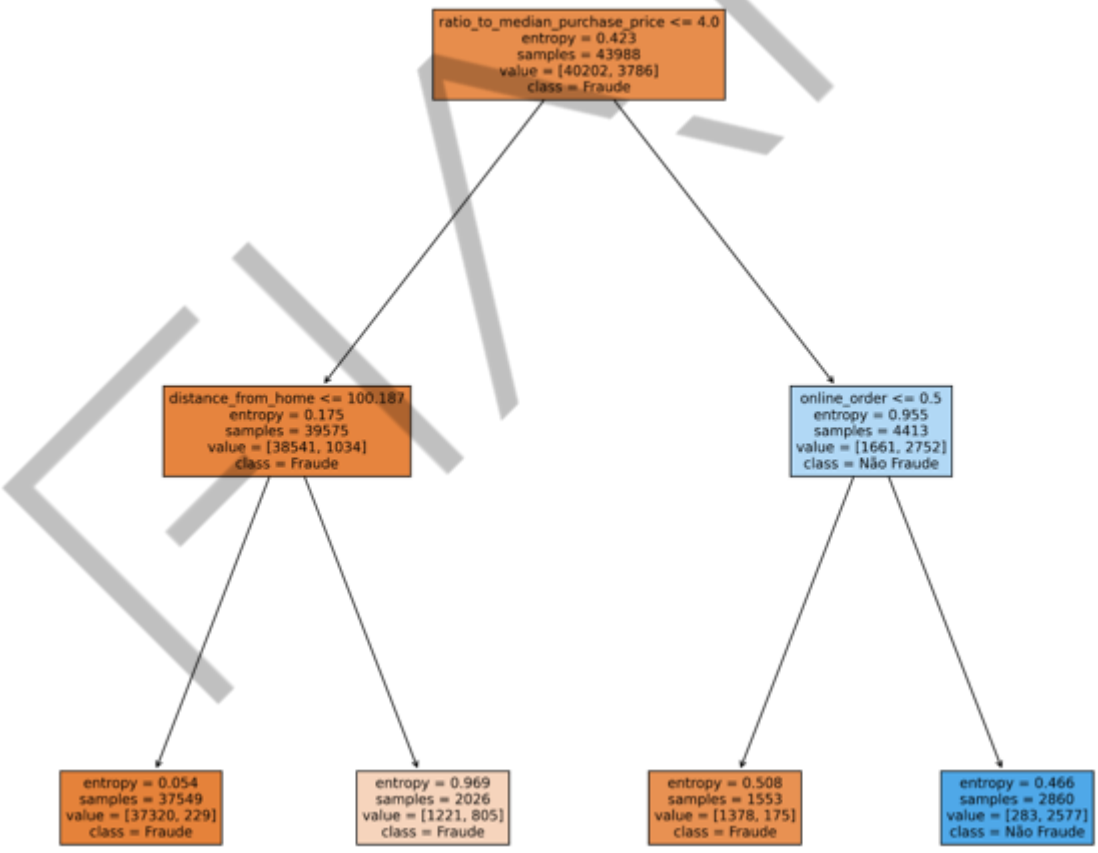

Figura 2 - Árvore de Decisão gerada pelo DecisionTreeClassifier.
Fonte: Elaborado pela autora (2024)

Suponha que você irá prever se uma certa transação pode ser fraudulenta ou não dadas as seguintes “regras” geradas pela árvore.

Você começa no nó da raiz (profundidade 0 da parte superior da árvore): este nó pergunta se a relação ao preço médio de compra é menor ou igual a 0.423. Caso o preço médio da compra atenda a essa regra, vamos para o próximo nó, filho esquerdo da raiz (profundidade 1, esquerda). Dentro desse nó, é realizada mais uma validação, caso o local da compra tenha sido em uma distância menor ou igual a 100.187; então, a regra nos leva para mais um nó da árvore, nos dizendo que essa transação pode ser fraudulenta (class = Fraude).

Uma das muitas qualidades das Árvores de Decisão é que elas exigem pouca preparação dos dados. Em particular, elas não requerem o escalonamento ou a centralização das características. Assim, vamos aprender o que seria cada atributo de uma folha da árvore:

- **Samples**: quantidade de instâncias de treinamento que se aplica ao nó (folha).
- **Value:** quantidade de instâncias de treinamento de cada classe que se aplica ao nó (folha).
- **Entropy:** medida da impureza de um nó (entropy=0) se todas as instâncias de treinamento pertencem à mesma classe a qual se aplica.

Mas o que seria a medida da impureza de um nó da árvore?

Para medir a impureza de um nó da árvore, podemos utilizar os coeficientes gini ou entropy. O coeficiente gini é aplicado por padrão no Sklearn, mas você também pode selecionar a entropia (entropy) ao configurar o parâmetro **“criterion”**.

A entropia aproxima-se de zero quando as moléculas ainda estão paradas e bem ordenadas, ou seja: a entropia é zero quando todas as instâncias são idênticas. Observe que, nos últimos nós da árvore, classificando quando temos uma classe fraude ou não fraude, a medida de entropia é quase zero, ou seja, temos as classes quase totalmente puras. Devo escolher o critério gini ou entropia? Na verdade, na maioria das vezes não temos muita diferença nos resultados. Podemos colocar aqui que o coeficiente gini é um pouco mais rápido para calcular, o que torna este um bom padrão.

Há também outros tipos de hiperparâmetros na árvore de decisão, os chamados hiperparâmetros de regularização. No Scikit- Learn, temos:

- **max_depth:** quantidade de ramificações (profundidade da árvore) a serem criadas. Reduzir max_depth regulariza o modelo, evitando assim o overfitting.
- **min_samples_split:** número mínimo de amostras que um nó deve ter antes de ser dividido.
- **min_samples_leaf:** expressa como uma fração do número total de instâncias ponderadas.
- **max_leaf_nodes:** número máximo de nós da folha.
- **max_features:** número máximo de características que vão ser avaliadas em cada divisão do nó.

Random Forest

O modelo de Random Forest (Florestas Aleatórias) é um método de ensemble de Árvores de Decisão. Métodos de ensemble são algoritmos que se utilizam da combinação de outros algoritmos para construir um modelo mais preciso e acurado.

A Random Forest cria um conjunto de árvores de decisões e utiliza a moda das predições de todas as árvores. O algoritmo, de forma inteligente, inicializa várias árvores de decisões, calcula o ganho de informação (entropia) e, posteriormente, é calculada uma moda com base na árvore que possui a maior frequência de nós, dada a análise de todas as possibilidades possíveis. Veja a aplicação do modelo Random Forest no Python:

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=5, max_depth = 2,  random_state=7)

rf.fit(x_train, y_train)

y_predito_random_forest = rf.predict(x_test)

from sklearn.tree import export_graphviz



fig, axes = plt.subplots(nrows = 1,ncols = 5,figsize = (10,2), dpi=900)
```

```
for index in range(0, 5):
    tree.plot_tree(rf.estimators_[index],
                   feature_names = label_names,
                   class_names=class_names,
                   filled = True,
                   ax = axes[index]);

    axes[index].set_title('Estimator: ' + str(index), fontsize = 11)
fig.savefig('rf_5trees.png')
```

O QUE VOCÊ VIU NESTA AULA?

Árvore de Decisão e Random Forest.

Daqui em diante, é importante que você replique os conhecimentos adquiridos para fortalecer mais suas bases e conhecimentos.

IMPORTANTE: não esqueça de praticar com o desafio da disciplina, para que, assim, você possa aprimorar seus conhecimentos!

Você não está só nesta jornada! Te esperamos no Discord e nas *lives* com especialistas, onde você poderá tirar dúvidas, compartilhar conhecimentos e estabelecer conexões!

REFERÊNCIAS

BRUCE, A. **Estatística Prática para Cientistas de Dados**. 1. ed. [s.l.]: O'Reilly Media, Inc., 2019.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. 2. ed. [s.l.]: O'Reilly Media, Inc., 2019.

SCIKIT-LEARN. **Documentação Scikit-Learn**. 2023. Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 13 abr. 2023.

PALAVRAS-CHAVE

Palavras-chave: Decision Tree. Random Forest. Gini. Entropy.