Instituto Tecnológico de Aeronáutica - ITA
ELE-32 - Introdução a Comunicações
Aluno: Samir Nunes da Silva

Trabalho Extra - Implementação de Árvore de Decisão com o Algoritmo ID3

---

1 Descrição do Problema

Em um problema de previsão, no campo de ciência de dados, tem-se variáveis chamadas de features e uma variável-alvo chamada de target - os valores das features são passados ao modelo, no momento da previsão, que então retorna o valor do target. Para ser capaz de realizar a previsão, o modelo deve ser treinado no conjunto de dados de treino, o qual deve ser composto de um número suficiente de amostras (conjunto de valores das features e do valor correspondente do target) para que o modelo aprenda algum padrão subjacente. Nesse contexto, a depender do modelo, há diferentes algoritmos utilizados para realizar o treinamento.

Árvore de decisão (decision tree) [1] é um método supervisionado não-paramétrico de aprendizado de máquina que se baseia em uma estrutura composta de nós nos quais se realizam decisões com base no valor de uma dada feature. A Figura 1 é uma ilustração de uma árvore de decisão no problema de decidir acerca da compra de um carro. Por meio dela, nota-se que a escolha é feita nos nós folhas com base nas decisões feitas em cada nó. As features, nesse caso, são a cor do carro, o ano do modelo, o modelo do carro e a quilometragem, e o target é a escolha de comprar ou não o carro.

Para construir (treinar) um modelo de árvore de decisão, pode-se utilizar algoritmos como o ID3 (Iterative Dichotomiser 3) [2], o C4.5, o C5.0 e o CART (Classification and Regression Tree). No presente trabalho, será implementado e explorado o algoritmo ID3, o qual se vale dos conceitos de entropia e de ganho de informação (informação mútua), comuns à disciplina de telecomunicações. Por meio dele, é possível criar modelos para problemas de classificação binária, quando o target possui apenas dois valores possíveis: 0 ou 1. Ao fim, o algoritmo desenvolvido será aplicado a um problema de previsão de bank churn, ou seja, de prever se um cliente irá ou não sair de um dado serviço bancário.

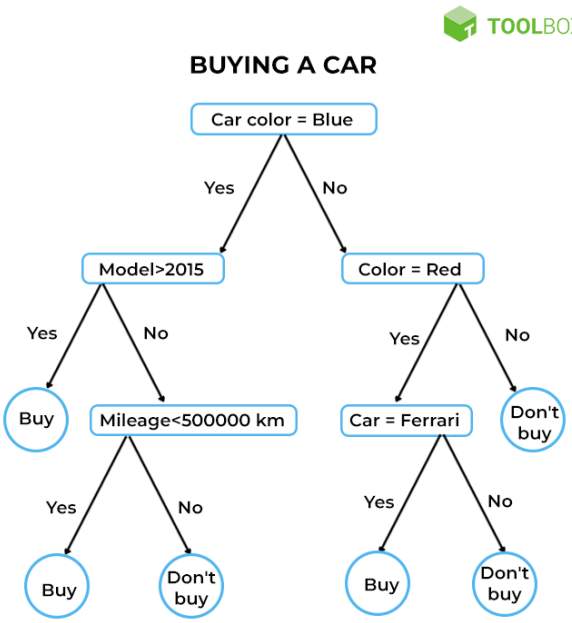


Figura 1: Exemplo de árvore de decisão. Imagem retirada de: `https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-decision-tree/`

2 Implementação

A implementação da árvore de decisão e do algoritmo ID3 foi feita na linguagem Python, com o auxílio das bibliotecas `Pandas` e `Numpy`. Basicamente, ela foi feita em três passos: definição de um nó da árvore, criação do algoritmo ID3 e criação da estrutura de árvore de decisão. Os códigos produzidos podem ser vistos em anexo.

Inicialmente, definiu-se a estrutura de dados Node, que representa um nó da árvore. Ela é responsável por armazenar informações como o conjunto de dados visto pelo nó, os dois nós filhos (chamados de `true_child` e `false_child`, sendo true o nó gerado pela decisão quando a condição da feature atende ao sinal de maior ou igual, e false sendo o oposto), o target, a previsão do nó, sua entropia e as informações acerca do split do nó nos seus filhos (feature usada para o split, valor da feature usado para a condição do split e ganho de informação do split). Essa é a estrutura de dados que funcionará como componente fundamental da árvore de decisão.

Como dito anteriormente, o algoritmo ID3 se baseia no uso de entropia e informação mútua para a criação da árvore de decisão. Conforme visto no contexto de telecomunicações e teoria da informação [3], a entropia de uma variável aleatória discreta X é uma grandeza definida pela Equação 1, sendo $p(x)$ a probabilidade de X valer x. A entropia representa a incerteza de uma dada variável aleatória.

$$H(x) = -\sum_{x \in X} p(x) log_2[p(x)] \tag{1}$$

Conhecendo-se X em um dado contexto, pode-se obter informações acerca de outra variável aleatória discreta Y. Tal situação aparece, por exemplo, no caso de um canal binário simétrico,

conforme a Figura 2. Nele, X é a variável aleatória que representa a entrada e Y a variável aleatória que representa a saída, tendo-se probabilidade p de errar a transmissão e 1-p de acertar. Conhecer X, nesse caso, fornece informações acerca de Y, e tal fato é representado pela grandeza chamada informação mútua entre X e Y, dada pela Equação 2, onde $H(Y|X)$ é a entropia condicional de Y dado X, que representa a incerteza sobre Y quando sabemos o valor de X.

$$I(Y, X) = H(Y) - H(Y|X) \tag{2}$$

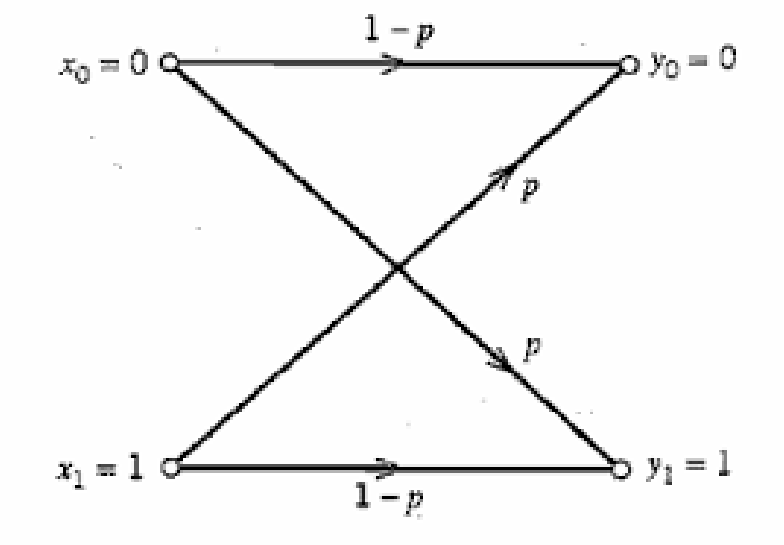


Figura 2: Canal binário simétrico.

No algoritmo ID3, temos, de maneira análoga, o conceito de entropia e de informação mútua, a qual é chamada, nesse contexto, de ganho de informação. Cada nó possui uma dada distribuição da variável target, resumida como a quantidade de valores zeros e uns dessa variável que o nó possui devido às decisões já feitas, e sua entropia relativa à variável target, aqui representada como a variável aleatória Y, a qual é calculada conforme a Equação 1, podendo y assumir os valores de 0 ou 1. Após um split em nós filhos baseado em uma feature, aqui representada pela variável aleatória X, e em um dado valor dela, cada um dos novos nós terá também uma entropia associada. O ganho de informação $I(Y, X)$ é então dado pela Equação 3 [2], que deriva da Equação 2. Nela, T representa o conjunto dos subconjuntos criados pelo split dos dados do nó em dois nós filhos e p(t) a proporção do número de elementos em t em relação ao conjunto de dados de antes do split. Aqui, o ganho de informação representa quanta informação a feature usada para o split forneceu acerca da variável target, de forma análoga ao que ocorre no canal binário. O objetivo do algoritmo ID3 é, portanto, escolher, de maneira gulosa, as features e seus respectivos valores de tal forma que os ganhos de informação sejam os maiores possíveis em cada split, minimizando a entropia dos nós folhas.

$$I(Y, X) = H(Y) - \sum_{t \in T} p(t)H(t) = H(Y) - H(Y|X) \tag{3}$$

Com isso em mente, implementou-se o algoritmo na classe `ID3`, com uma construção de árvore recursiva. Começa-se no nó raiz com todo o conjunto de dados e realiza-se a busca pela feature e seu valor para realizar o split dos dados em dois nós com o maior ganho de informação possível. Isso é feito iterando-se em todas as features e em todos seus valores possíveis, realizando-se os splits e calculando-se cada ganho de informação e, ao fim, selecionando o que gera o maior ganho. O split considera uma condição na forma $\geq$ (maior ou igual) ao valor da feature e $<$ (menor) que esse valor, gerando um `true_node` no primeiro caso e um `false_node` no segundo, e, quando uma feature é usada, ela não pode ser usada novamente para outro split. Em seguida, realiza-se a construção da mesma forma de maneira recursiva para os nós filhos, até que se atinjam as condições de parada: o nó possui apenas um valor do target (0 ou 1 apenas) ou não há mais features disponíveis para split. Quando uma delas for satisfeita, tem-se um nó folha, por meio do qual é possível realizar uma previsão, caso os valores das features dadas de input levem as decisões a ele. A construção é feita através da função `build_tree`.

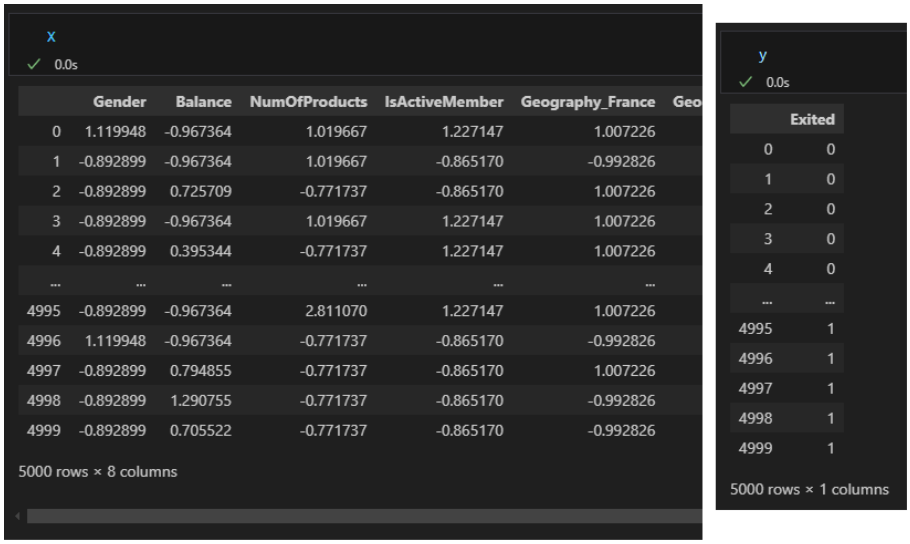
Finalmente, implementou-se a classe `DecisionTree`, por meio da qual pode-se gerar modelos de árvore de decisão construídos com o algoritmo ID3 e utilizá-los para realizar previsões. A função `train` realiza o treinamento de um modelo de árvore, fornecidos os dados de treino: X, o dataframe contendo as features, e y, o dataframe contendo as labels do target. Basicamente, ela realiza a chamada de `build_tree` para, a partir de seu nó raiz, construir a árvore.

Com a árvore construída, é possível usar a função `predict` para realizar previsões, dados os valores das features. A previsão é feita de forma recursiva, percorrendo-se a árvore conforme os valores de entrada das features satisfazem ou não as condições de split de cada um dos nós. Finalmente, quando se atinge o nó folha, retorna-se a previsão dada por ele.

Também criou-se um método para realizar o `print` da árvore gerada. Ele mostra cada nó com seus respectivos pais e filhos, bem como com sua entropia, ganho de informação do split nos seus filhos, feature usada para o split e valor limiar da feature usado no split.

3 Resultados

Primeiramente, realizou-se a redução aleatória do conjunto de dados de bank churn [4], para 5000 amostras, para que o treinamento não fosse muito demorado; essas amostras são enviadas em anexo. Parte dos dados pode ser vista na Figura 3. Na seleção das amostras, manteve-se a proporção das classes no target (50% cada, conforme o dataset original). Com os dados em mãos, realizou-se o scaling dos dados através de um `StandardScaler` [5] para transformar todos os dados para a mesma escala (essa é uma boa prática, pois em alguns algoritmos a escala dos dados afeta a velocidade de convergência e a qualidade do modelo) e, em seguida, o treino de uma árvore de decisão, considerando o target "Exited"e as features "Gender", "Balance", "NumOfProducts", "IsActiveMember", "Geography_France", "Geography_Germany", "Geography_Spain"e "Age_bin", cujas explicações podem ser encontradas na página de onde os dados foram retirados. O treinamento levou cerca de 35 s.



x

✓ 0.0s

	Gender	Balance	NumOfProducts	IsActiveMember	Geography_France	Geo
0	1.119948	-0.967364	1.019667	1.227147	1.007226	
1	-0.892899	-0.967364	1.019667	-0.865170	-0.992826	
2	-0.892899	0.725709	-0.771737	-0.865170	1.007226	
3	-0.892899	-0.967364	1.019667	1.227147	1.007226	
4	-0.892899	0.395344	-0.771737	1.227147	1.007226	
...	...	...	...	...	...	
4995	-0.892899	-0.967364	2.811070	1.227147	1.007226	
4996	1.119948	-0.967364	-0.771737	-0.865170	-0.992826	
4997	-0.892899	0.794855	-0.771737	-0.865170	1.007226	
4998	-0.892899	1.290755	-0.771737	-0.865170	-0.992826	
4999	-0.892899	0.705522	-0.771737	-0.865170	-0.992826	

5000 rows × 8 columns

y

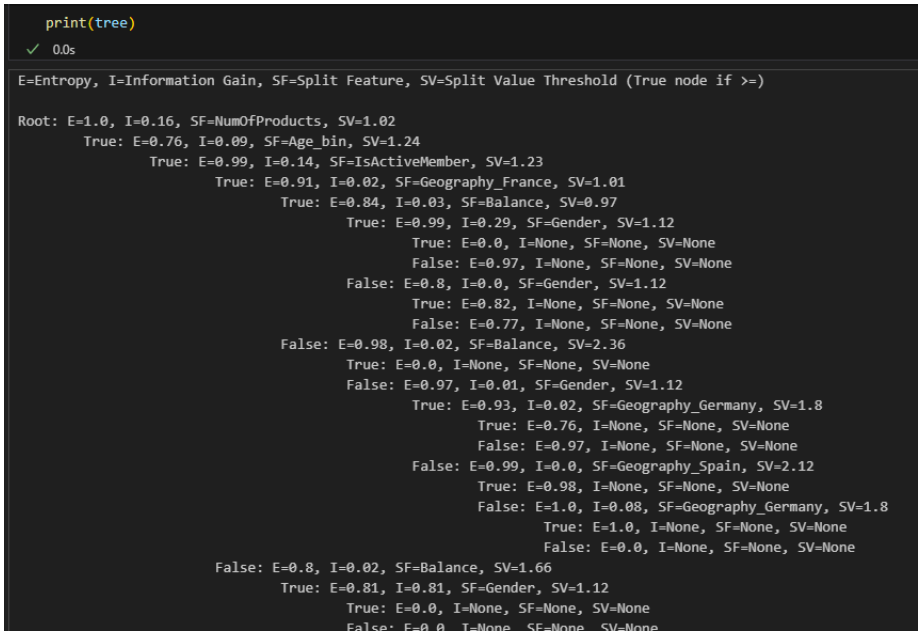
✓ 0.0s

	Exited
0	0
1	0
2	0
3	0
4	0
...	...
4995	1
4996	1
4997	1
4998	1
4999	1

5000 rows × 1 columns

Figura 3: Parte dos dados. X representa o dataframe das features e y o dataframe do target.

A árvore gerada pode ser vista, em partes (pois ficou consideravelmente grande), na Figura 4, na qual se tem todas as informações essenciais: relações entre os nós (percebidas pelos espaçamentos), entropias, ganhos de informação, features usada no splits e valores das features usados de limiares para o split.



```
print(tree)
```

✓ 0.0s

```
E=Entropy, I=Information Gain, SF=Split Feature, SV=Split Value Threshold (True node if >=)

Root: E=1.0, I=0.16, SF=NumOfProducts, SV=1.02
        True: E=0.76, I=0.09, SF=Age_bin, SV=1.24
                True: E=0.99, I=0.14, SF=IsActiveMember, SV=1.23
                        True: E=0.91, I=0.02, SF=Geography_France, SV=1.01
                                True: E=0.84, I=0.03, SF=Balance, SV=0.97
                                        True: E=0.99, I=0.29, SF=Gender, SV=1.12
                                                True: E=0.0, I=None, SF=None, SV=None
                                                False: E=0.97, I=None, SF=None, SV=None
                                        False: E=0.8, I=0.0, SF=Gender, SV=1.12
                                                True: E=0.82, I=None, SF=None, SV=None
                                                False: E=0.77, I=None, SF=None, SV=None
                                False: E=0.98, I=0.02, SF=Balance, SV=2.36
                                        True: E=0.0, I=None, SF=None, SV=None
                                        False: E=0.97, I=0.01, SF=Gender, SV=1.12
                                                True: E=0.93, I=0.02, SF=Geography_Germany, SV=1.8
                                                        True: E=0.76, I=None, SF=None, SV=None
                                                        False: E=0.97, I=None, SF=None, SV=None
                                                False: E=0.99, I=0.0, SF=Geography_Spain, SV=2.12
                                                        True: E=0.98, I=None, SF=None, SV=None
                                                        False: E=1.0, I=0.08, SF=Geography_Germany, SV=1.8
                                                                True: E=1.0, I=None, SF=None, SV=None
                                                                False: E=0.0, I=None, SF=None, SV=None
                        False: E=0.8, I=0.02, SF=Balance, SV=1.66
                                True: E=0.81, I=0.81, SF=Gender, SV=1.12
                                        True: E=0.0, I=None, SF=None, SV=None
                                        False: E=0.0, I=None, SF=None, SV=None
```

Figura 4: Parte da árvore de decisão construída com o algoritmo ID3.

O modelo foi então utilizado para realizar a previsão no próprio conjunto de treino, para visualizar se, de fato, o modelo aprendeu (não necessariamente generalizou, mas o objetivo é o aprendizado em si). Nesse sentido, calculou-se três métricas de classificação binária: acurácia,

revocação e precisão [6]. Os resultados podem ser vistos na Figura 5 e confirmam que o modelo de fato aprendeu, já que todas as métricas ficaram na faixa de 83%, significando que, no geral, a árvore acertou as previsões.

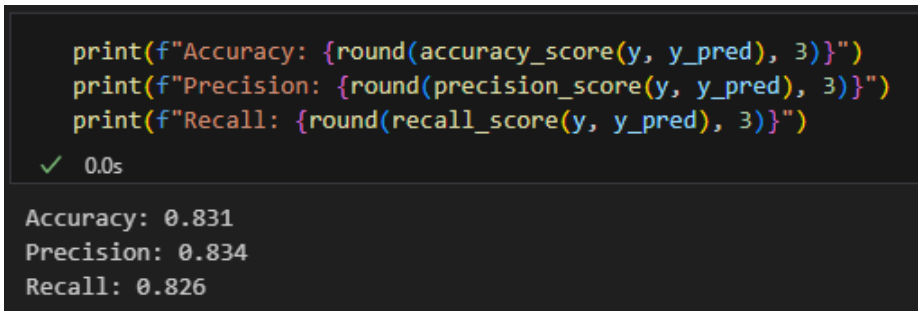


Figura 5: Métricas de classificação binária obtidas para o modelo de árvore de decisão treinado.

Com isso, criou-se um possível modelo de árvore de decisão para previsão de bank churn usando o algoritmo ID3, baseando-se em conceitos fundamentais de teoria da informação que também se apresentam em telecomunicações. Assim, verifica-se, de fato, uma interseção entre as áreas de aprendizado de máquina e telecomunicações, a qual foi possível explorar no presente trabalho.

4 Conceitos Utilizados

- Entropia de Variáveis Discretas
- Entropia Condicional
- Informação Mútua
- Canal Binário Simétrico

5 Referências

[1] https://towardsdatascience.com/decision-trees-for-classification-id3-algorithm-explained-89df76e72df1
[2] https://en.wikipedia.org/wiki/ID3_algorithm
[3] https://sgfin.github.io/files/notes/Cover_and_Thomas_ch2_entropy.pdf
[4] https://www.kaggle.com/datasets/prishasawhney/binary-classification-bank-churn-dataset-cleaned?select=train_cleaned.csv
[5] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
[6] https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall