Inteligência Artificial

CC2006

Árvores de Decisão

Catarina Monteiro - 202105279

Lara Sousa - 202109782

Mariana Serrão - 202109927

2022 / 2023

Índice

Índice de Ilustrações

Índice de Tabelas

Índice de Gráficos

1. Introdução

Uma árvore de decisão é um modelo de classificação que funciona através da criação de uma estrutura hierárquica em forma de árvore direcionada. É uma estrutura na qual o nó inicial é designado por raiz e os restantes possuem uma única aresta de entrada. A um nó com arestas de saída dá se o nome de nó interno ou de teste. Todos os outros são designados de folhas (também conhecidos como nós terminais ou de decisão).

Numa árvore de decisão, cada nó interno representa um teste realizado a um atributo de entrada. Este teste divide o conjunto de dados em dois ou mais subconjuntos, tendo por base um valor limite ou uma condição definida para esse atributo. Cada um desses subconjuntos representa um ramo da árvore que conduz a outro nó interno ou a uma folha. As folhas representam classes ou categorias previstas pelo modelo para os dados de entrada. Alternativamente, podem conter um vetor de probabilidade, que indica a probabilidade de o atributo-alvo ter um determinado valor.

No caso mais simples e mais frequente, cada teste considera um único atributo, para que o conjunto de dados seja dividido de acordo com o valor do atributo. Para atributos numéricos, a condição refere-se a um intervalo. As amostras são classificadas a partir da raiz da árvore até uma folha, seguindo o resultado dos testes em cada nó do percurso.

O objetivo principal de uma árvore de decisão é ajudar na tomada de decisões, fornecendo uma representação visual deste processo e permitindo que o utilizador do modelo siga um caminho específico com base nas informações disponíveis. As árvores de decisão são utilizadas em diversas áreas, como a financeira, na medicina, em marketing, entre outras, uma vez que permitem a identificação de padrões e relações complexas entre atributos e classes, tornando-se um modelo de grande utilidade para previsão e tomada de decisões em diferentes contextos. Além disso, a sua facilidade de interpretação e capacidade de lidar com dados categóricos e numéricos fazem das árvores de decisão uma técnica de aprendizagem computacional muito popular e versátil.

A Figura 1 trata-se de um exemplo de uma árvore de decisão, usada na tomada de decisão para um novo cliente de um banco. É feita uma hierarquia de testes a algumas das variáveis envolvidas no problema de decisão, ou seja, o montante, a idade, o salário e a conta.

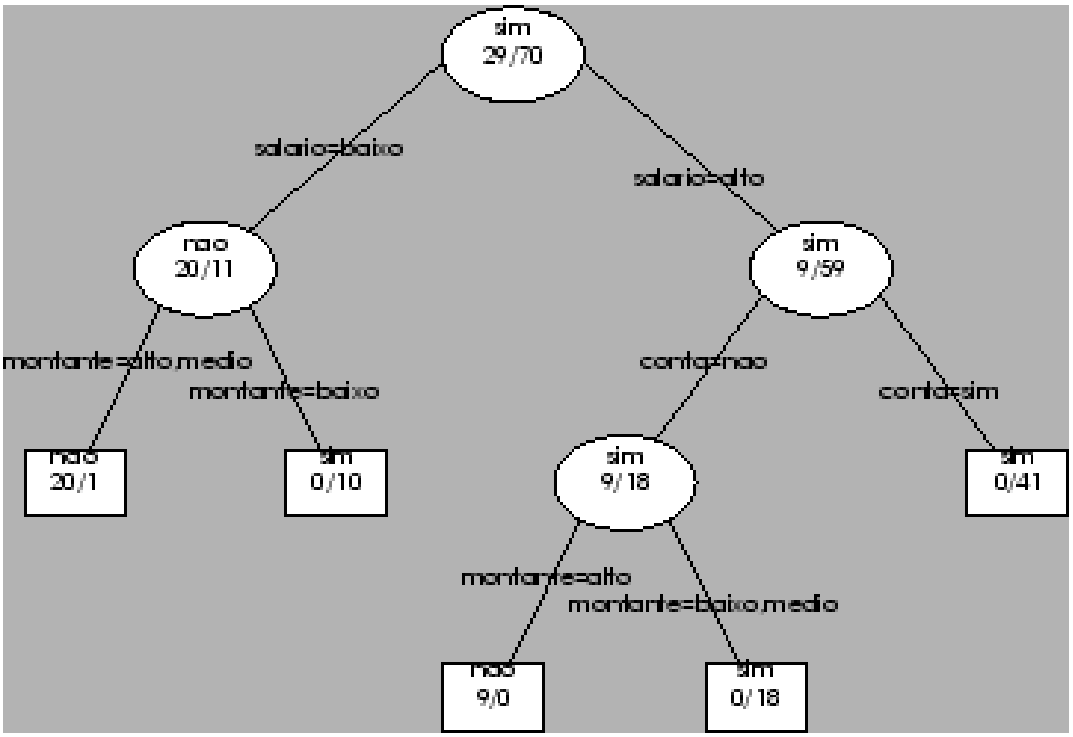


Figura 1 - Árvore Decisão para Cliente do Banco

Relativamente à tomada de decisão, deve ser iniciada na raiz, atendendo ao salário e, a partir desse nó, deve proceder-se à verificação do salário, que pode ser baixo, alto ou médio. Se for baixo, é solicitada a verificação do montante, e, se for alto ou médio não será então aconselhado a conceção do crédito. Resumindo o referido, começa-se por fazer o teste do nó raiz, e obtém-se a sua veracidade. Se for verdadeiro segue-se para o *ramo* esquerdo da árvore, executando o teste destinado; se for falso segue-se o ramo direito, e assim sucessivamente até se alcançar um nó folha ou terminal, no qual a decisão do modelo para o caso em análise é encontrada. É, então, possível traduzir uma árvore de decisão para um conjunto de regras de decisão, facilmente implementável numa função que toma decisões para novos casos, pois é possível, por exemplo, descrever o percurso desde o nó raiz, e seguir sempre o ramo esquerdo até se encontrar uma folha, através da regra: 'SE salário alto E montante alto ou médio ENTÃO não conceder crédito'.

As árvores de decisão são úteis em cenários nos quais as instâncias são descritas por pares atributo-valor e a função alvo é de valor discreto. Consegue lidar com hipóteses disjuntas e são capazes de lidar com exemplos de treino que contenham erros (ruído) ou valores ausentes nos atributos.

Este algoritmo destaca-se pela sua simplicidade na representação do conhecimento proposicional usado essencialmente para fazer decisões e classificar objetos. A estrutura

de uma árvore é construída com base em atributos relevantes e testes que permitam classificar objetos ou tomar decisões com base nos dados disponíveis.

Portanto, envolve a análise dos dados de treino e a seleção dos melhores atributos para dividir os dados de forma eficiente. Nesse sentido, a curva de aprendizagem é uma ferramenta que pode ser usada para avaliar o desempenho do modelo de árvore de decisão à medida que mais dados são utilizados. Inicialmente, quando há menos dados, a árvore pode não ter informações suficientes para fazer decisões precisas e classificar corretamente os objetos. No entanto, conforme a quantidade de dados aumenta, a árvore pode aprender mais padrões e melhorar seu desempenho, tornando-se mais confiável na tomada de decisões e classificação.

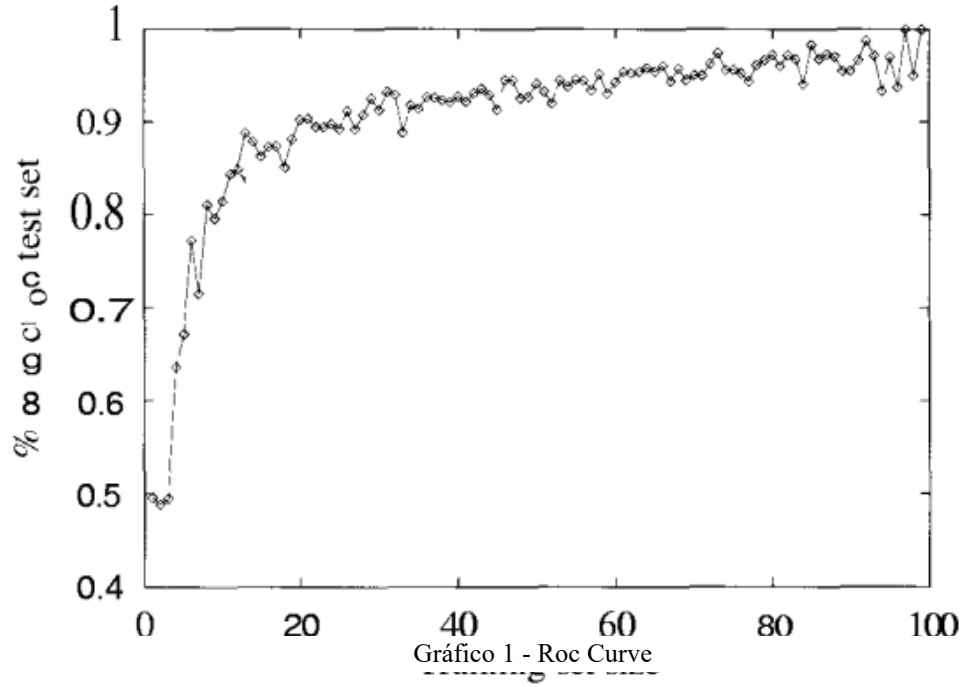


Gráfico 1 - Roc Curve

2. Algoritmos para indução de árvores de decisão

As árvores de decisão podem ser induzidas a partir de exemplos, isto é, dos valores dos atributos e do valor da classe prevista, ou seja, do classificador do exemplo. Se a classe for verdadeira para algum exemplo, o exemplo denomina-se de positivo, caso contrário é um exemplo negativo. Na tabela 1 está representado um conjunto de exemplos X_1,…...,X_{12} para o domínio restaurante, na qual se observa que os exemplos positivos são os que apresentam a classe *WillWait* como *Yes* (por exemplo, o caso do X_1 e do X3) e que os exemplos negativos foram classificados como *No*,(X_2 e X5). Todo o conjunto completo de exemplos denomina-se de training set.

Tabela 1 - Exemplo Restaurante

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0-10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0-10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0-10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0-10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0-10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10-30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0-10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30-60*	*Yes*

As árvores de decisão são uma técnica popular para modelar e resolver problemas de classificação e regressão, oferecendo uma representação visualmente intuitiva e de fácil interpretação, além de serem capazes de lidar com conjuntos de dados complexos. A construção de árvores de decisão envolve a escolha criteriosa de atributos para dividir o conjunto de dados e tomar decisões precisas. Nesse contexto, diferentes algoritmos de indução de árvores de decisão foram desenvolvidos, cada um com as suas próprias abordagens e critérios de seleção, destacando-se os algoritmos: ID3, C4.5 e CART. Compreender estes algoritmos é de crucial importância no que diz respeito à utilização de árvores de decisão como ferramentas eficazes de modelagem e tomada de decisão em diversos domínios.

Antes de falar mais detalhadamente sobre cada algoritmo de indução de árvores de decisão, é importante explorar um pouco de toda a história por detrás da sua criação. O primeiro sistema para construção de árvores de decisão de que se tem conhecimento

foi desenvolvido na década de 1950 por Edward Feigenbaum, como parte da sua tese de Doutoramento. Tinha o nome de Elementary Perceiver and Memorizer, ou EPAM, e o foco era, essencialmente, explicar e prever o fenómeno da aprendizagem verbal humana.

O próximo avanço na área seria dado na década de 1960 na Universidade de Yale, onde o psicólogo Carl I. Hovland e o seu aluno de doutoramento Earl B. Hunt desenvolveram um modelo computorizado da aprendizagem de conceitos por parte de seres humanos. Com futuras contribuições de Janet Martin e Philip Stone, nasce o Concept Learning System, ou CLS.

No final da década de 1970, J. Ross Quinlan desenvolveu o Iterative Dichotomiser 3 (ID3), na Universidade de Stanford, onde estava de licença sabática por parte da Universidade de Sidney. Quinlan tinha sido o primeiro aluno de doutoramento de Earl Hunt. Aquando da sua criação, o objetivo seria aprender uma regra que decidisse o resultado de um jogo de xadrez e tinha apenas cerca de 600 linhas de Pascal.

Quinlan continuaria a desenvolver sistemas de construção de árvores de decisão e surgiu o C4.5 com cerca de 9000 linhas de C, onde se podia trabalhar com atributos numéricos e não apenas com atributos categóricos, entre outras melhorias relativamente ao ID3. Uma versão melhorada do C4.5 passa a ser comercializada em 1983. Ainda na década de 1980 é também introduzido o algoritmo CART (Classification and Regression Trees), onde se introduzem importantes conceitos sobre árvores de regressão e de classificação, por Leo Breiman et al.

O algoritmo **Iterative Dichotomiser 3**, ID3, proposto por J. Ross Quinlan em 1986, destaca-se pela sua simplicidade e eficiência, tendo como objetivo criar uma árvore de decisão que possa ser utilizada em previsões ou tomada decisões com base em atributos dos dados de input, frequentemente conjuntos de dados rotulados. Na sua abordagem *top-down* (de cima para baixo), inicia com o conjunto de dados completo e, em cada nó da árvore, analisa os atributos de forma a eleger a raiz da árvore como o atributo que possui o maior ganho de informação. De forma recursiva, seleciona os restantes atributos repetindo o procedimento para cada subconjunto de dados criado pela divisão dos exemplos de acordo com o valor do atributo selecionado tal como ilustrado na figura 2 e atendendo sempre a um maior ganho de informação ou a uma maior redução na entropia.

Este conceito de entropia é utilizado pelo ID3 na medição da impureza de um conjunto de dados. Trata-se de uma medida de incerteza ou desordem. Quanto maior a entropia, maior a incerteza. Assim, o objetivo do algoritmo é encontrar a melhor divisão dos dados que reduza a entropia e aumente a pureza das subárvores resultantes.

Contudo, existem algumas limitações associadas ao algoritmo ID3, tais como a tendência a criar árvores muito profundas e complexas, o que pode conduzir ao *overfitting* (modelo excessivamente ajustado aos dados de treinamento); e o facto de não lidar bem com atributos contínuos, uma vez que requer que os atributos sejam discretos.

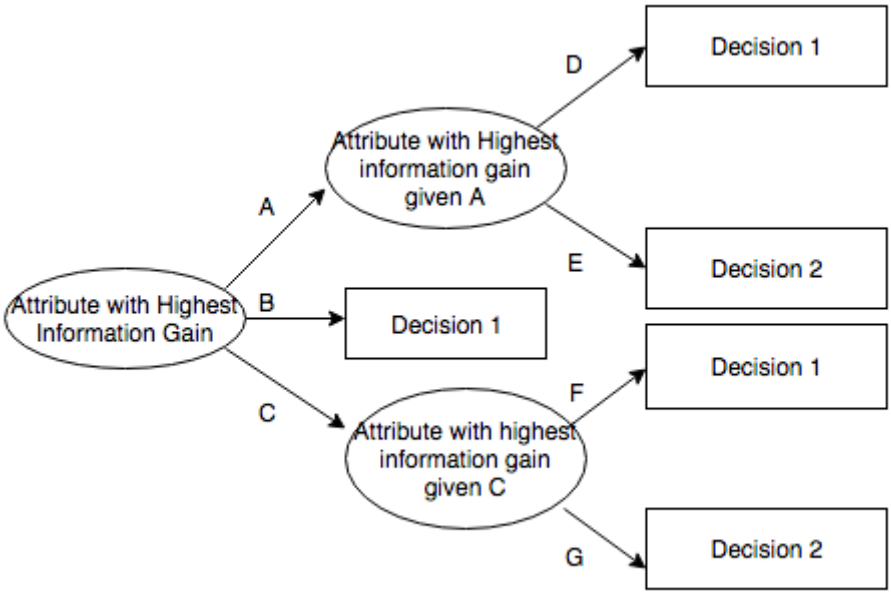


Figura 2 - Algoritmo ID3

O algoritmo **Classification and Regression Tree 4.5.** (C4.5), desenvolvido por Ross Quinlan em 1993, surge como uma evolução do ID3, face às suas limitações, nomeadamente pela sua capacidade de lidar com atributos contínuos, pois permite que sejam tratados de forma mais eficiente, usando discretização, o processo de conversão de eatributos contínuos em atributos discretos, ou seja, transformação de valores numéricos em categorias ou intervalos discretos.

Da mesma forma que o algoritmo anterior, o C4.5 recorre à abordagem *top-down* e utiliza o conceito de ganho de informação para selecionar o melhor atributo em cada nó da árvore introduzindo, adicionalmente, *Normalized Information Gain* (NIG). Esta medida considera o número de valores distintos de um atributo e a quantidade de dados em cada valor, mitigando a tendência do ID3 em selecionar atributos com muitos valores

distintos. Para além destes aspetos, o algoritmo C4.5 controla dados ausentes e introduz o conceito de *pruning* (simplificação da árvore de decisão, através da remoção de ramos ou nós que não contribuem para a precisão), com o intuito de contornar o *overfitting*.

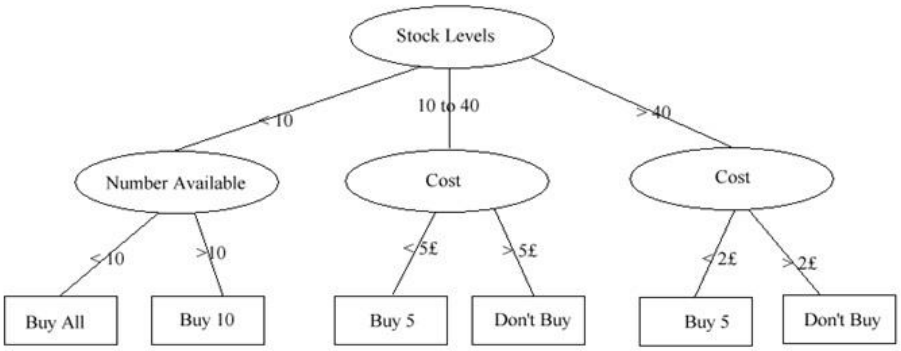


Figura 3 - Algoritmo C4.5

O algoritmo **Classification and Regression Trees** (CART) é adequado tanto para problemas de classificação, como para problemas de regressão. De forma recursiva, divide o conjunto de dados em subconjuntos, até que uma condição o obrigue a terminar, tendo em conta os valores dos atributos.

De forma mais aprofundada, o algoritmo começa por selecionar o atributo de divisão, isto é, o atributo que será usado para dividir o conjunto de dados em subconjuntos, recorrendo a uma métrica de impureza, como o Índice de Gini e a entropia. O objetivo desta abordagem de seleção é obter o atributo resultante na maior redução de impurezas.

De seguida, é realizada a divisão dos dados em subconjuntos, sendo que cada subconjunto representa um ramo na árvore de decisão, de acordo com os diferentes valores do atributo eleito.

Os dois passos mencionados até então são repetidos de forma recursiva até ao momento em que a recursividade é interrompida, por exemplo, quando a profundidade máxima pré-definida é atingida ou então quando não existem mais atributos disponíveis para dividir. Na interrupção da recursão, a cada folha da árvore é atribuído a classe ou o valor de regressão correspondente ao subconjunto de dados associado. Se for necessário, com o objetivo de minimizar o *overfitting,* é ainda possível, após a construção da árvore,

aplicar *prunning*. No final, a árvore fica apta à classificação de novas instâncias (classificação) ou previsão de valores (regressão).

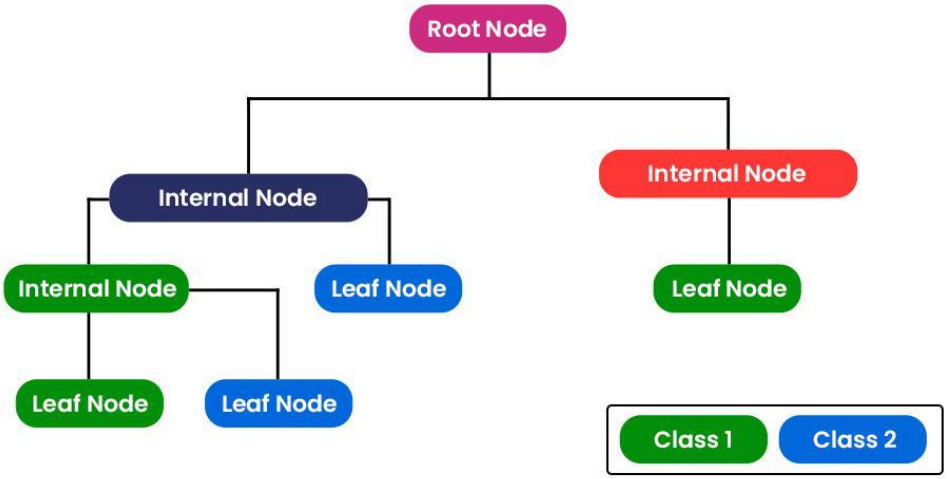
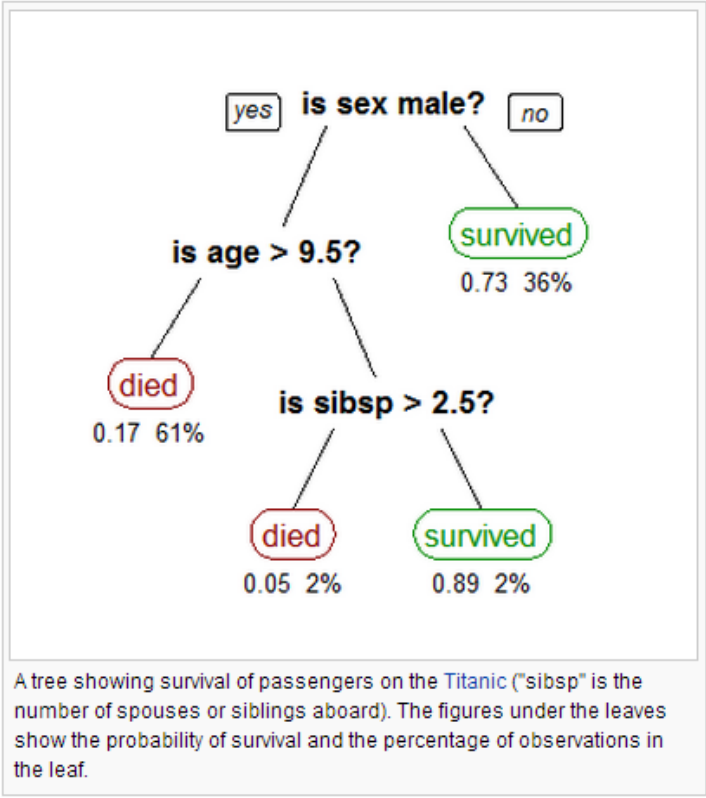


Figura 4 - Algoritmo CART 1



A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf.

Figura 5 - Algoritmo CART 2

3. Implementação

O algoritmo utilizado para a implementação da árvore de decisão foi o ID3 (Iterative Dichotomiser 3), um algoritmo clássico de aprendizagem computacional, que constrói árvores de decisão a partir de conjuntos de dados rotulados. O objetivo da implementação é realizar a classificação de dados, onde um conjunto de exemplos é utilizado para construir uma árvore de decisão, que pode ser utilizada para prever a classe de novos exemplos. A implementação foi realizada em Python, com recorrência às bibliotecas pandas e numpy para manipulação e análise de dados.

O algoritmo desenvolvido solicita ao utilizador, inicialmente, que insira o nome do arquivo CSV que contém o conjunto de dados. Em seguida, lê o arquivo e realiza o pré-processamento dos dados. Realiza, então, a construção e consequente impressão da árvore de decisão. Oferece ainda a opção de classificar exemplos específicos fornecidos pelo utilizador.

A classe principal da implementação é a classe TreeNode, que representa um nó da árvore de decisão. Cada nó contém um atributo associado, um rótulo da classe (apenas para folhas), a contagem de elementos na folha e um dicionário de ramos, que representa os ramos da árvore.

O algoritmo ID3 começa com o conjunto de dados completo e seleciona recursivamente o melhor atributo para dividir o conjunto em subconjuntos mais puros. A pureza é medida pela entropia, que é uma medida de incerteza. Quanto menor a entropia, maior a pureza. O ganho de informação é usado para escolher o atributo com o maior impacto na redução da entropia.

Assim, criou-se a função *entropia*, que realiza o cálculo da mesma num conjunto de dados, em relação a uma classe específica. Conta a frequência de cada classe no conjunto de dados e utiliza a fórmula da entropia de Shannon ($H(S) = -\Sigma (p(i) * \log2(p(i))$; onde $H(S)$ é a entropia do conjunto de dados S; $p(i)$ é a proporção de exemplos em S que pertencem à classe i; e log2 representa o logaritmo na base 2) para calcular a entropia total.

Foi criada também a função *ganho*, que calcula o ganho de informação obtido ao dividir o conjunto de dados com base num atributo específico, através da subtração da soma ponderada das entropias aos subconjuntos da entropia inicial.

A função *seleciona_melhor_atributo* é a que se encarrega de selecionar o atributo que possui o maior ganho de informação em relação à classe, tendo em conta a entropia, ou seja, seleciona o melhor atributo para dividir o conjunto de dados.

No processo de construção da árvore, criou-se a função *constroi,* que, numa primeira fase, verifica casos de paragem, como quando todos os exemplos possuem a mesma classe ou quando não há mais atributos para dividir. Caso contrário, seleciona o melhor atributo, através da função *seleciona_melhor_atributo*, e cria subárvores para cada valor desse atributo. A construção é realizada recursivamente, até que todos os exemplos sejam classificados corretamente.

De forma a imprimir a árvore no formato pedido, a função *print_tree* foi implementada. Percorre a árvore de forma recursiva, e imprime os atributos e valores dos ramos em níveis diferentes, o que facilita a visualização da estrutura da árvore.

Para os casos em que o utilizador tem intenções de classificar exemplos específicos, a função *classify* realiza esse mesmo processo, utilizando a árvore de decisão. Percorre a árvore de forma recursiva, comparando os valores do exemplo com os valores dos ramos. Caso o valor do atributo não esteja presente na árvore, a função retorna a classe mais frequente encontrada nos ramos, através da função *maior_classe*.

O algoritmo ID3 utilizado nesta implementação é uma abordagem clássica e eficiente para construção de árvores de decisão. Utiliza o ganho de informação para selecionar os melhores atributos e construir uma árvore que seja capaz de classificar exemplos com base nesses atributos. A implementação fornece uma ferramenta útil para a análise e classificação de conjuntos de dados.

4. Resultados

Após a implementação da árvore, como descrito no capítulo anterior, obteve-se as seguintes árvores para os data sets: weather.csv, restaurant.csv e iris.csv respetivamente.

4.1. Data Set Weather

```
Atributo: Temp
|
|----Valor: 85
|   Classe: no (Count: 1)
|
|----Valor: 80
|   Classe: no (Count: 1)
|
|----Valor: 83
|   Classe: yes (Count: 1)
|
|----Valor: 70
|   Classe: yes (Count: 1)
|
|----Valor: 68
|   Classe: yes (Count: 1)
|
|----Valor: 65
|   Classe: no (Count: 1)
|
|----Valor: 64
|   Classe: yes (Count: 1)
|
|----Valor: 72
|   Atributo: Weather
|   |
|   |----Valor: sunny
|   |   Classe: no (Count: 1)
|   |
|   |----Valor: overcast
|   |   Classe: yes (Count: 1)
|
|----Valor: 69
|   Classe: yes (Count: 1)
|
|----Valor: 75
|   Classe: yes (Count: 2)
|
|----Valor: 81
|   Classe: yes (Count: 1)
|
|----Valor: 71
|   Classe: no (Count: 1)
```

4.2. Data Set Restaurant

```
Atributo: Pat
|
|----Valor: Some
|   Classe: Yes (Count: 4)
|
|----Valor: Full
|   Atributo: Type
|   |
|   |----Valor: Thai
|   |   Atributo: Fri
|   |   |
|   |   |----Valor: No
|   |   |   Classe: No (Count: 1)
|   |   |
|   |   |----Valor: Yes
|   |   |   Classe: Yes (Count: 1)
|   |
|   |----Valor: French
|   |   Classe: No (Count: 1)
|   |
|   |----Valor: Burger
|   |   Atributo: Alt
|   |   |
|   |   |----Valor: No
|   |   |   Classe: No (Count: 1)
|   |   |
|   |   |----Valor: Yes
|   |   |   Classe: Yes (Count: 1)
|   |
|   |----Valor: Italian
|   |   Classe: No (Count: 1)
|
|----Valor: None
|   Classe: No (Count: 2)
```

4.3 Data Set Iris

```
Atributo: petallength
|
|----Valor: 1.4
|   Classe: Iris-setosa (Count: 12)
|
|----Valor: 1.3
|   Classe: Iris-setosa (Count: 7)
|
|----Valor: 1.5
|   Classe: Iris-setosa (Count: 14)
|
|----Valor: 1.7
|   Classe: Iris-setosa (Count: 4)
|
|----Valor: 1.6
|   Classe: Iris-setosa (Count: 7)
|
|----Valor: 1.1
|   Classe: Iris-setosa (Count: 1)
|
|----Valor: 1.2
|   Classe: Iris-setosa (Count: 2)
|
|----Valor: 1.0
|   Classe: Iris-setosa (Count: 1)
|
|----Valor: 1.9
|   Classe: Iris-setosa (Count: 2)
|
|----Valor: 4.7
|   Classe: Iris-versicolor (Count: 5)
|
|----Valor: 4.5
|   Atributo: sepallength
|   |
|   |----Valor: 6.4
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 5.7
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 5.6
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 6.2
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 6.0
|   |   Classe: Iris-versicolor (Count: 2)
|   |
|   |----Valor: 5.4
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 4.9
|   |   Classe: Iris-virginica (Count: 1)
|
|----Valor: 4.9
|   Atributo: sepalwidth
|   |
|   |----Valor: 3.1
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 2.5
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 2.8
|   |   Classe: Iris-virginica (Count: 1)
|   |
|   |----Valor: 2.7
|   |   Classe: Iris-virginica (Count: 1)
|   |
|   |----Valor: 3.0
|   |   Classe: Iris-virginica (Count: 1)
|
|----Valor: 4.0
|   Classe: Iris-versicolor (Count: 5)
|
|----Valor: 4.6
|   Classe: Iris-versicolor (Count: 3)
|
|----Valor: 3.3
|   Classe: Iris-versicolor (Count: 2)
|
|----Valor: 3.9
|   Classe: Iris-versicolor (Count: 3)
|
|----Valor: 3.5
|   Classe: Iris-versicolor (Count: 2)
|
|----Valor: 4.2
|   Classe: Iris-versicolor (Count: 4)
|
|----Valor: 3.6
|   Classe: Iris-versicolor (Count: 1)
```

```
|
|----Valor: 4.4
|   Classe: Iris-versicolor (Count: 4)
|
|----Valor: 4.1
|   Classe: Iris-versicolor (Count: 3)
|
|----Valor: 4.8
|   Atributo: sepallength
|   |
|   |----Valor: 5.9
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 6.8
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 6.2
|   |   Classe: Iris-virginica (Count: 1)
|   |
|   |----Valor: 6.0
|   |   Classe: Iris-virginica (Count: 1)
|
|----Valor: 4.3
|   Classe: Iris-versicolor (Count: 2)
|
|----Valor: 5.0
|   Atributo: sepallength
|   |
|   |----Valor: 6.7
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 5.7
|   |   Classe: Iris-virginica (Count: 1)
|   |
|   |----Valor: 6.0
|   |   Classe: Iris-virginica (Count: 1)
|   |
|   |----Valor: 6.3
|   |   Classe: Iris-virginica (Count: 1)
|
|----Valor: 3.8
|   Classe: Iris-versicolor (Count: 1)
|
|----Valor: 3.7
|   Classe: Iris-versicolor (Count: 1)
|
|----Valor: 5.1
|   Atributo: sepallength
|   |
|   |----Valor: 6.0
|   |   Classe: Iris-versicolor (Count: 1)
|   |
|   |----Valor: 5.8
|   |   Classe: Iris-virginica (Count: 3)
|   |
|   |----Valor: 6.5
|   |   Classe: Iris-virginica (Count: 1)
|   |
|   |----Valor: 6.3
|   |   Classe: Iris-virginica (Count: 1)
|   |
|   |----Valor: 6.9
|   |   Classe: Iris-virginica (Count: 1)
|   |
|   |----Valor: 5.9
|   |   Classe: Iris-virginica (Count: 1)
|
|----Valor: 3.0
|   Classe: Iris-versicolor (Count: 1)
|
|----Valor: 6.0
|   Classe: Iris-virginica (Count: 2)
|
|----Valor: 5.9
|   Classe: Iris-virginica (Count: 2)
|
|----Valor: 5.6
|   Classe: Iris-virginica (Count: 6)
|
|----Valor: 5.8
|   Classe: Iris-virginica (Count: 3)
|
|----Valor: 6.6
|   Classe: Iris-virginica (Count: 1)
|
|----Valor: 6.3
|   Classe: Iris-virginica (Count: 1)
|
|----Valor: 6.1
|   Classe: Iris-virginica (Count: 3)
|
|----Valor: 5.3
|   Classe: Iris-virginica (Count: 2)
|
|----Valor: 5.5
|   Classe: Iris-virginica (Count: 3)
|
|----Valor: 6.7
|   Classe: Iris-virginica (Count: 2)
|
```

|----Valor: 6.9
| Classe: Iris-virginica (Count: 1)
|
|----Valor: 5.7
| Classe: Iris-virginica (Count: 3)
|
|----Valor: 6.4
| Classe: Iris-virginica (Count: 1)
|
|----Valor: 5.4
| Classe: Iris-virginica (Count: 2)
|
|----Valor: 5.2
| Classe: Iris-virginica (Count: 2)

4.4 Classificação Individual

Para a classificação individual de um dado exemplo é possível obter vários inputs, sendo estes uma das classes possíveis, assim como uma mensagem de alerta juntamente com a classe prevista aquando de um valor que não se encontra no data set. Algumas das possibilidades encontram-se a seguir descritas, usando o data set weather.csv como exemplo.

Exemplo 1:

Terminal: Deseja classificar um exemplo? (Responda Y/N)
Input: y
Terminal: Quantos exemplos deseja classificar?
Input: 1
Terminal: Insira os exemplos, um de cada vez.
Use este formato: <ID>,<Weather>,<Temp>,<Humidity>,<Windy>
O programa vai prever o valor para a classe [Play].
Exemplo 1:
Input: 1,sunny,85,85,FALSE
Terminal: Classe: no

Exemplo 2:

Terminal: Deseja classificar um exemplo? (Responda Y/N)
Input: y
Terminal: Quantos exemplos deseja classificar?
Input: 1
Terminal: Insira os exemplos, um de cada vez.
Use este formato: <ID>,<Weather>,<Temp>,<Humidity>,<Windy>
O programa vai prever o valor para a classe [Play].
Exemplo 1:
Input: 3,sunny,90,90,FALSE
Terminal: Valor do atributo Temp não encontrado na árvore. Classe prevista: yes

5. Comentários Finais e Conclusões

Aprendizagem computacional e, em particular, as árvores de decisão, desempenham um papel fundamental no que toca à capacidade de máquinas na aprendizagem com dados e tomada de decisões inteligentes. Citando um grande cientista de Inteligência Artificial, Geoffrey Hinton, "As árvores de decisão são como um lego: estas permitem construir coisas maiores e mais complexas a partir de blocos simples e reutilizáveis". Assim, é realçada a simplicidade e flexibilidade das árvores de decisão como uma ferramenta poderosa na representação de conhecimento e tomada de decisões em diferentes domínios.

Ao longo do trabalho, explorou-se os mais diversos fundamentos das árvores de decisão, abrangendo a sua estrutura em forma de árvore, os testes para divisão de dados e a seleção de atributos relevantes, sendo que todo o conhecimento adquirido sobre o tópico foi projetado na sua implementação em Python. A referida implementação abrangeu a exploração das bibliotecas e ferramentas disponíveis na linguagem de programação escolhida, dado que foram usadas para construir, treinar e fazer previsões com base em modelos de árvore de decisão. Discutiu-se, ainda, algumas formas que poderiam conduzir a um melhor desempenho do modelo, o que iria contornar, deste modo, problemas como o *overfitting*, assim como a necessidade de avaliar adequadamente modelos, a partir da análise da curva de aprendizagem, uma vez que permite identificar possíveis problemas de BIAS ou variância e otimizar a sua capacidade de generalização para dados desconhecidos

Adicionalmente, abordou-se os conceitos de algoritmos indutivos, responsáveis por construir modelos de árvores de decisão a partir de dados de treino, finalizando com o aprofundamento individual dos mais populares algoritmos indutivos usados nesta abordagem, ID3, CART e C4.5. Destacou-se, nesse tópico, o inferir das regras e padrões a partir dos dados, técnicas de aprendizagem supervisionada, utilizadas pelos mesmos.

Para concluir, as árvores de decisão oferecem uma abordagem de fácil interpretação e eficiente no que toca à resolução de problemas de classificação e regressão. A compreensão dos princípios subjacentes aos algoritmos indutivos e a implementação adequada das árvores de decisão são elementos cruciais na exploração do potencial desses modelos e na aplicação com sucesso a uma ampla variedade de problemas no mundo real.

6. Referências Bibliográficas

Artificial Intelligence: A Modern Approach, 1st ed. by Stuart Russell and Peter Norvig (2009).

Artificial Intelligence: Foundations of Computational Agents Book by Alan Mackworth and David Lynton Poole (2010).

Avinash Navlani (02/2023). *Decision Tree Classification in Python Tutorial*. Consulta realizada a 01 de maio de 2023, a partir de https://rb.gy/3466j

Abhishek Sharma 44 (10/01/2023). *Python | Decision tree implementation.* Consulta realizada a 01 de maio de 2023, a partir de https://www.geeksforgeeks.org/decision-tree-implementation-python/

Eligijus112 (15/06/2021). *decision-tree-python*. Consulta realizada a 01 de maio de 2023, a partir de https://github.com/Eligijus112/decision-tree-python

milaan9 (09/12/2022). *Python_Decision_Tree_and_Random_Forest*. Consulta realizada a 01 de maio de 2023, a partir de https://github.com/milaan9/Python_Decision_Tree_and_Random_Forest

gaganmalhotra (23/05/2018). *Decision-Tree-Implementation-in-Python*. Consulta realizada a 01 de maio de 2023, a partir de https://github.com/gaganmalhotra/Decision-Tree-Implementation-in-Python

Lior Rokach, Oded Maimon - Department of Industrial Engineering (03/10/2009). *DECISION TREES*. Consulta realizada a 05 de maio de 2023, a partir de https://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf

Luis Torgo (03/10/2003). *Árvores de Decisão*. Consulta realizada a 20 de maio de 2023, a partir de https://www.dcc.fc.up.pt/~ltorgo/SebentaR/HTML/node25_ct.html

tirumalachandraveni (23/09/2022). *CART (Classification And Regression Tree) in Machine Learning*. Consulta realizada a 22 de maio de 2023, a partir de https://www.geeksforgeeks.org/cart-classification-and-regression-tree-in-machine-learning/

TheMainstreamSeer (13/01/2013). *Introduction to Classification & Regression Trees (CART)*. Consulta realizada a 26 de maio de 2023, a partir de http://themainstreamseer.blogspot.com/2013/01/introduction-to-classification.html