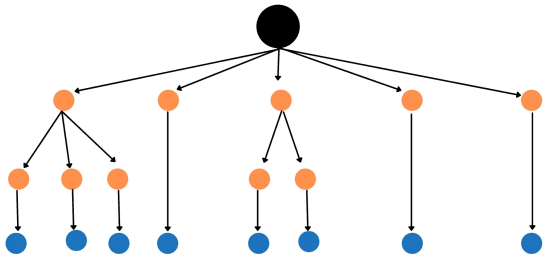
U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Faculdade de Ciências da Universidade do Porto

Relatório Final

Terceiro Trabalho de IA: Árvores de Decisão



Margarida Vila Chã, 202107923
Sebastião Santos Lessa, 202103238
Alexandre Marques, 202106956
28 de maio 2023, Porto

Índice

1. Introdução

O que é uma Árvore de Decisão?

Uma árvore de decisão é um modelo preditivo e interpretável muito utilizado na área de Ciência de Computadores e Inteligência Artificial. Uma árvore de decisão pode ser descrita como um algoritmo de aprendizagem supervisionada que constrói uma estrutura em forma de árvore para tomar decisões ou fazer previsões com base em um conjunto de dados variáveis de entrada.

Num contexto científico, uma árvore de decisão é construída a partir de um conjunto de dados de treino, onde cada instância contém um conjunto de características ou variáveis independentes e uma variável alvo ou dependente que se deseja prever. O algoritmo de construção da árvore analisa as características das instâncias de treino e, com base nessa análise, cria um modelo em forma de árvore, onde cada nó interno representa uma decisão com base numa variável, e cada ramo representa um possível valor dessa variável.

Para construir a árvore de decisão, são utilizados critérios de divisão, como o Gini Index ou Entropy, que medem a impureza ou a homogeneidade dos dados em relação à variável alvo. O algoritmo tem como objetivo encontrar as melhores divisões que maximizam a pureza das subpopulações resultantes em cada ramo da árvore.

Uma vez que a árvore de decisão esteja construída, é possível utilizá-la para fazer previsões ou tomar decisões com base nas características de uma nova instância. A nova instância é percorrida na árvore a partir do nó raiz até chegar a um nó folha, que contém a previsão ou a decisão final.

Para que serve uma Árvore de Decisão?

Uma árvore de decisão tem várias características que lhe permite ter várias funções. Seguidamente, encontram-se algumas das diferentes finalidades:

- ➔ **Tomada de Decisões**: Uma árvore de decisão pode ser usada para tomar decisões em vários contextos analisando as características relevantes de uma situação e seguindo um conjunto de regras para chegar a uma decisão. Por exemplo, num ambiente empresarial, este algoritmo pode ajudar a decidir se uma empresa deve investir num determinado projeto com base em critérios como retorno do investimento, custo e riscos.
- ➔ **Previsão**: As árvores de decisão também podem ser usadas para fazer previsões em diferentes áreas. Por exemplo, em finanças, elas podem ser aplicadas para prever o comportamento do mercado de ações com base em indicadores económicos. Da mesma forma, em medicina, uma árvore de decisão pode ajudar a prever a

probabilidade de um paciente desenvolver certa doença com base em seus sintomas, histórico médico e outros fatores.

- ➔ **Classificação**: Uma árvore de decisão pode ser usada para classificar instâncias em diferentes categorias ou grupos. Por exemplo, em análise de crédito, uma árvore de decisão pode ser construída para determinar se um cliente é de baixo risco ou alto risco com base na sua pontuação de crédito, histórico de pagamentos e outras variáveis relevantes.
- ➔ **Deteção de Padrões**: Árvores de decisão podem ser usadas para identificar padrões em conjuntos de dados complexos.

2. Algoritmos para indução de Árvores de Decisão

Algoritmos populares relacionados com Árvores de Decisão

Algoritmos de indução de árvores de decisão são técnicas e métodos utilizados para criar árvores de decisão a partir de dados de treino. O objetivo destes algoritmos é construir uma árvore de decisão precisa e eficiente que possa prever corretamente a variável alvo ou classe com base nas características fornecidas.

Estes algoritmos empregam diferentes estratégias para seleção de atributos, critérios de divisão, poda e tratamento de valores em falta, entre outros aspectos, para construir árvores de decisão que capturem eficazmente os padrões e relações nos dados. A escolha do algoritmo depende dos requisitos específicos do problema em questão e das características do conjunto de dados. Seguidamente, encontram-se alguns exemplos.

- ➔ **Algoritmo CART:**

O CART (Classificação e Regressão de Árvores) é um algoritmo amplamente utilizado na área de Inteligência Artificial e Ciência de Computadores para construir árvores de decisão que podem ser aplicadas tanto em problemas de classificação como de regressão. A abordagem do CART é baseada na construção recursiva de uma árvore binária, onde cada nó representa uma regra de divisão em uma determinada característica. A medida de impureza usada pelo CART é o Gini Index por default. O algoritmo realiza uma busca exaustiva por todas as possíveis divisões e seleciona aquela que resulta na maior diminuição do Gini Index, permitindo que o CART encontre as características mais discriminativas para classificar ou prever corretamente os dados.

Uma das vantagens do CART é sua capacidade de lidar com atributos numéricos e categóricos, discretizando as características numéricas para criar divisões binárias. No entanto, uma dificuldade do mesmo algoritmo é sua tendência a criar árvores profundas e complexas, o que pode levar ao overfitting e dificultar a interpretação dos resultados.

➔ **Algoritmo C4.5:**

O C4.5 é uma extensão do algoritmo ID3 e tem a capacidade de lidar com características tanto categóricas como contínuas.

Este algoritmo utiliza o ganho de informação normalizado para selecionar a melhor característica para divisão em cada nó da árvore. O ganho de informação normalizado leva em consideração o número de valores possíveis de uma característica, garantindo uma seleção equilibrada de características categóricas e contínuas. Além disso, o C4.5 introduz um mecanismo para lidar com dados ausentes durante a construção da árvore. Ele atribui pesos às amostras com dados ausentes e calcula o ganho de informação ponderado para evitar viés em direção a características com valores ausentes.

Após a construção da árvore, o C4.5 realiza uma etapa de poda pós-construção para evitar overfitting, removendo ramos da árvore que não contribuem significativamente para a melhoria da precisão. Essas características tornam o C4.5 um algoritmo robusto e amplamente utilizado.

➔ **Algoritmo Gradient Boosting:**

O Gradient Boosting é um algoritmo de aprendizagem que constrói árvores de decisão sequencialmente, com o objetivo de melhorar a precisão das previsões ao longo do tempo. A abordagem do Gradient Boosting é baseada em aprender a partir dos erros cometidos pelas árvores anteriores.

Começando com uma única árvore, o algoritmo avalia os erros residuais e cria uma nova árvore para corrigir esses erros. Essa árvore subsequente é ajustada para minimizar os erros residuais da árvore anterior. Em cada iteração, o Gradient Boosting adiciona uma nova árvore ao conjunto, onde cada árvore é treinada para capturar padrões nos erros residuais deixados pelas árvores anteriores. Esta abordagem sequencial de construção das árvores permite que o Gradient Boosting melhore gradualmente a precisão das previsões.

Uma das principais características do Gradient Boosting é sua capacidade de generalização. Cada árvore subsequente é ajustada para corrigir as deficiências das árvores anteriores, tornando o algoritmo mais capaz de capturar relacionamentos complexos nos dados. Além disso, o Gradient Boosting é robusto em relação a outliers e ruídos nos dados de treinamento, pois os erros residuais ajudam a direcionar o aprendizado em direção aos exemplos mais difíceis.

No entanto, o Gradient Boosting pode ser computacionalmente intensivo, pois requer o treinamento de várias árvores sequenciais. Cada árvore é construída de forma iterativa e depende dos resultados das árvores anteriores, o que pode aumentar o tempo de treinamento. Além disso, é importante ajustar os hiperparâmetros adequadamente para evitar overfitting. Um ajuste inadequado dos hiperparâmetros pode levar a um modelo super ajustado aos dados de treinamento.

➔ **Algoritmo Random Forest:**

O Random Forest é um algoritmo de aprendizagem que combina várias árvores de decisão para realizar classificação ou regressão. Cada árvore no Random Forest é treinada num subconjunto aleatório dos dados de treino, usando a técnica de bagging (bootstrap aggregating). Essa abordagem cria uma diversidade entre as árvores e reduz a variância do modelo final.

Durante o treino do Random Forest, para cada nó de divisão, apenas um subconjunto aleatório de características é considerado, o que ajuda a evitar a dependência de características específicas. A árvore resultante é uma combinação das previsões individuais de cada árvore no conjunto, geralmente por meio de votação majoritária no caso de classificação ou média no caso de regressão.

Uma das principais características do Random Forest é sua capacidade de lidar com grandes conjuntos de dados e alta dimensionalidade, além de ser robusto a outliers e ruídos. Também fornece uma estimativa da importância das características, permitindo a seleção das mais relevantes.

No entanto, uma das dificuldades na utilização do Random Forest é a interpretabilidade do modelo resultante, uma vez que é uma combinação de várias árvores. Além disso, o treino de múltiplas árvores pode exigir mais recursos computacionais em comparação com um único modelo de árvore de decisão.

Esses algoritmos de indução de árvores de decisão são amplamente utilizados na área de aprendizagem computacional e fornecem abordagens diferentes para a construção de modelos preditivos.

Quais as diferenças entre Algoritmos?

- **Estrutura e Funcionamento**: As árvores de decisão são estruturas em forma de árvore, onde cada nó representa uma decisão com base em uma variável. Já os Random Forests e o Gradient Boosting são algoritmos que combinam várias árvores de decisão em um único modelo. Já a estrutura do CART é baseada numa árvore hierárquica, onde cada nó representa uma decisão com base numa variável.

- **Método de Treino**: As árvores de decisão são construídas através de um processo de divisão recursiva, selecionando características com base em critérios como o Gini Index ou Entropy. O Random Forests cria várias árvores de decisão aleatórias a partir de subconjuntos diferentes dos dados de treino, e as previsões são feitas agregando as previsões individuais de cada árvore. O Gradient Boosting também constrói várias árvores de decisão, mas utiliza um processo de treino iterativo em que a divisão da nova

árvore tenta corrigir os erros das árvores anteriores. Por último, o CART e o C4.5 possuem métodos parecidos, um processo de divisão recursivo.

- **Tratamento de Dados Desequilibrados**: Os algoritmos baseados em árvores de decisão, como Random Forest e Gradient Boosting, tendem a lidar bem com conjuntos de dados desequilibrados, nos quais as classes possuem quantidades diferentes de exemplos. Eles podem equilibrar automaticamente a importância das classes durante o treino das árvores.

Diferenças Métricas entre Algoritmos

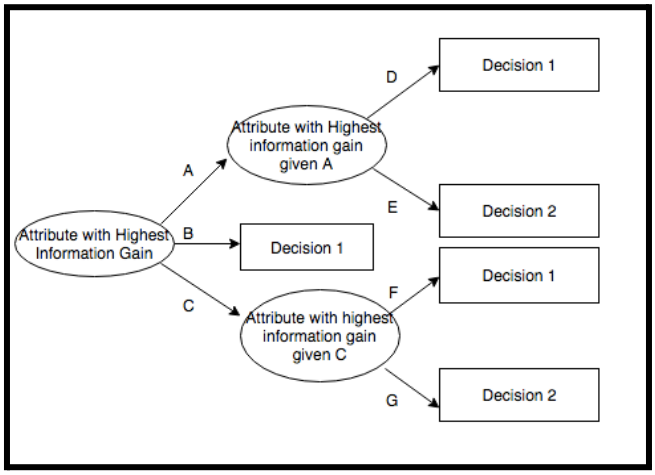
As diferenças métricas entre os algoritmos CART (Classification and Regression Trees), C4.5, Random Forest e Gradient Boosting são baseadas nas abordagens de construção de modelos e nas métricas usadas para avaliar a qualidade desses modelos. Seguidamente, existem alguns exemplos de métricas associadas a cada algarismo:

1. **CART (Classification and Regression Trees):**
    - A principal métrica usada para dividir os nós durante a construção da árvore é o Gini Index ou a entropia.
    - A qualidade do modelo é frequentemente avaliada usando a precisão (no caso de classificação) ou o erro médio quadrático (no caso de regressão).
2. **C4.5:**
    - Utiliza a entropia ou o índice de ganho de informação para decidir quais atributos usar para a divisão de nós.
    - Além da precisão e do erro médio quadrático, o C4.5 também pode usar outras métricas, como a taxa de erro de classificação.
3. **Random Forest:**
    - A métrica comum usada para avaliar a qualidade de um Random Forest é a precisão para problemas de classificação e o erro médio quadrático para problemas de regressão. Além disso, o Random Forest também pode fornecer importância relativa de atributos.
4. **Gradient Boosting:**
    - As métricas usadas para avaliar a qualidade do Gradient Boosting geralmente são o erro médio quadrático ou a log loss para problemas de classificação. Além disso, pode fornecer importância relativa de atributos.

Algoritmo ID3

Definição e como funciona?

O algoritmo ID3 é um algoritmo utilizado para construir árvores de decisão. Foi proposto por J. Ross Quinlan em 1986 e é uma abreviação para Iterative Dichotomiser 3.



A ideia central do ID3 é construir uma árvore de decisão a partir de um conjunto de dados de treino, onde cada exemplo é representado por um conjunto de atributos e uma classe ou valor a ser previsto. O objetivo é encontrar os atributos mais relevantes para a classificação ou previsão dos exemplos.

Este algoritmo segue uma abordagem top-down, recursivamente dividindo os dados com base em atributos, até que seja criada uma árvore completa. A construção da árvore ocorre de maneira iterativa, utilizando o conceito de ganho de informação para decidir qual atributo será utilizado para dividir os dados em cada nó da árvore.

O ganho de informação é uma medida que avalia a relevância de um atributo na classificação dos exemplos. É calculado a partir da entropia, que mede a impureza dos dados. Quanto maior o ganho de informação ao dividir os dados com base num determinado atributo, maior a relevância desse atributo na classificação.

Durante a construção da árvore, o ID3 escolhe o atributo com o maior ganho de informação para fazer a divisão em cada nó. Essa divisão cria ramos na árvore que representam os possíveis valores desse atributo. O processo é repetido recursivamente em cada ramo, até que sejam considerados todos os atributos ou que os exemplos de treino sejam classificados.

Uma vez que a árvore de decisão é construída, pode ser utilizada para classificar novos exemplos, percorrendo os ramos de acordo com os valores de seus atributos. Cada folha da árvore representa uma classe ou valor de previsão.

O algoritmo ID3 possui algumas limitações, como a tendência de criar árvores muito complexas ou sensíveis a ruídos nos dados de treino. No entanto, é um dos algoritmos mais conhecidos, importantes e serviu de base para o desenvolvimento de outras técnicas, como o algoritmo C4.5.

Quais as suas limitações?

O ID3, tal como todos os algoritmos, possui limitações específicas. Algumas das principais limitações são:

1. **Sensibilidade a atributos com muitos valores distintos**: O ID3 pode ter dificuldade em lidar com atributos que possuem muitos valores distintos, pois cada valor pode ser tratado como um ramo separado na árvore, levando a árvores complexas e potencialmente super ajustadas aos dados de treino.

2. **Incapacidade de lidar com atributos contínuos**: Como originalmente este algoritmo foi desenvolvido para atributos categóricos, não é capaz de lidar diretamente com atributos contínuos. É necessário realizar pré-processamento para discretizar os atributos contínuos em valores discretos ou utilizar variações do algoritmo, como o C4.5, que permite o tratamento de atributos contínuos.

3. **Tendência para Overfitting**: O ID3 tem uma tendência natural a criar árvores complexas que se ajustam muito bem aos dados de treino, o que pode levar a um desempenho reduzido em dados não vistos. Árvores demasiado ajustadas podem capturar ruídos ou variações irrelevantes nos dados de treino, resultando em baixa generalização para novos exemplos, e por isso, maus resultados finais.

4. **Ausência de poda (pruning)**: O ID3 não inclui um mecanismo de poda automática na construção da árvore, sendo que é uma técnica importante para evitar o sobreajuste, removendo ramos desnecessários da árvore que não contribuem significativamente para a melhoria da precisão do modelo. A ausência de poda no ID3 pode levar a árvores grandes e complexas, o que pode diminuir a eficiência computacional e a capacidade de generalização do modelo.

5. **Sensibilidade a ruído**: Tal como outros algoritmos, o ID3 não lida de forma robusta com ruídos ou com valores ausentes. A presença de ruído nos dados de treino pode levar a árvores incorretas ou menos precisas. Além disso, se um exemplo possuir um valor ausente, o ID3 não oferece uma abordagem padrão para lidar com essa situação.

3. Implementação

Linguagem utilizada

A linguagem utilizada foi Python devido a várias razões. Primeiramente, possuímos mais facilidade com esta linguagem e por isso ao escrever o código houve uma melhor compreensão, facilitando e simplificando mais o projeto. Seguidamente, como o primeiro e segundo trabalho desta cadeira realizamos em python, era apenas natural continuarmos e fazermos este segundo trabalho na mesma linguagem. Por último, apesar de python não ser a linguagem mais rápida, achamos que iria ser a melhor opção já que a probabilidade de o nosso código ser mais simples e eficiente iria ser maior, sendo esse facto um ponto positivo.

Estrutura de Dados utilizada

As estruturas utilizadas durante o desenvolvimento deste projeto foram principalmente classes e dicionários, pois achamos que desta maneira iríamos ter mais facilidade para escrever o código.

Organização do código

O nosso código está organizado em 2 módulos:

- **DecisionTree.py**: implementação de uma árvore de decisão em Python. Possui métodos para construir a árvore, realizar previsões, converter a árvore numa representação de string. Além disso, existe um método estático para calcular a precisão (score) de previsões em relação aos valores verdadeiros. A árvore de decisão é construída utilizando o algoritmo ID3 e realiza divisões com base em ganho de informação.
- **main.py**: módulo utilizado para correr todo o programa. Faz verificação se os parâmetros escritos no terminal estão corretos.

4. Resultados

Para cada dataset obtivemos as seguintes árvores com os respectivos contadores para cada exemplo nos seus ramos:

- restaurant:

```
Pat:
  Some: Yes  (4)
  Full:
    Hun:
      Yes:
        Type:
          French: No  (0)
          Thai:
            Fri:
              No: No  (1)
              Yes: Yes  (1)
          Burger: Yes  (1)
          Italian: No  (1)
      No: No  (2)
  None: No  (2)
```

- weather:

```
Temp:
  <= 83:
    Humidity:
      <= 86:
        Weather:
          sunny: yes  (2)
          overcast: yes  (3)
          rainy:
            Windy:
              False: yes  (2)
              True: no  (1)
      > 86:
        Weather:
          sunny: no  (2)
          overcast: yes  (1)
          rainy:
            Windy:
              False: yes  (1)
              True: no  (1)
  > 83: no  (1)
```

- iris:

```
petallength:
  <= 1.9: Iris-setosa  (50)
  > 1.9:
    petalwidth:
      <= 1.7:
        sepallength:
          <= 7.0:
            sepalwidth:
              <= 2.8: Iris-versicolor  (31)
              > 2.8: Iris-versicolor  (22)
          > 7.0: Iris-virginica  (1)
      > 1.7:
        sepallength:
          <= 5.9:
            sepalwidth:
              <= 3.0: Iris-virginica  (6)
              > 3.0: Iris-versicolor  (1)
          > 5.9: Iris-virginica  (39)
```

- connect4: como a árvore é muito complexa, decidimos colocar no *readMe.md* e não aqui.

5. Comentários Finais e Conclusões

Para confirmar que as árvores foram bem criadas, decidimos testar a precisão dos resultados obtidos pelo nosso *DecisionTreeClassifier*. Testamos utilizando os próprios *datasets* usados para treinar/criar as suas respectivas árvores. Observamos que, nos *datasets* "restaurant.csv" e "weather.csv", todos os resultados foram corretos, resultando numa precisão de 100%. Entretanto, para o conjunto de dados "iris.csv", a precisão foi ligeiramente inferior, atingindo 97,33%. Essa diferença pode ser atribuída ao facto do conjunto de dados "iris.csv" possuir um tamanho consideravelmente maior em relação aos outros dois conjuntos, bem como a presença de alguns outliers.

É válido destacar que os resultados obtidos demonstram um desempenho satisfatório, considerando que as árvores se adaptaram bem aos dados, sem se tornarem excessivamente complexas, independentemente da natureza dos atributos, sejam eles numéricos ou categóricos. Quanto aos atributos booleanos, decidimos realizar a transformação para valores categóricos, alterando True para "True" e False para "False". Essa abordagem foi adotada com o objetivo de simplificar e otimizar a previsão das classes num segundo conjunto de dados.

Em suma, os resultados alcançados demonstram a eficiência do nosso *DecisionTreeClassifier* na tarefa de previsão, com a capacidade de lidar adequadamente com diferentes tipos de atributos.

6. Referências Bibliográficas

1. Wikipedia. (s.d.). Algoritmo ID3. Obtido de https://pt.wikipedia.org/wiki/Algoritmo_ID3
2. GeeksforGeeks. (s.d.). CART (Árvore de Classificação e Regressão) em Aprendizagem de Máquina. Obtido de https://www.geeksforgeeks.org/cart-classification-and-regression-tree-in-machine-learning/
3. Wikipedia. (s.d.). Gradient boosting. Obtido de https://pt.wikipedia.org/wiki/Gradient_boosting
4. YouTube. (s.d.). Introdução a Árvores de Decisão e ao Algoritmo ID3. Obtido de https://www.youtube.com/watch?v=v6VJ2RO66Ag
5. Universidade do Porto. (s.d.). [Tarefa no Moodle]. Obtido de https://moodle.up.pt/mod/assign/view.php?id=126305
6. Universidade do Porto. (s.d.). [Moodle]. Obtido de https://moodle.up.pt/course/view.php?id=580
7. Chat.OpenAI. (s.d.). Obtido de https://chat.openai.com/
8. ThoughtCo. (s.d.). Definição de Entropia. Obtido de https://www.thoughtco.com/definition-of-entropy-604458