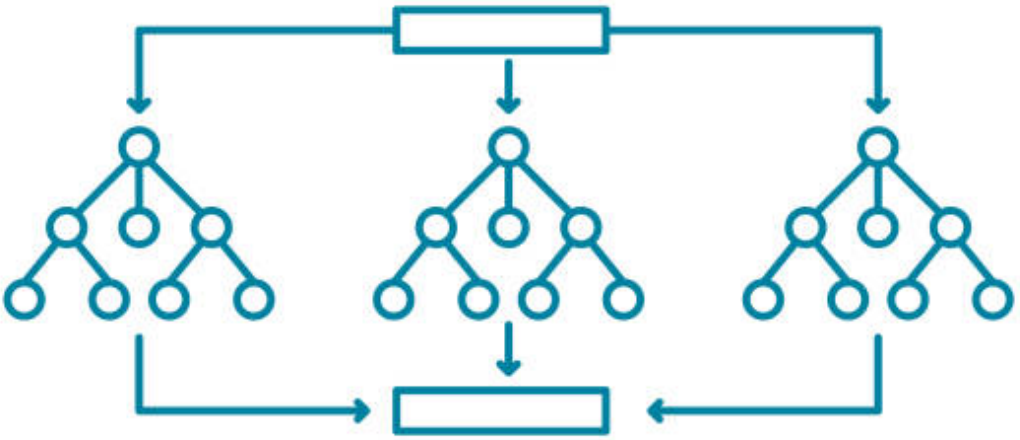
U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Árvores de decisão
Relatório Final



Unidade Curricular: Inteligência Artificial

Regentes: Inês Dutra

Francesco Renna

Elementos: Beatriz Marques de Sá - up202105831

Marisa Peniche Salvador Azevedo - up202108624

Marta Luísa Monteiro Pereira - up202105713

28 de maio de 2023

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Índice

1. Introdução

1.1. O que é uma árvore de decisão

Uma árvore de decisão é uma representação visual de uma tabela de decisão, organizada na forma de uma árvore. Esta abordagem é simples e fácil de implementar, tornando-a uma ferramenta útil em diversos contextos. Ao receber como entrada um objeto ou uma situação descrita por um conjunto de propriedades, a árvore de decisão produz um resultado binário, geralmente representado como "Sim" ou "Não".

Essa estrutura possibilita que um indivíduo compare diferentes ações com base nos seus custos, benefícios e probabilidades associadas. Ao seguir os caminhos na árvore, é possível visualizar as diferentes opções disponíveis e avaliar as consequências de cada escolha.

A construção de uma árvore de decisão começa com um único nó, conhecido como o nó raiz. A partir desse nó, a árvore divide-se em ramos representando os possíveis resultados ou decisões. Cada ramo cria nós adicionais, que por sua vez se ramificam em outras possibilidades. Esse processo continua até que se alcancem os nós finais, chamados de folhas, onde são apresentados os resultados ou conclusões.

As árvores de decisão são amplamente utilizadas em diversos campos, como ciência de dados, aprendizagem computacional, inteligência artificial, entre outros. A sua simplicidade e interpretabilidade são vantagens importantes, permitindo que especialistas e não especialistas compreendam e utilizem as informações geradas por essas árvores para tomar decisões informadas.

Em resumo, uma árvore de decisão é uma representação gráfica que possibilita a análise e comparação de possíveis ações com base nos seus custos, benefícios e probabilidades. A sua estrutura ramificada e intuitiva torna-a uma ferramenta valiosa para a tomada de decisões em diversos campos.

1.2. Para que serve?

As árvores de decisão são modelos de aprendizagem automática que ajudam no momento de tomar decisões em problemas complexos.

As árvores de decisão servem para analisar e classificar dados com base num conjunto de regras e critérios predefinidos. Estes modelos são especialmente úteis quando se lida com grandes e complexos conjuntos de dados, nos quais é necessário tomar decisões com base em múltiplas variáveis e condições.

Estas estruturas permitem uma análise sistemática e estruturada dos dados, oferecendo uma visão clara das diferentes opções e das suas implicações. Para além disso, as árvores de decisão podem ser interpretadas e compreendidas facilmente, tanto por especialistas da área como por pessoas sem conhecimentos especializados, devido à sua representação visual em forma de diagrama de árvore.

2. Algoritmos de indução de árvores de decisão

2.1. Algoritmos mais populares e as suas principais diferenças

2.1.1. Algoritmo CART

O algoritmo CART (Classification and Regression Trees) é um método de aprendizagem computacional utilizado para realizar tarefas de classificação e regressão. Ele cria uma árvore de decisão binária, onde cada nó interno representa um teste a um atributo e cada ramo representa o resultado do teste. Os nós folha da árvore contém as classes de destino para problemas de classificação ou valores de regressão para problemas de regressão.

O processo de construção da árvore CART começa com o conjunto de dados de treinamento e, em cada etapa, seleciona o melhor atributo para dividir os dados com base num critério de impureza, como o índice de Gini ou a entropia. A divisão é feita de forma a maximizar a pureza dos subconjuntos resultantes.

O índice de Gini é uma métrica utilizada no CART para tarefas de classificação. Este mede a impureza de um nó através do cálculo da soma das probabilidades ao quadrado para cada classe e quantifica a probabilidade de uma instância selecionada aleatoriamente ser classificada de forma incorreta em relação a uma variável específica. No CART, ele realiza divisões binárias, resultando em resultados rotulados como "sucesso" ou "falha".

O índice de Gini varia de 0 a 1, onde:

- Um valor de 0 indica que todos os elementos do nó pertencem a uma única classe ou que apenas uma classe existe.
- Um índice de Gini de 1 indica que os elementos do nó estão distribuídos aleatoriamente entre várias classes.
- Um valor de 0,5 sugere que os elementos estão uniformemente distribuídos entre diferentes classes.

2.1.2. C4.5

O algoritmo C4.5 é uma extensão e melhoria do algoritmo ID3. Este algoritmo é amplamente utilizado para construir árvores de decisão em problemas de aprendizagem computacional. O C4.5 aborda algumas limitações do ID3 e introduz recursos adicionais.

Construção inicial da árvore: Assim como o ID3, o C4.5 começa com o conjunto de dados original como o nó raiz da árvore de decisão.

Seleção de atributos: O C4.5 utiliza a medida de ganho de informação para selecionar o atributo que fornece a maior quantidade de informações relevantes para a classificação. Esta medida tem em conta o número de valores distintos do atributo e reduz o viés em relação aos atributos com muitos valores.

Tratamento de atributos contínuos: Ao contrário do ID3 o C4.5 pode lidar diretamente com atributos contínuos. Ele realiza uma discretização dos atributos contínuos, convertendo-os em atributos discretos usando técnicas como o particionamento binário ou o particionamento baseado em intervalos.

Tratamento de valores ausentes: O C4.5 possui mecanismos para lidar com valores ausentes nos dados. Ele calcula o ganho de informação ponderado para cada atributo, considerando os exemplos com valores ausentes e atribui um peso proporcional à probabilidade de cada valor ausente.

Poda da árvore: Após a construção inicial da árvore, o C4.5 realiza uma etapa de poda para evitar o overfitting. Ele avalia a importância estatística de cada subárvore por meio de testes estatísticos, como o teste do qui-quadrado, e poda as subárvores que não fornecem benefícios significativos em termos de desempenho.

Avaliação de confiança: O C4.5 atribui uma medida de confiança a cada instância classificada na árvore de decisão. Essa medida é baseada na

percentagem de instâncias da mesma classe no conjunto de treinamento que alcançam um determinado nó da árvore.

Assim, este algoritmo oferece melhorias em relação ao ID3, incluindo o tratamento de atributos contínuos, valores ausentes e a etapa de poda da árvore. Estas melhorias visam aumentar a precisão e a robustez do modelo gerado pela árvore de decisão.

2.1.3. ID3

O algoritmo ID3 (Iterative Dichotomiser 3) é um algoritmo de aprendizagem computacional usado para construir árvores de decisão a partir de conjuntos de dados. Ele é amplamente utilizado para problemas de classificação, onde o objetivo é atribuir uma classe ou categoria a uma instância com base em seus atributos.

O ID3 começa com o conjunto original S como nó raiz. É um algoritmo recursivo e em cada iteração do algoritmo, percorre todos os atributos não utilizados do conjunto S e calcula a entropia desse atributo. Em seguida, seleciona o atributo que possui o menor valor de entropia (ou o maior ganho de informação), e por isso usa uma estratégia greedy (ao selecionar o melhor atributo localmente para dividir o dataset em cada iteração). O conjunto S é então dividido ou particionado pelo atributo selecionado para produzir subconjuntos de dados.

Em suma, o ganho de informação calcula a eficácia de um atributo para classificar os seus valores. O atributo com maior ganho será o que tem mais eficácia e o que mais reduz a entropia. Logo, este atributo vai gerar uma árvore menos profunda, com menos nós e ligações.

O resultado final do algoritmo ID3 é uma árvore de decisão que pode ser usada para classificar novas instâncias, percorrendo os atributos na árvore até chegar a uma folha que representa a classe prevista.

Vantagens e desvantagens dos algoritmos

CART

Vantagens	Desvantagens
Resultados simplificados: Os resultados do algoritmo CART são fáceis de interpretar e compreender, pois são apresentados de forma simplificada em uma estrutura hierárquica.	Overfitting: Existe o risco de overfitting, quando a árvore de decisão se ajusta demais aos dados de treinamento e não consegue generalizar bem para novos dados.
Não paramétrico e não linear: Ao contrário de métodos estatísticos tradicionais, as árvores de classificação e regressão do CART não têm suposições sobre a distribuição dos dados ou a relação entre as variáveis, o que as torna flexíveis e adequadas para diferentes tipos de problemas.	Alta variância: O modelo pode ser sensível aos dados de treinamento, o que pode levar a grandes variações nas previsões quando os dados mudam. Isso pode resultar em um desempenho inconsistente do modelo.
Seleção implícita de recursos: O algoritmo CART decide quais atributos usar e como dividir os dados com base em critérios de impureza, selecionando automaticamente as variáveis mais importantes e relevantes.	
Robusto a outliers: As árvores de decisão do CART são robustas a valores extremos nos dados, pois conseguem se adaptar aos padrões gerais.	

C4.5

Vantagens	Desvantagens
Interpretabilidade: As árvores de decisão construídas pelo C4.5 são fáceis de entender e interpretar.	Sensibilidade a outliers: O C4.5 é sensível a valores discrepantes nos dados, o que pode impactar a precisão dos resultados.
Manipulação de atributos categóricos e numéricos: O C4.5 é capaz de lidar com diferentes tipos de dados, sejam eles categóricos ou numéricos.	Viés para atributos com muitos valores: O algoritmo tende a favorecer atributos com um grande número de valores, o que pode resultar em árvores de decisão mais complexas e propensas ao overfitting.
Lida com dados incompletos: O algoritmo é capaz de trabalhar com conjuntos de dados que possuem valores ausentes, sem a necessidade de pré-processamento dos dados.	Alto custo computacional: O tempo de execução do C4.5 pode ser elevado para conjuntos de dados grandes, aumentando o custo computacional.
Seleção automática de atributos: O C4.5 realiza a seleção automática de atributos relevantes, o que simplifica a etapa de pré-processamento dos dados.	Dificuldade na manipulação de dados contínuos: O C4.5 trata atributos numéricos como discretos, o que pode resultar em perda de informações importantes nos dados contínuos.

ID3

Vantagens	Desvantagens
As árvores de decisão geradas pelo ID3 são visualmente compreensíveis e podem ser facilmente interpretadas, o que torna o ID3 uma ótima escolha quando a interpretabilidade do modelo é uma prioridade.	O ID3 foi projetado para lidar principalmente com atributos categóricos, e não lida naturalmente com atributos contínuos. É necessário realizar uma discretização dos atributos contínuos antes de aplicar o ID3, o que pode introduzir perdas de informação.
O ID3 seleciona automaticamente os atributos mais relevantes para a tarefa de classificação, utilizando medidas como a entropia e o ganho de informação para determinar a importância dos atributos, o que facilita a identificação dos fatores que mais influenciam na tomada de decisão.	Como o algoritmo tem como objetivo otimizar o ganho de informação em cada etapa, ele pode facilmente ser influenciado por exemplos atípicos ou valores discrepantes, resultando em uma árvore de decisão menos generalizável.
O ID3 pode enfrentar dificuldades quando lida com atributos que possuem muitos valores distintos. A criação de partições para cada valor do atributo pode levar a uma árvore de decisão muito profunda e complexa, aumentando o risco de overfitting e tornando a árvore difícil de interpretar.	Dependendo das características do conjunto de dados, o ID3 pode gerar árvores de decisão muito profundas. Isso pode dificultar a interpretação da árvore e aumentar a complexidade do modelo, especialmente quando há atributos com muitos valores distintos ou quando há sobreajuste aos dados de treinamento.

2.2. Métricas

2.2.1. Ganho de informação

O ganho de informação é uma métrica utilizada nas árvores de decisão para medir a quantidade de informação obtida ao dividir um conjunto de dados com base em um determinado atributo. Ele é calculado pela diferença entre a entropia do conjunto de dados original e a entropia ponderada das partições resultantes da divisão.

O ganho de informação é calculado usando a entropia. A fórmula para calcular a entropia é a seguinte:

$$Entropia(S) = -\Sigma(p(i) \times log_2(p(i)))$$

Onde:

S - é o conjunto de dados.
$p(i)$ - é a proporção dos exemplos de classe i em relação ao conjunto de dados S.

O ganho de informação é calculado como a diferença entre a entropia do conjunto de dados inicial $Entropia(S)$ e a entropia ponderada dos subconjuntos resultantes da divisão em um atributo específico.

$$Ganho\ de\ Informação(A) = Entropia(S) - \Sigma((|Sv| / |S|) \times Entropia(Sv))$$

Onde:

A - é o atributo de divisão.
Sv - são os subconjuntos resultantes da divisão pelo atributo A.
$|Sv|$ - é o número de exemplos no subconjunto Sv.
$|S|$ - é o número total de exemplos no conjunto de dados S.

2.2.2. Razão de ganho

A razão de ganho é uma variação do ganho de informação que leva em consideração o ganho de informação intrínseco. O ganho de informação intrínseco é calculado usando a entropia dos possíveis valores do atributo A.

A fórmula para calcular o ganho de informação intrínseco é a seguinte:

$$Ganho\ de\ Informação\ Intrínseco(A) = -\Sigma((|Sv| / |S|) \times log_2(|Sv| / |S|))$$

Onde:

A - é o atributo de divisão.
$|Sv|$ - é o número de exemplos no subconjunto Sv.
$|S|$ - é o número total de exemplos no conjunto de dados S.

A razão de ganho é calculada dividindo o ganho de informação pelo ganho de informação intrínseco:

$$Razão\ de\ Ganho(A) = Ganho\ de\ Informação(A) / Ganho\ de\ Informação\ Intrínseco(A)$$

2.2.3. Gini

O índice de Gini é uma métrica que mede a impureza de um conjunto de dados, ou seja, a probabilidade de um elemento escolhido aleatoriamente ser classificado incorretamente ao ser rotulado de acordo com a distribuição de classes no conjunto. Nas árvores de decisão, o Gini é usado para avaliar a qualidade da divisão de um atributo.

O índice de Gini é calculado usando as proporções dos exemplos de classe em um conjunto de dados. A fórmula para calcular o índice de Gini é a seguinte:

$$Gini(S) = 1 - \Sigma((p(i))^2)$$

Onde:

S - é o conjunto de dados.
$p(i)$ - é a proporção dos exemplos de classe i em relação ao conjunto de dados S.

O índice de Gini é usado para medir a impureza de um conjunto de dados. Ao realizar a divisão em um atributo específico, é calculado o índice de Gini ponderado dos subconjuntos resultantes da divisão.

$$Gini(A) = \Sigma((|Sv| / |S|) \times Gini(Sv))$$

Onde:

A - é o atributo de divisão.
Sv - são os subconjuntos resultantes da divisão pelo atributo A.
$|Sv|$ - é o número de exemplos no subconjunto Sv.
$|S|$ - é o número total de exemplos no conjunto de dados S.

3. Implementação

3.1. Escolha da linguagem de programação utilizada

Para este trabalho, optamos por utilizar a linguagem Python, pois esta é uma linguagem de programação de código aberto, orientada a objetos e de alto nível. Assim, podemos usá-la sem nenhum custo adicional e temos acesso ao seu código fonte para personalizá-la de acordo com as nossas necessidades.

Para além disso, Python é uma linguagem bastante popular com uma ampla disponibilidade de bibliotecas que nos pode ajudar na análise e no processamento dos dados recolhidos.

Outra vantagem é existir uma grande comunidade de desenvolvedores nesta linguagem de programação, o que significa que podemos encontrar facilmente suporte e documentação para resolver problemas e dúvidas que possam surgir durante o desenvolvimento do projeto.

Em suma, consideramos que a escolha de Python para a implementação dos algoritmos é uma escolha sólida.

3.2. Estruturas de dados usadas

As estruturas utilizadas foram listas, dicionários e tuplos, que desempenham papéis importantes no código fornecido.

Listas foram utilizadas para armazenar diferentes conjuntos de dados, como a matriz de dados do conjunto de treino (“data_matrix”), a lista de rótulos (“labels”), os valores únicos de um determinado atributo (“unique_values”) e subconjuntos de dados (“subset”). Estas listas são cruciais para realizar iterações, filtragens e manipulações dos dados ao longo do código.

Os dicionários desempenham um papel fundamental no armazenamento e acesso eficiente de informações. O dicionário attributes mapeia o nome do atributo para a sua posição na matriz de dados, permitindo um acesso mais direto aos atributos.

O dicionário “reverse_attributes” realiza o mapeamento inverso, associando a posição do atributo ao seu nome. O dicionário transformations é usado para armazenar os intervalos de cada coluna (variável) por ordem crescente. Além disso, os dicionários “count_classes” e “class_counts” são utilizados para contabilizar a frequência das classes nos dados.

Tuplos são usados para representar intervalos de valores em combinação com as suas classes correspondentes. O tuplo “intervalo” armazena os valores mínimo e máximo de um intervalo, juntamente com a classe correspondente.

Estas estruturas de dados desempenham papéis específicos no código, permitindo a manipulação, organização e armazenamento eficiente dos dados, facilitando assim a construção da árvore de decisão.

3.3. Organização do código

A função principal **main()**:

- Abre o ficheiro.csv pretendido.
- Cria uma lista vazia, chamada *“data_matrix”* onde será armazenado o Conteúdo do ficheiro .csv.
- Cria um dicionário vazio, chamado *“attributes”* que será usado para mapear o nome de cada variável (coluna) para sua posição (índice) na matriz de dados.
- O nome da última coluna é armazenado em *“label_class”*.
- Fecha o ficheiro.csv.
- Cria um objeto Tree chamado “*tree*”, passando *“data_matrix”*, *“attributes”* e *“label_class”* como argumentos para o construtor da classe Tree. Isso cria uma árvore de decisão com base na matriz de dados, nos atributos e na classe.
- A árvore de decisão é impressa no formato ID3.

Classe **Node**:

- Esta classe representa um nó da árvore de decisão.

Construtores:

- *"attribute"*: Atributo do nó.
- *"label"*: Classe a que o nó pertence.
- *"is_leaf"*: Indica se o nó é uma folha.
- *"count"*: Dicionário que armazena o número de ocorrências de cada classe.
- *"children"*: Dicionário que armazena os filhos do nó.

Métodos:

- *recursive(self, node, indent)*: Método auxiliar para representação da árvore em string..
- *__str__(self)*: Retorna a representação em string do nó.

Classe **Tree**:

- Representa a árvore de decisão.

Construtores:

- *"dataset"*: Matriz de dados inicial.
- *"classe"*: Nome da coluna que representa a classe.
- *"entropy_class"*: Entropia da classe.
- *"entropy_attributes"*: Dicionário que armazena a entropia de cada atributo.
- *"attributes"*: Dicionário que mapeia os atributos para suas posições na matriz de dados.
- *"reverse_attributes"*: Dicionário invertido do attributes para facilitar a tradução de posição para nome do atributo.
- *"root"*: Raiz da árvore de decisão.

Métodos:

- *build_tree(self, data_matrix, attributes)*: Constrói a árvore de decisão recursivamente.
- *most_common_class(self, labels)*: Retorna a classe mais comum.
- *calcule_best_split(self, data_matrix, best_attribute):* Returna o melhor valor para dividir os valores
- *split_info_gain(self, dataset, label_menor, label_maior):* Returna o ganho de informação
- *entropy(self, data_label)*: Calcula a entropia de um conjunto de classes.
- *attribute_entropy(self, data_matrix)*: Calcula a entropia de cada atributo.
- *information_gain(self, attributes)*: Calcula o ganho de informação para cada atributo e retorna o atributo com maior valor de ganho de informação.
- *_count_classes(self, dataset)*: Conta o número de ocorrências de cada classe em um conjunto de dados.
- *__str__(self)*: Retorna a representação em string da árvore de decisão.

4. Resultados

4.1. restaurant.csv

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

4.2. weather.csv

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

4.3. iris.csv

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

5. Comentários Finais e Conclusões

Com a implementação do algoritmo ID3, compreendemos de forma mais aprofundada o processo detalhado de organização da informação contida nas tabelas de decisão. Essa compreensão nos permitiu extrair informações úteis por meio da criação de árvores de decisão correspondentes, as quais são fundamentais para a tomada de decisões finais.

Durante nossa pesquisa, exploramos diversos algoritmos de indução de árvores de decisão, o que nos proporcionou uma visão mais abrangente das diferentes abordagens disponíveis para operar sobre tabelas e árvores. Descobrimos que alguns desses métodos, como o C4.5, são mais otimizados em termos de desempenho, enquanto outros, como o CART, apresentam resultados inferiores.

Em resumo, ao finalizar todas as nossas investigações acerca dos algoritmos de árvores de decisão e ao implementar o algoritmo ID3, adquirimos um entendimento

mais profundo do processo de organização da informação nas tabelas de decisão. Essa experiência permitiu obter informações valiosas a partir da criação das respectivas árvores de decisão, que desempenham um papel crucial na tomada de decisões finais.

Quanto ao desenvolvimento do projeto, na nossa opinião, foi um trabalho com um nível de dificuldade maior que os anteriores. Infelizmente não conseguimos acabar o trabalho com tudo o que foi pedido, mas achamos que a nossa postura foi positiva.
No entanto, achamos importante mencionar que encontramos alguns obstáculos como o tempo para desenvolver o projeto e a maneira como foi feita a divisão de trabalho, que não foi a melhor.

6. Webgrafia e Bibliografia

WebGrafia:

http://web.tecnico.ulisboa.pt/ana.freitas/bioinformatics.ath.cx/bioinformatics.ath.cx/indexf23d.html?id=199
https://pt.wikipedia.org/wiki/%C3%81rvore_de_decis%C3%A3o
https://pt.wikipedia.org/wiki/Aprendizagem_de_%C3%A1rvore_de_decis%C3%A3o
https://medium.com/@cm.oeiras01/compreendendo-algoritmos-de-heur%C3%ADsticas-id3-15b0a75d26ef

Bibliografia:

Dissertação por Carine Halmenschlager:
https://www.lume.ufrgs.br/bitstream/handle/10183/2755/000325797.pdf?sequence=1
Russell, S. J., & Norvig, P. (2010). Artificial intelligence: a modern approach.
Elaine Rich and Kevin Knight (2017). Artificial Intelligence.