INSTITUTO FEDERAL DE EDUCAÇÃO,
CIÊNCIA E TECNOLOGIA GOIANO
Campus URUTAÍ
NÚCLEO DE INFORMÁTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

KAROLAYNE RODRIGUES RAMOS DOS SANTOS

CLASSIFICAÇÃO COM ÁRVORES DE DECISÃO EM PARALELO

Urutaí, 15 de Março de 2021

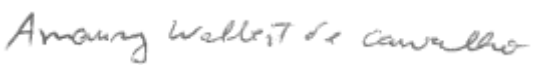
KAROLAYNE RODRIGUES RAMOS DOS SANTOS

CLASSIFICAÇÃO COM ÁRVORES DE DECISÃO EM PARALELO

Monografia, defendida por Karolayne Rodrigues Ramos dos Santos, apresentado ao Instituto Federal de Educação, Ciência e Tecnologia Goiano, como parte das exigências para a obtenção do título de Bacharel em Sistemas de Informação, aprovados pela banca examinadora.

COMISSÃO EXAMINADORA

Prof. Dr. Júnio César de Lima
Orientador



Prof. Me. Amaury Walbert de Carvalho
Avaliador

Profa. Dra. Cristiane de Fátima dos Santos Cardoso
Avaliadora

Urutaí (GO), 15 de março de 2021.

Sistema desenvolvido pelo ICMC/USP
Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas - Instituto Federal Goiano

Santos, Karolayne
S237c Classificação com Árvores de Decisão em Paralelo / Karolayne Santos; orientador Dr. Júnio César de Lima; co-orientador Me. Julio Cesar Batista Pires. -- Urutaí, 2021.
30 p.

Monografia (Graduação em Sistemas de informação) -- Instituto Federal Goiano, Campus Urutaí, 2021.

1. Aprendizado de máquina. 2. árvores de decisão. 3. classificação. 4. programação paralela. I. Lima, Dr. Júnio César de, orient. II. Pires, Me. Julio Cesar Batista, co-orient. III. Título.

Responsável: Johnathan Pereira Alves Diniz - Bibliotecário-Documentalista CRB-1 n°2376

INSTITUTO FEDERAL
Goiano

Repositório Institucional do IF Goiano - RIIF

Goiano

Sistema Integrado de Bibliotecas

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano, a disponibilizar gratuitamente o documento no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

Identificação da Produção Técnico-Científica

[ ] Tese	[ ] Artigo Científico
[ ] Dissertação	[ ] Capítulo de Livro
[ ] Monografia – Especialização	[ ] Livro
[X] TCC - Graduação	[ ] Trabalho Apresentado em Evento
[ ] Produto Técnico e Educacional - Tipo: _____	

Nome Completo do Autor: Karolayne Rodrigues Ramos dos Santos
Matrícula: 201611202010127
Título do Trabalho: Classificação com Árvores de Decisão em Paralelo

Restrições de Acesso ao Documento

Documento confidencial: [X] Não [ ] Sim, justifique: _____

Informe a data que poderá ser disponibilizado no RIIF Goiano: 09/04/2021
O documento está sujeito a registro de patente? [ ] Sim [X] Não
O documento pode vir a ser publicado como livro? [ ] Sim [X] Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

1. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
2. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
3. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Uruta, 09/04/2021
Local Data

Karolayne R. Ramos dos Santos
Assinatura do Autor e/ou Detentor dos Direitos Autorais

Ciente e de acordo:

_____

Assinatura do(a) orientador(a)

INSTITUTO FEDERAL GOIANO – CAMPUS URUTAÍ
DIRETORIA / GERÊNCIA DE GRADUAÇÃO E PÓS-GRADUAÇÃO
COORDENAÇÃO DOS CURSOS DA ÁREA DE INFORMÁTICA
CURSO SUPERIOR DE SISTEMAS DE INFORMAÇÃO

ATA DE APRESENTAÇÃO DE TRABALHO DE CURSO

Aos quinze dias do mês de março de dois mil e vinte e um, reuniram-se os professores: Júnio César de Lima, Amaury Walbert de Carvalho e Cristiane de Fátima dos Santos Cardoso nas dependências do Instituto Federal Goiano - Campus Urutaí, para avaliar o Trabalho de Curso do(s) acadêmico(s): **Karolayne Rodrigues Ramos dos Santos**, como requisito necessário para a conclusão do Curso Superior de Sistemas de Informação desta Instituição. O presente TC tem como título: **CLASSIFICAÇÃO COM ÁRVORES DE DECISÃO EM PARALELO,** foi orientado por Júnio César de Lima.

Após análise, foram dadas as seguintes notas:

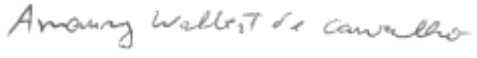
Professores	**Aluno / Notas**
	Karolayne Rodrigues Ramos dos Santos
1. Júnio César de Lima	9,1
2. Amaury Walbert Carvalho	8,1
3. Cristiane de Fátima dos Santos Cardoso	9,0

MÉDIA FINAL:	**8,8**

OBSERVAÇÕES: aprovação condicionada a realização das correções sugeridas pela banca e entrega da versão corrigida no prazo determinado pela coordenação de TC e/ou de curso.

Por ser verdade firmamos a presente:

Júnio César de Lima 

Amaury Walbert de Carvalho 

Cristiane de Fátima dos Santos Carvalho 

RESUMO

Este trabalho apresenta uma síntese do funcionamento de um algoritmo de classificação de dados, que classifica itens ou amostras de acordo com as características adquiridas por meio de treinos realizados sobre uma base de dados. Esses treinos são feitos utilizando modelos de árvores de decisão como ferramenta para analisar os dados. O processo de classificação pode ser aplicado quando o objetivo é prever ocorrências futuras, como por exemplo para uma empresa, na melhora no plano de vendas ou no relacionamento com os clientes. O projeto teve como resultado a demonstração de funcionamento de um classificador serial e um paralelo, apresentando sua metodologia, estrutura, o modelo matemático que o compõem e suas aplicações. Também foi feito a comparação dessas duas perspectivas diferentes de construção de software (serial e paralelo), constatando que, no algoritmo paralelo houve uma redução de tempo de execução considerável comparado ao serial e evidenciando quando esse ganho pode ser vantajoso em uma aplicação.

Palavras-chave:

Aprendizado de máquina, árvores de decisão, classificação, programação paralela.

This work presents a synthesis of the functioning of a data classification algorithm, which classifies items or samples according to the characteristics acquired through training carried out on a database. These trainings are done using decision tree models as a tool to analyze the data. The classification process can be applied when the objective is to predict future occurrences, such as for a company, in improving the sales plan or in the relationship with customers. The project resulted in the demonstration of the functioning of a serial and a parallel classifier, presenting its methodology, structure, the mathematical model that composes it and its applications. A comparison was also made of these two different perspectives of software construction (serial and parallel), noting that, in the parallel algorithm, there was a considerable reduction in execution time compared to the serial and showing when this gain can be advantageous in an application.

Key-words: Machine Learning, decision trees, classification, parallel programming.

LISTA DE FIGURAS

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

Na sociedade de informação moderna, a análise de dados é um termo que vem sendo cada vez mais discutido. Segundo He e Garcia (2009) os avanços da tecnologia possibilitaram maior disponibilidade e amplificação de dados. Com isso a importância dos dados como ferramenta de auxílio à gestão e na tomada de decisão para as empresas ganhou muita importância. Para estas aplicações, utiliza-se diversos recursos da tecnologia da informação como por exemplo, o Aprendizado de Máquina, cujo estudo é o desenvolvimento de técnicas computacionais com o propósito de implementar algoritmos capazes de "aprender" de forma autônoma, interpretando dados específicos. Um dos ramos do Aprendizado de Máquina é a Classificação, que pertence ao grupo de aprendizado supervisionado, que é muito utilizado na análise de dados. Segundo a definição de Nasridinov, Lee e Park (2014, p. 405) "A Classificação categoriza os objetos em suas devidas classes, tendo como exemplo um conjunto de objetos e os colocando em grupos, baseando nas suas similaridades."(NASRIDINOV; LEE; PARK, 2014)

A classificação pode ser aplicada tanto em coisas simples, como por exemplo um detector de spam, onde todas as palavras relacionadas com emails do tipo spam são colhidas a fim de alimentar o algoritmo para que ele seja capaz de prever futuros emails indesejados. Como também em pesquisas mais profundas, por exemplo, na pesquisa sobre diabetes, que utiliza uma base de dados de pacientes com pré diabetes para então predizer o diagnóstico de pessoas com diabetes ajudando no retardamento ou tratamento da doença.

As árvores de decisão são os algoritmos mais práticos e usados em inferência indutiva, utilizados para a classificação e previsão de dados". Estes modelos utilizam a estratégia de

dividir para conquistar: um problema complexo é decomposto em subproblemas mais simples e, recursivamente essa técnica de divisão é aplicada a cada subproblema gerado"(GAMA, 2009). Entretanto, esses algoritmos demandam muito tempo e poder de processamento a algumas aplicações, tais como processamento de vídeo, análises sísmicas, simulações de dinâmica dos fluidos e previsão do tempo, uma vez que esses problemas geram enormes quantidades de dados. (RUDAKOV, 2019), (WRONA et al., 2018),(Hasan; Uddin; Chowdhury, 2016).

O propósito geral desse trabalho é expor o funcionamento de um algoritmo de classificação, utilizando o modelo de árvores de decisão para estruturalização, a fim de demonstrar seus benefícios nesse tipo de aplicação e também a construção desse mesmo algoritmo mas utilizando a abordagem da paralelização para demonstrar alguns recursos da CPU. As abordagens (sequencial e paralela) serão comparadas a fim de analisar o desempenho dos dois algoritmos e explorar as vantagens e aplicações que cada um teve.

Todas as definições e temáticas referenciadas no projeto estão retratadas no capítulo Fundamentos Teóricos que contém todas as pesquisas e embasamento teórico desse projeto. Na seção Aprendizado de Máquina será explicado a definição do tema, sendo subdividos em outros duas seções, classificação e árvores de decisão, juntamente com seus conceitos e aplicações. Na seção Programação Paralela será explicada a definição, exemplos e como seus recursos foram utilizados no projeto. O capítulo Desenvolvimento conta com 3 seções, Dataset, Algoritmo Sequencial e Algoritmo Paralelo. A seção Dataset foi encarregada de descrever a definição de um dataset a apresentar o conjunto de dados escolhido, assim como esclarecimento de seus dados e importância. As duas últimas seções, Algoritmo Sequencial e Algoritmo Paralelo, descrevem as ferramentas, técnicas, e detalhamento de cada etapa de implementação dos algoritmos. No capítulo resultados é exposto os resultados obtidos e os métodos escolhidos para avaliação dos testes realizados. Por fim temos a conclusão onde serão descritas as expectativas, pontos positivos e negativos e sugestões de continuação do projeto para projetos futuros.

CAPÍTULO 2

FUNDAMENTOS TEÓRICOS

2.1 Aprendizado de Máquina

O termo Aprendizado de Máquina - AM (*Machine Learning*) foi criado por Artur Samuel em 1959, ele afirmava que "o aprendizado de máquina é um campo de estudo que dá aos computadores a capacidade de aprender sem precisar ser programado para tal" (Samuel, 1959). Em outras palavras o AM é a vertente da inteligência artificial que permite aos computadores maior flexibilidade, assertividade e autonomia em suas tomadas de decisões, isso porque se trata de um processo que aprende por experiência e análise e busca melhorar suas performances de maneira evolutiva. Dado uma linguagem de descrição usada para expressar possíveis hipóteses, um conhecimento preliminar, um conjunto de exemplos positivos, e um conjunto de exemplos negativos, Aprendizado Indutivo de Conceitos pode ser entendido como o problema de se encontrar uma hipótese que cubra todos os exemplos positivos e não cubra nenhum dos exemplos negativos. Dentre os paradigmas de aprendizado, existem duas no ramo do aprendizado indutivo: supervisionado e não-supervisionado.

Segundo a definição de (REZENDE; MONARD; CARVALHO, 1999) "No supervisionado é fornecido ao algoritmo de aprendizado, ou indutor, um conjunto de exemplos de treinamento para os quais o rótulo da classe associada é conhecido.[...]. Já no aprendizado não-supervisionado, o indutor analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters. Problemas de aprendizagem supervisionados são classificados em problemas de "regressão" e "classificação".

Classificação é a forma de mapear elementos iguais em categorias específicas como na classificação de emails como "spam" e "não spam". Já a regressão é responsável por induzir

conceitos a partir de exemplos que estão pré-classificados, ou seja, exemplos que estão rotulados com uma classe conhecida. Se as classes possuem valores discretos, o problema é categorizado como classificação. Caso as classes possuam valores contínuos, o problema é categorizado como regressão.

Será utilizada a classificação neste projeto, pois necessitava-se de um algoritmo que tenha a capacidade de prever a partir de valores binários para definir o rótulo de classe e não identificar uma categoria em escala contínua. Trata-se de uma abordagem mais simples e que produz insumos suficientes para justificar a proposta de otimização de tempo por meio da paralelização.

2.1.1 Classificação

A classificação é uma subcategoria de aprendizagem supervisionada, ou seja é técnica de obter dados de entrada e tentar determinar corretamente a classe de novos exemplos ainda não rotulados. A Figura 2.1 detalha um modelo de construção de um algoritmo de classificação.

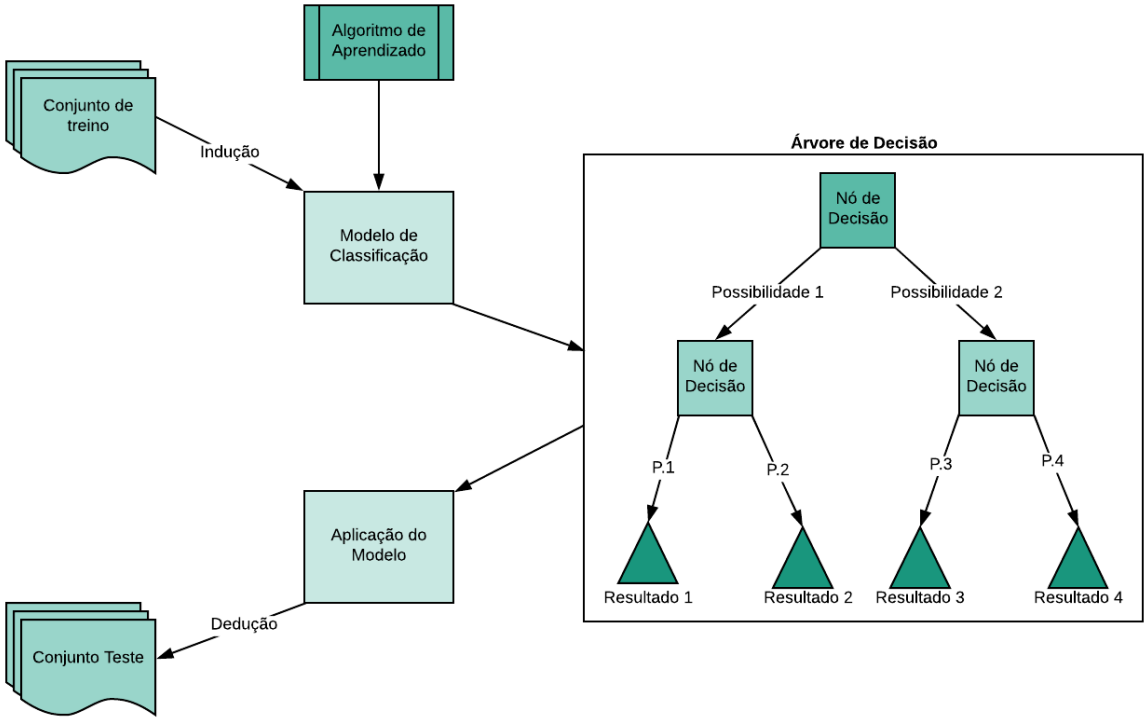


Figura 2.1: Abordagem Geral para construir um modelo de classificação
Fonte: Adaptado de Introduction to data mining, p.148 (TAN; STEINBACH; KUMAR, 2006)

Para gerar um modelo de classificação é necessário realizar a indução. Ela representa a inferência de conhecimento a partir de um conjunto de treino, em que as informações desse

conjunto são analisadas a fim de encontrar tendências e padrões. Esses padrões são objetos com características similares, que logo serão agrupados em classes e formuladas regras para prever a classe dos objetos analisados futuramente. Esse processo e as informações obtidas nele resultam no algoritmo de aprendizado. Para estruturar o algoritmo de classificação e testar os novos dados, existem alguns modelos/técnicas de classificação são eles, KNN, SVM, Regressão Logística e Árvores de Decisão. O modelo escolhido para esse projeto, como representa a Figura 2.1 foi a árvore de decisão. A definição e justificativa dessa escolha serão apresentadas na próxima seção.

2.1.2 Árvores de Decisão

Uma árvore de decisão é uma árvore onde cada nó interno (não terminal) representa um teste ou decisão sobre o item de dado considerado (GOEBEL; GRUENWALD, 1999). Seu sucesso pode ser explicado por diversos fatores, como (TAN; STEINBACH; KUMAR, 2005) cita:

- interpretabilidade do conhecimento adquirido - uma árvore de decisão usa uma representação gráfica e pode ser facilmente convertida em regras;
- robustez na presença de ruídos;
- algoritmos de indução de árvores de decisão não são computacionalmente custosos, mesmo para grandes conjuntos de dados;
- capacidade de lidar com atributos redundantes e irrelevantes, os quais, se não tratados adequadamente, podem prejudicar o desempenho do classificador;

Alguns algoritmos bem conhecidos para indução de árvores de decisão são o ID3 (QUINLAN, 1986), o C4.5 (QUINLAN, 2014) e o CART (MOLA, 1998). O algoritmo de base escolhido para esse projeto foi o ID3, que usa a abordagem top-down em que o melhor atributo é selecionado e usado como teste na raiz da árvore.

No geral as árvores de decisão podem ser explicadas como uma série de declarações *if-elses*, que quando aplicados a um registro de uma base de dados, resultam na classificação daquele registro. O mais interessante sobre o programa de árvores de decisão não é a sua construção a partir de classificação de um conjunto de treinamento, e sim a sua habilidade de aprendizado. Quando o treinamento é finalizado, é possível armazenar os dados na árvore de decisão construída a partir de exemplos com novos casos a fim de classificá-los.

Um exemplo de árvore de decisão pode ser visto na Figura 2.2, que demonstra a decisão de se jogar ou não tênis, baseado nas caraterísticas representadas.

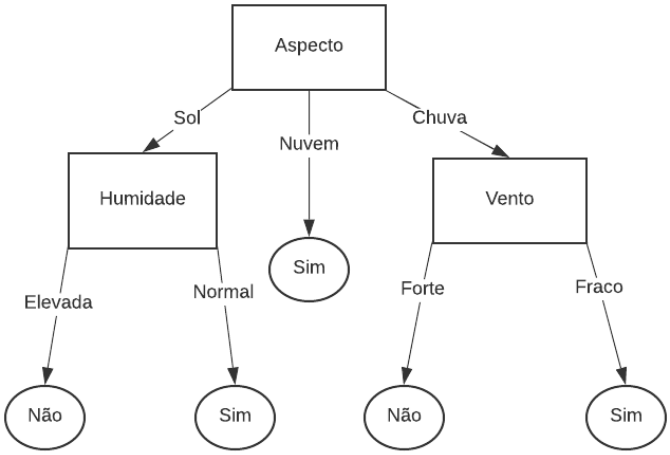


Figura 2.2: Exemplo de Árvore de Decisão
Fonte: Adaptado de Machine Learning (MITCHELL, 1997)

Cada retângulo representa um nó que será um atributo. Esse nó que pode ser dividido para gerar outros nós. Essas divisões são os possíveis valores que ele pode ter. Para demonstrar as divisões temos as setas representando cada possível valor do atributo. Já os círculos são os nós-folhas que não possuem nós filhos e representam classe/resultado da amostra.

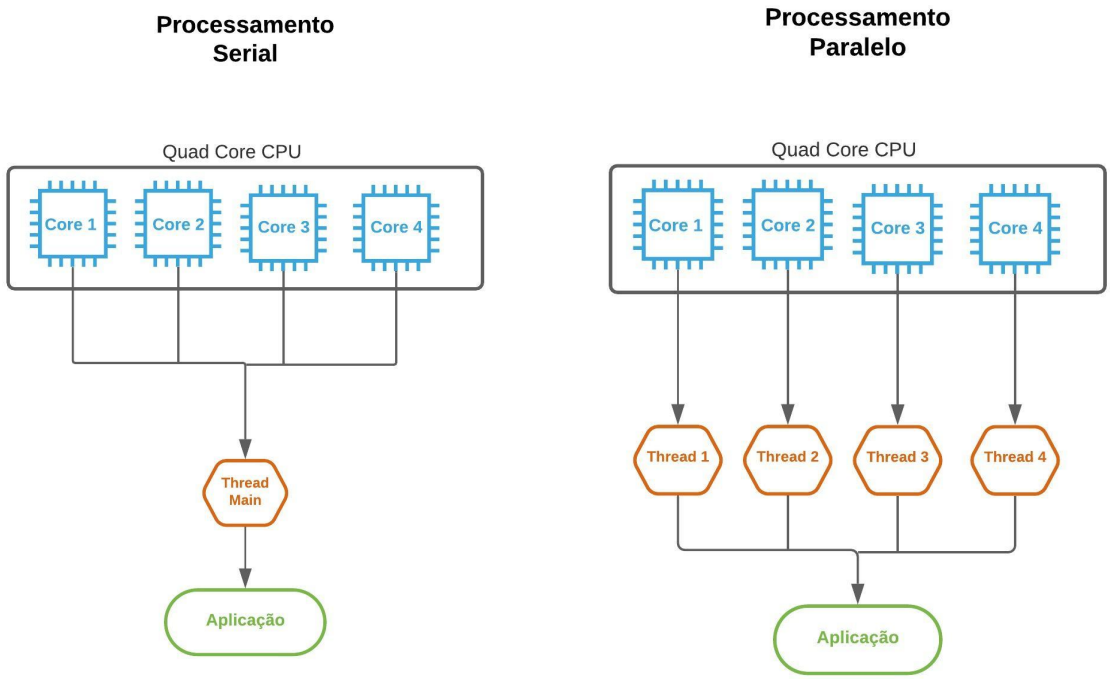
Na Figura 2.2, temos o nó "Aspecto"como raiz da árvore tendo 3 ramificações (sol, nuvem e chuva) que resultaram em 3 outros nós filhos. Esses nós podem assumir o valor de atributo (retângulo) ou de folha (círculo). Pode-se observar que se o atributo "aspecto"resulta na ramificação de valor "nuvem", essa condição é decisiva o bastante para designar chegar em um nó folha cujo valor é "sim"evidenciando que o jogador poderia jogar tênis sob aquela condição. As ramificações de valor "sol"e "chuva"resultaram nos nós "humidade"e "vento", cada qual também tiveram seus nós-filhos. Isso acontece porque mais atributos tiveram que ser analisados para chegar a um resultado ou nós-folhas.

2.2 Programação Paralela

Programação paralela consiste em dividir uma determinada aplicação em partes e, utilizando recursos como comunicação e sincronização, elas possam ser executadas simultaneamente por meio de vários elementos de processamento específicos e previamente configurados, tendo como consequência o ganho de desempenho quando comparado aos programas sequenciais.

Por meio dos núcleos de uma CPU, cada tarefa poderá ser designada a um número específico de threads - responsável por processar essas tarefas. Elas podem trocar dados e informações entre si e compartilhar os mesmos recursos do sistema, incluindo o mesmo espaço de memória. Sistemas de hardwares equipados com apenas uma CPU são chamados de monothread. Já para os hardwares que possuem mais de um core em sua CPU, as threads são feitas concorrencialmente e recebem o nome de multithread.

A figura 2.3 apresenta uma abordagem simplificada de um processo serial e um paralelo, ambos utilizando uma CPU *Quad Core* - 4 núcleos. Na 2.3a percebe-se que a aplicação que é o nosso programa é atribuído para uma *thread* - (*Thread Main*), esse é o processo serial, o processo convencional quando se executa uma aplicação na JVM sem alterações nas threads. Já na figura 2.3b várias *threads* foram atribuídas à aplicação e elas serão responsáveis pela sua execução. O número de threads e que instruções cada uma executará da aplicação, assim como sincronização dos dados e o acesso ao objeto, dentre outras especificações importantes na paralelização, cabe ao programador implementar.



(a) Serial **(b)** Paralelo

Figura 2.3: Representação dos modos de execução

O paradigma de paralelização é utilizado em diferentes áreas, tais como: aplicações de computação gráfica, simulações computacionais, pesquisa e classificação de dados, aplicações científicas e de engenharia, onde o rendimento da aplicação é fortemente dependente da eficiência computacional do hardware. Também se pode citar aplicações que envolvem pesquisa e análise de dados, medicina, geração energética e fatores climáticos (GRAMA et al., 2003) como exemplo de aplicações que utilizam o paralelismo.

CAPÍTULO 3

DESENVOLVIMENTO

Neste capítulo o desenvolvimento será abordado em 3 seções: dataset, algoritmo serial e algoritmo paralelo. Na primeira seção será apresentado a definição, utilidades e um exemplo de um dataset, pois a base de dados é de suma importância para um algoritmo de classificação, assim como o combustível é para um automóvel, o conjunto de dados abastece a implementação, para que ela tome forma e execute sua função. Já nas duas seções seguintes serão demonstrados os dois algoritmos propostos: serial e paralelo. Nelas serão apresentados os procedimentos e métodos para implementação, além de esclarecer as justificativas de porque o algoritmo paralelo tem um melhor desempenho de tempo do que o serial, juntamente com as aplicações de cada um.

3.1 Dataset

Em um algoritmo de classificação um dos primeiros passos é a definição do dataset. Datasets são arquivos que contém várias informações agrupadas sobre um determinado assunto. Eles podem ser no formato de planilha no Excel (XLS), um arquivo CSV, TXT, JSON ou XML. Na classificação utilizando árvores de decisão, existem determinadas especificações para se escolher um dataset, todas elas variando de acordo com o tipo de algoritmo e de árvore que se deseja implementar. Neste projeto as justificativas de escolha resumiam-se em:

- um dataset em que seus atributos fossem quantitativos; para que a árvore fosse extensa;
- um volume muito grande de amostras, para que as operações demandam um custo alto de processamento;
- as classes (saída de classificação) fossem binárias, ou seja com apenas duas classes, pois

cada classe representaria uma folha (nó terminal), fazendo com que apenas divisões binárias fossem sucedidas para as próximas novas divisões.

Baseado nas características citadas o dataset escolhido foi o *Banknote Authentication* (UCI, 2021). Esse dataset constitui-se de dados extraídos de imagens tiradas de amostras genuínas e falsas de notas de dinheiro, que são usadas para a avaliação de um procedimento de autenticação para detectar quais são fraudulentas com precisão. O dataset tem 1372 amostras e 5 atributos sendo eles:

- *variance of Wavelet Transformed image* (variância da transformada *Wavelet*);
- *skewness of Wavelet Transformed image* (distorção da transformada *Wavelet*);
- *curtosis of Wavelet Transformed image* (curtose da transformada *Wavelet*);
- *entropy of image* (entropia da imagem)
- *counterfeit* (classe que determina se a nota é falsificada) tendo apenas dois valores: 0 (não falsificada) ou 1 (falsificada).

Um exemplo das informações do dataset pode ser conferido na figura 3.1.

```
2.2546,8.0992,-0.24877,-3.2698,0
0.38478,6.5989,-0.3336,-0.56466,0
3.1541,-5.1711,6.5991,0.57455,0
2.3969,0.23589,4.8477,1.437,0
4.7114,2.0755,-0.2702,1.2379,0
4.0127,10.1477,-3.9366,-4.0728,0
2.6606,3.1681,1.9619,0.18662,0
3.931,1.8541,-0.023425,1.2314,0
0.01727,8.693,1.3989,-3.9668,0
3.2414,0.40971,1.4015,1.1952,0
2.2504,3.5757,0.35273,0.2836,0
-1.3971,3.3191,-1.3927,-1.9948,1
0.39012,-0.14279,-0.031994,0.35084,1
-1.6677,-7.1535,7.8929,0.96765,1
-3.8483,-12.8047,15.6824,-1.281,1
-3.5681,-8.213,10.083,0.96765,1
-2.2804,-0.30626,1.3347,1.3763,1
-1.7582,2.7397,-2.5323,-2.234,1
```

Figura 3.1: Fragmento do dataset
Fonte:
https://archive.ics.uci.edu/ml/machine-learning-databases/00267/

3.2 Algoritmo Serial

O algoritmo sequencial foi dividido em vários módulos, o fluxo de execução é representada na Figura 3.2. O passo 1 é a leitura e adequação da base de dados, onde o dataset foi convertido de seu formato (.txt) para uma lista de valores em java – *ArrayList* passo (1) e passo (2).

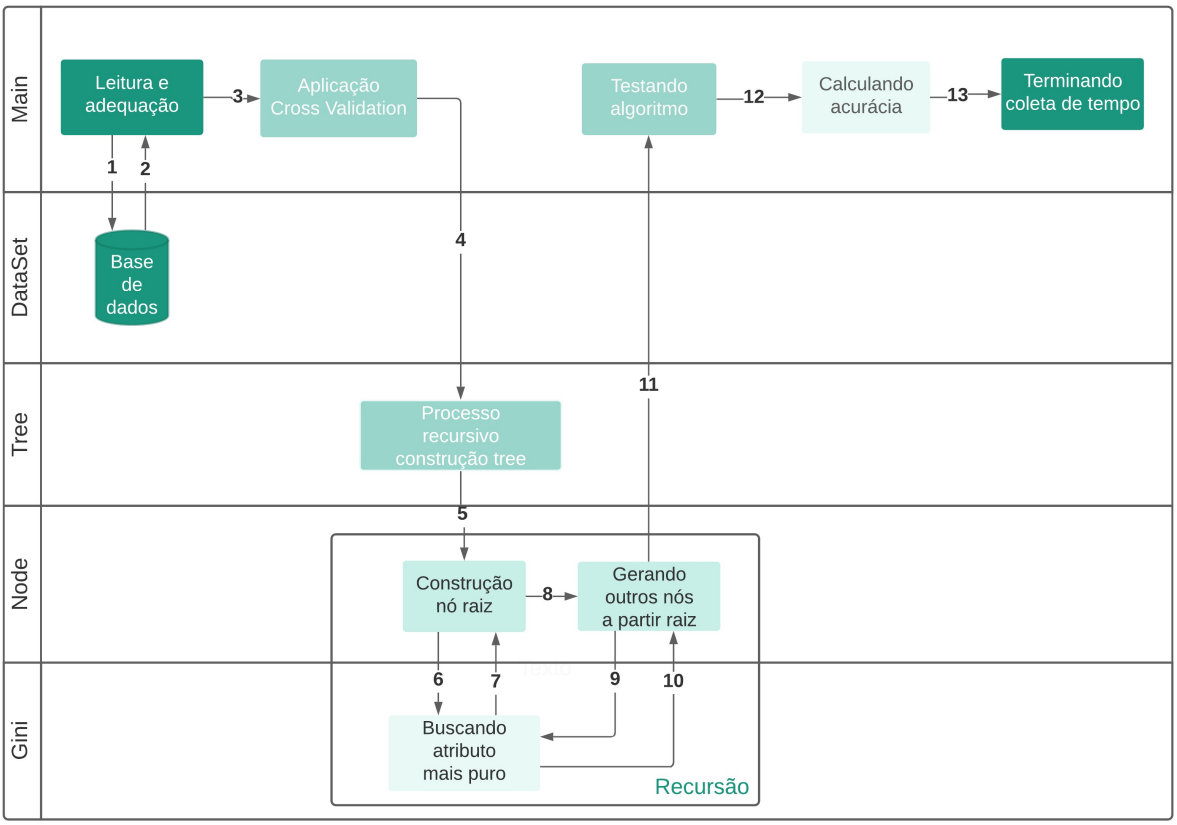


Figura 3.2: Fluxo algoritmo sequencial
Fonte: Arquivo Pessoal(2021).

O método de avaliação de dados utilizado é chamado de Validação Cruzada (Cross Validation) passo (3). "A validação cruzada é uma técnica para avaliar modelos de *Machine Learning* por meio de treinamento de vários modelos em subconjuntos de dados de entrada disponíveis e avaliação deles no subconjunto complementar dos dados.

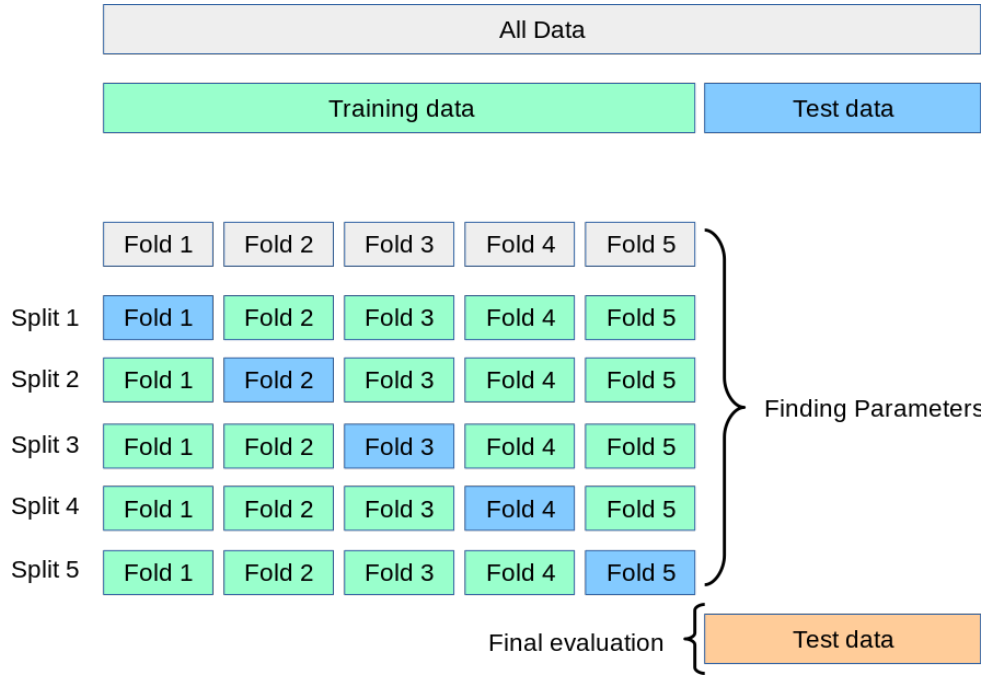


Figura 3.3: Cross Validation
Fonte: https://scikit-learn.org/stable/modules/cross_validation.html

A validação cruzada é usada detectar sobreajuste, ou seja, a não generalização de um padrão."(AMAZON, 2021). Um exemplo de representação de validação cruzada pode ser visto na 3.3, onde dados de treino são divididos em *folds*, que são subconjuntos ou partes extraídos do dataset e *split* são o número de iterações que devem ser realizadas o processo de classificação (treinamento e teste). Cada *split* reflete em um *fold*, ou seja, para cada *fold* é necessário um *split*. No primeiro *split*, o treinamento é feito utilizando os *folds* 2, 3, 4 e 5 (destacados na cor verde) e o teste é feito sob o *fold* 1 (destacado na cor azul), o segundo split repete o processo do *split* 1, utilizando os *folds* 1, 3, 4 e 5 e testado no *fold* 2, e assim sucessivamente. Após esse processo, todo os dados disponíveis do dataset terão sido treinados com diferentes subconjuntos de si próprio e o *fold* designado para o teste, nunca será usado no treinamento.

Uma vez que a abordagem para treino e teste foram definidas, parte-se para construção da árvore passo (5). Cada nó de decisão contém um teste para algum atributo, cada ramo descendente corresponde a um possível valor deste atributo, o conjunto de ramos são distintos, cada folha está associada a uma classe e, cada percurso da árvore, da raiz à folha corresponde uma regra de classificação.(QUINLAN, 1986). Dessa forma é possível usar a técnica de recursividade para construção e criação da árvore passo (4), tendo em vista que a partir de cada nó gerado, outros nós serão criados até alcançar uma folha que retorna a classificação da amostra.

Para avaliar qual atributo será definido como critério das partições, é necessário definir

qual dos atributos contém o menor grau de impureza, ou seja, o que determina o quão bem cada atributo irá dividir as amostras do conjunto trazendo mais ganho de informação. Por se tratar de um critério de partição bastante conhecido para árvores de decisão o índice escolhido foi o índice de Gini (ONODA; EBECKEN, 2001).

O índice Gini foi desenvolvido por Conrado Gini em 1912. Ele é responsável por medir o grau de heterogeneidade dos dados e pode ser utilizado para medir a impureza de um nó, representada na equação 3.1.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2 \tag{3.1}$$

Onde: p_i é a frequência relativa de cada classe e c é o número de classes.

Quando este índice é igual a zero, o nó é puro. Por outro lado, quando ele se aproxima do valor um, o nó é impuro (aumenta o número de classes uniformemente distribuídas neste nó). Quando, nas árvores de classificação com partições binárias, se utiliza o critério de Gini tende-se a isolar num ramo os registros que representam a classe mais frequente. Baseado nessa medida de impureza as partições são realizadas. Na primeira divisão, para encontrar qual atributo será atribuído à raíz da árvore passo (7), é necessário calcular o índice de gini considerando todos os atributos. A partir da raiz as ramificações serão construídas dando origem aos ramos que irão compor a árvore passo (8). Esse processo de divisões baseando-se no menor grau de impureza encontrado pelo gini, será feito inúmeras vezes passo (9) e passo (10) até que todos os galhos cheguem ao seu destino que são as folhas ou o resultado da classificação.

Após a árvore ser construída e os nós estabelecidos, podem ser iniciados os testes (11). Os testes serão feitos com a parte do dataset já separado previamente, onde cada amostra será comparada de acordo com as informações de seus atributos, seguindo as condições de cada nó, começando pela raiz, até encontrar a folha da árvore - determinada pelo treino - corresponde a classe real da amostra. A partir desse processo é possível estimar a proporção de erros e acertos obtido e assim calcular a sua acurácia passo (12). A acurácia indica uma performance geral do modelo ou seja ela diz quantas amostras o modelo classificou corretamente. Finalizando os testes inicializa-se a coleta de tempo passo (13), que consistem em armazenar o tempo de duração da execução do algoritmo a fim de comparar posteriormente com o algoritmo paralelo.

3.3 Algoritmo Paralelo

O algoritmo paralelo consiste em alterar o modo de execução do modelo sequencial para paralelo, ou seja, nessa parte do projeto, os modos serão executados através do uso simultâneo de várias unidades de processamento (CPUs).

Nesse projeto foi utilizada uma máquina com as seguintes especificações: Asus VivoBook S, i7-7500 CPU@2.70GHz QuadCore, GTx 930MX, 8GB RAM, Windows 10 x64. O objetivo dessa abordagem é demonstrar que, quanto mais árvores construídas mais threads podem ser utilizadas e consequentemente menos tempo de execução.s O esquema de implementação do algoritmo pode ser visto na figura 3.4.

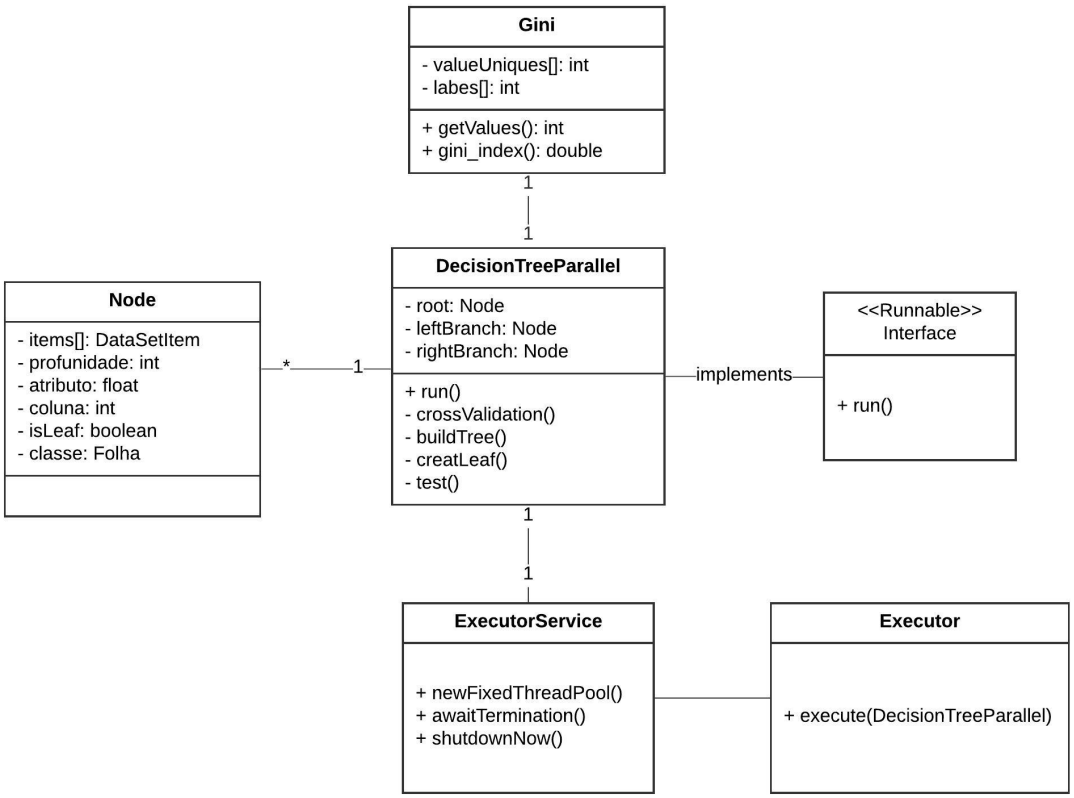


Figura 3.4: Diagrama de classe Paralelo
Fonte: Arquivo Pessoal(2021)

O processo do algoritmo paralelo segue o mesmo fluxo do algoritmo sequencial na leitura do dataset e criação dos *folds*, porém na construção da árvore, a implementação tem uma rota diferente. No algoritmo paralelo, a construção de uma árvore é regida por uma thread, sendo assim, uma thread é gerada a cada nova árvore. Por meio da classe *ExecutorService*, um recurso da plataforma Java e sua *JVM (Java Virtual Machine)*, foi possível gerenciar as execuções em paralelo, já que ela recebe o resultado do método *newFixedThreadPool* da classe *Executors* que cria uma *pool* de *threads* fixas, recebendo como parâmetro a quantidade de threads que se deseja manipular. Na figura 3.5 observa-se um exemplo, onde a variável *nThreads* é usada tanto na quantidade de threads que se deseja criar como também na iteração que define a quantidade de árvores criadas, observada no segundo argumento do *for*.

```
int nTrheads = 128;
start = System.currentTimeMillis();
ExecutorService exec = Executors.newFixedThreadPool(nTrheads);
DecisionTreeParallel dtP = null;
for (int i = 0; i < nTrheads; i++) {
    for (int j = 0; j < folds.size(); j++) {
        // Create n Decision Trees
        dtP = new DecisionTreeParallel(folds, j);
        exec.execute(dtP);
    }
}
try {
    // Wait a while for existing tasks to terminate
    while (!exec.awaitTermination(1200, TimeUnit.SECONDS)) {
        System.err.println("Pool did not terminate");
    }
} catch (InterruptedException ie) {
    // (Re-)Cancel if current thread also interrupted
    exec.shutdownNow();
    // Preserve interrupt status
    Thread.currentThread().interrupt();
}
```

Figura 3.5: Fragmento implementação
Fonte: Arquivo Pessoal(2021)

Para garantir que a coleta de tempos e cálculo da acurácia seja realizado apenas quando todas as threads tenham finalizado, é feita uma verificação usando o método *awaitTermination* que verifica se todas as threads finalizaram ou se o tempo máximo foi atingido. Esse tempo máximo é definido pelo programador para garantir que a aplicação não ultrapasse um limite esperado. Nessa aplicação esse tempo foi escolhido arbitrariamente, desde que seja suficiente para execução do algoritmo. Também é necessário um tratamento de exceção para garantir que, caso algum problema ocorra na execução todas as threads serão devidamente encerradas, esse encerramento é feito no método *shutDownNow*.

A classe *DecisionTreeParallel* apresentada na figura 3.4 é a classe principal, ela representa a árvore de decisão e é responsável pela validação cruzada, construção da árvore, criação de nós e teste dos dados. Ela também implementa a interface Runnable, consequentemente implementa também seu método *run*. O método run é chamado assim que o método *execute* da classe *ExecutorService* é executado, que por sua vez invoca o método *buildTree*. Esse método irá iniciar o processo de construção da árvore.

É feito uma composição com a classe *Gini* pelo método *giniIndex* que recebe e retorna um objeto da classe *Node*. A classe *Node* contém todas as informações sobre um nó, esse nó terá como atributo vinculado aquele que a classe *Gini* classificou como sendo o melhor para as divisões. A *DecisionTreeParallel* terá *n* objetos do tipo *Node* pois eles representam todos os nós que a compõem. Após a conclusão da construção da árvore, são realizados os testes que também seguem o mesmo fluxo do teste no algoritmo serial. Já o cálculo da acurácia, existe uma única distinção, pois como se tratam de várias árvores criadas, faz se necessário calcular a média aritmética para chegar em um único resultado.

CAPÍTULO 4

RESULTADOS

Os parâmetros determinados para os testes (número de threads/árvores) foram escolhidos arbitrariamente numa progressão aritmética de razão 2, resultando no intervalo (2,4,8,...,256). A escolha do 256 como fator máximo de avaliação se dá graças ao processamento máximo que o hardware utilizado para o projeto é capaz de suportar. Para saber quantas threads o hardware é capaz de processar foi implementado um *loop* que cria novas threads até que estoura-se a memória recebendo na exceção *OutOfMemoryError*. E como havia sido previamente dito, o número de threads acompanhará o número de árvores implementadas para uma melhor demonstração de uso de poder de processamento e como sua escalabilidade irá se comportar. Após os algoritmos executados usando o conjuntos de dados escolhido, percebe-se a diferença considerável entre o algoritmo serial em comparação com o paralelo.

Tabela 4.1: Representação dos Dados

Árvores	Sequencial	Paralelo
2	100.995	63.533
4	201.235	93.222
8	409.355	186.149
16	750.453	327.452
32	1.791.806	765.058
64	3.318.964	1.455.055
128	5.593.628	2.382.333
256	10.844.563	4.816.222

Na Tabela 4.1 a primeira coluna refere-se a quantidade de árvores que serão criadas e as colunas sequencial e paralelo representam o tempo de processamento que cada tipo de algoritmo teve para executar a classificação.

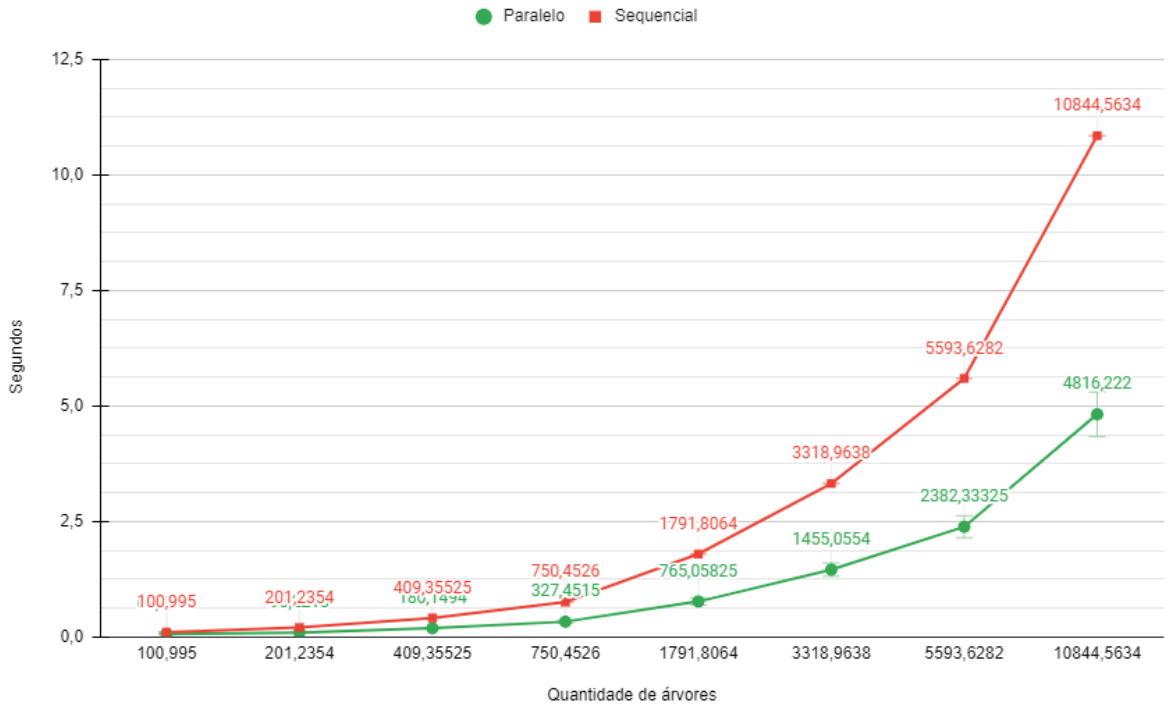


Figura 4.1: Gráfico de linhas
Fonte: Arquivo Pessoal(2021)

Na Figura 4.1 a linha verde representa o desempenho de execução em segundos da implementação com o sequencial e a linha vermelha em paralelo. É importante destacar que, no sequencial, apenas uma thread realiza todo o processo, enquanto que no paralelo, que cada árvore representa também o número de threads atribuídas para executar o algoritmo. Os dois gráficos demonstram que o algoritmo paralelo levou um tempo consideravelmente menor para a execução, chegando assim ao resultado esperado.

Para avaliar o desempenho unicamente da aplicação paralela era necessário utilizar métricas que permitissem avaliar a performance de um processador tendo por base a velocidade/número de operações que este consegue realizar num determinado espaço temporal. Para isso foi escolhido o *speedup*, medida do grau de desempenho que mede a razão entre o tempo de execução sequencial e o tempo de execução em paralelo pela fórmula 4.1.

$$S(p) = \frac{T(1)}{T(p)} \tag{4.1}$$

Onde, *T(1)* é o tempo de execução com um processador e *T(p)* é o tempo de execução com p processadores.

Tabela 4.2: Speedup

Threads	Speedup
2	1,589636317
4	2,158673186
8	2,199068329
16	2,291797717
32	2,342052255
64	2,280987927
128	2,347962108
256	2,251674321

Na tabela 4.2 os valores da segunda coluna demonstra a proporção de vezes que o algoritmo paralelo foi melhor do que o serial.

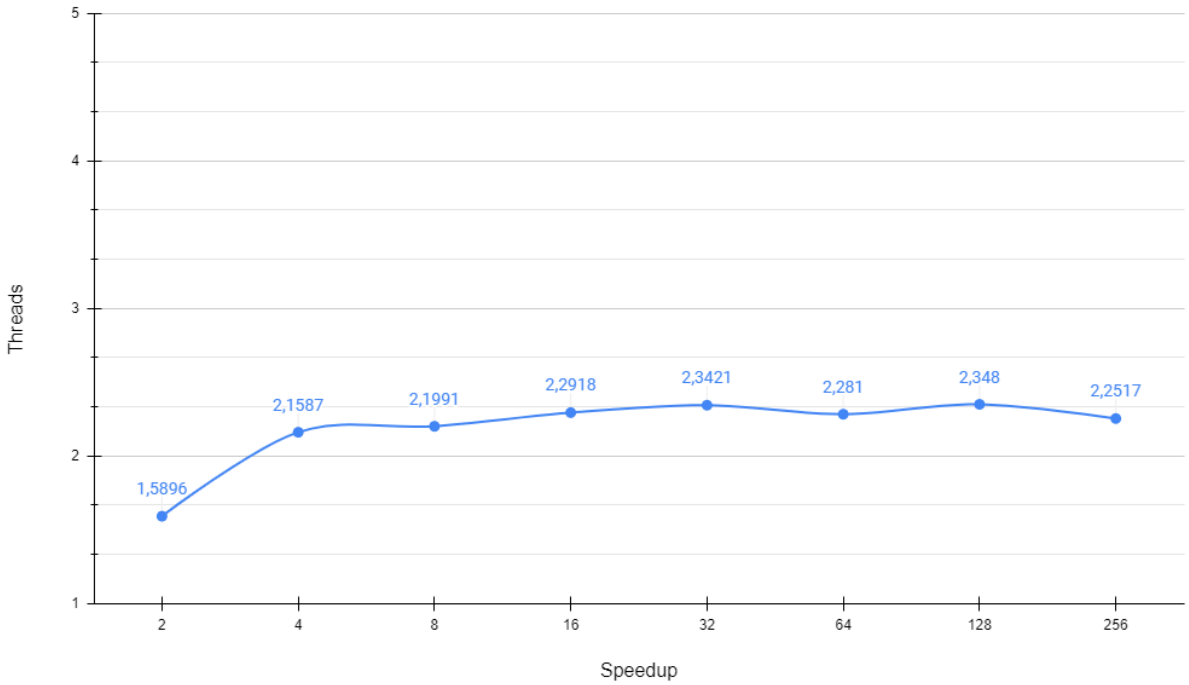


Figura 4.2: Gráfico de Speedup
Fonte: Arquivo Pessoal(2021)

Como pode-se observar na Figura 4.2, quando se ultrapassa o teste feito com mais duas de threads, o algoritmo paralelo é em média o dobro mais rápido que a versão sequencial do algoritmo.

CONCLUSÃO

Em virtude dos fatos mencionados, observa-se a capacidade da paralelização na redução do tempo para resolução de problemas à medida que os recursos computacionais aumentam. Destaca-se a importância que os recursos da CPU tiveram na otimização do tempo de execução desse tipo de implementação, principalmente para grandes números de árvores. Apesar da solução apresentada, não ser genérica, ou seja, dedicar-se a um ramo de processamento específico (classificação), nada impede que essa abordagem seja utilizada para outras evoluções desse ramo de estudo como por exemplo usando *Random Forest* (floresta aleatória), pois nesse modelo, quando está criando as árvores, ao invés de procurar pela melhor característica ao fazer a partição dos nós, ele busca a melhor característica em um subconjunto aleatório das características. Este processo irá gerar muito custo de processamento, mas tem como vantagem a geração de diversidade, o que geralmente leva a geração de modelos melhores e com melhores acurácias, uma vez que esses fatores não foram o objetivo central desse projeto.

Alguns parâmetros escolhidos para as avaliações foram arbitrários devido aos muitos obstáculos consequentes do hardware disponível para o estudo, portanto abriram-se muitas lacunas para futuras pesquisas no assunto. Por exemplo, adaptar esse processo porém utilizando-se dos poderes computacionais da GPU, afim de compará-los, que foi o tema escolhido inicialmente, mas pela alta curva de aprendizagem optou-se por utilizar apenas CPU. Em suma, os objetivos foram alcançados, foi demonstrado as diferenças do modelo serial vs paralelo e que os recursos computacionais disponíveis em cada CPU podem ser muito efetivos em algoritmos cujo poder de processamento exige muito da máquina.

REFERÊNCIAS BIBLIOGRÁFICAS

AMAZON. *Amazon Machine Learning Guia do desenvolvedor*. 2021. Disponível em: <https://docs.aws.amazon.com/pt_br/machine-learning/latest/dg/machinelearning-dg.pdf#cross-validation>. Acesso em: 0 fevereiro 2021.

GAMA, J. Árvores de decisão. *Machine Learning*, 2009.

GOEBEL, M.; GRUENWALD, L. A survey of data mining and knowledge discovery software tools. *SIGKDD Explor. Newsl.*, Association for Computing Machinery, New York, NY, USA, v. 1, n. 1, p. 20–33, jun. 1999. ISSN 1931-0145. Disponível em: <https://doi.org/10.1145/846170.846172>.

GRAMA, A.; KARYPIS, G.; KUMAR, V.; GUPTA, A. *Introduction to Parallel Computing*. Second. [S.l.]: Addison-Wesley, 2003. ISBN 0201648652 9780201648652.

Hasan, N.; Uddin, M. T.; Chowdhury, N. K. Automated weather event analysis with machine learning. In: *2016 International Conference on Innovations in Science, Engineering and Technology (ICISET)*. [S.l.: s.n.], 2016. p. 1–5.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2.

MOLA, F. Classification and regression trees software and new developments. In: RIZZI, A.; VICHI, M.; BOCK, H.-H. (Ed.). *Advances in Data Science and Classification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 311–318. ISBN 978-3-642-72253-0.

NASRIDINOV, A.; LEE, Y.; PARK, Y.-H. Decision tree construction on gpu: ubiquitous parallel computing approach. *Computing*, Springer, v. 96, n. 5, p. 403–413, 2014.

ONODA, M.; EBECKEN, N. Implementação em java de um algoritmo de Árvore de decisão acoplado a um sgbd relacional. In: . [S.l.: s.n.], 2001. p. 55–64.

QUINLAN, J. R. Induction of decision trees. *Machine learning*, Springer, v. 1, n. 1, p. 81–106, 1986.

QUINLAN, J. R. *C4. 5: programs for machine learning*. [S.l.]: Elsevier, 2014.

REZENDE, S. O.; MONARD, M. C.; CARVALHO, A. C. P. L. Sistemas inteligentes para engenharia: Pesquisa e desenvolvimento. In: *Anais III Workshop de Sistemas Inteligentes para Engenharia*. Belo Horizonte: Editora UFMG, 1999.

RUDAKOV, K. Mathematical foundations for processing high data volume, machine learning, and artificial intelligence. *Pattern Recognition and Image Analysis*, Springer, v. 29, n. 3, p. 339–343, 2019.

Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, v. 3, n. 3, p. 210–229, 1959.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining, (First Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. Classification: basic concepts, decision trees, and model evaluation. *Introduction to data mining*, Pearson Addison Wesley, v. 1, p. 148, 2006.

UCI, M. L. R. *Banknote Authentication Data Set*. 2021. Disponível em: <https://developers.google.com/transit/gtfs/?hl=pt-br>. Acesso em: 07 fevereiro 2021.

WRONA, T.; PAN, I.; GAWTHORPE, R. L.; FOSSEN, H. Seismic facies analysis using machine learning. *Geophysics*, Society of Exploration Geophysicists, v. 83, n. 5, p. O83–O95, 2018.