isec
Engenharia

MESTRADO EM INFORMÁTICA E SISTEMAS

DEFINITIVO

Métodos baseados em Árvores para Meta-Aprendizagem

Autor

João Miguel Catalão Antunes

Orientador

Francisco Pereira

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, Julho de 2022

isec
Engenharia

DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

Métodos baseados em Árvores para Meta-Aprendizagem

Relatório de Trabalho de Projeto para a obtenção do grau de Mestre em Informática e Sistemas

Especialização em Desenvolvimento de Software

Autor

João Miguel Catalão Antunes

Orientador

Francisco Pereira

Coimbra, Julho de 2022

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Agradecimentos

A elaboração da presente dissertação de mestrado, não seria possível sem o apoio de algumas pessoas e entidades. Assim sendo, pretendo agradecer de forma individual a todos os que sempre me apoiaram e, de alguma forma, contribuíram para a realização e concretização desta etapa final do meu mestrado.

Deste modo, resta-me agradecer:

À minha família, sem exceção, pelo apoio e confiança que sempre me transmitiram. Queria deixar um agradecimento especial aos meus pais e irmão pela sua dedicação, esforço e conforto que me proporcionaram ao longo de toda esta caminhada.

Aos meus amigos, pelo seu companheirismo, paciência e preocupação. Sem eles, o percurso teria sido muito mais difícil e desgastante. Foram um pilar do meu sucesso.

Ao meu orientador do ISEC, Prof Francisco Pereira, pela sua preocupação e ajuda em todas as questões que surgiram durante e sobre a dissertação. Agradecer pela sua motivação e profissionalismo que de facto se mostrou fulcral em todo este processo.

A todos os docentes que contribuíram para a minha formação pessoal e educacional ao longo de todos estes anos. Foi, sem dúvida alguma, importantíssimo a dedicação e o contributo de cada um. Deixo, aqui, mais uma vez, uma palavra de apreço pela instituição que é o ISEC e os meus sinceros votos de agradecimento.

A todos os enunciados, o meu muito obrigado!

Abstract

This project intends to create an automatic method for the selection of the data analysis algorithm which will be used to solve classification problems. Being that for the development of this study several steps were taken. The first stage was essentially about research and investigation on the topic so that knowledge was acquired.

The objective is to implement a new idea for the meta-learning through the development of a classifier that can classify new problems. However, and so that it's possible to move forward with this implementation it will be necessary to collect data in order to "feed" the classifier. The collection of the data is performed by the analysis of several datasets and by the extraction of knowledge/characteristics of each one of them. The objective of this is to create a database which will be the input of the classifier. The data collection phase will be described and detailed in the following chapters of this project.

Once the classifier's training database is created, this is followed by the next phase which is the creation of the classifier. It will have at its core the algorithm of the decision tree being target of parameter processes in order to reach optimization.

After the development of a meta-dataset and a meta-model it will be possible to assess the behaviour once it starts being produced. There exists a new problem of classification which will be "consumed" by the meta-model and based in historic data will classify the problem. This means that it will suggest the best data analysis algorithm.

This way it will be possible to develop/contribute with a data base with information/characteristics from several datasets as well as a classifier able to suggest the best data analysis algorithm.

Keywords: meta-learning, decision tree, dataset, data analysis

Resumo

O presente projeto visa criar um método automático para seleção do algoritmo de análise de dados que servirá para resolver problemas de classificação. Sendo que para a realização do presente estudo várias etapas foram realizadas. A primeira etapa centrou-se essencialmente na pesquisa e investigação do tema no sentido de adquirir conhecimento na área.

O objetivo passa por implementar uma ideia nova para fazer meta-aprendizagem através do desenvolvimento de um classificador capaz de classificar novos problemas. No entanto e para que seja possível esta implementação será necessário recolher dados no sentido de "alimentar" o classificador. A recolha dos dados passa por analisar diversos *datasets* e extrair conhecimento/características de cada uma deles com o objetivo de criar uma base de dados que servirá de input ao classificador. A fase de recolha de dados será descrita e detalhada nas secções seguintes deste projeto.

Uma vez criado a base de dados de treino do classificador segue-se a próxima etapa que se centra na criação deste. O classificador terá como base o algoritmo árvore de decisão sendo este alvo de parametrizações no sentido de o otimizar.

Numa fase posterior ao desenvolvimento de um *meta-dataset* e de um meta-modelo será possível verificar o comportamento deste uma vez colocado em produção. Existe um problema de classificação novo que será "consumido" pelo meta-modelo e este baseado em dados históricos classificará o problema, ou seja, sugere o melhor algoritmo de análise de dados.

Desta forma será possível desenvolver/contribuir com uma base de dados com informação/características de diversos *datasets* assim como um classificador capaz de sugerir o melhor algoritmo de análise de dados.

Palavras-chave: meta-aprendizagem, árvore de decisão, *dataset*, análise de dados.

Índice

Agradecimentos i
Abstract iii
Resumo v
Lista de Figuras ix
Lista de Tabelas xi
Lista de Equações xiii
Definições e Acrónimos xv
1 Introdução 1
1.1 Contextualização do problema 1
1.2 Objetivo 1
1.3 Contribuições 2
1.4 Plano de trabalhos 2
2 Aprendizagem de Máquina 5
2.1 Categorias de aprendizagem de máquina 6
2.2 Aprendizagem de máquina automatizada 7
2.3 Modelo CRISP-DM 11
2.4 Algoritmo - Árvore de decisão 17
2.5 Exemplos da aplicação de aprendizagem de máquina. 27
3 Arquitetura do Modelo 31
3.1 Construção do meta-modelo 33
3.2 Utilização do meta-modelo/Produção 34
3.3 Ambiente de desenvolvimento 36
4 Construção do *Meta-dataset* 37
4.1 Seleção e processamento de *datasets* 38
4.2 Extração de informação de cada *dataset* 39

4.2.1 Descrição detalhada dos elementos caracterizadores extraídos. 41
4.3 Criação do *meta-dataset* 46
4.4 Formato de um *dataset* 47
4.5 Apresentação do *meta-dataset* 48
4.6 Classificação das amostras do *meta-dataset* 51
5 Construção do Meta-modelo 55
5.1.1 Resultados do meta-modelo 1 e análise crítica 56
5.1.2 Resultados do meta-modelo 2 e análise crítica 58
5.1.3 Resultados do meta-modelo 3 e análise crítica 61
5.1.4 Conclusões dos meta-modelos desenvolvidos 63
6 Utilização do Meta-modelo para Classificação 65
6.1 Preparação dos problemas 66
6.2 Classificação do meta-modelo 67
6.3 Validação da classificação do meta-modelo 68
7 Conclusões e Trabalho Futuro 71
Referências 75
Anexo A: Lista de *datasets* usados A-1
Anexo B: *Dataset* gerado B-3
Anexo C: Apresentação das amostras do *dataset* C-23
Anexo D: Estudo Inicial / Caso de teste D-39

Lista de Figuras

Figura 1 - Contextualização da área da Inteligência artificial. Fonte: [1] 6
Figura 2 - Diagrama CRISP-DM. Fonte: [11] 11
Figura 3 - Validação Cruzada. Fonte: [13] 14
Figura 4 - Árvore de decisão - Tempo Chuva 19
Figura 5 - Árvore de decisão - Humidade Normal 21
Figura 6 - Árvore de decisão - Vento Fraco e Tempo Nuvens 22
Figura 7 - Veículos Autónomos. Fonte: [20] 27
Figura 8 - Assistentes virtuais e chatbots. Fonte [21] 28
Figura 9 - Sistemas Bancários - Fraude Fiscal. Fonte [23] 29
Figura 10 - Treino do meta-modelo 33
Figura 11 - Utilização do modelo 34
Figura 12 - Processo de criação do *meta-dataset.* Fonte [29] 37
Figura 13 - Método de deteção de outliers. Fonte: [36] 43
Figura 14 - Total das amostras 48
Figura 15 - Total das amostras da classe 1 48
Figura 16 - Total das amostras da classe 2 49
Figura 17 - Número de Características 49
Figura 18 - Valores em falta nos *datasets* 50
Figura 19 - Correlação do grupo 1 e 2 50
Figura 20 - Número de *Outliers* 51
Figura 21 - Atribuição de *label/target* aos diferentes *datasets* 52
Figura 22 - Tipos de árvores de decisão 52
Figura 23 - Parâmetros de criação de uma Árvore de decisão 52
Figura 24 – Número de amostras do *Meta-dataset* gerado 53
Figura 25 - Parametrização do Algoritmo 56
Figura 26 – Junção de duas classes 61
Figura 27 - Classificação de *datasets* 68

Lista de Tabelas

Tabela 1 - Matriz de Confusão. Fonte: [14] 15
Tabela 2 - Tomada de decisão sobre jogo de golfe [18] 18
Tabela 3 - Importância das características 40
Tabela 4 - Elementos Extraídos – Informação das amostras 41
Tabela 5 - Elementos Extraídos – Distribuição das duas classes e dos dois grupos 42
Tabela 6 - Elementos Extraídos - *Outliers* 43
Tabela 7 - Elementos Extraídos – Valores em falta 44
Tabela 8 - Elementos Extraídos – Correlação dos dois grupos 44
Tabela 9 - Elementos Extraídos – Importância das características dos dois grupos 45
Tabela 10 - *Meta-dataset* gerado 46
Tabela 11 - *Dataset* Crioterapia 47
Tabela 12 - *Dataset* Sangue Doado 47
Tabela 13 - Classificação das amostras do meta-dataset 54
Tabela 14 - Matriz de confusão - Meta-Modelo 1 56
Tabela 15 - Meta-Modelo 1 - Métricas de avaliação 57
Tabela 16 - Matriz de confusão – Meta-Modelo 2 59
Tabela 17 – Meta-Modelo 2 - Métricas de avaliação 60
Tabela 18 - Matriz de confusão – Meta-Modelo 3 62
Tabela 19 – Meta-Modelo 3 - Métricas de avaliação 63
Tabela 20 - *Dataset* Incêndios Florestais 66
Tabela 21 - *Dataset* Incêndios Florestais - Análise 67
Tabela 22 - Classificações do meta-modelo 68
Tabela 23 - Classificação dos *datasets* através das 4 Árvore de decisão 69

Lista de Equações

Equação 1 - Fórmula de Cálculo da *Accuracy* 15
Equação 2 - Fórmula de Cálculo da *Precision* 16
Equação 3 - Fórmula de Cálculo da *Recall* 16
Equação 4 - Fórmula de Cálculo da F1 16
Equação 5 - Fórmula de Cálculo da Entropia 19
Equação 6 - Fórmula de Cálculo da Entropia entre duas variáveis 19
Equação 7 - Fórmula de Cálculo do Ganho de Informação 20
Equação 8 - Fórmula de Cálculo da Importância das Características 40
Equação 9 - Fórmula de Cálculo do *nij* 40

Definições e Acrónimos

AM/ML – *Aprendizagem de Máquina*

API – *Interface de Programação de Aplicações*

AutoML – *Automated Machine Learning*

CRISP-DM – *Cross Industry Standard Process for Data Mining*

CRISP-ML(Q) – *Cross Industry Standard Process for the development of Machine Learning applications with Quality assurance methodology*

CV – *Validação Cruzada*

FN – *Falsos Negativos*

FP – *Falsos Positivos*

H2O – *Plataforma de algoritmos de código aberto para aprendizagem de máquina*

IA – *Inteligência Artificial*

KNN – *k-Nearest Neighbors*

NLP – *Natural Language Processing/Processamento de Linguagem Natural*

SVM – *Support Vector Machine*

TN/VN – *Verdadeiros Negativos*

TP/VP – *Verdadeiros Positivos*

TPOT – *Tree-Based Pipeline Optimization Tool*

WEKA – *Waikato Environment for Knowledge Analysis*

1 Introdução

A presente secção pretende introduzir e enquadrar o estudo a realizar durante o desenvolvimento do presente projeto. Inicia-se com a contextualização do problema existente assim como uma estratégia que o permita resolver. Em seguida serão apresentados os objetivos do projeto que visam resolver o problema em questão. Algumas contribuições serão apresentadas com o objetivo de partilhar conhecimento com a comunidade científica. Por fim, um plano de trabalhos será apresentado com os passos necessários no sentido de realizar o projeto.

1.1 Contextualização do problema

Atualmente com o crescente volume de informação e dados, a tomada de decisões por parte das empresas tornou-se complexa e é cada vez mais difícil de saber quais as decisões a tomar para ter sucesso organizacional a longo prazo. Os algoritmos baseados em árvores de decisão são métodos de aprendizagem supervisionados cada vez mais utilizados e que têm como vantagem serem bastante fiáveis. Os métodos baseados em árvores de decisão permitem desenvolver modelos preditivos de alta precisão, estabilidade e de fácil interpretação. Uma vez definido e treinado corretamente um algoritmo, este será capaz de resolver problemas de forma rápida, mesmo tendo em conta a enorme complexidade que este possa vir a apresentar.

Um dos problemas existentes prende-se no facto de dado um *dataset* com as suas características inerentes, sejam eles a sua dimensão, distribuição dos dados, dados em falta, *outliers* entre outros selecionar o algoritmo de análise de dados mais adequado que o permita analisar. Automatizando este processo pouparia recursos na escolha do algoritmo de análise de dados uma vez que o investimento em busca do melhor algoritmo realizado por engenheiros seria substituído por um classificador.

1.2 Objetivo

O objetivo deste projeto passa por criar um método automático para seleção do algoritmo de análise de dados. No sentido de concretizar e implementar o método referenciado várias etapas foram realizadas.

- Criar uma base de dados que permita treinar o classificador. Diversos *datasets* (27) foram estudados e processados com o objetivo de extrair conhecimento e informação de cada um, ou seja, as características que cada um possui.
- Criar e testar o classificador. Após a criação da base de dados, segue-se agora a próxima fase que visa criar o classificador que terá como *inputs* os dados recolhidos anteriormente.

O classificador terá como base o algoritmo árvore de decisão. O processo de implementação e desenvolvimento da árvore de decisão consiste essencialmente em parametrizar e otimizar o algoritmo no qual foi necessário realizar alguns testes.

- Exemplificação de utilização do classificador. Numa fase posterior à recolha de dados e implementação do classificador é possível verificar como este se comporta uma vez colocado em produção. O objetivo passa por demonstrar como é possível utilizar o classificador. O classificador recebe um problema/*dataset* no qual são extraídas as suas características inerentes que servirão de input ao classificador. O classificador baseado em dados históricos, e tendo em conta o problema em questão, irá sugerir o algoritmo de análise de dados que permita analisar um determinado *dataset*/problema.

1.3 Contribuições

Uma vez desenvolvido o presente projeto será possível contribuir para a comunidade científica algumas ideias e implementações que resultaram no projeto final, sendo elas:

- Uma metodologia de extração de informação/características presentes num *dataset*, adquirindo desta forma mais contexto sobre o seu conteúdo.
- Uma base de dados contendo informação extraída de 27 *datasets*. É possível em apenas um local observar 27 amostras sendo que cada uma contém as características extraídas de cada *dataset*.
- Um método de classificação de dados usando o algoritmo árvore de decisão.
- Um classificador que permite resolver problemas de classificação. Este classificador terá a capacidade de dado um problema (dado um *dataset*) conseguir sugerir o melhor algoritmo de análise de dados para a respetiva análise. Assim como uma técnica de utilização uma vez colocado em produção.

1.4 Plano de trabalhos

Esta dissertação seguirá a seguinte estrutura:

- **Estudo e enquadramento do projeto:** Esta fase é dedicada essencialmente ao estudo da envolvência da inteligência artificial, aprendizagem de máquina, algoritmos de inteligência artificial, ferramentas existentes e tratamento dos dados. É nesta fase que se começa a adquirir o conhecimento necessário para que mais tarde este possa ser aplicado durante o desenvolvimento do projeto.
- **Desenvolvimento de um caso de teste:** Após o estudo e a aquisição de conhecimento, foi desenvolvido um caso de teste onde o objetivo foi analisar uma grande quantidade de dados e através destes conseguir identificar e extrair conhecimento aplicando técnicas e

algoritmos de aprendizagem de máquina. É possível observar o estudo realizado no anexo D.

- **Arquitetura do modelo:** Este capítulo permite perceber e identificar os principais componentes estruturais do sistema e o relacionamento entre eles. Permite entender melhor o relacionamento entre o *meta-dataset* e o meta-modelo assim como o comportamento do meta-modelo quando surge um problema novo.
- **Construção do *meta-dataset*:** Durante este capítulo vai ser possível observar o processo de criação do *meta-dataset*. Relembro que a criação deste *meta-dataset* tem como prioridade ajudar a alimentar o modelo aquando da sua criação. Esta fase tem uma grande componente de análise, processamento e limpeza de dados.
- **Construção do meta-modelo:** É neste capítulo que será apresentado o processo de construção do meta-modelo que servirá como ferramenta para resolver problemas de classificação assim como conclusões e análise crítica.
- **Previsão e utilização do modelo:** Esta será a última fase deste estudo uma vez que visa demonstrar em termos práticos a utilização do modelo uma vez colocado em produção.

2 Aprendizagem de Máquina

A realização deste capítulo pretende enquadrar e contextualizar o tema do projeto. O objetivo passa por enquadrar a aprendizagem de máquina assim como as categorias associadas e a aprendizagem de máquina automatizada. Será também possível verificar um processo de criação de um modelo de padronização de processos de análise de dados tendo como base a aprendizagem de máquina. É possível encontrar também, um processo de criação de um algoritmo de aprendizagem de máquina -Árvore de decisão – assim como a sua forma de atuar. Alguns exemplos da aplicação de aprendizagem de máquina serão apresentados e contextualizados.

Durante a apresentação deste capítulo é possível observar alguns dados típicos de um problema supervisionado. Nesse sentido serão apresentadas algumas definições importantes mencionadas durante a presente secção.

- *Dataset* – Um conjunto de dados que podem ser apresentados em forma de tabela. Essencialmente constituído por linhas e colunas.
- Instâncias/Exemplos – Representa as linhas de um *dataset* que contém registos de determinados acontecimentos.
- *Feature* – Representa as colunas de um *dataset* que descreve as *features*/características dos acontecimentos.
- *Target* – Representa uma coluna, podendo ser ou não a última que contém a classificação atribuída a um determinado acontecimento.

Atualmente, com o crescente volume de informação e dados, a tomada de decisões por parte das empresas tornou-se complexa e é cada vez mais difícil saber quais as decisões a tomar para ter sucesso organizacional a longo prazo. A **inteligência artificial** entra neste capítulo, uma vez que é uma área integrante na ciência da computação responsável por simular inteligência e o comportamento humano. Um dos objetivos da inteligência artificial passa por executar atividades humanas desde as mais simples até às mais complexas, como é o caso do raciocínio, aprendizagem, planeamento e criatividade. A IA permite ainda que os seus próprios sistemas consigam identificar o ambiente que os rodeia, o intérprete e que consiga apresentar uma solução no sentido de alcançar um objetivo específico.

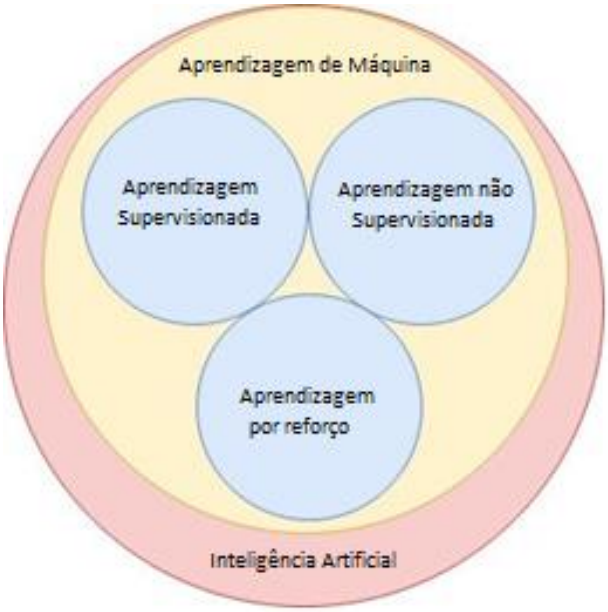

Figura 1 - Contextualização da área da Inteligência artificial. Fonte: [1]

A **aprendizagem de máquina** é um campo da inteligência artificial (ver figura 1) que automatiza a construção de modelos analíticos e permite que se adaptem a novos cenários de forma independente [2]. Uma vez desenvolvido um modelo inteligente, este tem de ser capaz de executar tarefas complexas e dinâmicas. Estes sistemas podem aprender através de dados, identificar padrões e tomar decisões com uma reduzida intervenção humana. Essencialmente a aprendizagem de máquina deveu-se ao facto de conseguir resolver problemas de uma forma mais rápida, simples e eficaz, desta forma provando que as máquinas podem ter a capacidade de aprender com dados. A aprendizagem de máquina serve essencialmente para desenvolver sistemas inteligentes em que estes possam ser usados de maneira a servir a sociedade, comunidade académica ou a indústria.

2.1 Categorias de aprendizagem de máquina

As tarefas relacionadas com a aprendizagem de máquina podem ser classificadas em três categorias diferentes como a aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem por reforço.

A **aprendizagem supervisionada** ocorre quando um modelo é treinado com dados/amostras já classificadas. Uma vez o modelo treinado com base nos dados pré-definidos (classificados) o sistema será capaz de tomar decisões quando receber novos dados de entrada e desta forma atribuir uma saída (classificação) [3]. Existem dois tipos de subcategorias dentro da aprendizagem supervisionada, a classificação e regressão. Os sistemas baseados em classificação têm como objetivo identificar a que classe pertence uma determinada amostra do problema, ou seja, por exemplo se um determinado e-mail é considerado spam ou não. Já os sistemas baseados na regressão focam-se essencialmente em prever um valor numérico, ou seja, o modelo pode aprender

uma função para prever o preço de um imóvel. Associamos os sistemas baseados em regressão a problemas com respostas quantitativas e associamos os sistemas baseados em classificação a problemas com respostas qualitativas.

Já a **aprendizagem não supervisionada** ao contrário da aprendizagem supervisionada não tem amostras classificadas e tem de ser capaz de descobrir novos padrões nos dados, ou seja, é efetuada com base em observação e descoberta. Este tipo de aprendizagem permite abordar problemas onde existe pouco conhecimento dos dados apresentados, agrupando-os (*clustering*) com base em relações entre as diversas variáveis presentes [3].

A **aprendizagem por reforço** permite que um agente num ambiente incerto e complexo consiga encontrar uma solução para um problema através de um sistema de tentativa erro. Este sistema permite ao agente receber "recompensas" ou "penalizações" por cada uma das ações efetuadas. O objetivo desta aprendizagem é maximizar a recompensa total [4].

2.2 Aprendizagem de máquina automatizada

No entanto com o passar do tempo e com a necessidade de acelerar o processo de criação de um modelo uma vez que a aplicação desta pode requerer muito tempo e recursos [5] apareceu um novo conceito de automatização de processos. Conceitos como AutoML, ajuste de híper-parâmetros e meta-aprendizagem amplamente usados em aprendizagem de máquina automatizada serão abordados durante este capítulo.

O termo *AutoML Automated Machine Learning*/Aprendizagem de Máquina Automatizada refere-se à automatização em larga escala de um amplo espectro de processos de aprendizagem de máquina além de criação de modelos, como o pré-processamento de dados, meta-aprendizagem, aprendizagem de características, pesquisa de modelos, otimização de híper-parâmetros, classificação/regressão, aquisição de dados e relatórios [6].

A aprendizagem de máquina automatizada (AutoML) é o processo de automatização de tarefas de aprendizagem de máquina em contexto real. Permite então cobrir o processo de construção de um modelo desde os dados por tratar até ao uso do modelo em produção. Através desta técnica será então possível que indivíduos que não tenham grande conhecimento e *know-how* da área consigam usar estes modelos sem se tornarem especialistas. Com a aplicação deste conceito automático será agora possível automatizar o processo de aplicação de AM desde o início até à sua conclusão de um modo mais simples e rápido uma vez que os processos estão automatizados não sendo necessário intervenção humana ou muito reduzida [7].

Os algoritmos de aprendizagem de máquina são funções estatísticas destinadas a minimizar a variante de uma função de custo. A função de custo depende de um conjunto de parâmetros chamados híper-parâmetros que compõem o algoritmo. Dado um conjunto de dados e com o objetivo de obter o modelo de aprendizagem de máquina mais eficiente, os parâmetros precisam

de ser definidos para o valor ideal específico desse conjunto de dados. No entanto, e até há bem pouco tempo este processo era realizado por especialistas na área manualmente. Tendo em conta que os híper-parâmetros podem assumir qualquer valor, a escolha dos parâmetros tornou-se bastante desgastante ao nível do tempo dedicado em busca do parâmetro ideal. O aparecimento de técnicas de otimização de híper-parâmetros permitiram automatizar o processo de ajuste tais como as árvores de regressão, processos gaussianos e estimativas de densidade. Este conceito pode ser usado também em domínios de aprendizagem de máquina com poucos híper-parâmetros uma vez que os conjuntos de dados tendem-se a tornar muito grandes para serem ajustados manualmente [6].

A meta-aprendizagem é um conceito também este amplamente utilizado e referenciado em aprendizagem de máquina automatizada. Cada modelo treinado num conjunto de dados contribui para a compreensão dos dados. Dado um modelo que represente um desempenho abaixo das expectativas é possível aprender algo com isso, uma vez que esses resultados combinados podem criar um sistema de conhecimento que pode ser usado em conjuntos de dados semelhantes. A meta-aprendizagem usa diversas características existentes num conjunto de dados tais como o número de características, dados em falta, número de amostras, *outliers* entre outros, assim como os dados de desempenho. O objetivo passa por criar um ecossistema de conhecimento que pode ajudar a construir um fluxo de trabalho de aprendizagem de máquina porque uma vez extraído essa informação será possível associar determinados conjuntos de dados à respetiva performance baseado em meta-aprendizagem [6].

Um sistema AutoML contém 4 componentes importantes a ter em conta aquando da sua implementação, que uma vez realizadas permitem incrementar exponencialmente a sua performance.

- **Mecanismo de pré-processamento** – O pré-processamento de dados é a primeira operação a ser realizada. É uma fase importante uma vez que os dados de entrada podem ser variados assim como a existência de discrepâncias. Este mecanismo foca-se em organizar e cuidar dos dados assim como efetuar poucas transformações no sentido de tarefas posteriores possam ser executadas sem problemas. Algumas transformações tais como a normalização, padronização de características e correção de valores são realizadas independentemente do tipo de dados. No entanto pode haver um pré-processamento mais focado ou específico consoante o tipo de dados em análise [6].

- **Mecanismo de características** – Esta fase visa identificar/transformar características dado um conjunto de dados. É uma fase importante uma vez que obter um conjunto de características adequado influencia diretamente o sucesso do modelo. As operações mais comuns realizadas nesta fase são a extração de características, seleção de características, redução de dimensionalidade, transformações lineares e agrupamento para aprendizagem não supervisionada [6].

- **Mecanismo de previsão** – Este mecanismo é o componente mais importante de um sistema de autoML. É nesta fase que o modelo é criado sendo depois treinado e avaliado durante o processo de automatização. O objetivo deste mecanismo é encontrar os melhores híper-parâmetros e algoritmos de aprendizagem para serem utilizados numa próxima fase [6].

- **Seleção do modelo e construção** – Fase em que é selecionado o algoritmo mais adequado tendo em conta um conjunto de algoritmos pré-selecionados na fase de mecanismo de previsão. A escolha do algoritmo pode requerer bastante tempo uma vez que existe toda uma panóplia de algoritmos e nesse sentido é necessário refinar a escolha através de um processo de meta-aprendizagem. Para a seleção do modelo mais adequado técnicas como a validação cruzada e tabelas de classificação podem ser utilizadas [6].

Existem já várias *frameworks* que lidam com este processo através da criação de *pipelines* que permitem otimizar o processo. As *frameworks* permitem que os cientistas de dados, engenheiros assim como os utilizadores não técnicos tenham a possibilidade de automaticamente criarem modelos de aprendizagem de máquina. No entanto podem existir duas abordagens aquando da automatização de aprendizagem de máquina:

- **Abordagem automatizada** – Tal como o nome indica esta abordagem tem como meta automatizar completamente o processo de aprendizagem de máquina. Sendo este o objetivo final – automatizar o processo de criação de aprendizagem de máquina – no entanto é um processo bastante desafiador mesmo com o avanço tecnológico existente [6].

- **Abordagem semiautomatizada** – Esta abordagem permite servir de "assistente" aos cientistas de dados nas tarefas de criação da aprendizagem de máquina como ajudar na implementação do modelo inicial ao invés de o fazerem de raiz. É uma consequência de que os sistemas autoML não estão maduros o suficiente para funcionarem de forma independente, no entanto podem contribuir no desempenho quando coordenados com o envolvimento humano [6].

No sentido de facilitar e otimizar a implementação de uma aprendizagem de máquina automatizada existem diversas frameworks que permitem responder a esse desafio. Sendo algumas delas:

- ***Auto-ml*** – Responsável por automatizar muitas tarefas de uma *pipeline* de aprendizagem de máquina, tais como engenharia de recursos, processamento de datas, codificação categórica e dimensionamento de recursos numéricos. Bastante útil para ambientes de produção uma vez que permite às empresas extrair conhecimento dos dados para os clientes num curto espaço de tempo [8].

- ***Auto-sklearn*** – Utiliza a *framework sklearn* para criar automaticamente *pipelines* de aprendizagem de máquina. A *framework* inclui métodos de engenharia de recursos como codificação *one-hot*, padronização de recursos numéricos entre outros. Os modelos usam

estimadores *sklearn* para problemas de classificação e regressão. As *pipelines* criadas pelo *auto-sklearn* são otimizadas com recurso a pesquisas *bayesianas* [8].

- ***AutoWeka*** – É uma *framework* de aprendizagem de máquina no qual pode ser consultado e usado através de uma interface gráfica, linha de comandos ou através de uma API em Java. O WEKA pode ser integrado com as ferramentas e bibliotecas amplamente utilizadas de ciência de dados, tais como *scikit-learn*, *R* e *Python*. *Auto-WEKA* é uma ferramenta de autoML no WEKA destinada a algoritmos de classificação e regressão. Essencialmente a ferramenta permite, dado um conjunto de dados explorar configurações de híper-parâmetros para os algoritmos suportados e desta forma recomendar ao utilizado qual destes apresenta um melhor desempenho.

- ***TPOT*** – TPOT ou "*Tree-Based Pipeline Optimization Tool*" é um otimizador baseado em programação genética que gera pipelines de aprendizagem de máquina. Estende a *framework scikit-learn* com os seus próprios métodos de classificação e regressão [8]. TPOT explora milhares de pipelines possíveis e encontra aquele que melhor se ajusta aos dados.

- ***H2O*** – É uma *framework* de aprendizagem de máquina semelhante à *framework scikit-learn* que aglomera toda uma panóplia de algoritmos de aprendizagem de máquina que são executados em *cluster* de servidores acessíveis por uma variedade de interfaces e linguagens de programação. A presente *framework* contém ainda um módulo de aprendizagem de máquina automática que usa os seus próprios algoritmos para construir uma *pipeline* [8].

No decorrer da apresentação desta dissertação e sendo que o tema central se foca na criação de métodos baseados em árvores para meta-aprendizagem, haverá uma fase que se foca na implementação de um classificador que terá como base o algoritmo árvore de decisão. Após uma análise criteriosa foi selecionada uma *framework* com o intuito de automatizar o processo de criação do classificador. A biblioteca *scikit-learn* foi a selecionada no sentido de auxiliar na implementação do algoritmo árvore de decisão. Esta biblioteca apresenta grande flexibilidade utilizando a linguagem em *python* que permite testar ideias e casos de estudo muito mais rapidamente assim como um grande controlo mais refinado ao contrário da *framework* WEKA que é uma abordagem do estilo *black-box*/caixa preta que não apresenta tanta flexibilidade. O *Scikit-learn* providencia uma interface orientada a objetos centrada no conceito de estimador. Um estimador é um qualquer objeto que aprende com dados, podendo ser este um algoritmo de classificação [9] sendo este um ponto importante visto que o classificador tem como objetivo resolver um problema de classificação. Uma das razões que motivaram a seleção desta biblioteca ao invés das anteriormente apresentadas prende-se no facto e baseado no estudo realizado pelos autores *Adithya Balaji* e *Alexander Allen* no estudo "Benchmarking Automatic Machine Learning Frameworks" [8] que concluíram que o a *framework auto-sklearn* revelou melhores resultados em

problemas de classificação e que a *framework* TPOT revelou melhores resultados em problemas de regressão. Este estudo tinha como objetivo fazer uma análise de diversas soluções *AutoML* e avaliar o seu desempenho usando *datasets* de classificação e regressão sendo algumas destas soluções as *frameworks Auto-ml, Auto-sklearn, TPOT* e *H2O.*

2.3 Modelo CRISP-DM

Criar um modelo envolve mais do que selecionar um algoritmo, treiná-lo e aplicá-lo a novos dados. Existem outras fases necessárias, que uma vez executadas corretamente permitem desenvolver um modelo capaz de responder às necessidades de cada problema. Em seguida será apresentado um processo de criação de um modelo tendo como base a aprendizagem de máquina assim como os seus passos associados. O diagrama presente na figura número 2 conhecido por CRISP-DM publicado em 1999/2000 para padronizar os processos de análise de dados [10]. Em seguida será então apresentado e detalhado os passos necessários à aprendizagem de máquina. Realçar que a apresentação do seguinte modelo foi baseado no estudo realizado pelos autores *Christoph Schröerab, Felix Kruseb* e *Jorge Marx Gómezb* realizado em 2020 denominado por "*A Systematic Literature Review on Applying CRISP-DM Process Model*" [11].

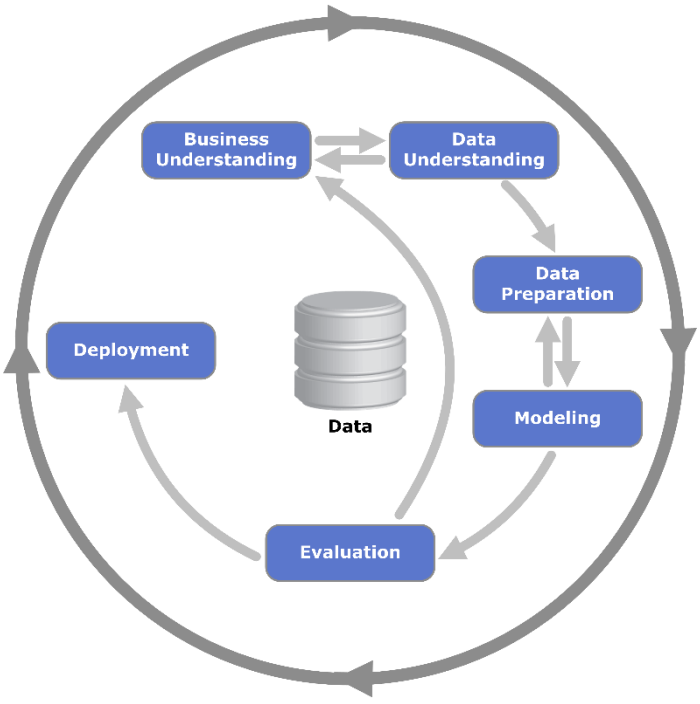

Figura 2 - Diagrama CRISP-DM. Fonte: [11]

I. Compreensão de negócio – Primeira fase do processo de construção, uma vez que é necessário perceber os objetivos do projeto assim como os requisitos necessários na perspetiva de negócio.

Nesta fase é crucial o levantamento de requisitos, riscos e contingências do projeto de maneira a definir bem os objetivos a alcançar. Uma vez que o planeamento já está em marcha, será então possível definir algumas ferramentas e tecnologias a serem usadas durante o desenvolvimento do projeto. Não deve ser descurado um investimento nesta fase uma vez que uma forte compreensão do problema permitirá atingir mais facilmente os objetivos propostos [10]-[11].

II. Compreensão dos dados – É neste momento que se inicia o processo de identificação, recolha e análise dos dados que permitirá atingir os objetivos do projeto. O processo de recolha dos dados dependerá de caso para caso visto que existe toda uma panóplia de dados disponíveis para posterior análise. É importante uma primeira análise dos dados, como a sua observação e compreensão, assim como identificar possíveis relações entre dados no sentido de iniciar o processo de familiarização entre os dados. Através desta análise será então possível começar a identificar a qualidade dos dados, ou seja, selecionar dados em falta ou dados que não sejam relevantes para o algoritmo que se pretende implementar para uma posterior correção. Este processo depois de analisado deve ser documentado no sentido de reter o máximo de informação possível sobre os dados [10]-[11].

III. Preparação dos dados – A presente fase permite preparar os dados para que então estes possam ser usados posteriormente. Se necessário efetuar a limpeza dos dados pré-selecionados proceder à sua correção ou remoção. Esta limpeza pode ser realizada nos seguintes casos:

- **Valores em falta:** Durante o processo de recolha de dados pode acontecer haver valores em falta em algumas amostras. Um dos problemas associados aos valores em falta consiste na perda de representatividade de algumas amostras uma vez que essas amostras podem ser removidas ou pode até mesmo ameaçar a veracidade dos dados uma vez que os dados estão distorcidos levando assim a conclusões inválidas. No entanto, existem já técnicas que permitem lidar com dados em falta tais como remover a amostra que contém dados em falta, substituir pela média, moda ou mediana ou usar algoritmos que permitam lidar com valores em falta. Importa dizer que cada caso é um caso e é necessário adaptar tendo em conta cada situação.

- **Outliers:** Valores inesperados, ou seja, valores que se encontram fora do padrão global de uma distribuição (tendência normal). Podem ser considerados *outliers*, quaisquer pontos que não estejam de acordo com o padrão predominante observado nos dados, no qual podem causar disrupções, uma vez que se abstraem do padrão normal. Os *outliers* podem ser detetados e tratados com recurso a *box-plots* uma vez que é através desta técnica que é possível calcular o valor máximo, mínimo, média e mediana assim como os quartis associados. Uma vez identificado um possível *outlier,* é possível recorrer a algumas estratégias ou abordagens que permitem lidar com estes casos, através da remoção da amostra ou alteração do valor considerado como *outlier*.

Com uma boa familiarização e compreensão dos dados será então possível construir novos dados através dos dados já existentes, através da duplicação dos dados ou previsão do valor em falta baseado em métodos estatísticos [10]-[11].

IV. Modelação – Existe uma panóplia de possibilidades no que respeita a seleção da técnica de modelagem também conhecido pela seleção do algoritmo. Um algoritmo é um conjunto de instruções que um sistema pode executar. O algoritmo tem a capacidade de aprender de uma forma autónoma através de dados históricos com o objetivo de conseguir prever acontecimentos futuros. A seleção do algoritmo não é um processo linear e igual para cada estudo uma vez que está diretamente relacionado com os dados a serem usados assim como os requisitos e igualmente importante o objetivo a atingir. Uma fase importante aquando da construção do algoritmo é a hiper parametrização do algoritmo que visa encontrar as melhores "combinações" de parametrização do modelo [12]. A correta parametrização de um modelo está diretamente relacionada com a qualidade do modelo gerado. Este pode ser um processo moroso e complexo que visa essencialmente encontrar a melhor combinação de parâmetros de um modelo. Existem várias opções no sentido de implementar um algoritmo desde a construção deste a partir do zero ou então usar diversas *frameworks* que já disponibilizam os métodos necessários para a construção do mesmo. Pode acontecer verificar-se que o algoritmo escolhido não apresenta bons resultados tendo em conta os critérios de sucesso definidos em cada projeto, sendo então necessário iniciar-se um novo estudo mas com um algoritmo diferente sendo que o objetivo passa por atingir os critérios de sucesso definidos no início de cada estudo.

Dependendo da abordagem a ser implementada na parte de modelagem pode ser necessário separar os dados em treino e teste. Relativamente ao treino do modelo o que acontece tipicamente é dividir o *dataset* em conjuntos de dados destinado ao treino e ao teste do modelo. O conjunto de dados destinado ao treino contém um output conhecido onde o modelo aprende com esses mesmos dados que podem ser generalizados para outros dados posteriormente. Já o conjunto de dados destinados a testes pode ser utilizado para previsão do modelo nesse subconjunto. Normalmente na comunidade académica o que acontece é dividir um *dataset* em 70% dos dados para treino do modelo e 30% para testar o modelo.

Figura 3 - Validação Cruzada. Fonte: [13]

Existe também uma outra técnica que permite avaliar modelos denominada por validação cruzada. Como ilustrado na figura 3 existe um conjunto de dados que é dividido em 5 subconjuntos do mesmo tamanho sendo que existe sempre um subconjunto dedicado ao teste do modelo (k-1) e os restantes 4 subconjuntos ao treino do modelo. Este processo é realizado 5 vezes (k=5) alternando de forma circular o subconjunto de teste e através deste método permite que o algoritmo não se ajuste demasiado aos dados. Essencialmente a validação cruzada é uma técnica bastante utilizada para avaliar a capacidade de generalização de um modelo a partir de um conjunto de dados.

V. Avaliação – Essencialmente é neste momento que é possível fazer uma retrospetiva e criticamente observar o trabalho desenvolvido nas etapas anteriores e retirar algumas ilações. Um dos objetivos passa por verificar se o modelo responde aos critérios de negócio inicialmente propostos para o problema. O sentido crítico é importante na presente fase uma vez que é necessário fazer uma observação detalhada do trabalho realizado e verificar se todas as etapas foram executadas corretamente e possíveis erros inseridos para que possam ser corrigidos. Após esta retrospetiva é então o momento de prosseguir com a implementação do projeto, repensar na estratégia modificando ou alterando algum passo desenvolvido ou até mesmo iniciar outro estudo.

No sentido de determinar a capacidade que o modelo tem de generalizar para novos casos é necessário testá-lo com novas amostras. No sentido de realizar este processo o *dataset* é dividido em dois conjuntos, sendo eles o conjunto de treino e o conjunto de teste. Os dados de treino serão utilizados para treinar o modelo sem que este tenha acesso aos dados destinados ao teste do modelo. Assim como os dados de teste serão utilizados para testar o modelo. De maneira a medir a qualidade do modelo tendo em conta o objetivo do estudo existem diversas métricas que ajudam a detetar a capacidade de erro e acerto destes. No que respeita a modelos de classificação as métricas no qual podemos analisar são as seguintes:

Tabela 1 - Matriz de Confusão. Fonte: [14]

		Valor Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Tendo em conta os dados presentes na tabela 1 é possível verificar:

- Verdadeiros Positivos (TP) - Amostra que foi predita como positiva e que na realidade era positiva.
- Verdadeiros Negativos (TN) - Amostra que foi predita como negativa e que na realidade era negativa.
- Falsos Positivos (FP) - Amostra predita como positiva mas que na realidade era negativa.
- Falsos Negativos (FN) - Amostra predita como negativa mas que na realidade era positiva.

Confusion matrix - Uma matriz de confusão é uma tabela no qual apresenta as frequências de classificações para cada classe do modelo. Através da matriz de confusão é possível verificar o número de previsões corretas e incorretas dividido por classes classificadas pelo modelo. Desta forma é possível visualizar o desempenho de um algoritmo de classificação.

- *Accuracy* - É uma métrica de avaliação que permite analisar a percentagem de amostras que o sistema foi capaz de calcular corretamente. Fórmula de cálculo presente na equação 1.

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Equação 1 - Fórmula de Cálculo da *Accuracy*

- *Precision* - É a razão dos casos positivos que foram identificados corretamente pelo modelo tendo em conta todos os casos positivos previstos, ou seja, todos os casos corretamente e incorretamente previstos como sendo positivos. Fórmula de cálculo presente na equação 2.

$$\frac{TP}{TP + FP}$$

Equação 2 - Fórmula de Cálculo da *Precision*

- *Recall* - É a razão entre os casos positivos identificados corretamente pelo modelo tendo em conta os casos positivos reais. Fórmula de cálculo presente na equação 3.

$$\frac{TP}{TP + FN}$$

Equação 3 - Fórmula de Cálculo da *Recall*

- *F1* - Esta métrica combina as métricas *precision* e *recall* com o objetivo de identificar a qualidade geral do modelo gerado, pode também ser interpretada como uma média ponderada entre a *precision* e *recall*. Permite medir o quão precisos os resultados do classificador são para os dados de teste, ou seja, é a média harmónica das métricas *precision* e *recall*. Fórmula de cálculo presente na equação 4.

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

Equação 4 - Fórmula de Cálculo da F1

Caso seja um estudo abordando multiclasses cada uma das métricas mencionadas anteriormente exceto a matriz de confusão podem ser calculadas tendo em conta dois critérios:

- **'micro':** Referente às métricas *F1/Recall/Precision* agregará as contribuições de todas as classes para calcular a média. Numa configuração de classificação de várias classes, a micro-média é preferível num episódio de desequilíbrio nas classes.
- **'macro'** Referente às métricas *F1/Recall/Precision* calculará a métrica independentemente para cada classe e, em seguida, obterá a média (portanto tratando todas as classes igualmente).

VI. Lançamento do modelo – Última fase de implementação deste processo uma vez que visa documentar um plano de lançamento do modelo. O modelo gerado e uma vez em produção necessita de um plano de manutenção de maneira a evitar problemas operacionais que possam ocorrer no futuro assim como uma monitorização para possíveis ajustes.

Existe, no entanto, um outro processo muito semelhante ao apresentado anteriormente denominado por CRISP-ML(Q) sendo que este processo permite adicionar em cada fase uma metodologia de qualidade que tem como objetivo mitigar riscos que poderiam afetar o sucesso e eficiência da aprendizagem de máquina. Uma nova contribuição que o processo apresenta prende-se no facto

de adicionar uma nova fase denominado por monitoramento e manutenção no sentido de lidar com os riscos e degradação do modelo perante um ambiente em constante mudança [15].

2.4 Algoritmo - Árvore de decisão

Deve ser mencionado que existem diversos algoritmos para construir modelos tais como redes neuronais, SVM (*Support Vector Machine*), KNN (*k- Nearest Neighbors*) e Árvores de decisão que é o algoritmo que vai ser utilizado neste projeto e que será descrito no presente capítulo [16].

A presente secção visa essencialmente apresentar o algoritmo Árvore de decisão, iniciando por uma introdução ao algoritmo acompanhado pela demonstração do seu processo de construção assim como a forma de atuar.

O algoritmo de árvore de decisão pertence à família dos algoritmos de aprendizagem supervisionada. A árvore de decisão pode ser usada para resolver problemas de classificação assim como problema de regressão. Sendo que o objetivo deste algoritmo é essencialmente treinar modelos de maneira a prever classes (classificar) através de regras e decisões [17].

Basicamente uma árvore de decisão é uma tabela de decisão sob forma de árvore. Uma árvore de decisão, devido às suas características, pode ser utilizada para diversas finalidades, tais como, a análise e pesquisa, estratégia e planeamento e até mesmo como ferramenta de tomada de decisão. As árvores de decisão têm como umas das suas melhores características o facto de serem de fácil compreensão.

Em seguida vai ser apresentado um exemplo de tomada de decisão de ir ou não jogar golfe, tendo em conta diversos fatores. Para a construção deste modelo é necessário ter em conta alguns fatores, tais como, o tempo (sol, chuva ou nuvens), temperatura (suave, agradável e quente), humidade (normal e alta) e o vento (fraco e forte). Cada um destes atributos tem vários valores. A decisão sim significa que estão reunidas todas as condições para se jogar golfe e não significa que as condições para jogar golfe não estão reunidas. Estas possíveis respostas são o resultado da classificação do modelo.

Tabela 2 - Tomada de decisão sobre jogo de golfe [18]

Dia	Tempo	Temperatura	Humidade	Vento	Jogo?	Importância
D1	Chuva	Quente	Alta	Fraco	Não	0.26
D2	Chuva	Quente	Alta	Forte	Não	0.21
D3	Nuvens	Quente	Alta	Fraco	Sim	0
D4	Sol	Suave	Alta	Fraco	Sim	0
D5	Sol	Agradável	Normal	Fraco	Sim	0
D6	Sol	Agradável	Normal	Forte	Não	0
D7	Nuvens	Agradável	Normal	Forte	Sim	0
D8	Chuva	Suave	Alta	Fraco	Não	0.33
D9	Chuva	Agradável	Normal	Fraco	Sim	0
D10	Sol	Suave	Normal	Fraco	Sim	0.20

Como é possível verificar pela tabela 2 existem diversos elementos a ter em consideração que influenciam diretamente sobre a realização do jogo de golfe. Tendo em conta a informação disponível na tabela podemos concluir que sempre que o tempo está nebuloso haverá jogo de golfe. No entanto, sempre que o estado do tempo é de sol ou chuva a realização do jogo está sempre diretamente relacionado com a temperatura, a humidade e o vento. É tendo em conta estas regras em formato de condições apresentado na tabela que a árvore de decisão irá ser construída. Ainda durante a apresentação do *dataset* e com recurso à biblioteca *sklearn* foi possível verificar quais as instâncias que permitem oferecer mais informação aquando da atribuição do target (jogo ou não jogo). Essencialmente a biblioteca permite avaliar todas as divisões possíveis para a criação da árvore e consegue detetar quais as instâncias que ofereceram mais informação (última coluna do *dataset*).

No sentido de criar uma árvore de decisão (recorrendo à biblioteca *sklearn*) e tendo em conta os dados disponíveis na tabela 2, é altura de selecionar a coluna que ficará na raiz e que será a base da decisão. O passo seguinte prende-se no cálculo da entropia no sentido de verificar qual o atributo com menor entropia, ou seja, com maior ganho de informação. O cálculo da entropia é uma forma de analisar a organização dos dados uma vez que quanto maior a entropia mais misturados estão os dados ao contrário de uma baixa entropia que representa uniformização e homogeneização dos dados [19].

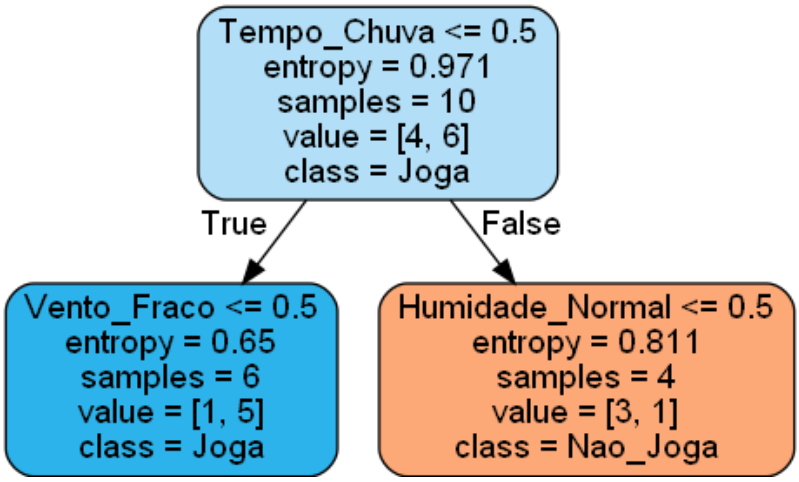

Figura 4 - Árvore de decisão - Tempo Chuva

O processo de criação da árvore de decisão passa por selecionar o atributo com menor entropia de entre os atributos Tempo, Temperatura, Humidade e Vento.

Tendo em conta que a fórmula de cálculo da entropia é dada por:

$$Entropia = -\sum_{i=1}^{n} p(x_i) log2p(x_i)$$

Equação 5 - Fórmula de Cálculo da Entropia

De acordo com a equação número 5 apresentada a entropia é calculada através do somatório de todos os $p(x_i)log2p(x_i)$ onde $p(x_i)$ é a proporção de exemplos em relação ao conjunto de dados a multiplicar pelo logaritmo de base 2 de $p(x_i)$.

E(JogarGolfe) = E(6,4) = $= -\left(\frac{6}{10} log_2 \frac{6}{10}\right) - \left(\frac{4}{10} log_2 \frac{4}{10}\right)$

$$= -(0.60\ log_2\ 0.60) - (0.4\ log_2\ 0.4) = 0.97$$

Agora é o momento de calcular a entropia para cada um dos restantes atributos, ou seja:

$$Entropia(S,T) = \sum_{C \in T} P(c)E(c)$$

Equação 6 - Fórmula de Cálculo da Entropia entre duas variáveis

Tendo em conta a equação 6, $P(c)$ é a proporção de frequência de 'c' em relação ao conjunto de dados a multiplicar pela entropia do conjunto 'c'. A entropia do conjunto 'c' é calculada através da aplicação da fórmula da entropia descrita na equação 5.

- E(JogarGolfe, Tempo) = P(Sol)E(Sol) + P(Nuvens)E(Nuvens) + P(Chuva)E(Chuva)

$$= \frac{4}{10}E(3,1) + \frac{2}{10}E(2,0) + \frac{4}{10}E(1,3) = 0.4 * 0.81 + 0 + 0.4 * 0.81 = 0.65$$

- E(JogarGolfe, Temperatura) = P(Quente)E(Quente) + P(Agradável)E(Agradável) + P(Suave)E(Suave)

$$= \frac{3}{10}E(1,2) + \frac{4}{10}E(3,1) + \frac{3}{10}E(2,1) = 0.3 * 0.92 + 0.4 * 0.81 + 0.3 * 0.92 = 0.88$$

- E(JogarGolfe, Humidade) = P(Normal)E($Normal$) + P(Alta)E(Alta)

$$= \frac{5}{10}E(4,1) + \frac{5}{10}E(2,3) = 0.5 * 0.72 + 0.5 * 0.97 = 0.85$$

- E(JogarGolfe, Vento) = P(Fraco)E($Fraco$) + P(Forte)E(Forte)

$$= \frac{7}{10}E(5,2) + \frac{3}{10}E(1,2) = 0.7 * 0.86 + 0.3 * 0.92 = 0.88$$

Verificamos pelos cálculos efetuados que o atributo com menor entropia é o atributo Tempo sendo este a base da divisão. No entanto e uma vez que o atributo tempo contém 3 valores possíveis sendo eles Sol, Nuvens e Chuva é agora necessário selecionar um destes. O atributo a selecionar é aquele que apresenta um maior ganho de informação à criação da árvore uma vez selecionado para a divisão.

$$Ganho\ de\ Informação = Entropia(S) - Entropia(S,T)$$

Equação 7 - Fórmula de Cálculo do Ganho de Informação

Sendo que o cálculo do ganho de informação é realizado tendo em conta a entropia do atributo 'S' ($Entropia(S)$) calculado na equação 5 quando subtraído com a entropia do atributo 'S' e 'T' ($Entropia(S,T)$) calculado na equação 6.

Neste caso será a chuva como é possível ver pela tabela 2 que demonstra que o atributo chuva adiciona mais informação quando comparado com os restantes, especialmente a amostra 1, 2 e 8. Verificamos pela figura 4 o nó principal contendo as 10 amostras a sofrer a separação através do atributo Tempo representado pela Chuva. Neste caso a amostra total foi dividida com 4 amostras cumprindo o requisito de separação (tempo chuvoso) e 6 amostras que não cumpriam o critério de separação (tempo com sol e nuvens).

Uma vez que ainda existem divisões por realizar é necessário agora verificar os atributos a dividir, calculando novamente a entropia para cada um destes mas agora já com o número de amostras mais reduzido.

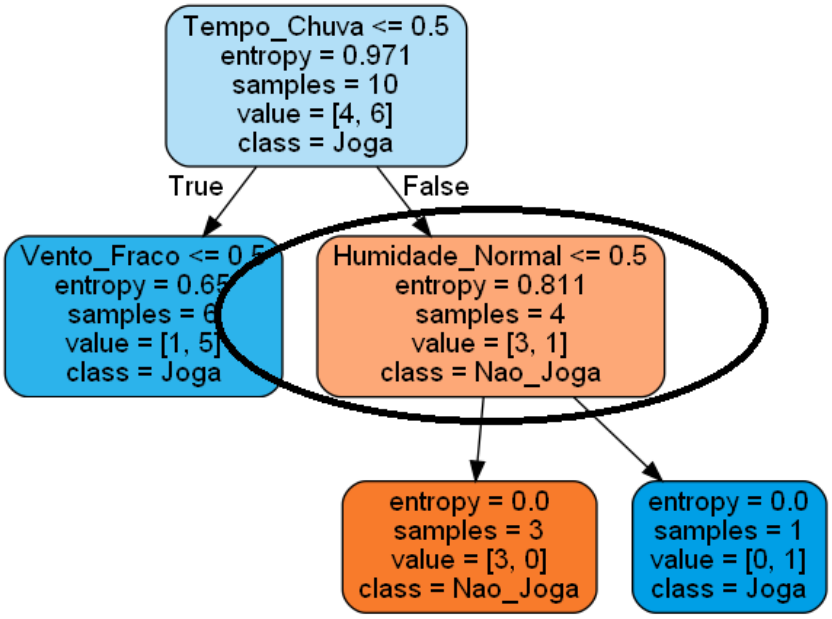

Figura 5 - Árvore de decisão - Humidade Normal

Tendo em conta as 4 amostras restantes após a aplicação do primeiro critério de divisão é o momento de calcular novamente a entropia e verificar qual o atributo a dividir.

E(JogarGolfe) = E(1,3) = $= -\left(\frac{1}{4} log_2 \frac{1}{4}\right) - \left(\frac{3}{4} log_2 \frac{3}{4}\right)$

$$= -(0.25\ log_2\ 0.25) - (0.75\ log_2\ 0.75) = 0.81$$

- E(JogarGolfe, Tempo) = P(Sol)E(Sol) + P(Nuvens)E(Nuvens) + P(Chuva)E(Chuva)

$$= 0 + 0 + \frac{4}{4} E(1,3) = 1 * 0.81 = 0.81$$

- E(JogarGolfe, Temperatura) = P(Quente)E(Quente) + P(Agradável)E(Agradável) + P(Suave)E(Suave)

$$= 0 + 0 + 0 = 0 + 0 + 0 = 0$$

- E(JogarGolfe, Humidade) = P(Normal)E($Normal$) + P(Alta)E(Alta)

$$= 0 + 0 = 0 + 0 = 0$$

- E(JogarGolfe, Vento) = P(Fraco)E($Fraco$) + P(Forte)E(Forte)

$$= \frac{3}{4} E(1,2) + 0 = 0.75 * 0.92 + 0 = 0.69$$

Pelos cálculos efetuados é possível verificar que existem dois atributos (Temperatura e Humidade) com entropia 0 verificando-se um empate no atributo com menor entropia. Neste caso é possível selecionar um destes 2 aleatoriamente no qual neste estudo foi selecionado a Humidade. O atributo Humidade faz-se representar por normal e alta e sendo que ambas as amostras contêm a mesma importância tendo em conta a tabela 2 é possível efetuar a divisão escolhendo uma destas no qual foi selecionado Humidade normal. Como é possível verificar pela figura 5. Uma vez o critério de seleção definido (Humidade Normal) é realizado a separação das 4 amostras restantes. Sendo que existe 1 amostra que cumpre o critério Tempo=Chuva e Humidade=Normal com a classe "Joga" (D9) e 3 amostra que cumpre o critério Tempo=Chuva e Humidade!=Normal com a classe "Não_Joga" (D1, D2 e D8). Verificando-se assim dois nós folha com as classes "Joga" e "Não_Joga".

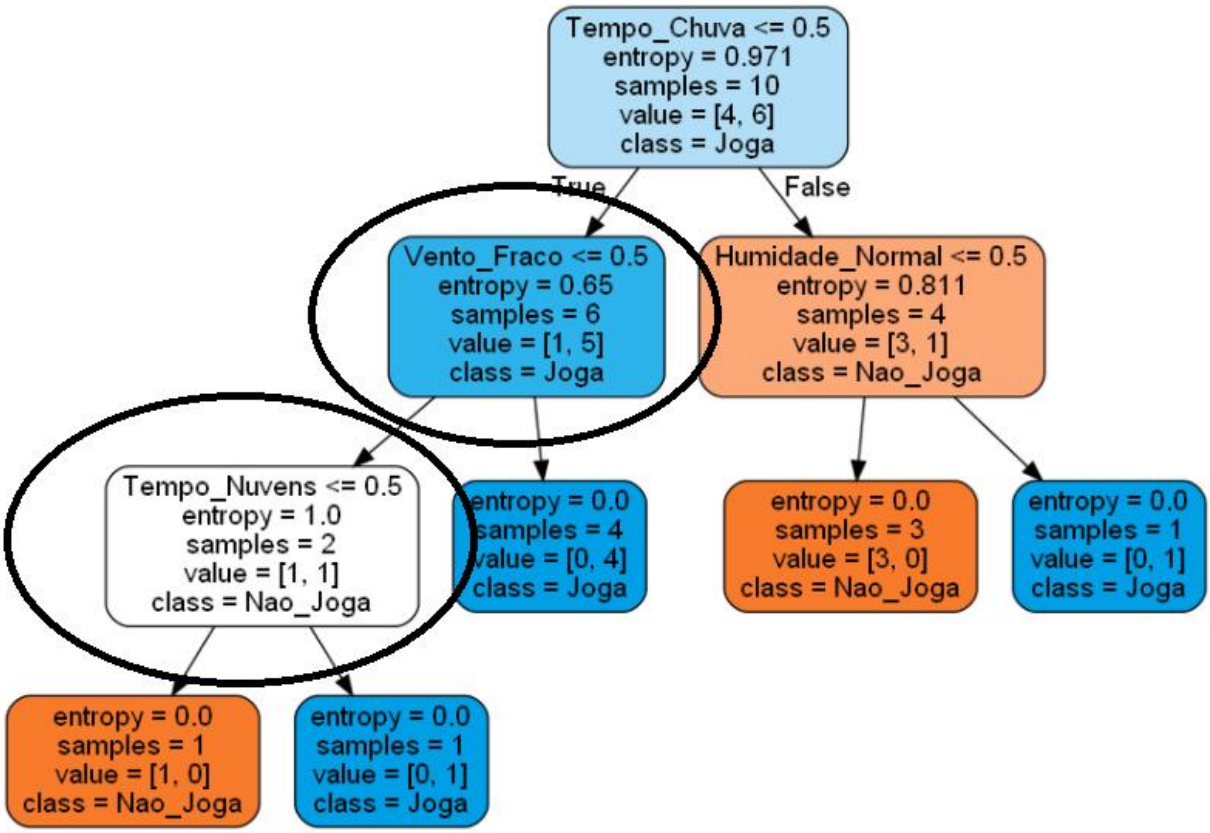

Figura 6 - Árvore de decisão - Vento Fraco e Tempo Nuvens

Resta então continuar a divisão dos restantes critérios. O primeiro critério como mencionado na figura 6 apenas conta já com 6 amostras uma vez que as restantes já foram classificadas anteriormente. É necessário selecionar o atributo com menor entropia que posteriormente irá ser sujeito à sua divisão.

E(JogarGolfe) = E(5,1) = $= -\left(\frac{5}{6} log_2 \frac{5}{6}\right) - \left(\frac{1}{6} log_2 \frac{1}{6}\right)$

$$= -(0.83\ log_2\ 0.83) - (0.17\ log_2\ 0.17) = 0.65$$

- E(JogarGolfe, Tempo) = P(Sol)E(Sol) + P(Nuvens)E(Nuvens) + P(Chuva)E(Chuva)

$$= \frac{4}{6}E(3,1) + \frac{2}{6}E(2,0) + 0 = 0.66 * 0.81 + 0 + 0 = 0.54$$

- E(JogarGolfe, Temperatura) = P(Quente)E(Quente) + P(Agradável)E(Agradável) + P(Suave)E(Suave)

$$= \frac{1}{6}E(1,0) + \frac{3}{6}E(2,1) + \frac{2}{6}E(2,0) = 0 + 0.5 * 0.92 + 0 = 0.46$$

- E(JogarGolfe, Humidade) = P(Normal)E($Normal$) + P(Alta)E(Alta)

$$= \frac{4}{6}E(3,1) + \frac{2}{6}E(2,0) = 0.66 * 0.81 + 0 = 0.53$$

- E(JogarGolfe, Vento) = P(Fraco)E($Fraco$) + P(Forte)E(Forte)

$$= \frac{4}{6}E(4,0) + \frac{2}{6}E(1,1) = 0 + 0.3 * 1 = 0.3$$

Pelos cálculos efetuados o atributo com menor entropia é o Vento que se faz representar por fraco e forte. Neste estudo o critério de separação será através do atributo vento fraco uma vez que tanto escolhendo vento fraco ou vento forte adicionariam a mesma informação à construção da árvore. Verificando pela figura 6 existem já 4 amostras que respeitam as condições de divisão Tempo!=Chuva e Vento=Fraco sendo elas as amostras D3, D4, D5 e D10 com a classe "Joga" constituindo desta forma um nó folha.

No entanto, ainda existem duas amostras por classificar, sendo necessário proceder a mais divisões uma vez que a entropia presente nos dados é de 1.

E(JogarGolfe) = E(1,1) = $= -\left(\frac{1}{2} log_2 \frac{1}{2}\right) - \left(\frac{1}{2} log_2 \frac{1}{2}\right)$

$$= -(0.5\ log_2\ 0.5) - (0.5\ log_2\ 0.5) = 1$$

- E(JogarGolfe, Tempo) = P(Sol)E(Sol) + P(Nuvens)E(Nuvens) + P(Chuva)E(Chuva)

$$= \frac{1}{2}E(0,1) + \frac{1}{2}E(1,0) + 0 = 0$$

- E(JogarGolfe, Temperatura) = P(Quente)E(Quente) + P(Agradável)E(Agradável) + P(Suave)E(Suave)

$$= 0 + \frac{2}{2}E(1,1) + 0 = 0 + 1 * 1 + 0 = 1$$

- E(JogarGolfe, Humidade) = P(Normal)E($Normal$) + P(Alta)E(Alta)

$$= \frac{2}{2}E(1,1) + 0 = 1 * 1 + 0 = 1$$

- E(JogarGolfe, Vento) = P(Fraco)E($Fraco$) + P(Forte)E(Forte)

$$= 0 + \frac{2}{2}E(1,1) = 0 + 1 * 1 = 1$$

Uma vez realizados os cálculos o atributo que apresenta menor entropia é o atributo Tempo sendo este sujeito à sua divisão. Neste estudo a divisão a ser realizada será através do atributo Tempo=Nuvens, no entanto poderia ser também Tempo=Sol uma vez que ambos adicionariam a mesma informação na construção da árvore. Não poderia ser Tempo=Chuva uma vez que as amostras em análise (D6 e D7) não contêm este atributo. O passo seguinte é dividir as restantes amostras pelas condições existentes tais como Tempo!=Chuva, Vento=Forte e Tempo=Nuvens para a amostra D7 e Tempo!=Chuva, Vento=Forte e Tempo!=Nuvens para a amostra D6. Através deste processo, dois nós folha são formados com a amostra D6 contendo a classe "Não_Joga" e a amostra D7 contendo a classe "Joga".

É possível verificar pela figura 6 a árvore de decisão gerada uma vez concluído o processo detalhado anteriormente.

A criação de uma árvore de decisão pode ser influenciada através da sua parametrização. Por exemplo, a divisão de um nó pode ser influenciada através de dois critérios sendo eles a *entropia* ou o critério *gini* tal como o nível de profundidade que a árvore poderá atingir. Tendo este conceito em mente será apresentado uma lista de parâmetros que podem ser manipulados durante a construção de uma árvore de decisão.

***criterion*: string, optional (default='gini')**

Parâmetro que permite medir a qualidade da separação dos atributos. O critério Gini tem como objetivo separar todas as classes no sentido de diminuir a impureza, considerando que uma população é pura quando pertencem todas à mesma classe. Outro critério possível de aplicar é o critério *entropy*, que tem como objetivo maximizar a pureza dos dados, é usada cada vez que é criado um nó da árvore. Em cada separação o objetivo é diminuir a entropia entre os dados, uma vez que estes cálculos são feitos antes e depois da separação e, caso a entropia diminua, prossegue-se para a divisão seguinte. No exemplo anterior, aquando da criação da árvore foi utilizado o critério entropia.

***splitter*: string, optional (default='best')**

Existem duas técnicas de separação, a técnica *best*, que é aquela que oferece melhores valores depois de medir a qualidade da separação. Em cada nó o algoritmo considera todos os atributos e

escolhe o melhor a dividir, ou seja, escolhe aquele que uma vez dividido oferece mais informação à criação da árvore. Já o parâmetro definido como *random* irá selecionar o atributo a sofrer a separação de forma aleatória. No exemplo anterior de criação da árvore foi usado o parâmetro "best" uma vez que o objetivo passava por medir todas as divisões e selecionar o atributo capaz de diminuir a impureza dado um critério de separação.

***max_depth*: int or None, optional (default=None)**

Define a profundidade máxima que a árvore pode atingir. Se o parâmetro "max_depth" não for especificado a árvore irá crescer até que todas as folhas sejam puras. Quanto mais profunda a árvore for, mais divisões terá, assim como a retenção de mais informação sobre os dados.

***min_samples_split*: int, float, optional (default=2)**

Número mínimo de características necessário para dividir um nó. Sempre que necessário proceder a uma divisão em um determinado nó o presente parâmetro permite definir o número mínimo de características a ter em consideração para proceder a sua separação e definir um critério de divisão.

***min_samples_leaf*: int, float, optional (default=1)**

Número mínimo de amostras necessário para ser considerado um nó folha. Por exemplo, imagine-se um exemplo em que *min_samples_split*=5 e existem 7 amostras num determinado nó, a divisão será permitida resultando em duas folhas sendo que uma delas com 1 amostra e a segunda com 6 amostras por exemplo. Se *min_samples_leaf*=2 a divisão não será permitida mesmo o nó contendo 7 amostras porque uma das folhas resultantes terá menos que o número mínimo de amostras necessário para criar um nó folha.

***min_weight_fraction_leaf*: float, optional (default=0.)**

O mínimo da soma total dos pesos de todas as amostras de entrada necessário para estar num nó-folha. As amostras têm um peso igual quando *sample_weight* não é fornecido. É semelhante ao parâmetro anterior "*min_samples_leaf*" mas ao invés de ser necessário ter um número mínimo de amostras necessário para ser considerado um nó folha é necessário conter uma fração de amostras (pesos) para ser considerado nó folha.

***max_features*: int, float, string or None, optional (default=None)**

Número máximo de características a serem consideradas para efetuar a separação de um nó. Imagine-se um exemplo com 50 atributos com *max_features*=10, cada vez que é necessário proceder à divisão de um nó, 10 atributos são selecionados aleatoriamente no intuito de encontrar o melhor atributo a ser dividido. Este processo é repetido sempre que necessário dividir um nó.

***random_state*: int, RandomState instance or None, optional (default=None)**

Controla a obrigatoriedade que é aplicado aos dados antes de ser feita a divisão. Sempre que o parâmetro *random_state* não é especificado os dados destinados ao treino e teste do modelo, permitem conter valores diferente em cada execução ao contrário de uma definição deste parâmetro, que uma vez definido os dados dedicados ao treino e ao teste do modelo serão sempre os mesmos.

***max_leaf_nodes*: int or None, optional (default=None)**

A árvore de decisão vai crescer/formar até existir um máximo de nós folha. Ao definir este parâmetro o algoritmo irá gerar primeiramente o nó folha com maior diminuição de impureza. É possível observar numa árvore de decisão com diferentes profundidades, onde é possível priorizar aqueles que diminuem mais a impureza existente nos dados.

***min_impurity_decrease*: float, optional (default=0.)**

O nó é dividido apenas se essa divisão adicionar uma diminuição da impureza maior ou igual a valor providenciado. A divisão de um nó é realizada tendo em conta o ganho de informação gerado ao fazer a divisão, através deste processo é possível fazer a melhor divisão significando isto que quando a impureza é 0 todas as amostras no nó folha estão corretamente classificadas. O que este parâmetro faz é precisamente impedir a divisão de um nó sempre que o valor da diminuição da impureza gerada com essa divisão seja inferior ao valor fornecido.

***class_weight*: dict, list of dicts, “balanced” or None, default=None**

Atribuição de pesos às classes. Por padrão, todas as classes têm o mesmo peso, se necessário é possível atribuir diferentes pesos a cada classe. Existindo duas classes sendo que a classe 1 faz se representar com mais amostras que a classe 2, podemos concluir que a classe 1 terá mais peso e influência uma vez que contém mais amostras e informação associada. Este parâmetro permite atribuir, se necessário, mais peso à classe 2 e igualando a influência que ambas as classes podem conter. Os pesos que cada classe possa conter influenciará a classificação das classes na fase de treino do algoritmo.

Os híper-parâmetros são parâmetros de modelos que devem ser definidos antes de treinar o modelo tais como os parâmetros mencionados anteriormente para a criação de uma árvore de decisão. O objetivo passa pela seleção dos parâmetros mais adequados ao modelo de maneira a otimizar e maximizar a performance deste. Os parâmetros mencionados anteriormente são disponibilizados pela biblioteca scikit-learn. A híper-parametrização da árvore vai influenciar diretamente a criação da árvore uma vez que existem parâmetros que uma vez conjugados uns com os outros permitem otimizar a árvore de decisão. No sentido de encontrar a melhor combinação entre os parâmetros, a ferramenta *Gridsearch* permite automatizar o processo de ajuste de parâmetros de um algoritmo, uma vez que esta ferramenta de uma maneira sistemática permite realizar diversas combinações

de parâmetros e avaliar essas mesmas combinações posteriormente. A biblioteca *scikit-learn* facilita o processo de criação da árvore de decisão assim como a híper-parametrização deste, sendo que todos os parâmetros mencionados anteriormente podem ser alvo de seleção uma vez que são disponibilizados pela biblioteca.

O presente capítulo é de elevada relevância uma vez que está relacionado com a criação do classificador árvore de decisão e no sentido de adquirir mais conhecimento e *know-how* sobre o algoritmo, foi realizado um estudo. A prova de conceito realizada presente no anexo D aborda o processo de criação de um classificador – assim como a sua híper-parametrização - usando a biblioteca *scikit-learn*. Através deste estudo foi possível adquirir algumas competências que ajudaram a realizar o presente projeto.

2.5 Exemplos da aplicação de aprendizagem de máquina.

Durante esta secção alguns exemplos práticos reais de utilização de sistemas construídos e desenvolvidos com recurso a aprendizagem de máquina.

Figura 7 - Veículos Autónomos. Fonte: [20]

Veículos Autónomos: Veículos autónomos já incorporam sistemas baseados em aprendizagem de máquina no sentido de melhorar a segurança e confiabilidade. Atualmente os veículos baseados em modelos têm a capacidade de tomar decisões de uma forma completamente autônoma tal como os humanos fazem. Como ilustra a figura 7 os modelos têm de ter a capacidade de tomar decisões baseado no comportamento de outros veículos ou no meio ambiente em que estão inseridos. Tudo isto é possível através da aprendizagem de máquina que gere este processo. Estes veículos terão a capacidade de identificar determinados eventos ou objetos, seja através de câmaras ou sensores e num reduzido espaço de tempo conseguir tomar uma decisão. O comportamento de um veículo autónomo será diferente consoante o ambiente em que está inserido, ou seja, um veículo será mais

cauteloso dentro de uma localidade não só pela velocidade ser mais reduzida mas pela imprevisibilidade da ocorrência de eventos que a aprendizagem de máquina terá de gerir e reagir baseado em dados históricos.

Estes veículos estão equipados com um conjunto de tecnologias tais como sensores, radares e câmaras que permitem e facilitam a recolha de dados associados ao meio em que estão envolvidos e desta forma gerando uma grande quantidade de dados. O processamento e interpretação destes dados através de técnicas de aprendizagem de máquina permitem criar modelos que uma vez colocados em prática conseguem tomar decisões baseado em dados históricos. Uma vez o modelo gerado através da recolha de dados permite de uma forma autónoma gerar rotas, definir manobras de condução ajustadas à informação sensorial recolhida anteriormente relativa a objetos, tais como sinais de trânsito, outras viaturas ou até peões.

O sucesso da implementação desta tecnologia está associada à correta extração de informação – recolha de dados – que seja possível fazer. Uma vez recolhido/extraído uma grande quantidade de dados e uma vez processados o reconhecimento de novos eventos e objetos torna-se mais simples uma vez que eventos semelhantes já foram detetados anteriormente no qual permite ao modelo tomar decisões baseados em eventos passados. Essencialmente os carros autónomos têm a capacidade de aprender com a experiência. A tomada de decisão baseado na experiência e dados históricos permite também melhorar a componente de segurança uma vez conseguem antecipar movimentações e eventos que possam ocorrer.

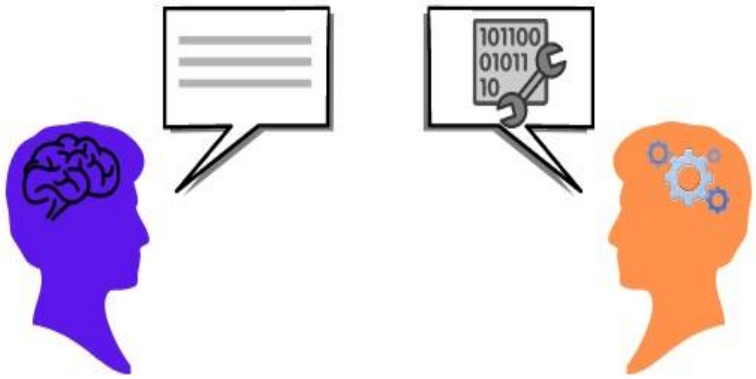

Figura 8 - Assistentes virtuais e chatbots. Fonte [21]

Essencialmente os assistentes virtuais são dispositivos inteligentes que têm como função executar tarefas através de comandos, sejam eles de texto ou de voz. Os comandos realizados pelo utilizador são processados através da aplicação de algoritmos de aprendizagem de máquina que uma vez interpretado executará uma determinada tarefa. Alguns assistentes virtuais tais como a Alexa, da Amazon, a Siri, da Apple, e a Cortana da Microsoft já têm a capacidade de receber pedidos de voz ou texto e executar tarefas tais como pesquisas no google, tocar músicas ou ler notícias.

Os chatbots são robôs que permitem realizar uma conversação. É possível comunicar com estes sistemas através de mensagens escritas ou de voz. Estes tipos de sistemas são frequentemente usados em empresas que recebem grandes quantidades de contactos e precisam de automatizar o processo de resposta em tempo útil. Os chatbots têm como base técnicas de aprendizagem de máquina que permitem interpretar um pedido do utilizador e responder automaticamente. Um dos objetivos destes sistemas é responder a pedidos usando algoritmos de aprendizagem de máquina sem necessidade de intervenção humana.

Os assistentes virtuais e *chatbots* fazem parte dos sistemas NLP que é um ramo da inteligência artificial que combina a linguística computacional (modelos baseados em regras de linguagem humana) e modelos estatísticos de aprendizagem de máquina e aprendizagem profunda. Uma vez juntas as presentes tecnologias permitem que os sistemas computacionais consigam processar a linguagem humana através de texto ou voz e que desta forma a consigam interpretar e compreender o seu significado (figura 8) [22].

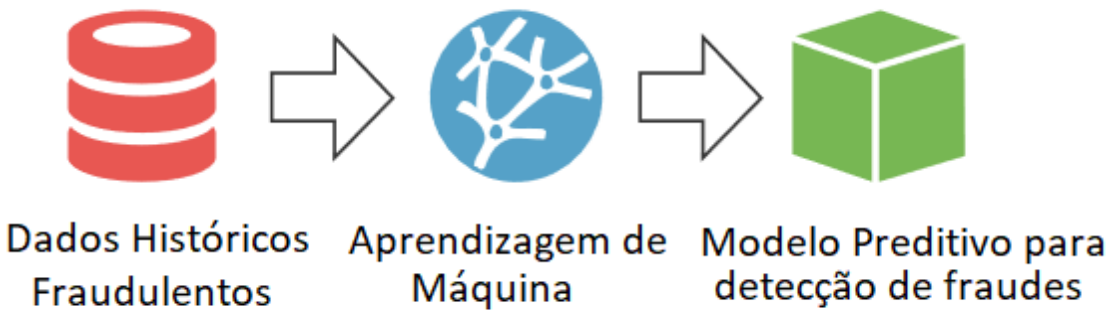

Figura 9 - Sistemas Bancários - Fraude Fiscal. Fonte [23]

A aprendizagem de máquina oferece um grande valor para empresas, indústrias ou parte interessada que tentam lidar e processar grandes volumes de dados. Permite também ajudar a entender melhor eventuais mudanças de comportamento, preferências ou até mesmo irregularidades, essencialmente permite encontrar padrões e extrair conhecimento dos dados. Ao longo do tempo responsáveis por empresas e indústrias aperceberam-se de determinados fenómenos, dos quais não eram preceptivos por meio humano uma vez que são padrões ocultos ou anomalias existentes entre os dados, podendo estes serem benéficos ou prejudiciais. Uma das grandes vantagens da aprendizagem de máquina é o facto de providenciarem soluções através da implementação de algoritmos e modelos de maneira a prever acontecimentos que de forma humana seria impossível ou bastante complexo. Na área da banca e outros negócios na indústria financeira já usam tecnologia baseada em aprendizagem de máquina no sentido de identificar e detetarem possíveis irregularidades entre os dados e desta forma prevenirem fraudes. Este tipo de informação permite identificar oportunidades de investimentos, ou ajudar investidores a definir a melhor altura

para realizar determinados negócios. Através da exploração de dados - *data mining* – será possível identificar determinados clientes com perfis de alto risco ou usar cyber-vigilância no sentido de detetar sinais de alerta de fraude [24]. O sistema de deteção de fraudes é baseado em regras estipuladas ou um conjunto de condições definidas pelo utilizador ou pelo sistema bancário através da aplicação da aprendizagem de máquina. Tal como referenciado na figura 9 existem dados históricos contendo registos com determinados padrões, regras e orientações que permitem gerar um modelo preditivo de deteção de fraudes. O algoritmo de aprendizagem de máquina terá a capacidade de detetar as variáveis que resultam de um maior impacto e através da relação existente entre estas pode indicar a probabilidade de uma determinada transação ser ou não legítima. Estes conseguem identificar e detetar situações fraudulentas aprendendo com erros do passado, tal como o ser humano. Uma grande vantagem de utilizar sistemas baseados em aprendizagem de máquina para deteção de fraudes é o facto de terem a capacidade de aprender com mudanças existentes entre os dados ao longo do tempo. Sempre que a estratégia fraudulenta muda, o modelo terá de ser reativo e capaz de identificar essas mudanças através do treino do modelo. Uma vez que este tipo de atividades associadas a serviços financeiros e económicos é de elevada importância, tanto a nível económico como empresarial, o investimento na prevenção aumentou exponencialmente através da aquisição de sistemas baseados em aprendizagem de máquina.

Foram apresentados alguns exemplos de utilização da aprendizagem de máquina, no entanto existe toda uma panóplia de casos em que esta pode ser aplicada. Existiam outros exemplos que utilizam sistemas baseados em aprendizagem de máquina que podiam ter sido referenciados tais como nos transportes, emails, compras online, aviação, sistemas de recomendações entre outros. Essencialmente é um método de análise de dados que já está tão inserido no nosso quotidiano que nem nos apercebemos da sua utilização, dado a já familiaridade existente. Uma vez que este subcampo da inteligência artificial tem como objetivo colocar em prática a ideia de que máquinas e sistemas podem aprender de forma autónoma a partir de um grande volume de dados, e tomar decisões tendo como consequência um maior investimento na área.

3 Arquitetura do Modelo

É durante este capítulo que irão ser abordadas as duas grandes etapas a desenvolver durante o desenrolar do projeto através de dois diagramas de alto nível. Durante este capítulo será apresentada a arquitetura que permite então perceber melhor o fluxo dos processos assim como as etapas a serem realizadas. É importante também realçar que a arquitetura a ser apresentada é apenas um exemplo de construção de um meta-modelo sendo que cada caso é um caso e suscetível a alteração conforme o problema em causa.

A relevância na apresentação da arquitetura do modelo prende-se no facto de descrever a estrutura, comportamento e a visão do modelo a implementar. O presente modelo consiste em vários componentes onde cada um tem uma função específica que visam implementar o sistema. É importante ter uma real perceção dos componentes que englobam o modelo e como estes comunicam com o intuito de atingir um determinado objetivo. Cada componente existente tem a sua finalidade e o seu próprio comportamento, no entanto, existem dependências entre cada um dos componentes, não só da maneira como estes comunicam, mas da permuta de dados que possam existir entre cada um.

A presente arquitetura visa resolver o problema de meta-aprendizagem, ou seja, os componentes existentes nesta arquitetura baseiam-se em meta-aprendizagem uma vez que executam tarefas e têm a capacidade de aprender com elas. Por exemplo, algoritmos de aprendizagem supervisionada aprendem a encontrar padrões nos dados de entrada para que consequentemente consigam encontrar esses mesmo padrões nos dados de saída com o intuito de classificar novas amostras em problemas de classificação e regressão.

A arquitetura proposta tem toda a sua relevância uma vez que será a base de implementação do projeto. Permite identificar cada componente assim como estes se relacionam. Existe um problema de classificação e a arquitetura proposta visa resolver esse mesmo problema. Sendo todos os passos necessários implementar para resolver um problema de classificação mais detalhado nas secções seguintes. Como já realçado este estudo pretende criar um método automático para seleção do algoritmo de análise de dados, sendo que a arquitetura apresentada tem como objetivo providenciar uma solução de maneira a garantir a disponibilidade e integração dos dados a serem explorados futuramente. É nesta altura que há uma interseção entre a visão técnica e estratégica do projeto, uma vez que o intuito deste é planear e manter alinhados os processos de extração, manipulação dos dados tendo sempre em conta os objetivos do negócio. A arquitetura é um dos componentes "invisíveis" mas que quando falham podem comprometer seriamente a performance do projeto, sendo uma secção onde deve ser definida uma estratégia que permitirá resolver o problema existente.

Os vários componentes existentes na arquitetura uma vez conjugados tem como objetivo sugerir o melhor algoritmo de análise de dados tendo em conta os dados de entrada, ou seja, as características de um *dataset* sejam eles a sua dimensão, distribuição dos dados, dados em falta, *outliers* entre outros. A arquitetura proposta por esta dissertação tem como objetivo responder a uma necessidade, providenciando uma alternativa de seleção de algoritmo de análise de dados mais adequado para analisar um determinado *dataset*. Será apresentado uma série de exemplos de utilização desta alternativa no sentido de comprovar o seu correto funcionamento utilizando a arquitetura proposta.

A apresentação desta arquitetura terá duas fases como mencionado anteriormente, sendo que a primeira fase abordará o comportamento do modelo na fase de treino e a segunda fase abordará o comportamento numa fase futura, ou seja, como o modelo se comporta quando estiver em produção.

A razão pela qual foi decidido aplicar esta arquitetura e não outra prende-se no facto da arquitetura proposta ter a capacidade de resolver problemas de classificação. No sentido de criar um meta-modelo que permite classificar novos problemas são necessários dados que o permitam "alimentar". Nesse sentido e durante o processo de criação foi adicionado um componente responsável pela criação/extração dos dados que permitissem alimentar o classificador, fazendo-se este representar pelo *meta-dataset*. No entanto, e após a "base de dados" destinada ao treino do meta-modelo estar concluído, é posteriormente necessário implementar o meta-modelo, sendo este "alimentado" pelo *meta-dataset*, que será responsável por classificar novos problemas baseado em dados históricos. Desta forma deu origem a dois componentes, sendo estes o *meta-dataset* e o meta-modelo. Já a arquitetura proposta para a segunda fase foca-se na demonstração uma vez colocado em produção. A arquitetura proposta pretende simular o comportamento deste em produção, existe um meta-modelo que vai receber um problema e vai sugerir a melhor árvore de decisão (classificação).

Relembro que este estudo visa essencialmente criar um método automático para a seleção do algoritmo de análise de dados. Este sistema tem de ser capaz de criar uma árvore de decisão que permita classificar dados, ou seja, dado um problema o sistema deverá ser capaz de selecionar de entre várias possibilidades a melhor árvore de decisão. Essencialmente existe um problema de classificação que poderá ser resolvido através de 4 árvores de decisão. Para isso será necessário criar um meta-classificador que escolhe a melhor árvore de decisão para o problema original.

Neste capítulo será apresentado um diagrama de maneira a apresentar o fluxo dos dados para cada etapa a realizar. Cada diagrama tem como objetivo apresentar de uma forma simples a visão geral do sistema para cada etapa, assim como a demonstração de informação que entrará e sairá do sistema.

3.1 Construção do meta-modelo

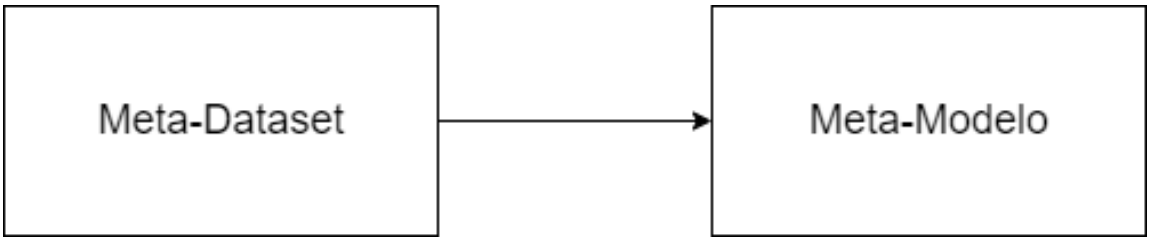

Figura 10 - Treino do meta-modelo

Etapa número 1: Este processo inicia-se na construção do *meta-dataset* que essencialmente se centra na recolha de dados e respetivo tratamento destes para que possam ser processados e servir de alimento aquando da fase de testes e construção do meta-modelo como é possível verificar pela figura 10. Cada instância do *meta-dataset* representa cada um dos 27 *datasets* analisados, ou seja, cada amostra representa a informação extraída (características) de cada um dos *datasets*. Os dados a recolher de cada *dataset* são elementos caracterizadores de cada um que permite perceber melhor a sua constituição e informação existente como por exemplo o número de amostras, número de *outliers* e valores em falta sendo que no decorrer deste projeto mais elementos serão apresentados e detalhados. Essencialmente este processo consiste na capacidade de o meta-modelo utilizar os 27 *datasets* já classificados e aprender regras e descobrir padrões para que mais tarde possa ser utilizado como ferramenta de classificação. Os dados recolhidos serão centralizados em apenas um local (*meta-dataset*).

A fase da construção do meta-modelo revela um problema de meta-aprendizagem uma vez que tem a capacidade de aprender com os dados (no caso do *meta-dataset*), por outras palavras, extrair informação e conhecimento assim como descobrir padrões entre os dados, desta forma tem a capacidade de aprender aprendendo com os dados. Verificamos o mesmo problema de meta-aprendizagem no meta-modelo uma vez que será sujeito a aprendizagem no sentido de encontrar o meta-modelo que consiga responder à necessidade de cada projeto. É um processo de meta-aprendizagem que tem como objetivo ir aprendendo durante o seu processo.

Essencialmente é possível verificar duas fases do processo de implementação de um método automático de classificação. Sendo duas fases de elevada relevância uma vez que a qualidade final do classificador gerado vai estar diretamente relacionado. A qualidade dos dados é um processo importante uma vez que será o "alimento" que permitirá treinar o meta-modelo. Tal como ilustrado na figura 10 e devido à relevância do processo de criação dos dados e implementação do meta-modelo foi decidido criar dois componentes com esse mesmo propósito.

Existe um processo de construção de um *meta-dataset* que permite a criação de dados que posteriormente possam ser analisados e processados pelo meta-modelo. Desta forma é bastante percetível como os componentes se interligam assim como a informação que um transmite ao outro.

É a realização de dois dos objetivos propostos inicialmente tais como a criação de uma base de dados que permita treinar o classificador e a implementação de um meta-modelo no qual é possível agora verificar através da figura 10 representados pelos dois componentes. O processo de criação/extração dos dados faz-se representar pelo componente "*meta-dataset*", e a fase/processo de implementação do meta-modelo faz se representar pelo componente meta-modelo. O fluxo existente é realizado entre o *meta-dataset* e o meta-modelo sendo que o *meta-dataset* irá providenciar toda a informação de maneira a criar/alimentar o meta-modelo.

A etapa apresentada será detalhada pormenorizadamente nos capítulos seguintes.

3.2 Utilização do meta-modelo/Produção

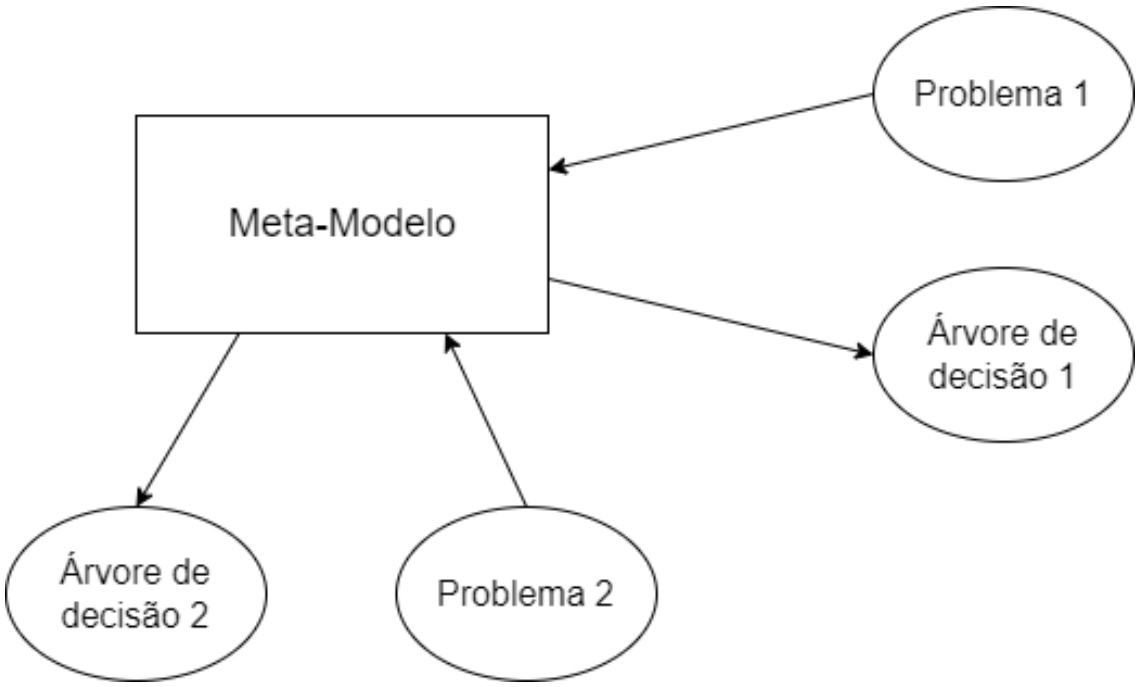

Figura 11 - Utilização do modelo

Etapa número 2: Uma vez o meta-modelo gerado, este servirá de uma maneira muito prática como “ferramenta” para resolver problemas. O meta-modelo tem conhecimento de um problema de classificação e tem de ser capaz de apresentar a árvore de decisão mais otimizada possível para resolver esse mesmo problema (ver figura 11).

Como é possível verificar pela figura 11, o fluxo é muito simples de perceber, existe um meta-modelo já treinado que recebe um problema e tem de ser capaz de o analisar e processar e consequentemente apresentar uma solução que neste caso será uma das 4 árvores mais otimizada. Este será o comportamento em termos práticos do modelo de ML uma vez colocado em produção.

Existirão problemas de classificação distintos, diga-se com características diferentes, onde o meta-modelo terá de ter a capacidade de o processar sempre com o objetivo de apresentar a melhor árvore de decisão.

É na presente fase que é realizado mais um dos objetivos propostos no início deste projeto que seria a exemplificação de utilização do classificador do meta-modelo no qual é possível observar através da figura 11. Este será o comportamento que o meta-modelo permite executar uma vez que este seja colocado em produção. Pela figura acima é possível ver o fluxo dos dados, tendo sido definida a presente arquitetura não só pela sua simplicidade mas também pela lógica que apresenta. O processo é iniciado com a utilização do meta-modelo gerado na secção anterior, onde receberá um novo pedido para a respetiva classificação (problema novo). O novo pedido encapsula as características extraídas de um *dataset* tais como o número de amostras, *outliers*, dados em falta, distribuição dos dados entre outros, para que posteriormente o meta-modelo possa processar o problema assim como encontrar padrões nos dados de maneira a classificá-lo baseado em dados históricos.

Desta forma é possível encontrar um método automatizado de classificação de problemas. Problemas esses que se centravam na seleção do melhor algoritmo de análise de dados, dado um *dataset*/problema novo. A arquitetura apresenta uma solução que visa resolver este problema. Tal como apresentado a solução tem 3 fases de elevado relevo. A primeira foca-se na criação de dados que permite posteriormente "alimentar o meta-modelo", deve ser dito que a qualidade dos dados extraídos terá uma grande preponderância na classificação de amostras. A segunda fase centra-se na implementação do meta-modelo que utiliza os dados anteriormente recolhidos para o seu treino. Durante a geração do meta-modelo existe uma particularidade que se foca na híper-parametrização do algoritmo que revela alguma complexidade devido ao seu número de combinações possíveis mas que uma vez otimizado este processo permite aumentar o número de classificações corretas. Sendo a terceira fase responsável pela demonstração e utilização do meta-modelo uma vez colocado em produção, essencialmente recebe um problema/características de um *dataset* e baseado em padrões e conhecimentos que o meta-modelo identificou atribui uma classificação.

É tendo em conta as fases mencionadas que foi proposta a arquitetura apresentada, uma vez que representa uma solução para um problema de classificação. Tem em conta as 3 fases mencionadas e apresenta uma solução para cada uma destas. A arquitetura apresenta também a abordagem implementada aquando da interligação das fases e a permuta de dados que é necessário no normal fluxo dos dados. No entanto, o presente estudo que implementa a arquitetura apresentada serve como exemplo da correta funcionalidade, colocando-a em prática demonstrando assim o seu

propósito. Os próximos capítulos têm como objetivo a apresentação detalhada de cada uma das fases.

3.3 Ambiente de desenvolvimento

Em seguida serão apresentadas algumas ferramentas adotadas para o desenvolvimento do meta-modelo, tais como bibliotecas para processamento de dados, apresentação gráfica dos dados e uma biblioteca de aprendizagem automática. Durante o desenvolvimento deste projeto será usado a linguagem *Python* uma vez que para análise de dados é uma das linguagens mais utilizadas e devido ao facto das ferramentas e bibliotecas utilizadas neste estudo o permitirem.

Pandas - Esta biblioteca será utilizada para o processamento e manipulação dos dados presentes nos *datasets*. Deve ser dito também que esta biblioteca é *open source* e muito usada na comunidade académica. As razões pela qual esta biblioteca foi escolhida prende-se no facto de ser relativamente simples de aprender e de oferecer estrutura de dados e operações para manipular dados numéricos. Para este estudo vão ser apenas usados dados numéricos [25].

Matplotlib- No desenvolvimento deste estudo foi necessário demonstrar graficamente o resultado das execuções do algoritmo de aprendizagem automática. Neste caso em concreto esta biblioteca será responsável por mostrar em forma de grafos as árvores de decisão criadas pelos algoritmos. A biblioteca escolhida para este estudo foi a *matplotlib* uma vez que permite a criação de diversos gráficos de uma forma simples e com poucos comandos [26].

Scikit-learn- Foi a biblioteca de aprendizagem automática escolhida para o desenvolvimento do modelo de ML. Esta biblioteca disponibiliza diversos algoritmos de classificação, regressão e de clustering. A biblioteca *Scikit-learn* permite implementar o algoritmo árvore de decisão usado neste projeto de uma forma muito simples e eficiente, sendo apenas necessário introduzir dados que vão ser canalizados para os testes e treino do meta-modelo e parametrizar o algoritmo de acordo com o problema a resolver [27].

Jupyter Notebook – *Jupyter notebook* cria um documento virtual que permite executar código de uma linguagem de programação assim como a disponibilização de ferramentas para a edição de texto tornando assim o conceito de programar mais interativo e dinâmico uma vez que oferece ao utilizado o respetivo output do código implementado. O processamento dos dados, criação dos gráfico e construção do modelo de ML foi desenvolvido no *jupyter notebook* [28].

4 Construção do *Meta-dataset*

Neste capítulo será apresentado o processo de desenvolvimento/construção do *meta-dataset* que será utilizado para "alimentar/treinar" o classificador do meta-modelo.

O objetivo deste capítulo centra-se essencialmente em detalhar o processo de criação do *meta-dataset*. Numa fase inicial foram identificados, estudados e pré-selecionados um conjunto de *datasets* (27) com características semelhantes, características essas que serão apresentadas e detalhadas durante a apresentação deste capítulo. Uma vez selecionados todos os *datasets*, segue-se então uma nova fase que se centra fundamentalmente no processamento e limpeza dos dados que cada um contém. Numa fase posterior ao pré-processamento dos dados é então o momento de analisar/estudar cada um dos *datasets* (já processados) de maneira a retirar/extrair informação relevante e importante e em seguida registá-la num novo *dataset*. Um novo *dataset* (*meta-dataset*) será criado, sendo que cada amostra presente corresponderá à informação extraída de cada *dataset* analisado, ou seja, os *datasets* em análise serão utilizados como exemplos de modo a construir o *meta-dataset*.

Figura 12 - Processo de criação do *meta-dataset.* Fonte [29]

Em suma, três grandes etapas serão apresentadas durante este capítulo tal como ilustra a figura 12.

- Etapa 1: Seleção e processamento dos *datasets* selecionados.
- Etapa 2: Estudo e extração de informação de cada um dos *datasets* identificados.
- Etapa 3: Registo da informação extraída na etapa número 2 no sentido de criar o *meta-dataset*.

Relembro que o propósito deste capítulo passa de uma forma muito simples pela criação do *meta-dataset* sendo que as 3 etapas necessárias a realizar serão aprofundadas e detalhadas durante este capítulo.

4.1 Seleção e processamento de *datasets*

É neste momento que se inicia o processo de identificação e seleção de *datasets*. Uma vez os *datasets* selecionados e identificados, vai ser possível numa fase posterior extrair informação de cada um de maneira que esses dados possam ser canalizados para alimentar o classificador.

Todos os *datasets* analisados são provenientes de repositórios diferentes tais como os repositórios "*Kaggle*" [30], "*Machine Learning Repository*" [31] e "*data world*" [32]. Foi decidido analisar *datasets* de repositórios diferentes para que exista uma maior diversidade e variedade de informação visto que são originários de fontes de dados diferentes.

Como mencionado anteriormente foram analisados e estudados diversos *datasets* sendo que apenas alguns destes foram escolhidos, o critério de seleção essencialmente prendeu-se no facto dos dados serem maioritariamente numéricos, uma percentagem de valores em falta bastante reduzido e problemas de classificação binária. Estes foram os requisitos impostos para a seleção de um *dataset* a entrar para este estudo no sentido de uniformizar e homogeneizar todos os *datasets*. Esta homogeneização ocorre no sentido de incentivar a que o classificador processe tipos de dados estruturados (numéricos) ao invés de por exemplo dados não estruturados tais como imagens ou linguagens de texto. Deve ser dito, no entanto, que alguns *datasets* requereram pequenas intervenções no sentido de uniformizar convenientemente os dados, nomeadamente no tratamento de dados em falta assim como processar valores categóricos. O impacto final destas intervenções foi bastante reduzido, uma vez que a informação intervencionada representa claramente uma minoria entre os dados.

> **Valores em falta** – Alguns *datasets* continham valores em falta. As alternativas que existem para lidar com esta situação são a remoção da amostra, manter os valores em falta não fazendo nada ou a imputação de dados (previsão do valor em falta baseado em métodos estatísticos). Neste caso, a estratégia usada passou pelo cálculo da média total dos valores correspondentes a essa característica e que pertençam à mesma classe, substituindo depois os valores em falta pelo valor da média [33].
>
> **Valores categóricos** –Acontece que alguns dos *datasets* continham valores categóricos e desta forma necessitaram de ser intervencionados uma vez que ficou decidido manter a uniformidade entre os *datasets* no sentido de que cada um destes apenas iria conter dados numéricos. Em alguns *datasets* analisados a representação do sexo de um individuo fazia-se representar por *female* ou *male* e que, depois de intervencionado, as características referentes a *female* passaram a ser representadas pelo valor 1 (um) e *male* pelo valor 0 (zero) [33]. Tendo sido apenas esta a única intervenção relativa a valores categóricos.

No total foram analisados 27 *datasets* (ver anexo A) binários e numéricos. Todas as amostras presentes em cada um dos *datasets* contêm uma *label* associada (0 ou 1) daí serem problemas de classificação binária.

4.2 Extração de informação de cada *dataset*

É então nesta fase onde irão ser usados os *datasets* pré-processados na fase anterior detalhada, pré-processamento esse que permitiu uniformizar e homogeneizar os *datasets* através de substituição de valores em falta assim como o tratamento de valores categóricos. O presente processo essencialmente tem como objetivo a extração de informação que cada um dos *datasets* contém. A informação a extrair está diretamente relacionada com o conteúdo de cada um. Existem vários elementos caracterizadores e simultaneamente comuns entre os diversos *datasets* a análise como:

- Número de amostras por *dataset*.
- Número de características por *dataset*.
- Número de amostras da classe 0 e classe 1.
- Distribuição dos valores de cada uma das características, nomeadamente a média, desvio padrão, mínimo, máximo e quartis de cada característica tendo em conta as duas classes e cada classe em individual.
- Será também calculada a percentagem de características com valores em falta, ou seja, se existirem 4 características em que apenas uma delas contém valores em falta a percentagem de características com valores em falta será de 25%.
- Matriz de correlação que permitirá então perceber melhor como as características se irão relacionar.
- Será também extraído de cada *dataset* o número de *outliers* existentes de cada uma das classes assim como a percentagem de características que contêm *outliers*.

Serão estes os elementos a extrair de cada um dos *datasets* uma vez que estavam disponíveis para a sua extração, por outras palavras, são elementos comuns entre cada um dos *datasets*. São elementos caracterizadores que uma vez conjugados ou analisados individualmente permitirão encontrar padrões existentes nos dados [34]. Através dos padrões identificados será possível conhecer melhor os dados e enquadrar num determinado perfil. É precisamente esse o objetivo do classificador, ou seja, baseado em dados históricos (*meta-dataset*) ter a capacidade de encontrar padrões e enquadrá-lo num determinado perfil. Recordo que nesta fase apenas será extraída informação detalhada anteriormente de cada um dos *datasets*.

Para existir homogeneidade entre os diversos *datasets* aquando da extração dos elementos caracterizadores dos *datasets* foi decidido agrupar esta informação em dois grupos. A razão que motivou a criação dos grupos passou essencialmente pela simplificação e homogeneização dos dados, ou seja, tendo em conta que cada *dataset* contém um número diferente de características a criação dos grupos permite agrupar informação de um conjunto de características não sendo necessário verificar/registar essa mesma informação individualmente por característica.

Essencialmente uma característica é considerada importante ou não baseado no contributo que esta ofereceu para a previsão da variável "*target*". Em seguida é apresentado a fórmula de cálculo da importância das características através da equação 8 e da equação 9.

$$fi_i = \frac{\sum_{j:nó\ j\ estratificado\ em\ caracteristicas\ i} ni_j}{\sum_{k \in todos\ os\ nós} ni_k}$$

Equação 8 - Fórmula de Cálculo da Importância das Características

- fi_i = Importância da característica i
- ni_j = Importância do nó j

No sentido de calcular a importância das características é necessário calcular também o ni_j que se faz representar pela seguinte equação (número 9).

$$ni_j = w_j C_j - w_{esquerda(j)} C_{esquerda(j)} - w_{direita(j)} C_{direita(j)}$$

Equação 9 - Fórmula de Cálculo do ni_j

- ni_j = Importância do nó j
- w_j = Número ponderado de amostras que chegaram ao nó j
- C_j = O valor da impureza do nó j
- $esquerda_j$ = Nó filho da esquerda dividido no nó j
- $direita_j$ = Nó filho da direita dividido no nó j

A importância de cada característica é calculada como a diminuição na impureza do nó folha ponderada pela probabilidade de alcançar esse nó. A probabilidade pode ser calculada pelo número de amostras que chegam a esse nó, dividido pelo número total de amostras [35]. Ver equação 8.

Tabela 3 - Importância das características

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0.05	0.31	0.08	0.01	0.04	0.22	0.13	0.12

Na tabela 3 é possível ver um exemplo prático retirado de um dos *datasets* sujeitos a estudo. Neste caso de estudo, o *dataset* contém dados sobre diversos indivíduos que uma vez conjugados permitem determinar se um indivíduo tem ou não diabetes. Neste *dataset* existem 8 características em que cada uma contribuiu de maneira diferente para atribuição da classificação final (tem ou

não diabetes). Sendo que cada uma das características contribuiu de maneira diferente para essa classificação e analisando a tabela 3 verificamos que a característica mais importante é a glucose e a menos importante é a espessura da pele. Para verificar se um indivíduo tem ou não diabetes a característica "glucose" tem uma influência muito grande na decisão final. Foi através de métodos disponibilizados pela biblioteca *Scikit-learn* que foi possível extrair estas métricas que permitem verificar a importância de cada uma das características.

O grupo 1 vai conter a informação e valores extraídos associados às características menos importantes e o grupo 2 vai conter a informação relativa às características mais importantes. A estratégia da criação de grupos é colocada em prática sempre que estejam a ser analisados elementos que não sejam comuns entre os diversos *datasets* em análise, como a distribuição (média, desvio padrão, mínimo, máximo e quartis), correlação entre características assim como a sua importância. Tornando a explicação mais clara para o exemplo da distribuição dos dados, existirá o grupo 1 com a distribuição das características menos importantes e o grupo 2 irá conter os dados referentes à distribuição dos dados das características mais importantes. Em termos práticos e recorrendo à tabela 3 verificamos que as características *SkinThickness*, *Insulin*, *Pregnancies*, *BloodPressure* iriam pertencer ao grupo 1 e as características *Glucose*, *BMI*, *DiabetesPedigreeFunction*, *Age* iriam pertencer ao grupo 2. É através do processo de seleção da importância de cada uma das características detalhado mais à frente que vai ser possível definir os dois grupos.

4.2.1 Descrição detalhada dos elementos caracterizadores extraídos.

Tabela 4 - Elementos Extraídos – Informação das amostras

Atributo / Elemento caraterizador	Justificação
NSamples	Número de amostras que o *dataset* contém.
NFeatures	Número de características que o *dataset* contém.
NSamplesClass1	Número de amostras que o *dataset* contém referentes à classe 1.
NSamplesClass2	Número de amostras que o *dataset* contém referentes à classe 2.
ClassBalance	Percentagem referente ao balanceamento entre as duas classes.

É através do número de amostras que o *dataset* possui que é possível perceber a dimensão deste assim como a quantidade de dados que este possui. É possível verificar o número de características que cada *dataset* possui através do atributo/elemento número de características. É extraído também o número de amostras que cada classe contém assim como a dimensão de amostras que existe no *dataset*. Outra métrica extraída de cada *dataset* foi o balanceamento das classes, ou seja, é neste momento que é recolhida a informação associada ao balanceamento das classes através do número

de amostras que existe para a classe 1 e classe 2. Através da extração desta informação permite já ter uma perceção inicial da dimensão dos dados a analisar. (Ver tabela 4)

A distribuição dos valores das características foram também extraídos, nomeadamente a média, desvio padrão, mínimo, máximo e quartis de cada característica tendo em conta as duas classes e cada classe em individual que permite então perceber se os dados seguem uma certa tendência, assim como a extração de padrões que possam existir nos dados. É possível também observar como os dados estão distribuídos por cada característica. Relembro que nesta etapa foram criados 2 grupos sendo que o primeiro grupo contém a informação de todas as características com importância menor ao contrário do grupo 2 que contém as características com importância mais elevada. É possível verificar os elementos caracterizadores extraídos nas tabelas 5, 6 e 7.

Tabela 5 - Elementos Extraídos – Distribuição das duas classes e dos dois grupos

Atributo / Elemento caraterizador	Justificação
DistAllClassesGroup1Mean	Média dos valores das duas classes do grupo 1, ou seja, a média dos valores das características menos importantes.
DistAllClassesGroup1Std	Desvio padrão dos dados do *dataset* das duas classes pertencentes ao grupo 1.
DistAllClassesGroup1Min	Valor mínimo das duas classes do grupo 1.
DistAllClassesGroup125	Valor pertencente ao primeiro quartil das duas classes do grupo 1.
DistAllClassesGroup150	Valor pertencente ao segundo quartil ou mediana das duas classes do grupo 1.
DistAllClassesGroup175	Valor pertencente ao terceiro quartil das duas classes do grupo 1.
DistAllClassesGroup1Max	Valor máximo das duas classes do grupo 1.
DistAllClassesGroup2Mean	Média dos valores das duas classes do grupo 2, ou seja, a média dos valores das características mais importantes.
DistAllClassesGroup2Std	Desvio padrão dos dados do *dataset* das duas classes pertencentes ao grupo 2.
DistAllClassesGroup2Min	Valor mínimo das duas classes do grupo 2.
DistAllClassesGroup225	Valor pertencente ao primeiro quartil das duas classes do grupo 2.
DistAllClassesGroup250	Valor pertencente ao segundo quartil ou mediana das duas classes do grupo 2.
DistAllClassesGroup275	Valor pertencente ao terceiro quartil das duas classes do grupo 2.
DistAllClassesGroup2Max	Valor máximo das duas classes do grupo 2.

É possível verificar pela tabela 5 a extração dos elementos caracterizadores/atributos associados à distribuição dos valores das duas classes e dos dois grupos. Sendo o mesmo processo realizado para a classe 1 do grupo 1 e 2 assim como para a classe 2 do grupo 1 e 2.

Tabela 6 - Elementos Extraídos - *Outliers*

Atributo / Elemento caraterizador	Justificação
OutliersAllClasses	Valores considerados *outliers* nas duas classes.
OutliersPercAllClasses	Percentagem de características com valores considerados *outliers* nas duas classes.
OutliersClass1	Valores considerados *outliers* na classe 1.
OutliersPercClass1	Percentagem de características com valores considerados *outliers* na classe 1.
OutliersClass2	Valores considerados *outliers* na classe 2.
OutliersPercClass2	Percentagem de características com valores considerados *outliers* na classe 2.

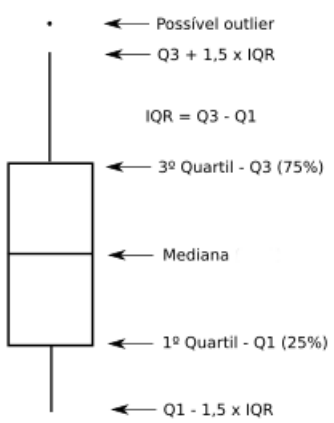

Figura 13 - Método de deteção de outliers. Fonte: [36]

Foi registado também o número de dados que se encontram fora do padrão global de uma distribuição (tendência normal) denominado *outlier*. Tipicamente um dado é identificado como sendo um *outlier* quando estiver a mais de 1,5*AQ (Terceiro Quartil – Primeiro Quartil) acima do 3ºQuartil (*outliers* superiores) ou abaixo do 1ºQuartil (*outliers* inferiores). Ver figura 13. Os dados associados a cada uma das características existentes foram processados no sentido de identificar *outliers*. Permitem então verificar a qualidade da distribuição dos dados. Foram extraídas também as percentagens de características que não contêm qualquer tipo de valor considerado como *outlier*. (Ver tabela 6)

Tabela 7 - Elementos Extraídos – Valores em falta

Atributo / Elemento caraterizador	Justificação
MissgingValuesAllClasses	Valores em falta em cada característica nas duas classes.
MissgingValuesPercAllClasses	Percentagem de características com valores em falta nas duas classes.
MissgingValuesClass1	Valores em falta em cada característica na classe 1.
MissgingValuesPercClass1	Percentagem de características com valores em falta na classe 1.
MissgingValuesClass2	Valores em falta em cada característica na classe 2.
MissgingValuesPercClass2	Percentagem de características com valores em falta na classe 2.

O número de valores em falta de cada uma das características foi também extraído, através deste método foi possível também registar em percentagem o número de características com pelo menos 1 valor em falta. A partir desta abordagem será possível perceber a quantidade de dados que estão em falta tendo em conta os dados existentes. Relembro que os valores em falta foram preenchidos na totalidade como mencionado anteriormente. (Ver tabela 7).

Tabela 8 - Elementos Extraídos – Correlação dos dois grupos

Atributo / Elemento caraterizador	Justificação
CorrelationGroup1	Correlação entre as características pertencentes ao grupo 1.
CorrelationGroup2	Correlação entre as características pertencentes ao grupo 2.

A correlação entre as características existentes num determinado *dataset* foi igualmente registado no sentido de perceber e entender melhor a interdependência entre duas variáveis. O coeficiente de correlação entre duas variáveis pode assumir um valor entre -1 e +1. Se uma variável tende a aumentar à medida que outras diminuem, o coeficiente é negativo, caso contrário, se as duas variáveis tendem a aumentar em conjunto o coeficiente de correlação é positivo. O primeiro grupo contém a soma das correlações das características menos importantes sendo que o segundo grupo contém a soma das correlações das características mais importantes. (Ver tabela 8). Através deste processo é possível observar como as características se relacionam, seja através de uma tabela ou processo semelhante é possível verificar os coeficientes de correlação entre características. A correlação é calculada aos pares, ou seja, para a característica X será possível observar como esta se vai relacionar com a característica Y e com as restantes características. Será então identificada como correlação positiva quando estas se movem na mesma direção e negativa quando em direções opostas, ou seja, mede a força da relação linear entre duas características.

Tabela 9 - Elementos Extraídos – Importância das características dos dois grupos

Atributo / Elemento caraterizador	Justificação
FeaturesImportanceGroup1	Percentagem da importância das características pertencentes ao grupo 1.
FeaturesImportanceGroup2	Percentagem da importância das características pertencentes ao grupo 2.

Por último foi recolhido a informação referente à importância de cada característica que o *dataset* contém. Essencialmente uma característica é considerada importante ou não baseada no contributo que esta ofereceu para a previsão da variável "*target*". (Ver tabela 9).

No anexo B é possível verificar o *dataset* base gerado em que cada uma destas características apresentadas anteriormente vai ter um valor associado, valor esse que foi extraído dos *datasets* em análise.

4.3 Criação do *meta-dataset*

Tabela 10 - *Meta-dataset* gerado

Dataset Number	*NSamples*	*NFeatures*	*NSamples Class1*	*NSamples Class2*	*DistAllClasses Group1Mean*	. . .	*Features Importance Group1*	*Features Importance Group2*
1	116,00	9,00	52,00	64,00	604,82	. . .	0,14	0,86
2	5300,00	2,00	2924,00	2376,00	0,00	. . .	0,37	0,63
3	5001,00	7,00	2500,00	2501,00	20,00	. . .	0,10	0,90
4	1151,00	19,00	540,00	611,00	115,55	. . .	0,19	0,81
5	1340,00	19,00	509,00	831,00	19,01	. . .	0,23	0,77
6	3020,00	4,00	1283,00	1737,00	5,25	. . .	0,17	0,83
7	569,00	5,00	212,00	357,00	14,22	. . .	0,14	0,86
8	80,00	5,00	34,00	46,00	1,01	. . .	0,26	0,74
9	14635,00	24,00	10945,00	3690,00	3,82	. . .	0,04	0,96
10	68783,00	11,00	34742,00	34041,00	3,52	. . .	0,10	0,90
11	1340,00	19,00	509,00	831,00	14,60	. . .	0,23	0,77
12	90,00	6,00	42,00	48,00	92,82	. . .	0,10	0,90
13	748,00	4,00	570,00	178,00	1384,19	. . .	0,31	0,69
14	690,00	14,00	383,00	307,00	14,15	. . .	0,09	0,91
15	306,00	3,00	225,00	81,00	4,03	. . .	0,28	0,72
16	1372,00	4,00	762,00	610,00	0,21	. . .	0,18	0,82
17	768,00	8,00	500,00	268,00	173,29	. . .	0,20	0,80
18	14980,00	14,00	8257,00	6723,00	30039,37	. . .	0,35	0,65
19	100,00	9,00	12,00	88,00	2,40	. . .	0,08	0,92
20	270,00	13,00	150,00	120,00	153,44	. . .	0,21	0,79
21	17898,00	8,00	16259,00	1639,00	127,55	. . .	0,08	0,92
22	90,00	7,00	19,00	71,00	9,40	. . .	0,05	0,95
23	583,00	10,00	416,00	167,00	121,27	. . .	0,27	0,73
24	19020,00	10,00	12332,00	6688,00	7,06	. . .	0,23	0,77
25	5000,00	13,00	3530,00	1470,00	61,57	. . .	0,18	0,82
26	245057,00	3,00	50859,00	194198,00	125,07	. . .	0,10	0,90
27	3333,00	10,00	2850,00	483,00	203,50	. . .	0,26	0,74

A presente secção visa fundamentalmente agregar as duas últimas secções, ou seja, reunir a informação estudada/extraída em apenas um local como representado na tabela 10 (*meta-dataset*).

Tendo em conta a figura acima ilustrada, este será o *meta-dataset* gerado. Cada amostra representa a informação extraída de cada um dos *datasets*, simplificando, no primeiro *dataset* analisado verificamos que tem 116 amostras, 9 *features*, 52 amostras da classe 1, 64 amostras da classe 2 e seguindo a mesma lógica para todos os atributos enumerados e detalhados na secção anterior.

Foram então analisados 27 *datasets*, onde foi possível recolher os dados associados aos atributos propostos. O processo de recolha de dados de cada *dataset* foi feito de forma individual e igual para todos os 27 *datasets* a análise.

4.4 Formato de um *dataset*

Esta secção tem como objetivo a demonstração e explicação de um *dataset* analisado.

Tabela 11 - *Dataset* Crioterapia

Sexo	*Idade*	*Tempo*	*Número de Verrugas*	*Tipo*	*Área*	*Resultado do tratamento*
1	35	12	5	1	100	0
1	29	7	5	1	96	1
1	50	8	1	3	132	0
1	32	11,75	7	3	750	0
1	67	9,25	1	1	42	0
1	41	8	2	2	20	1
1	36	11	2	1	8	0
1	59	3,5	3	3	20	0
1	20	4,5	12	1	6	1

O *dataset* da tabela 11 contém uma pequena amostra de um estudo da crioterapia que é um tratamento que usa baixas temperaturas para tratamentos estéticos e terapêuticos na pele. Para que seja possível verificar se o resultado de um determinado tratamento teve ou não efeito, torna-se necessário recolher alguns dados associados a algumas características, tais como o sexo da pessoa, a idade, o tempo do tratamento, número de verrugas, tipo de tratamento, área e, por fim, se o tratamento teve ou não efeito. Este *dataset* contém (apenas nesta figura ilustrada) 10 amostras de pessoas que tendo em conta estas características obtiveram o resultado ilustrado na figura (1-Sucesso, 0-Insucesso).

Tabela 12 - *Dataset* Sangue Doado

Meses desde última doação	*Número doações*	*Total sangue doado*	*Meses desde primeira doação*	*Sangue doado marco 2007*
2	50	12500	98	1
0	13	3250	28	1
1	16	4000	35	1
2	20	5000	45	1
1	24	6000	77	0
4	4	1000	4	0
2	7	1750	14	1
1	12	3000	35	0
2	9	2250	22	1
5	46	11500	98	1

Este *dataset* (tabela 12), contém 10 amostras (pequena amostra retirada) de um estudo feito que permite verificar, tendo em conta as amostras mencionadas no *dataset*, se um determinado indivíduo doou sangue em Março de 2007.

4.5 Apresentação do *meta-dataset*

Segue-se agora uma apresentação sumária de cada um dos 27 *datasets*. Essencialmente será apresentada a informação referente ao número de amostras que cada classe possui, número de características, distribuição dos dados nos diferentes *datasets*, valores em falta, correlação entre características, número de *outliers*, balanceamento das classes e a importância das características.

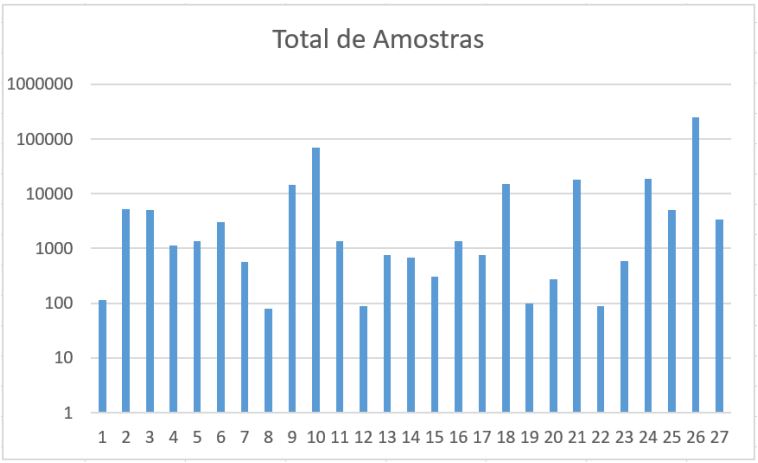

Figura 14 - Total das amostras

Pela figura 14 é possível analisar e verificar o total de amostras presentes em cada um dos 27 *datasets* a análise em escala logarítmica.

Analisando então o gráfico podemos observar que o *dataset* com menos amostras é o número 8 que contém 80 amostras, o *dataset* com mais amostra é o número 26 que conta com 245057 amostras sendo que a média de amostra se situa nas 15245.

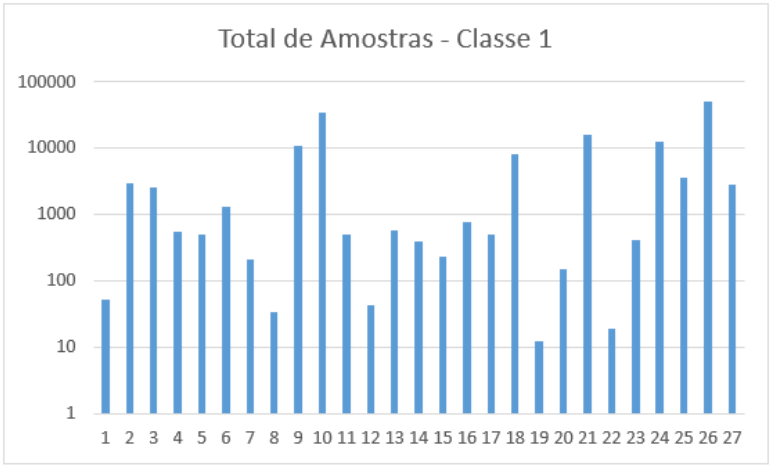

Figura 15 - Total das amostras da classe 1

No que respeita à figura 15 que visa apresentar o número de amostras que existem nos diversos *datasets* associados à classe número 1 podemos verificar que o *dataset* com menos amostras é o número 19 que "apenas" apresenta 12 amostras, no entanto o *dataset* com maior número de amostras é o número 26 que apresenta 50859 amostras, sendo que a média da amostra se situa nas 5608.

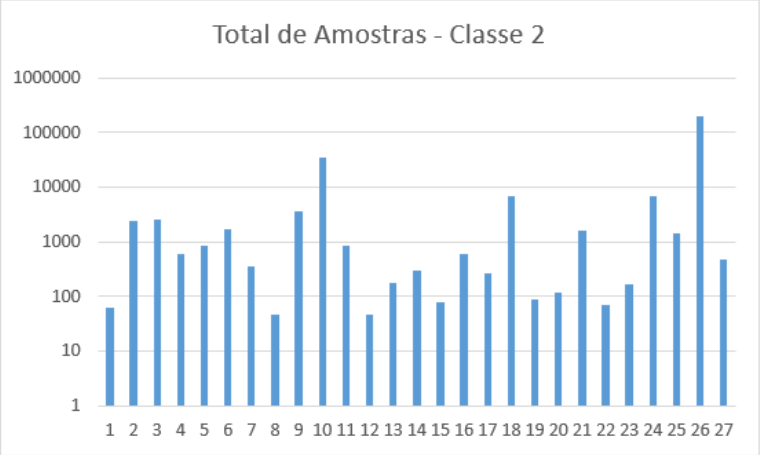

Figura 16 - Total das amostras da classe 2

Já a classe número 2 e como é possível verificar pela figura 16 o *dataset* com menos amostras é o número 8 que conta com 46 amostras e o número 26 com 194198 amostras sendo o que contém o número de amostras mais elevado, no entanto a média de amostras da classe número 2 é de 9637.

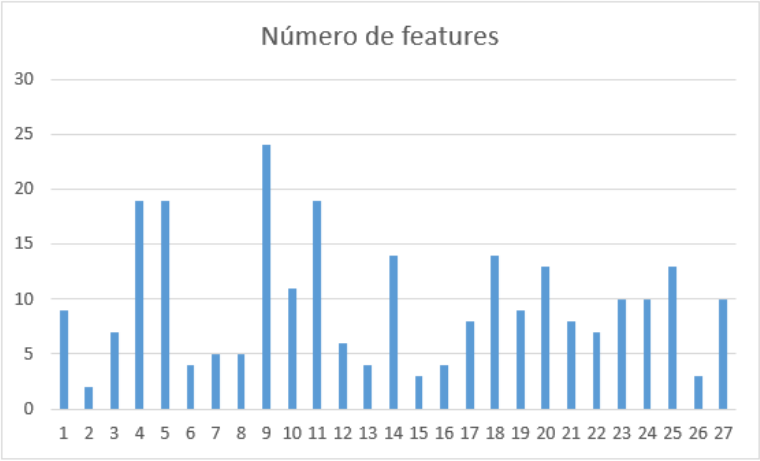

Figura 17 - Número de Características

Analisando então o número de características que cada *dataset* contém podemos verificar pela figura 17 que o *dataset* com menos características é o número 2 que contém "apenas" 2 e o *dataset* número 9 é o que contém mais características que conta com 24, no entanto a média de características dos diversos *datasets* é de 9.

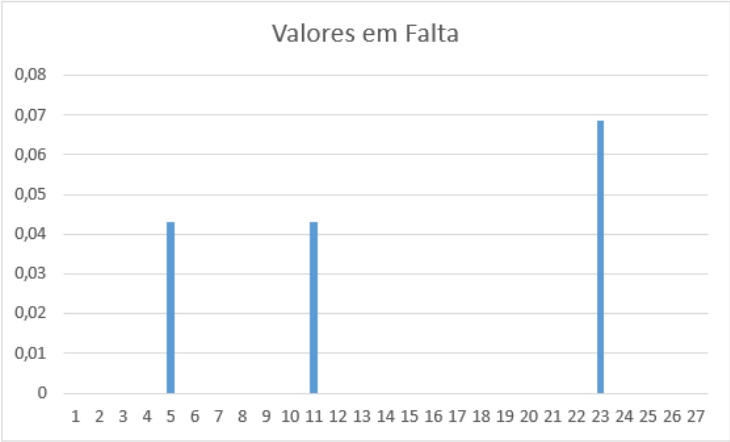

Figura 18 - Valores em falta nos *datasets*

Pela figura 18 podemos verificar o número de valores em falta existentes em cada um dos *datasets*. Apenas 3 *datasets* contêm valores em falta sendo que cada um destes sofreu um processamento de maneira a lidar com os valores em falta, e como justificado na secção "Construção do *Meta-dataset*" a estratégia passou por substituir os valores em falta.

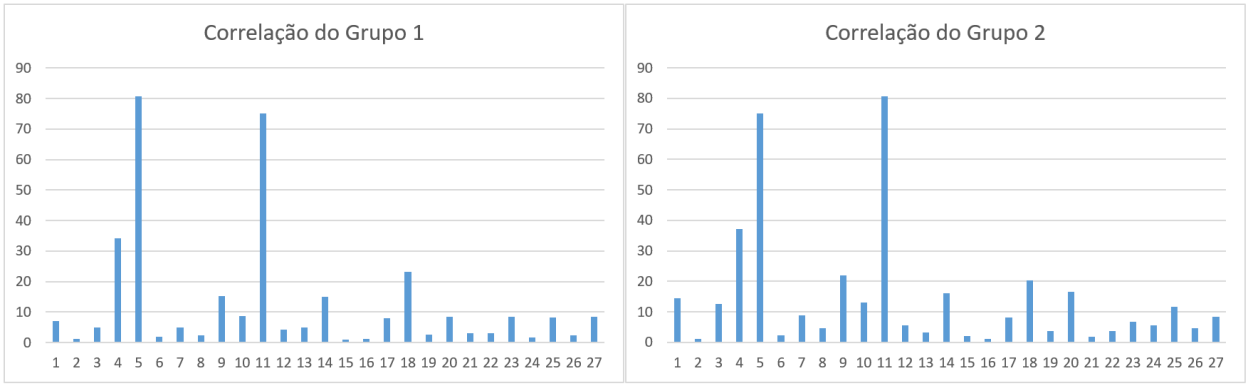

Figura 19 - Correlação do grupo 1 e 2

Através da figura 19 é possível verificar a correlação existente entre as características dos dois grupos. É um facto que existe uma relação desconhecida e complexa entre as características nos diversos *datasets*. E é então neste momento que se extraiu essa mesma correlação entre as diversas

características nos dois grupos. Através desta extração é possível então perceber os *datasets* que contêm características com correlação mais elevada e mais reduzida.

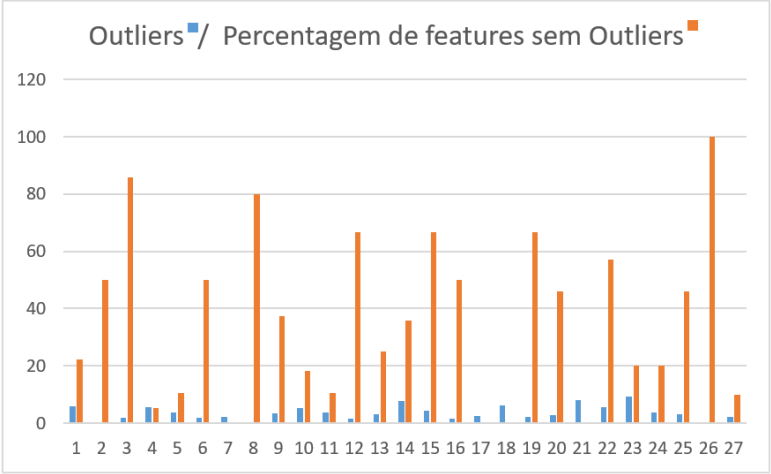

Figura 20 - Número de *Outliers*

É importante também demonstrar o número de *outliers* existentes entre cada um dos *datasets*. Analisando convenientemente a figura 20 podemos concluir que o número de *outliers* presente em cada um dos *datasets* é bastante reduzido. Dado o exemplo do *dataset* 26 é possível verificar a ausência de *outliers* significando isto que 100% das características presentes não contêm 1 único *outlier*.

4.6 Classificação das amostras do *meta-dataset*

Uma vez que este estudo se centra essencialmente num problema de aprendizagem supervisionada é necessário então atribuir uma *label* a cada amostra do *meta-dataset* gerado anteriormente. Relembrando que estamos perante um problema de classificação, querendo isto dizer, que neste momento temos um *dataset* constituído por diversas amostras (características de 27 amostras) no qual agora é o momento de classificar cada uma destas. Cada *dataset* (dos 27 existentes) irá ser processado por 4 árvores de decisão diferentes sendo que a árvore que apresentar melhores resultados será a escolhida para classificar a amostra. (Ver figura 21). Essencialmente queremos descobrir o classificador ideal para um dado problema.

Figura 21 - Atribuição de *label/target* aos diferentes *datasets*

Para que seja possível classificar cada um dos *datasets* é necessário criar diversos tipos de árvores de decisão diferentes (ver figura 22). Imagine-se que cada árvore de decisão representa um algoritmo, por outras palavras, que se um *dataset* obtiver melhores resultados na árvore de decisão número 1 significa que o algoritmo que melhor se adequa ao *dataset* em específico é o algoritmo número 1. O objetivo passa por colocar uma *label* (classificar) em todos os *datasets* através da escolha da árvore de decisão ideal.

Árvore 1
Critério: Gini
Profundidade da árvore: 8

Árvore 2
Critério: Gini
Profundidade da árvore: 12

Árvore 3
Critério: Entropy
Profundidade da árvore: 12

Árvore 4
Critério: Entropy
Profundidade da árvore: 8

Figura 22 - Tipos de árvores de decisão

O processo de criação de cada uma das árvores de decisão foi igual, ou seja, cada uma foi criada tendo como base os métodos disponibilizados pela biblioteca *Scikit-learn* que permite criar uma árvore de decisão onde apenas é necessário redefinir e otimizar alguns parâmetros.

```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0) ¶
```

Figura 23 - Parâmetros de criação de uma Árvore de decisão

Na figura 23 é possível então verificar os parâmetros *default* de criação de cada uma das árvores. Para este estudo em específico apenas foi necessário alterar o critério de separação assim como a

profundidade de cada uma das árvores como é possível verificar através da figura 22. Foram os 4 parâmetros selecionados presentes na figura 22 os escolhidos para alteração uma vez que são parâmetros simples que permitem criar árvores de decisão diferentes.

Depois de detetado qual a árvore de decisão com maior precisão apenas é necessário preencher o novo *dataset* gerado na secção anterior (criação do *meta-dataset*) na característica target e repetir o mesmo sistema para todos os *datasets* e desta forma classificar os *datasets*.

O presente processo foi realizado iterativamente sendo que em cada iteração cada um dos 27 *datasets* a estudo (já pré-processado anteriormente) como input onde foi posteriormente dividido em treino (67% dos dados) e teste (33% dos dados) do modelo. Parte dos *datasets* analisados tinham as classes balanceadas. A estratégia passou por não balancear os *datasets* "desequilibrados" uma vez que representavam uma minoria, adicionando o facto que o objetivo passava por analisar apenas os dados existentes. No entanto existe algum impacto na estratégia definida uma vez que para os *datasets* que não têm as classes balanceadas o modelo terá a tendência de classificar novos dados como pertencentes à classe que possui mais exemplos.

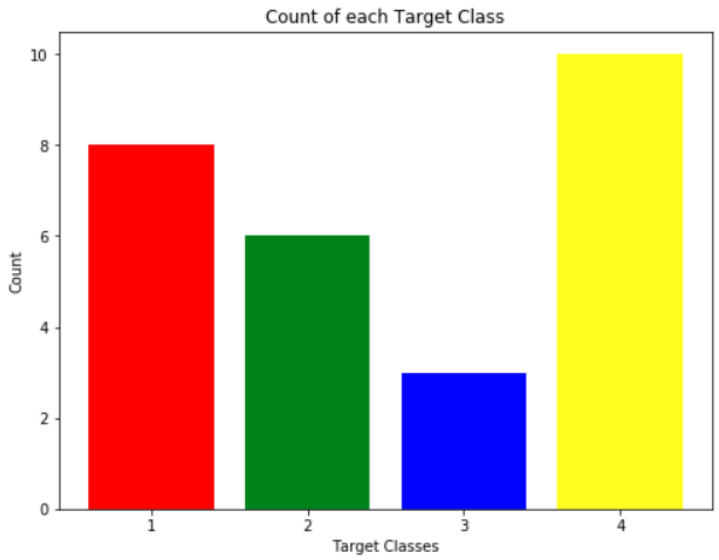

Figura 24 – Número de amostras do *Meta-dataset* gerado

Uma vez criado o *meta-dataset* é possível agora retirar algumas ilações dos dados obtidos. Existem então 4 classes diferentes com amostras também estas diferentes (ver figura 24). Cada classe contém um número de amostras onde todas somadas representam os 27 *datasets* a estudo. Podemos verificar que na classe número 4 existem 10 amostras(*datasets*), o que revela que cada uma destas após serem processadas pelas 4 árvores de decisão a árvore número 4 foi a que apresentou melhor *accuracy*, sendo aplicado o mesmo método para as restantes classes.

Existe uma clara diferença de representatividade entre as classes existentes sendo a mais evidente é a classe número 3 que apenas contém 3 amostras quando comparado com a classe número 4 que contém 10 amostras. É seguro concluir que existe um desequilíbrio entre as classes existentes.

Tabela 13 - Classificação das amostras do meta-dataset

Dataset Number	*NSamples*	*NFeatures*	*NSamples Class1*	*NSamples Class2*	*DistAllClasses Group1Mean*	*. . .*	*Features Importance Group1*	*Features Importance Group2*	*Target*
1	116,00	9,00	52,00	64,00	604,82	. . .	0,14	0,86	2
2	5300,00	2,00	2924,00	2376,00	0,00	. . .	0,37	0,63	1
3	5001,00	7,00	2500,00	2501,00	20,00	. . .	0,10	0,90	1
4	1151,00	19,00	540,00	611,00	115,55	. . .	0,19	0,81	2
5	1340,00	19,00	509,00	831,00	19,01	. . .	0,23	0,77	4
6	3020,00	4,00	1283,00	1737,00	5,25	. . .	0,17	0,83	4
7	569,00	5,00	212,00	357,00	14,22	. . .	0,14	0,86	4
8	80,00	5,00	34,00	46,00	1,01	. . .	0,26	0,74	1
9	14635,00	24,00	10945,00	3690,00	3,82	. . .	0,04	0,96	1
10	68783,00	11,00	34742,00	34041,00	3,52	. . .	0,10	0,90	4
11	1340,00	19,00	509,00	831,00	14,60	. . .	0,23	0,77	4
12	90,00	6,00	42,00	48,00	92,82	. . .	0,10	0,90	3
13	748,00	4,00	570,00	178,00	1384,19	. . .	0,31	0,69	4
14	690,00	14,00	383,00	307,00	14,15	. . .	0,09	0,91	4
15	306,00	3,00	225,00	81,00	4,03	. . .	0,28	0,72	1
16	1372,00	4,00	762,00	610,00	0,21	. . .	0,18	0,82	2
17	768,00	8,00	500,00	268,00	173,29	. . .	0,20	0,80	1
18	14980,00	14,00	8257,00	6723,00	30039,37	. . .	0,35	0,65	3
19	100,00	9,00	12,00	88,00	2,40	. . .	0,08	0,92	2
20	270,00	13,00	150,00	120,00	153,44	. . .	0,21	0,79	2
21	17898,00	8,00	16259,00	1639,00	127,55	. . .	0,08	0,92	4
22	90,00	7,00	19,00	71,00	9,40	. . .	0,05	0,95	1
23	583,00	10,00	416,00	167,00	121,27	. . .	0,27	0,73	2
24	19020,00	10,00	12332,00	6688,00	7,06	. . .	0,23	0,77	4
25	5000,00	13,00	3530,00	1470,00	61,57	. . .	0,18	0,82	4
26	245057,00	3,00	50859,00	194198,00	125,07	. . .	0,10	0,90	3
27	3333,00	10,00	2850,00	483,00	203,50	. . .	0,26	0,74	1

Pela tabela 13 podemos então verificar o *meta-dataset* com cada uma das amostras com a *label/target* correspondente. Desta forma foi possível criar um *meta-dataset* contendo problemas de classificação. No anexo B é possível ver o *meta-dataset* completo.

5 Construção do Meta-modelo

Este capítulo tem como objetivo construir um meta-modelo que visa escolher a árvore de decisão que irá resolver um problema de classificação. Os dados recolhidos, analisados e detalhados nas secções anteriores servirão de input à construção do meta-modelo (*meta-dataset*). O meta-modelo a ser gerado será submetido a métricas de avaliação no intuito de avaliar a sua qualidade. Em termos práticos um modelo de aprendizagem de máquina necessita de validação antes de ser colocado em produção.

O objetivo passa por perceber com os dados que existem até ao momento o comportamento do meta-modelo através das métricas de avaliação.

No sentido de iniciar a construção do meta-modelo foi necessário recorrer à biblioteca *Scikit-learn* que disponibiliza métodos que permitem auxiliar a construção do algoritmo a implementar, que neste caso é a árvore de decisão. Existe, no entanto, um desafio que se prende no facto de parametrizar o algoritmo de maneira a conseguir extrair o máximo de performance possível. Para responder a este desafio de parametrização do algoritmo, o recurso *Gridsearch* veio dar uma forte ajuda uma vez que tem a capacidade de selecionar os melhores valores aquando da parametrização do algoritmo. Para que esta operação tenha efeito é necessário selecionar quais os parâmetros de construção do algoritmo da árvore de decisão que é necessário otimizar, definir o número de vezes que queremos testar o meta-modelo (validação cruzada) e a métrica de avaliação. Vamos também usar a amostragem estratificada que tem como grande objetivo garantir que os subconjuntos do CV possuem uma amostragem representativa de cada uma das classes, bastante útil e utilizada quando estamos perante classes não balanceadas. Desta forma é possível garantir que os dados de treino contenham a mesma distribuição de dados, por outras palavras cada subconjunto terá amostras de todas as classes.

Como mencionado anteriormente o classificador foi construído recorrendo à biblioteca *Scikit-learn* que permite selecionar os parâmetros a serem usados. Para este teste e através da otimização de parâmetros efetuado pelo *Gridsearch* podemos concluir os seguintes valores para os seguintes parâmetros:

- *criterion*='*entropy*'
- *max_depth*=3
- *max_leaf_nodes*=5
- *min_samples_split*=2
- *splitter*='*best*'

```
dt_classifier = DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=3,
                    max_features=None, max_leaf_nodes=5,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, presort=False,
                    random_state=42, splitter='best')
```

Figura 25 - Parametrização do Algoritmo

Através da figura 25 podemos verificar a construção do classificador – já com os parâmetros otimizados – recorrendo à biblioteca *Scikit-learn*.

Uma vez o meta-modelo criado é altura de o avaliar. As técnicas de avaliação utilizadas foram a matriz de confusão, *accuracy*, *f1*, *precision* e *recall*. A técnica de validação cruzada também foi utilizada durante os testes do meta-modelo tendo sido apenas indicado o número de "partições" que após alguns testes se conclui que o valor mais indicado são 2 partições.

5.1.1 Resultados do meta-modelo 1 e análise crítica

Tabela 14 - Matriz de confusão - Meta-Modelo 1

		Classe Prevista			
		Classe 1	Classe 2	Classe 3	Classe 4
Classe Real	Classe 1	4	1	3	0
	Classe 2	1	1	4	0
	Classe 3	1	0	0	2
	Classe 4	3	2	0	5

Pela observação da matriz de confusão e fazendo uma análise crítica, o modelo tem uma grande dificuldade de classificar as amostras que pertençam a classes minoritárias, um comportamento bastante comum em conjuntos de dados desequilibrados. O modelo no que respeita às classes com mais amostras (1 e 4) conseguiu, embora com alguma confusão, classificar corretamente algumas amostras (50%). Observando a diagonal da matriz de confusão na tabela 14 podemos observar as amostras que o modelo classificou corretamente, no entanto, e devido às classificações de classes incorretas feitas pelo modelo, sobretudo aquando das classes minoritárias e as poucas que classificou corretamente nas classes maioritárias leva-nos a classificar com sendo um mau modelo, uma vez que nem metade das amostras classificou corretamente.

Tabela 15 - Meta-Modelo 1 - Métricas de avaliação

Métricas	Valor
Accuracy	37%
F1 micro	37%
F1 macro	31%
Precision micro	37%
Precision macro	35%
Recall micro	37%
Recall macro	29%

Discussão de algumas conclusões e consequente análise crítica dos resultados ilustrados na tabela 15.

A métrica "*accuracy*" é de 37%. É possível concluir que em cada 10 amostras o modelo consegue selecionar corretamente quase 4, ou seja, os verdadeiros positivos. Tendo em conta que o modelo apresenta uma taxa de 37% em selecionar corretamente as amostras pertencentes a cada classe, sendo que maioritariamente dessas amostras pertencem às classes mais representativas, posso concluir que estamos perante um modelo que não apresenta bons resultados. Visto que o objetivo passa por selecionar o melhor algoritmo de análise de dados e tendo em conta que nem 50% das amostras o modelo classifica corretamente não podemos considerar um bom modelo.

A métrica "*precision*" oscila também nos 35% a 37%. Através da métrica *precision* é possível observar de todos os dados classificados como positivos pelo modelo, quantos estão corretos. Podemos observar que o modelo consegue selecionar com uma percentagem a rondar os 36% as amostras positivas no universo das amostras consideradas pelo modelo como sendo positivas. O que nos leva a concluir que das 27 amostras existentes apenas 10 amostras foram classificadas corretamente. É uma taxa muito baixa uma vez que a escolha do algoritmo de análise de dados necessita de ser bem atribuído sob pena de no futuro ser necessário aplicar outro algoritmo para o problema em questão.

A métrica "*recall*" ronda entre os 29% a 37%. A presente métrica permite identificar a percentagem de dados classificados como positivos quando comparado com a quantidade real de positivos que existem na amostra total.

A métrica "*f1*" oscila nos 31% a 37%, significando então que a métrica que permite unir as métricas, precisão e *recall* em apenas uma, permitindo desta forma identificar a qualidade geral do modelo gerado.

Essencialmente as diferenças das métricas precision/recall/f1 micro é que permitem agregar as contribuições de todas as classes e calcular posteriormente a média ao contrário das métricas precision/recall/f1 macro que visam calcular as contribuições de cada classe de forma independente e posteriormente calcular a média no fundo trata as classes de igual forma e peso. Concluímos então que é possível prever com mais eficiência através das métricas precision/recall/f1 micro uma vez se trata de classes não balanceadas onde o algoritmo tende a classificar mais eficientemente as classes mais representadas.

Ou seja, tendo em conta todas as métricas mencionadas e observadas acima e sendo fundamental conseguir construir um modelo que seja capaz de identificar pelo menos a grande maioria das amostras corretamente é possível dizer que este não é o modelo indicado. Uma vez e como mencionado anteriormente este modelo apenas consegue identificar corretamente o algoritmo de decisão mais apropriado em cerca de 37% dos casos.

5.1.2 Resultados do meta-modelo 2 e análise crítica

A verdade é que os resultados do modelo apresentado anteriormente não foram os melhores tendo em conta as métricas de avaliação, como por exemplo o modelo gerado não conseguiu identificar corretamente cada uma das amostras (verdadeiros positivos) com pelo menos uma taxa de 50%. Uma das causas associadas a este "insucesso" pode estar relacionada com o facto de as classes não estarem balanceadas. No sentido de verificar se com os dados balanceados os resultados melhoram, um novo modelo será construído, mas agora com as classes balanceadas.

Existe um método que permite balancear as classes através da aplicação de técnicas como o *oversampling* e *undersampling*. Basicamente esta técnica permite equilibrar o número de amostras em cada classe adicionando ou removendo amostras das classes. Num caso onde existam duas classes desequilibradas, existem duas técnicas possíveis no sentido de as balancear. A primeira técnica passa por remover amostras da classe mais representativa aplicando assim a técnica de *undersampling*, sendo a segunda técnica a adição de amostras à classe menos representativa aplicando desta forma a técnica de *oversampling*.

Neste teste será aplicado a técnica de *oversampling*, ou seja, serão adicionadas novas amostras nas classes menos representativas de maneira que todas as classes fiquem com o mesmo número de amostras.

Importante relembrar que apenas foi aplicada a técnica de *oversampling* às amostras destinadas aos dados de treino do modelo de cada subconjunto separadamente, de maneira a não perder a capacidade de generalização para novos dados, o que poderia ocorrer se o *oversampling* fosse aplicado antes do CV. O processo mencionado é garantido através da aplicação de uma *pipeline* do pacote *imblearn* que de uma forma automática não permite realizar o *oversampling* nos dados de teste. Realçar que a técnica de *oversampling* aplicada neste teste passou por duplicar dados já existentes. O modelo tem de ser capaz de conseguir adaptar a novos dados, caso isso não aconteça

podemos estar perante *overfitting*, sendo que o *cross-validation* ajuda a evitar o *overfitting* uma vez que permite criar diversos subconjuntos de dados de treino(k-1) sendo que existe sempre um subconjunto dedicado aos testes do modelo, através deste método permite que o algoritmo não se ajuste demasiado aos dados. Para a construção do algoritmo e seguindo a lógica anterior os parâmetros usados pelo algoritmo já otimizados pelo *Gridsearch* foram os seguintes: *criterion='entropy'*, *max_depth*=5, *max_leaf_nodes*=7, *min_samples_split*=2, *splitter*='*best*'.

Tabela 16 - Matriz de confusão – Meta-Modelo 2

		Classe Prevista			
		Classe 1	Classe 2	Classe 3	Classe 4
Classe Real	Classe 1	4	0	2	2
	Classe 2	1	2	1	2
	Classe 3	0	1	1	1
	Classe 4	1	1	3	5

Podemos observar pela tabela 16 que identificou corretamente 50% das amostras correspondente à classe número 1 sendo que é visível ainda alguma confusão com as classes número 3 e 4. A classe número 2 conta com 2 amostras identificadas corretamente, no entanto, também esta apresenta alguma confusão com a classe 4 no qual o modelo identificou incorretamente 2 amostras como pertencentes à classe 2. Já a classe 3 o modelo claramente não conseguiu diferenciar a classe correta uma vez que identificou incorretamente 1 amostra como pertencente às classes 2 e 4 e apenas 1 amostra como pertencente à classe 3 corretamente. Já a classe 4 conseguiu identificar corretamente metade das amostras (5) mas é possível verificar alguma confusão com a classe número 3 que identificou incorretamente 3 amostras. Apesar do *oversampling* aplicado é percetível que o modelo continua a identificar corretamente mais amostras das classes maioritárias uma vez que estas continham amostras mais diversificadas permitindo extrair mais conhecimento. O modelo revelou a capacidade de identificar mais amostras pertencentes às classes 1 e 4, ao contrário das classes 2 e 3 que demonstraram a existência de alguma confusão entre classes.

Tabela 17 – Meta-Modelo 2 - Métricas de avaliação

Métricas	Valor
Accuracy	44%
F1 micro	44%
F1 macro	42%
Precision micro	44%
Precision macro	45%
Recall micro	44%
Recall macro	42%

Discussão de algumas conclusões e consequente análise crítica dos resultados obtidos ilustrados na tabela 17.

A métrica "*accuracy*" é de 44%, querendo isto dizer que o modelo conseguiu identificar corretamente 4 amostras em cada 10. Tendo em conta os resultados obtidos, o modelo tende a selecionar com uma taxa de 44% corretamente as amostras pertencentes a cada classe sendo que maioritariamente dessas amostras pertencem às classes mais representativas. É possível concluir que estamos perante um modelo com algumas deficiências mas que ainda assim ligeiramente melhor que o modelo anterior. Conclui-se também que a adição de novas amostras melhoraram o modelo, embora ligeiramente, uma vez que este dispunha de mais amostras/informação/conhecimento para a sua criação. É visível ainda uma ligeira redução da confusão entre classes quando comparado com o modelo anterior.

A métrica "*precision*" oscila entre os 44% a 45%, significando então que em todas as classificações de classes positivas que o modelo realizou, quantas estão efetivamente corretas. No entanto e analisando os resultados o presente modelo leva-nos a concluir que das 27 amostras existentes apenas 12 amostras foram classificadas corretamente. É então possível verificar um aumento de eficiência nesta métrica quando comparado com o modelo anterior.

A métrica "*recall*" oscila nos 42% a 44%, dito de outra forma, é a razão entre os casos positivos identificados corretamente pelo modelo tendo em conta os casos positivos reais. Uma vez mais comparando esta métrica com o modelo anterior houve novamente um aumento da performance. Este modelo construído ganhou a capacidade de conseguir identificar com mais eficácia a percentagem de dados classificados como positivos quando comparado com a quantidade real de positivos que existem na amostra total.

A métrica f1 oscila entre os 42% a 44%. A presente métrica tem como objetivo unir as métricas *precision* e *recall* que visam avaliar a qualidade do modelo.

Após a apresentação e análise dos dados apresentados por este modelo podemos concluir que em todos os métodos de avaliação a que o modelo foi sujeito apresentou um aumento de eficiência e eficácia na deteção das classes corretas. Permite então concluir que a abordagem sugerida de duplicar amostras existentes apresentou melhorias significativas na deteção e identificação das classes presentes na amostra.

5.1.3 Resultados do meta-modelo 3 e análise crítica

Após a apresentação dos dois modelos, uma nova abordagem será desenvolvida de maneira a construir um novo modelo que possa apresentar resultados mais positivos.

A implementação deste novo modelo tem como objetivo analisar o comportamento após a união de duas classes, ou seja, a junção de duas classes em que uma delas contém o menor número de amostras. Para implementar esta estratégia foi definido que a classe com menos amostras se vai fundir com a classe mais "semelhante", semelhança esta que foi detetada aquando da classificação dos *datasets*, por outras palavras na atribuição da *label*. Através deste processo as classes ficam um pouco mais balanceadas, e deixa de existir uma classe com um número tão reduzido de amostras, ao contrário da classe que recebeu estas amostras. As classes mais semelhantes detetadas foram as classes 2 e 3. Foi detetado que a classe 2 e 3 tinham uma grande semelhança na fase da atribuição da *label* (secção "Classificação das amostras do *meta-dataset*"). Esta semelhança foi observada devido à precisão apresentada por ambas, uma vez que era muito aproximada quando comparada com as restantes classes. No que respeita à criação do algoritmo e tendo em conta que o processo de criação é semelhante nos 3 modelos, os parâmetros utilizados pelo algoritmo já otimizados pelo *Gridsearch* foram os seguintes: *criterion*='*gini*', *max_depth*=1, *max_leaf_nodes*=2, *min_samples_split*=2, *splitter*='*random*'.

Figura 26 – Junção de duas classes

Depois de aplicada a técnica mencionada anteriormente nos dados a serem usados pelo algoritmo, é possível visualizar através da figura 26 a classe 1 que conta com 8 amostras, a classe 2 que conta agora com 9 amostras (classe 2 + classe 3) e a classe 4 que tem 10 amostras.

Tabela 18 - Matriz de confusão – Meta-Modelo 3

Classe Real		Classe Prevista		
		Classe 1	Classe 2	Classe 3
	Classe 1	2	5	1
	Classe 2	2	4	3
	Classe 3	2	3	5

Matriz de confusão: Após as modificações efetuadas anteriormente para a construção deste modelo (junção de duas classes), verificamos então, agora um universo de 27 amostras mas com apenas 3 classes. O modelo classificou corretamente 2 amostras como pertencentes à classe 1, no entanto classificou incorretamente 5 amostras como pertencentes à classe 2 e 1 amostra como pertencente à classe 3 mas que na realidade pertenciam à classe 1. No que respeita à classe 2 o modelo conseguiu identificar corretamente 4 amostras, no entanto classificou incorretamente 2 amostras como pertencentes à classe 1 e 3 amostras como pertencentes à classe 3. Já a classe 3 o modelo classificou corretamente 5 amostras mas, no entanto, classificou incorretamente 2 amostras como pertencentes à classe 1 e 3 amostras como pertencentes à classe 2. Ao analisar de uma forma mais aprofundada a matriz de confusão da tabela 18 é possível verificar pela sua diagonal que existe uma grande confusão entre classes. Na classe 1 o modelo identificou mais amostras como pertencentes à classe dois do que propriamente à classe 1. Na classe 2 e 3 o modelo conseguiu identificar mais amostras pertencentes à classe correta, no entanto existe ainda muita confusão com as demais amostras. Tendo em conta a matriz apresentada o modelo não consegue de uma forma clara distinguir e identificar corretamente as diversas amostras.

Tabela 19 – Meta-Modelo 3 - Métricas de avaliação

Métricas	Valor
Accuracy	41%
F1 micro	41%
F1 macro	40%
Precision micro	41%
Precision macro	41%
Recall micro	41%
Recall macro	40%

É possível também verificar pela tabela 19 as métricas resultado da avaliação do modelo de ML.

A métrica "*accuracy*" é de 41%. O modelo gerado conseguiu identificar corretamente os verdadeiros positivos com uma percentagem de 41%. Métrica bastante semelhante quando comparado com o modelo anterior. Uma vez mais este modelo apresenta resultados abaixo do esperado visto que nem 50% das amostras consegue classificar corretamente. Tendo em conta esta métrica é possível concluir que o modelo construído não apresenta bons resultados.

A métrica "*precision*" ronda os 41%, ou seja, permite observar as amostras que realmente são positivas dos dados classificados pelo modelo como sendo positivos.

A métrica "*recall*" ronda também entre os 40% e 41%. Permite perceber e entender melhor a razão entre os casos positivos identificados corretamente pelo modelo tendo em conta os casos positivos reais, dito de outra forma, é a soma dos falsos negativos e verdadeiros positivos.

A métrica f1 oscila entre os 40% a 41%. Relembro que a presente métrica visa unir as métricas *precision* e *recall* que têm como objetivo avaliar a qualidade do modelo.

A aplicação desta técnica que passou por unir as classes mais semelhantes não surtiu o efeito esperado que passava por unir as classes mais semelhantes e desta forma reduzir o número de classes com elevado grau de "confusão" entre si. No entanto e observando os resultados, é possível verificar alguma confusão entre as classes existentes, não sendo o modelo desta forma capaz de distinguir com uma percentagem positiva as classes corretas.

5.1.4 Conclusões dos meta-modelos desenvolvidos

Observando os resultados gerais, é possível verificar que os meta-modelos 2 e 3 apresentam métricas de avaliação muito semelhantes sendo que o meta-modelo 1 quando comparados com os outros dois meta-modelos fica aquém quando nos referimos à deteção e identificação dos

verdadeiros positivos (*accuracy*). Uma das razões para a baixa percentagem de classificações corretas determinadas no meta-modelo 1 prende-se no facto de não existiam amostras suficientes para que o meta-modelo possa aprender a classificar novas amostras, sempre que é necessário classificar um novo problema o meta-modelo tende a classificá-lo como pertencente à classe mais "abundante". No meta-modelo 2 e após se terem adicionado mais amostras (*oversampling*) permitiu ao meta-modelo ter mais informação e exemplos de maneira a classificar corretamente novas amostras, aumentando ligeiramente a percentagem de classificações corretas de novas amostras.

Nos 3 meta-modelos apresentados e observados anteriormente não permitem concluir que são bons meta-modelos porque nenhum dos 3 conseguiu identificar pelo menos 50% das amostras corretamente, por outras palavras, nenhum dos meta-modelos - dado um novo problema - conseguiria selecionar o melhor algoritmo de análise de dados com uma eficácia de pelo menos 50%. No entanto, e analisando cada um destes deve ser dito que o meta-modelo número 2 apresentou ligeiramente melhores resultados quando comparado com o meta-modelo 1 e 3. É possível concluir que usando esta população de amostras (27 amostras) e o facto de se terem realizado alterações nos dados, verificaram-se melhorias. O meta-modelo tinha ao seu dispor mais conhecimento/informação no qual permitiu classificar novas amostras e desta forma subtrair alguma confusão entre classes.

Uma das razões que poderia levar o meta-modelo a identificar corretamente mais amostras prende-se com o facto de se adicionar mais amostras. Uma vez adicionadas mais amostras o meta-modelo conseguiria desenvolver mais casos de teste e consequentemente identificar mais padrões entre os dados de maneira a classificar com uma maior eficácia e eficiência novos dados. Com os dados existentes e com o facto de existirem 4 classes, o meta-modelo tende a "confundir" algumas classes como pertencentes a uma classe incorreta, tipicamente à classe mais representativa.

6 Utilização do Meta-modelo para Classificação

O objetivo deste capítulo passa por demonstrar o comportamento do meta-modelo quando colocado em produção. Existe um problema de classificação novo que será "consumido" pelo meta-modelo gerado no capítulo anterior. O meta-modelo baseado em dados históricos pretende classificar um novo problema. Relembro que este estudo tem como objetivo criar um método automático para seleção do algoritmo de análise de dados.

O presente capítulo representa a realização de mais um dos objetivos que passou pela demonstração de utilização de um classificador com a capacidade de resolver problemas de classificação. Este classificador deve ser usado quando o objetivo passa por selecionar um algoritmo de análise de dados que permita analisar um determinado *dataset*. Desta forma o investimento despendido em busca do melhor algoritmo de análise de dados seria substituído por um classificador, diminuindo o tempo dedicado à investigação do melhor algoritmo. A presente abordagem pode ser usada pela parte interessada em automatizar o processo de seleção de algoritmo. Deve ser dito que o classificador classifica qualquer novo problema sempre baseado em dados históricos, validando e legitimando a classificação por este atribuído. Sendo que o objetivo do classificador é encontrar padrões nos dados semelhantes a padrões que o meta-modelo já aprendeu e desta forma conseguir fazer uma associação entre novos dados e os dados já treinados, validando a classificação final.

Fundamentalmente este capítulo será responsável por apresentar de uma maneira prática a utilização do meta-modelo gerado que servirá como ferramenta para resolver problemas. O meta-modelo número 2 foi o selecionado, uma vez que de entre os 3 meta-modelos estudados no capítulo anterior foi o que apresentou melhores resultados no sentido que quando comparado com os restantes conseguiu classificar mais amostras corretamente.

Outro objetivo passa por demonstrar a efetividade do classificador sempre que recebe um problema novo. Uma vez um novo problema classificado, o próximo passo passa por analisar se a classificação está ou não correta, caso não esteja, verificar se existe algum padrão de confusão entre classes. É isso que este capítulo pretende fazer, exemplificar o processo de classificação de um novo problema sendo em seguida alvo de validação e verificação.

Durante este capítulo, 6 problemas de classificação serão selecionados com o objetivo de o meta-modelo classificá-los, ou seja, sugerir a melhor árvore (classificação). Em cada um dos problemas (dataset) e após o seu processamento, segue-se a próxima fase que tem como objetivo a extração de conhecimento e informação de cada um, diga-se, as características a estes associados. A informação/conhecimento que cada *dataset* possui será posteriormente utilizado como dados de entrada ao meta-modelo. No sentido e com o intuito de validar as classificações realizadas pelo

meta-modelo cada um dos problemas mencionados serão classificados aplicando as 4 árvores de decisão ilustradas na figura 22. É um processo baseado na classificação do *meta-dataset* realizado no capítulo "Construção do *meta-dataset*". Este passo permite validar a classificação realizada pelo meta-modelo e verificar se a árvore escolhida foi ou não a mesma.

Cada um dos exemplos a utilizar de seguida têm como objetivo simular o mesmo processo e comportamento uma vez colocado em produção. Os 6 *datasets* escolhidos foram selecionados de um modo aleatório sem nenhuma objetividade ou preferência, sendo dos poucos requisitos o facto de serem numéricos, uma percentagem de valores em falta bastante reduzido assim como problemas de classificação binária. Sendo posteriormente alvo de análise e processamento no sentido de extrair informação/características de cada um destes simulando o processo referido no capítulo 4. É importante a simulação no sentido de demonstrar que é possível utilizar o classificador em produção assim como a sua capacidade de classificar novos problemas, essencialmente é uma validação e prova de conceito útil e relevante. Os exemplos implementados têm como objetivo dar a conhecer as capacidades do classificador, ou seja, uma vez "posto à prova" como este classifica novos dados e quantos destes estão ou não corretos.

É possível verificar os 6 *datasets* usados neste estudo nos Anexos A.

6.1 Preparação dos problemas

É durante esta fase que os 6 problemas/*datasets* serão sujeitos a extração da informação/características a estes associados, simulando desta forma 6 problemas a classificar.

Tabela 20 - *Dataset* Incêndios Florestais

Day	Month	Temperature	RH	Ws	Rain	FFMC	DMC	DC	ISI	BUI	FWI	Classes
1	6	29	57	18	0	65,7	3,4	7,6	1,3	3,4	0,5	0
2	6	29	61	13	1,3	64,4	4,1	7,6	1	3,9	0,4	0
3	6	26	82	22	13,1	47,1	2,5	7,1	0,3	2,7	0,1	0
4	6	25	89	13	2,5	28,6	1,3	6,9	0	1,7	0	0
5	6	27	77	16	0	64,8	3	14,2	1,2	3,9	0,5	0
6	6	31	67	14	0	82,6	5,8	22,2	3,1	7	2,5	1
7	6	33	54	13	0	88,2	9,9	30,5	6,4	10,9	7,2	1
8	6	30	73	15	0	86,6	12,1	38,3	5,6	13,5	7,1	1
9	6	25	88	13	0,2	52,9	7,9	38,8	0,4	10,5	0,3	0
…	…	…	…	…	…	…	…	…	…	…	…	…

É possível verificar pela tabela 20 um exemplo de um dos *datasets* que serão classificados pelo modelo. Este exemplo apresenta um conjunto de dados constituídos por 122 amostras representando informação sobre incêndios florestais na Argélia. O presente *dataset* inclui diversas

características que uma vez conjugadas entre si, permite revelar a presença ou ausência de incêndios florestais numa determinada zona.

Todo este processo inicia-se no estudo do *dataset* a classificar onde diversas características sobre o *dataset* serão extraídos tais como o número de amostras totais e por classe, número de características, distribuição dos valores, valores em falta, correlação entre características e outliers. É possível verificar este processo detalhado no capítulo: "Extração de informação de cada *dataset".*

Tabela 21 - *Dataset* Incêndios Florestais - Análise

NSamples	NFeatures	NSamplesClass1	NSamplesClass2	DistAllClassesGroup1Mean	DistAllClassesGroup1Std	...
122	12	63	59	120,998361	92,743475	...

O processo de extração dos elementos caracterizadores do *dataset* e o seu registo dará origem a um conjunto de informação devidamente organizada como é possível verificar pela tabela 21. A tabela armazena toda a informação que servirá como input ao classificador.

Exatamente o mesmo processo foi realizado para os restantes três *datasets* sendo possível verificar nos anexos A tanto os *datasets* como as características extraídas de cada um.

6.2 Classificação do meta-modelo

É neste momento que o meta-modelo número 2 mencionado no capítulo "Construção do meta-modelo" será utilizado para classificar novos dados/problemas, comportamento este que se pode verificar pela figura 11. Será então necessário providenciar novos dados de entrada para que este identifique a classe correspondente. Os dados que irão ser consumidos pelo meta-modelo são os elementos caracterizadores de cada *dataset* como representado na tabela 21. Relembro que o objetivo deste estudo passa pela criação de um método automático para a seleção do algoritmo de análise de dados. O meta-modelo 2 terá novos dados com o intuito de os consumir e processar com o objetivo de posteriormente identificar a classe correspondente, por outras palavras, o meta-modelo pretende selecionar o algoritmo de análise de dados mais apropriado para o problema em questão.

Aplicando aqui a técnica ilustrada pela figura 11, existe um "problema" / novo *dataset* que será consumido pelo meta-modelo no qual vai sugerir a árvore de decisão ou dito de outra forma o algoritmo de análise de dados mais adequado.

```
algerian_forest_fires_predicted = dt_classifier.predict([datasetAlgerianForestFiresToPredict])

cancer_classification_predicted = dt_classifier.predict([datasetCancerClassificationToPredict])

failure_heart_predicted = dt_classifier.predict([datasetFailureHeartToPredict])

raisin_predicted = dt_classifier.predict([datasetRaisinToPredict])

rice_predicted = dt_classifier.predict([datasetRiceToPredict])

vertebral_predicted = dt_classifier.predict([datasetVertebralToPredict])
```

Figura 27 - Classificação de *datasets*

A previsão da classe a que pertence cada *dataset* foi realizada com recurso à biblioteca *scikit-learn* que disponibiliza um método (figura 27) (*predict*) que dado um modelo treinado e dados de entrada prevê a classe pertencente, classificando-o desta forma.

Tabela 22 - Classificações do meta-modelo

Dataset	Valor
***Dataset* 1**	Árvore 3
***Dataset* 2**	Árvore 4
***Dataset* 3**	Árvore 4
Dataset 4	Árvore 4
Dataset 5	Árvore 3
Dataset 6	Árvore 2

Pela tabela 22 é possível verificar as classificações realizadas pelo meta-modelo, o meta-modelo classificou como pertencentes à classe 3 os *datasets* 1 e 5, como pertencentes à classe 4 os *datasets* 2, 3 e 4 sendo que o *dataset* 6 foi classificado como pertencente à classe 2.

Após as classificações realizadas pelo meta-modelo (tabela 22) é a altura de validar e verificar se as classificações são ou não as corretas. No subcapítulo seguinte tem como objetivo verificar as classificações efetuadas.

6.3 Validação da classificação do meta-modelo

No sentido de verificar as classificações realizadas pelo meta-modelo, cada um dos problemas será classificado por cada uma das 4 árvores de decisão descritas no capítulo "Construção do *Meta-dataset*", mais concretamente na figura 22. Este processo é o mesmo aquando da classificação das

amostras que serviram para criar o *meta-dataset.* Se as previsões efetuadas pela meta-modelo coincidirem com a árvore selecionada estamos perante uma classificação bem-sucedida.

Tabela 23 - Classificação dos *datasets* através das 4 Árvore de decisão

Dataset	**Valor**
***Dataset* 1**	Árvore 1
***Dataset* 2**	Árvore 4
***Dataset* 3**	Árvore 3
Dataset 4	Árvore 4
Dataset 5	Árvore 1
Dataset 6	Árvore 2

Através da análise da tabela 23 podemos verificar a classificação de cada um dos *datasets* quando submetido às 4 árvores de decisão, método semelhante aquando da classificação das amostras que deram origem ao meta-*dataset*. Dos 6 *datasets* classificados e quando comparados com as classificações do *meta-dataset* podemos verificar que o meta-modelo classificou corretamente três problemas/*dataset* sendo eles o *dataset* 2, 4 e 6. Neste estudo o meta-modelo classificou incorretamente o *dataset* 1 e 5 como pertencente à classe 3 mas que na realidade pertencia à classe 1 assim como o *dataset* 3 como pertencente à classe 4 mas que na realidade pertencia à classe 3. As causas associadas ao problema mencionado – como já referenciado no capítulo anterior – prendem-se na ausência de amostras de treino que permitem identificar padrões e extrair mais informação com o objetivo de originar um meta-modelo mais eficiente. Com os resultados verificamos que existe confusão entre classes, nomeadamente entre a classe 1 e 3 assim como entre a classe 3 e 4. Nos *datasets* 1 e 5 o meta-modelo classificou como pertencente à classe 3 mas que na realidade pertenciam à classe 1, verifica-se a existência de confusão entre classes e que os padrões existentes são muito semelhantes o que levou ao meta-modelo classificar incorretamente. Verificamos que o meta-modelo conseguiu com uma percentagem de 50% classificar corretamente as 6 amostras neste estudo, ou seja, 3 amostras.

Tal como qualquer abordagem existente, haverá sempre pontos mais fortes e fracos, sendo que o objetivo passa sempre por fortalecer os pontos fortes e melhorar os pontos fracos. A verdade é que depois de uma análise crítica e detalhada conclui-se que o classificador teve a capacidade de classificar corretamente 50% dos problemas em estudo. No entanto, analisando criticamente conclui-se que uma taxa de 50% para a seleção de um algoritmo de análise de dados é ainda um valor reduzido.

Enquadrando o problema no atual contexto, a verdade é que a incorreta classificação de um novo problema não coloca em causa vidas humanas ou um cenário semelhante, no entanto, a incorreta seleção de algoritmo pode fazer com que alguns recursos sejam despendidos. A incorreta seleção

do algoritmo de análise de dados que permita analisar um novo problema dará origem ao consumo de alguns recursos temporais e financeiros. Tendo em conta estes aspetos, um classificador com uma taxa de classificação de novos problemas mais elevada permitiria diminuir a percentagem de classificações incorretas e consequentemente a poupança de recursos.

Este será um ponto a melhorar, sendo que a adição de novos exemplos/amostras para que o meta-modelo/classificador possa aprender mais padrões e casos de teste no sentido de que no futuro a taxa de classificação de amostras corretas aumente através do processo de meta-aprendizagem. Um dos pontos fortes deste classificador é a capacidade de classificar novas amostras/problemas com rapidez, uma vez que a classificação de novos problemas manualmente iria exigir bastantes recursos.

Tal como já mencionado, qualquer parte interessada na utilização do classificador terá uma ferramenta capaz de selecionar o algoritmo de análise de dados com uma taxa de 50%. Desta feita o principal objetivo deste estudo foi concretizado que passava pela implementação de métodos baseados em árvores para meta-aprendizagem no qual deu origem a um classificador capaz de classificar novos dados com uma taxa de 50%.

7 Conclusões e Trabalho Futuro

Na presente dissertação foi apresentado um estudo sobre como criar um método automático para seleção do algoritmo de análise de dados usando árvores de decisão. O presente método visa resolver problemas de classificação dado um problema, diga-se, sugerir um algoritmo de análise de dados para um determinado *dataset*.

No sentido de implementar este método automático foi necessário criar uma base de dados que o permitisse "alimentar". Foram analisados e processados 27 *datasets* sendo que o objetivo passou por extrair informação/características de cada um destes. Uma vez a base de dados criada, com 27 amostras onde cada amostra representa as características de cada *dataset*, seguiu-se o próximo passo que foi classificar cada uma destas. No sentido de classificar cada amostra, foram criadas 4 árvores de decisão diferentes. Cada amostra presente no *dataset* gerado foi processada por cada uma das 4 árvores de decisão diferentes sendo que a árvore que apresentar melhores resultados será a escolhida para classificar a amostra. Realçar que cada árvore de decisão no presente contexto representa um algoritmo de análise de dados.

Uma vez criado o *meta-dataset*, a criação do meta-modelo foi a etapa seguinte. A árvore de decisão foi o algoritmo usado depois de definido e parametrizado. Foram implementados 3 meta-modelos neste estudo.

Primeiramente foi implementado um meta-modelo utilizando o *meta-dataset* gerado sem qualquer manipulação dos dados de maneira a observar os seus resultados e performance. No entanto, o meta-modelo não apresentou bons resultados uma vez que não conseguiu classificar corretamente pelo menos 50% das amostras. O meta-modelo apresenta uma capacidade de classificar corretamente amostras em 37% dos casos, ou seja, 3 a 4 amostras num universo de 10 amostras. Evidenciou a capacidade de classificar corretamente 50% das amostras da classe 1 e 4 ao contrário das classes 2 e 3 que apenas conseguiu classificar 1 amostra (classe 2) corretamente. As diferenças das classes 1 e 4 para as classes 2 e 3 é a representatividade de amostras que cada classe contém. Conclui-se que as classes com mais amostras permitem criar mais casos de testes e reconhecer padrões entre os dados nos quais estes possam ser canalizados para classificar novos dados.

No entanto, e uma vez que o primeiro meta-modelo não conseguiu de pelo menos classificar 50% das amostras corretamente, foram implementados dois novos meta-modelos representando duas novas abordagens.

A primeira abordagem (2º meta-modelo) passou por duplicar as amostras existentes, por outras palavras aplicar a técnica *over-sampling*. Uma vez aplicado esta técnica o classificador revelou melhorias no que diz respeito à correta classificação de amostras, no entanto, e ainda assim não

conseguiu classificar pelo menos metade das amostras corretamente (apenas 44%). O meta-modelo classificou corretamente 50% das amostras correspondentes às classes 1 e 4, no entanto classificou apenas corretamente 3 amostras nas classes 2 e 3. O segundo meta-modelo manteve a capacidade de classificação nas classes mais representativas, no entanto, revelou melhorias na classificação das amostras pertencentes às classes 2 e 3 quando comparado com o 1º meta-modelo. A causa associada a esta melhoria prende-se na adição de novas amostras (over-sampling) que permitiu ao meta-modelo "aprender" novos casos de testes e padrões.

Relativamente ao 3º meta-modelo que passou pela junção das classes mais semelhantes revelou melhorias quando comparado com o 1º meta-modelo mas não tão eficiente quanto o meta-modelo 2. Analisando os dados é possível verificar que na classe 3 conseguiu classificar corretamente 50% das amostras mas revelou confusão na classificação das amostras na classe 1 e 2.

Analisando os 3 meta-modelos é possível verificar que o segundo meta-modelo quando comparado com os restantes foi o que apresentou melhor eficiência uma vez que conseguiu com uma taxa de 44% a capacidade de classificar corretamente as amostras. No entanto, e como mencionado anteriormente, não conseguiu selecionar pelo menos 50% das classes. Uma das razões observadas para este problema deve-se ao facto de não existirem amostras suficientes para que o algoritmo fosse capaz de identificar padrões e extrair informação para criar um meta-modelo eficiente. Outra razão identificada passou por o meta-modelo tender a classificar uma nova amostra como pertencente à classe mais representativa existindo aqui alguma confusão entre classes assim como falta de informação referente às classes menos representativas (classes com menos número de amostras).

Uma vez selecionado o melhor classificador (meta-modelo 2) foram efetuados alguns testes que tinham como objetivo a demonstração prática da utilização do mesmo. Dado um novo problema (características de um *dataset*), o classificador baseado em dados históricos pretende atribuir uma classificação. Verificou-se que o meta-modelo apresentou uma capacidade de classificar corretamente novos problemas com uma taxa de 50% num universo de 6 amostras novas.

A realização deste projeto permite contribuir com uma abordagem de extração de informação/características de 27 *datasets* de maneira a criar uma base de dados reunindo toda esta informação. Uma outra contribuição passou pela realização de um classificador que tem a capacidade de classificar novas amostras, diga-se, resolver problemas de classificação de seleção do algoritmo de análise de dados.

Como trabalho futuro e no sentido de melhorar as previsões do modelo é sugerido reforçar o meta-*dataset* com novas amostras, sobretudo nas classes menos representativas e desta forma contribuir para que o classificador consiga adquirir mais conhecimento e identificar novos padrões entre os dados de maneira a classificar novas amostra com maior nível de eficiência. Uma nova sugestão

prende-se no facto de identificar novos algoritmos que possam ser usados no presente âmbito e verificar se estes apresentariam melhores resultados uma vez aplicados.

No entanto as ilações a retirar após esta dissertação maioritariamente são positivas uma vez que foi possível abordar diversas técnicas e conceitos durante o desenvolvimento da aprendizagem de máquina desde a exploração de dados até à implementação do modelo assim como a sua utilização.

Referências

[1] «Machine Learning & Deep Learning», *DataCamp Community*, 15 de Novembro de 2018. https://www.datacamp.com/community/tutorials/machine-deep-learning (acedido 30 de Março de 2022).

[2] «What is Machine Learning? | Glossary». https://www.hpe.com/pt/en/what-is/machine-learning.html (acedido 17 de Fevereiro de 2022).

[3] «Supervised and Unsupervised learning», *GeeksforGeeks*, 1 de Outubro de 2017. https://www.geeksforgeeks.org/supervised-unsupervised-learning/ (acedido 22 de Dezembro de 2021).

[4] B. Osiński e K. Budek, «What is reinforcement learning? The complete guide», *deepsense.ai*, 5 de Julho de 2018. https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/ (acedido 22 de Dezembro de 2021).

[5] «Automated Machine Learning», *DataRobot AI Cloud*. https://www.datarobot.com/wiki/automated-machine-learning/ (acedido 24 de Dezembro de 2021).

[6] N. Thiloshon e P. Guhanathan, «A Review on Automated Machine Learning (AutoML) Systems», 2019.

[7] «Automated machine learning», *Wikipedia*. 8 de Dezembro de 2021. Acedido: 24 de Dezembro de 2021. [Em linha]. Disponível em: https://en.wikipedia.org/w/index.php?title=Automated_machine_learning&oldid=1059235159

[8] A. Balaji e A. Hanuschkin, «Benchmarking Automatic Machine Learning Frameworks», Ago. 2018.

[9] «Statistical learning: the setting and the estimator object in scikit-learn», *scikit-learn*. https://scikit-learn/stable/tutorial/statistical_inference/settings.html (acedido 23 de Maio de 2022).

[10] «CRISP-DM», *Data Science Process Alliance*. https://www.datascience-pm.com/crisp-dm-2/ (acedido 22 de Dezembro de 2021).

[11] C. Schröerab, F. Kruseb, e J. Gómezb, «A Systematic Literature Review on Applying CRISP-DM Process Model», 2020.

[12] A. L. Vaz, «Otimizando os hiperparâmetros», *Data Hackers*, 19 de Agosto de 2019. https://medium.com/data-hackers/otimizando-os-hiperpar%C3%A2metros-621de5e9be37 (acedido 23 de Dezembro de 2021).

[13] E. B. Rabello, «Cross Validation: Avaliando seu modelo de Machine Learning», *Medium*, 8 de Julho de 2019. https://medium.com/@edubrazrabello/cross-validation-avaliando-seu-modelo-de-machine-learning-1fb70df15b78 (acedido 14 de Junho de 2022).

[14] V. Rodrigues, «Métricas de Avaliação: acurácia, precisão, recall… quais as diferenças?», *Medium*, 10 de Junho de 2021. https://vitorborbarodrigues.medium.com/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-recall-quais-as-diferen%C3%A7as-c8f05e0a513c (acedido 14 de Junho de 2022).

[15] S. Studer *et al.*, «Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology», Fev. 2021.

[16] «Commonly Used Machine Learning Algorithms | Data Science», *Analytics Vidhya*, 8 de Setembro de 2017. https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/ (acedido 24 de Março de 2022).
[17] «Decision Tree Algorithm, Explained», *KDnuggets*. https://www.kdnuggets.com/decision-tree-algorithm-explained.html/ (acedido 28 de Dezembro de 2021).
[18] by, «How to Build Decision Tree for Classification - (Step by Step Using Entropy and Gain)», *The Genius Blog*, 19 de Abril de 2018. https://kindsonthegenius.com/blog/how-to-build-a-decision-tree-for-classification-step-by-step-procedure-using-entropy-and-gain/ (acedido 30 de Março de 2022).
[19] «Árvore de decisão: entenda esse algoritmo de Machine Learning». https://blog.somostera.com/data-science/arvores-de-decisao (acedido 21 de Fevereiro de 2022).
[20] «Estudo: carros autônomos não acabarão com acidentes de trânsito». https://www.tecmundo.com.br/mobilidade-urbana-smart-cities/153849-estudo-carros-autonomos-nao-acabarao-acidentes-transito.htm (acedido 14 de Junho de 2022).
[21] «Aplicações de NLP no mercado de trabalho», *Alura*. https://www.alura.com.br/artigos/aplicacoes-nlp-mercado-de-trabalho (acedido 14 de Junho de 2022).
[22] «O que é processamento de linguagem natural? [NLP]», *Tecnoblog*. https://tecnoblog.net/responde/o-que-e-processamento-de-linguagem-natural-nlp/ (acedido 29 de Março de 2022).
[23] «Fraud prevention using machine learning». https://www.neuraldesigner.com/solutions/fraud-detection (acedido 14 de Junho de 2022).
[24] «Machine Learning: What it is and why it matters». https://www.sas.com/en_gb/insights/analytics/machine-learning.html (acedido 31 de Maio de 2022).
[25] «pandas - Python Data Analysis Library». https://pandas.pydata.org/ (acedido 19 de Janeiro de 2021).
[26] «Matplotlib: Python plotting — Matplotlib 3.3.3 documentation». https://matplotlib.org/ (acedido 19 de Janeiro de 2021).
[27] «scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation». https://scikit-learn.org/stable/ (acedido 19 de Janeiro de 2021).
[28] «Project Jupyter». https://www.jupyter.org (acedido 31 de Janeiro de 2021).
[29] «Short note on the Data Cleaning for Data Science!», *TechLeer*. https://www.techleer.com/articles/604-short-note-on-the-data-cleaning-for-data-science/ (acedido 14 de Junho de 2022).
[30] «Find Open Datasets and Machine Learning Projects | Kaggle». https://www.kaggle.com/datasets (acedido 5 de Dezembro de 2020).
[31] «UCI Machine Learning Repository: Data Sets». https://archive.ics.uci.edu/ml/datasets.php (acedido 25 de Setembro de 2021).
[32] «data.world | The Cloud-Native Data Catalog», *data.world*. https://data.world/ (acedido 25 de Setembro de 2021).
[33] S. Ray, «A Comprehensive Guide to Data Exploration.», Jan. 2016.
[34] N. Todd, «Descriptive Statistics.».

[35] S. Ronaghan, «The Mathematics of Decision Trees, Random Forest and Feature Importance in Scikit-learn and Spark», *Medium*, 1 de Novembro de 2019. https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3 (acedido 7 de Fevereiro de 2022).
[36] «How to interpret Box Plot || Python», *AI ASPIRANT*, 12 de Agosto de 2019. https://aiaspirant.com/box-plot/ (acedido 14 de Junho de 2022).

Anexo A: Lista de *datasets* usados

Lista dos 27 *datasets* usados durante o projeto.

Dataset 1 - https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra

Dataset 2 - https://www.kaggle.com/saranchandar/standard-classification-banana-dataset

Dataset 3 - https://www.kaggle.com/elakiricoder/gender-classification-dataset

Dataset 4 - https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set

Dataset 5 - https://data.world/exercises/logistic-regression-exercise-1

Dataset 6 - https://www.kaggle.com/omarrodriguez/bangladesh-wells

Dataset 7 - https://www.kaggle.com/merishnasuwal/breast-cancer-prediction-dataset

Dataset 8 - https://archive.ics.uci.edu/ml/datasets/Caesarian+Section+Classification+Dataset

Dataset 9 - https://www.kaggle.com/omnamahshivai/surgical-dataset-binary-classification

Dataset 10 - https://www.kaggle.com/aiaiaidavid/cardio-data-dv13032020

Dataset 11 - https://www.kaggle.com/sachinsharma1123/performance-prediction

Dataset 12 - https://archive.ics.uci.edu/ml/datasets/Cryotherapy+Dataset+

Dataset 13 - https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center

Dataset 14 - https://data.world/uci/statlog-australian-credit-approval

Dataset 15 - https://archive.ics.uci.edu/ml/datasets/Haberman%27s+Survival

Dataset 16 - https://archive.ics.uci.edu/ml/datasets/banknote+authentication

Dataset 17 - https://data.world/data-society/pima-indians-diabetes-database

Dataset 18 - https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State

Dataset 19 - https://archive.ics.uci.edu/ml/datasets/Fertility

Dataset 20 - https://data.world/uci/statlog-heart

Dataset 21 - https://archive.ics.uci.edu/ml/datasets/HTRU2

Dataset 22 - https://archive.ics.uci.edu/ml/datasets/Immunotherapy+Dataset

Dataset 23 - https://archive.ics.uci.edu/ml/datasets/ILPD+%28Indian+Liver+Patient+Dataset%29

Dataset 24 - https://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope

Dataset 25 - https://www.kaggle.com/teertha/personal-loan-modeling

Dataset 26 - https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation

Dataset 27 - https://www.kaggle.com/barun2104/telecom-churn

Lista de *datasets* classificados pelo meta-modelo

Dataset 1 - https://archive.ics.uci.edu/ml/datasets/Algerian+Forest+Fires+Dataset++

Dataset 2 - https://www.kaggle.com/abrahamanderson/cancer-classification

Dataset 3 - https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records

Dataset 4 - https://archive.ics.uci.edu/ml/datasets/Raisin+Dataset

Dataset 5 - https://www.kaggle.com/datasets/muratkokludataset/rice-dataset-commeo-and-osmancik

Dataset 6 - https://archive.ics.uci.edu/ml/datasets/Vertebral+Column

Anexo B: *Dataset* gerado

Em seguida será apresentado o *dataset* gerado. Sendo que cada amostra representa as características extraídas de cada *dataset* analisado.

Tabela B-1 - *Dataset* gerado (parte 1)

Nº	NSamples	NFeatures	NSamplesClass1	NSamplesClass2	DistAllClassesGroup1Mean	DistAllClassesGroup1Std
1	116	9	52	64	604,824586	372,510813
2	5300	2	2924	2376	0,000018	1,000038
3	5001	7	2500	2501	19,997421	1,985144
4	1151	19	540	611	115,552735	81,634928
5	1340	19	509	831	19,013881	13,268147
6	3020	4	1283	1737	5,251325	4,51141
7	569	5	212	357	14,223652	3,538113
8	80	5	34	46	1,0125	1,302284
9	14635	24	10945	3690	3,82173	4,219333
10	68783	11	34742	34041	3,519735	1,954586
11	1340	19	509	831	14,595074	11,019238
12	90	6	42	48	92,822222	135,802612
13	748	4	570	178	1384,191177	1465,666088
14	690	14	383	307	14,148045	11,896719
15	306	3	225	81	4,026144	7,189654
16	1372	4	762	610	0,20597	6,411043
17	768	8	500	268	173,286458	153,921605
18	14980	14	8257	6723	30039,36645	7537,109515
19	100	9	12	88	2,402	1,588669
20	270	13	150	120	153,440741	26,073051
21	17898	8	16259	1639	127,545944	146,661442
22	90	7	19	71	9,399999	5,537458
23	583	10	416	167	121,265522	294,0376
24	19020	10	12332	6688	7,05851	131,326943
25	5000	13	3530	1470	61,5734	104,792353
26	245057	3	50859	194198	125,065446	62,255653
27	3333	10	2850	483	203,496643	61,907135

Tabela B-2 - *Dataset* gerado (parte 2)

Nº	DistAllClassesGroup1Min	DistAllClassesGroup125	DistAllClassesGroup150	DistAllClassesGroup175	DistAllClassesGroup1Max
1	71,966429	321,370498	537,056131	785,758757	1850,530342
2	-2,39	-0,914	-0,0372	0,8225	3,19
3	16,5	18,7	20	21,4	23,6
4	3	48,24905	102,425026	166,809181	489,25545
5	1,6	9,6	14,95	24,6	85,5
6	0	0	5	9	18
7	7,03363	11,78637	13,46587	15,8853	28,2734
8	0	0	0	2	3
9	0	1	4	6	14
10	2	3	3	4	8
11	1	6,8	11,2	18,9	73,1
12	6	23	76	110	764
13	251	502	1004	1757	12550
14	3	8,165	10	16,625	112,5
15	0	0	1	4	52
16	-13,8343	-3,988425	0,02998	3,57406	20,3769
17	0	63	128,5	245,25	1084
18	16597,94	29896,42	29964,6	30034,36	942321,41
19	-0,8	1,8	2,8	4	4
20	72	134	158,5	172	210
21	-6,694921	42,476515	94,49862	156,404329	1517,034443
22	3	4	9	12,75	24
23	13,7	33,6	52	99,3	4964,8
24	-995,5774	-43,914425	20,544	71,63685	994,9809
25	2	2	5	108	645
26	0	68	139	176	255
27	1	162	203	244,78	415,4

Tabela B-3 - *Dataset* gerado (parte 3)

Nº	DistAllClassesGroup2Mean	DistAllClassesGroup2Std	DistAllClassesGroup2Min	DistAllClassesGroup225	DistAllClassesGroup250
1	176,728346	69,187006	88,323	132,277893	156,685656
2	0,000016	0,99988	-3,09	-0,75325	-0,01525
3	1,9938	2,000052	0	0	1
4	178,625902	147,171737	3,774942	65,89112	140,424147
5	221,943321	65,82152	38,7	167,675	224,85
6	49,988793	39,586062	0,897	21,93725	38,061499
7	766,147786	380,514146	197	511,64	656,18
8	30,35	6,524157	18	26,75	29
9	120,390261	46,130997	5,796118	84,693939	111,688679
10	501,17087	56,403823	197	473	492
11	226,484849	68,013615	39,3	170,475	228,6
12	37,966667	17,672555	16,25	23,5625	35
13	43,78877	32,47211	2	18,75	35
14	1247,050116	5403,771477	15,75	108,67	206,375
15	115,310457	14,052857	88	104	115
16	2,356088	8,71181	-20,8152	-3,4812	2,81583
17	186,59987	51,948339	21,078	150,54375	178,3725
18	30396,98195	17204,92924	12426,6667	30141,53	30223,08
19	1,0869	2,412055	-1,44	-1,19	-0,28
20	445,027776	83,636767	253	387	436,8
21	184,433872	53,030736	36,078963	157,770136	180,71016
22	148,308333	169,165951	24	65,75	96,25
23	420,281904	448,276219	77,7	233	289,93
24	299,71987	162,018918	7,5074	186,717075	266,55815
25	95794,21854	3636,322958	9336	93246,45	96068
26	255,684478	132,503362	0	157	281
27	257,931897	77,536464	14	206,53	254,27

Tabela B-4 - *Dataset* gerado (parte 4)

Nº	DistAllClassesGroup275	DistAllClassesGroup2Max	DistClasse1Group1Mean	DistClasse1Group1Std	DistClasse1Group1Min
1	199,564199	470,418759	569,688218	320,049693	72,504689
2	0,782	2,81	0,062025	1,082451	-2,04
3	4	4	20,5138	2,124351	16,6
4	251,891499	1026,009079	97,966392	71,516098	3
5	271,3	446,2	14,508055	9,708077	1,6
6	66,241	349,181006	4,915043	4,260858	0
7	908,6	2728,78	17,565728	3,216579	11,02371
8	35,25	46	0,970589	1,231881	0
9	150,235768	270,952802	3,876017	4,125926	0
10	542	948	3,486357	1,889244	2
11	277	458,6	11,275246	8,170373	1
12	48,6875	82	119,02381	188,685795	8
13	64	172	1205,24035	1191,438785	251
14	725,415	102125,25	10,861057	7,663473	3
15	126,5	152	2,791111	5,870318	0
16	9,6361	19,7764	-0,350922	5,364971	-13,4899
17	218,47625	349,52	159,938	134,835496	0
18	30322,57	2121986,18	30050,32747	7285,40568	16597,94
19	3,25	5	2,183333	1,650749	-0,4
20	494,6	861,2	161,453332	22,086044	97
21	207,010569	410,107832	131,72712	136,399756	-6,694921
22	141,4375	1038	9,631579	4,830676	3
23	420,2	4277,8	149,363943	342,893659	14,7
24	384,410625	1181,4433	21,835628	87,411012	-639,9513
25	98543,75	101996	62,046458	105,815731	2
26	341	510	113,869876	41,612799	26
27	308,47	509,29	203,169169	61,672955	1

Tabela B-5 - *Dataset* gerado (parte 5)

Nº	DistClasse1Group125	DistClasse1Group150	DistClasse1Group175	DistClasse1Group1Max	DistClasse1Group2Mean
1	308,820296	545,589958	730,52828	1390,234918	161,73462
2	-1,06	0,2745	0,98	3,19	0,053703
3	19,1	20,7	22,2	23,6	3,4944
4	40,347047	77,576306	151,803155	389,792398	160,741955
5	8,2	11,7	18	65,2	203,943615
6	0	5	8	17	55,031533
7	15,16901	17,4272	19,700925	28,2547	1115,346698
8	0	1	1,75	3	29,794118
9	1	4	5	14	120,736089
10	3	3	4	8	486,694548
11	5,8	8,9	14,3	55,8	207,352394
12	24	79	109	764	48,511905
13	502	753	1506	11044	45,542105
14	7,125	9,415	11,5	49,875	439,279034
15	0	0	3	46	114,88
16	-3,93795	0,148225	3,076183	11,2789	6,533313
17	63	132	219	939	171,903934
18	29894,37	29962,07	30025,13	682020,92	30451,50165
19	1,6	2,8	2,8	4	1,459167
20	149,25	163	177	210	433,302668
21	55,830582	102,194161	161,65282	1473,80525	187,400891
22	6	10	12	19	144,539473
23	38,15	62,55	121,375	4964,8	469,845361
24	-33,477125	25,4264	73,976025	538,7471	274,045279
25	2	5	107	645	95767,73982
26	80	117	147	225	350,593051
27	161	201	245	412,75	252,55532

Tabela B-6 - *Dataset* gerado (parte 6)

Nº	DistClasse1Group2Std	DistClasse1Group2Min	DistClasse1Group225	DistClasse1Group250	DistClasse1Group275
1	51,260539	88,97975	128,593276	150,601929	182,109216
2	0,915192	-3,09	-0,634	0,009645	0,794
3	1,329038	0	4	4	4
4	127,232168	3,838462	65,067091	127,642274	229,64239
5	62,258928	38,7	153,4	204,2	245,6
6	44,193684	1,436	23,027	41,027	75,799498
7	393,572101	443,88	823,3725	1067,66	1357,44
8	5,299366	19	27	28,5	33
9	46,321955	26,896118	87,893939	110,274936	164,963665
10	49,650717	212	449	488	507
11	63,711188	39,3	155,8	207	249,3
12	17,599407	20,75	37,3125	47,625	54,6875
13	33,030427	2	20	39	68
14	870,902017	17,17	126,96	211,88	385,79
15	14,235069	88	103	115	126
16	7,15814	-11,218	1,333408	8,2219	12,576425
17	45,797795	21,078	141,62975	164,386	197,86175
18	18323,16572	14755,3867	30138,46	30216,93	30304,61
19	2,364829	-1,44	-0,2975	1,255	2,7575
20	83,920259	253	379	424,2	475,675
21	40,644893	51,404073	162,785742	182,548544	204,511054
22	116,530024	42,75	78,625	109	161,125
23	504,200993	82,7	246,5	318,3	454,475
24	127,450152	19,9363	184,627775	252,14425	345,2747
25	3277,46406	90034	93123,2	96023,6	98523,35
26	73,536916	162	293	370	407
27	73,083667	15,7	205,455	251,38	300,635

Tabela B-7 - *Dataset* gerado (parte 7)

Nº	DistClasse1Group2Max	DistClasse2Group1Mean	DistClasse2Group1Std	DistClasse2Group1Min	DistClasse2Group125
1	348,371859	633,372884	408,273186	126,253956	350,709209
2	2,59	-0,076291	0,882449	-2,39	-0,68
3	4	19,481247	1,640138	16,5	18,4
4	869,850723	131,095492	85,923206	7	59,172599
5	424,4	21,773767	14,273016	2,1	11,5
6	347,321006	5,499712	4,66775	0	0
7	2728,78	12,239002	1,793958	7,03363	11,16306
8	41	1,043478	1,286474	0	0
9	267,772802	3,660703	4,314233	0	1
10	905	3,553803	2,009384	2	3
11	433,8	16,628521	11,92442	1,3	8,2
12	82	69,895833	47,847061	6	23
13	172	1957,235955	2017,259897	251	753
14	7675,585	18,248746	14,015098	3	8,75
15	146	7,45679	9,185654	0	1
16	19,7764	0,90163	7,332795	-12,8748	-3,815875
17	337,629	198,190299	181,601887	0	67,75
18	1672679,49	30025,90443	3353,457375	29341,53	29898,725
19	4,82	2,431818	1,581079	-0,8	1,8
20	838,2	143,425	25,893693	72	126,75
21	406,824096	86,068368	113,28734	-4,38945	17,093264
22	497	9,338029	5,731201	3	4
23	4277,8	51,271856	39,23698	15,2	30
24	993,81	-20,18901	183,211142	-995,5774	-82,494125
25	101989	60,437416	102,317634	2	2
26	485	127,997482	66,301507	0	64
27	471,89	205,428944	63,028847	1	163,5

Tabela B-8 - *Dataset* gerado (parte 8)

Nº	DistClasse2Group150	DistClasse2Group175	DistClasse2Group1Max	DistClasse2Group2Mean	DistClasse2Group2Std
1	528,872694	822,478552	1843,240342	188,910748	75,345006
2	-0,185	0,40525	2,19	-0,066054	1,091729
3	19,6	20,8	21,8	0,493802	1,315252
4	121,371439	188,349894	474,376682	194,431685	159,229902
5	17,8	28	85,5	232,968413	63,803455
6	5	9	18	46,264075	35,212525
7	12,29076	13,4707	18,0134	558,780364	150,089681
8	1	2	3	30,76087	7,319455
9	3	5	14	119,364485	42,8586
10	3	4	8	515,945301	57,523689
11	13	21,6	73,1	238,203764	66,111222
12	76	107	174	28,739584	11,060006
13	1506	2259	12550	38,174157	28,800046
14	14	26	112,5	2254,790586	7842,312637
15	4	11	52	116,506172	13,509255
16	-0,28793	5,96814	20,0627	-2,862019	7,286067
17	105	293,25	1076	214,017664	50,543197
18	29969,74	30047,68	299309,21	30330,02239	5673,496439
19	2,3	4	4	1,036137	2,40655
20	147,5	163,25	203	459,684168	80,046687
21	49,608099	120,375063	1315,946443	155,000824	59,645262
22	8	13,5	24	149,316902	180,270478
23	39,2	56,15	303,8	296,818444	184,336027
24	10,52585	66,948225	994,9487	347,061236	193,480373
25	5	109,75	597	95857,8035	4321,65581
26	151	179	255	230,828634	128,41751
27	207	244,795	397,4	289,65704	92,296169

Tabela B-9 - *Dataset* gerado (parte 9)

Nº	DistClasse2Group2Min	DistClasse2Group225	DistClasse2Group250	DistClasse2Group275	DistClasse2Group2Max
1	100,3459	139,711322	166,737502	216,216539	442,064675
2	-2,28	-0,98025	-0,04555	0,76725	2,81
3	0	0	0	0	4
4	5,505779	72,346795	150,077185	278,582725	956,772874
5	79,4	180,7	238,7	279,95	446,2
6	0,897	21,288999	36,133999	61,716001	282,030005
7	197	464,22	553,97	656,96	1140,51
8	18	26	31	36	46
9	18,106118	86,970182	117,676329	147,584932	254,992802
10	216	476	507	547	948
11	80,2	184	243,5	286,35	458,6
12	16,25	20,6875	25,625	37	55
13	2	17,25	32	49,25	124
14	15,75	33,71	388,13	1550,85	100961,75
15	92	105	116	126	152
16	-20,8152	-8,871475	-1,633325	2,647505	11,9931
17	21,088	178,0625	210,699	250,503	338,52
18	25207,17	30145,12	30233,34	30343,08	485306,69
19	-1,44	-1,19	-0,31	3,25	5
20	288	406,85	456,9	508,925	706,2
21	38,148277	108,951828	153,966518	198,609517	340,779588
22	24	67,375	91,25	135,5	1035,75
23	104,87	211,1	254,8	307,8	1855,2
24	7,5171	203,41295	311,33965	461,0181	1181,4433
25	9341	93386,45	96558	98587,25	101996
26	0	106	269	314	510
27	19,55	216,91	303,77	364,225	507,99

Tabela B-10 - *Dataset* gerado (parte 10)

Nº	MissgingValuesAllClasses	MissgingValuesPercAllClasses	MissgingValuesClass1	MissgingValuesPercClass1	MissgingValuesClass2
1	0	100	0	100	0
2	0	100	0	100	0
3	0	100	0	100	0
4	0	100	0	100	0
5	0,043205027	94,73684211	0,023566379	94,73684211	0,019638649
6	0	100	0	100	0
7	0	100	0	100	0
8	0	100	0	100	0
9	0	100	0	100	0
10	0	100	0	100	0
11	0,043205027	94,73684211	0,023566379	94,73684211	0,019638649
12	0	100	0	100	0
13	0	100	0	100	0
14	0	100	0	100	0
15	0	100	0	100	0
16	0	100	0	100	0
17	0	100	0	100	0
18	0	100	0	100	0
19	0	100	0	100	0
20	0	100	0	100	0
21	0	100	0	100	0
22	0	100	0	100	0
23	0,068610635	90	0,034305317	90	0,034305317
24	0	100	0	100	0
25	0	100	0	100	0
26	0	100	0	100	0
27	0	100	0	100	0

Tabela B-11 - *Dataset* gerado (parte 11)

Nº	MissgingValuesPercClass2	CorrelationGroup1	CorrelationGroup2	OutliersAllClasses	OutliersPercAllClasses
1	100	7,07626	14,381457	5,842911877	22,22222222
2	100	1,189423	1,189423	0,009433962	50
3	100	4,976223	12,603447	1,862484646	85,71428571
4	100	34,099551	37,174363	5,720426174	5,263157895
5	94,73684211	80,680378	75,130296	3,703849175	10,52631579
6	100	1,85259	2,27144	1,978476821	50
7	100	5,011073	8,944163	2,284710018	0
8	100	2,296125	4,575941	0,5	80
9	100	15,259518	21,874024	3,481380253	37,5
10	100	8,783032	13,007002	5,353066892	18,18181818
11	94,73684211	75,021903	80,732309	3,703849175	10,52631579
12	100	4,187561	5,523501	1,481481481	66,66666667
13	100	4,90439	3,225626	3,24197861	25
14	100	14,946111	16,122773	7,743271222	35,71428571
15	100	0,93306	2,112118	4,357298475	66,66666667
16	100	1,220433	1,110803	1,67638484	50
17	100	8,003455	8,229875	2,376302083	0
18	100	23,194266	20,316384	6,073335876	0
19	100	2,51439	3,75532	2,111111111	66,66666667
20	100	8,527675	16,647459	2,792022792	46,15384615
21	100	3,159318	1,955522	7,996703542	0
22	100	3,06652	3,79009	5,555555556	57,14285714
23	90	8,439627	6,816101	9,142367067	20
24	100	1,799798	5,494884	3,686119874	20
25	100	8,229186	11,75843	3,101538462	46,15384615
26	100	2,351626	4,671822	0	100
27	100	8,40226	8,322238	2,313231323	10

Tabela B-12 - *Dataset* gerado (parte 12)

Nº	OutliersClass1	OutliersPercClass1	OutliersClass2	OutliersPercClass2	ClassBalance
1	2,777777778	22,22222222	2,777777778	22,22222222	81,25
2	0,028301887	50	0,094339623	50	81,25854993
3	4,567657897	28,57142857	4,433399034	28,57142857	99,96001599
4	2,332068224	21,05263158	3,223741369	21,05263158	88,3797054
5	1,54752553	5,263157895	2,109190888	5,263157895	61,25150421
6	0,860927152	50	1,092715232	50	73,86298215
7	0,702987698	0	1,159929701	0	59,3837535
8	4,25	60	0,5	80	73,91304348
9	4,080970277	37,5	1,140245986	45,83333333	33,71402467
10	3,57223574	18,18181818	2,790726567	27,27272727	97,9822693
11	1,54752553	5,263157895	2,227022781	5,263157895	61,25150421
12	2,037037037	50	1,851851852	83,33333333	87,5
13	2,774064171	25	2,372994652	0	31,22807018
14	6,252587992	28,57142857	1,6873706	35,71428571	80,15665796
15	2,832244009	66,66666667	0,217864924	66,66666667	36
16	0,364431487	50	0,510204082	25	25
17	2,132161458	12,5	0,813802083	12,5	53,6
18	3,701602136	0	2,294964715	0	81,42182391
19	1,444444444	44,44444444	1,777777778	66,66666667	13,63636364
20	3,418803419	38,46153846	2,136752137	38,46153846	80
21	5,206587328	0	0,249329534	37,5	10,08057076
22	1,111111111	57,14285714	4,285714286	71,42857143	26,76056338
23	6,483704974	10	1,903945111	30	40,14423077
24	1,531019979	0	1,604626709	20	54,23289004
25	1,958461538	53,84615385	1,113846154	46,15384615	41,64305949
26	0	100	0	100	26,18925015
27	1,431143114	20	0,522052205	20	16,94736842

Tabela B-13 - *Dataset* gerado (parte 13)

Nº	FeaturesImportanceGroup1	FeaturesImportanceGroup2	target
1	0,1373044	0,8626956	2
2	0,36995027	0,63004973	1
3	0,10495833	0,89504167	1
4	0,19235105	0,80764895	2
5	0,22913859	0,77086141	4
6	0,16972009	0,83027991	4
7	0,14216969	0,85783031	4
8	0,26404953	0,73595047	1
9	0,04404971	0,95595028	1
10	0,09569996	0,90430005	4
11	0,23213522	0,76786477	4
12	0,10106647	0,89893353	3
13	0,30776456	0,69223543	4
14	0,08980327	0,91019673	4
15	0,27771672	0,72228328	1
16	0,17839028	0,82160972	2
17	0,19728469	0,80271532	1
18	0,35265433	0,64734568	3
19	0,08072256	0,91927744	2
20	0,21273476	0,78726524	2
21	0,08029838	0,91970161	4
22	0,05337287	0,94662713	1
23	0,26785226	0,73214775	2
24	0,23416689	0,76583311	4
25	0,18243761	0,81756237	4
26	0,0983979	0,9016021	3
27	0,26410287	0,73589714	1

Dataset Gerado pelo meta-modelo para classificar novos problemas

Tabela B-14 - Dataset a classificar pelo meta-modelo (parte 1)

Nº	NSamples	NFeatures	NSamplesClass1	NSamplesClass2	DistAllClassesGroup1Mean	DistAllClassesGroup1Std
1	122	12	63	59	120.998361	92.743475
2	569	30	212	357	872.97416	416.133813
3	299	12	203	96	63.004461	14.308925
4	900	8	450	450	179244.7059	79821.39042

Tabela B-15 - Dataset a classificar pelo meta-modelo (parte 2)

Nº	DistAllClassesGroup1Min	DistAllClassesGroup125	DistAllClassesGroup150	DistAllClassesGroup175	DistAllClassesGroup1Max
1	26.7	47.875	95.7	166.5	424.8
2	246.081688	594.214835	752.361297	1033.560964	2997.7001
3	40	51	61	75	100
4	51669.71087	121080.3611	160800.8484	213683.8886	513756.2753

Tabela B-16 - Dataset a classificar pelo meta-modelo (parte 3)

Nº	DistAllClassesGroup2Mean	DistAllClassesGroup2Std	DistAllClassesGroup2Min	DistAllClassesGroup225	DistAllClassesGroup250
1	183.062296	39.398344	95.6	156.55	185.55
2	983.74721	631.007744	222.040724	584.3917	771.61982
3	264246.2325	98869.42079	25254.5	212854.4	262541.1
4	1598.317636	389.943222	845.432127	1313.266283	1528.819164

Tabela B-17 - Dataset a classificar pelo meta-modelo (parte 4)

Nº	DistAllClassesGroup275	DistAllClassesGroup2Max	DistClasse1Group1Mean	DistClasse1Group1Std	DistClasse1Group1Min
1	212.825	259	74.953968	55.984157	26.7
2	1201.49593	4927.86515	1258.888378	425.729604	531.593898
3	304471.4	858383.4	60.889984	13.042528	40
4	1804.154	3696.843	129339.2	36767.72	51669.71

Tabela B-18 - Dataset a classificar pelo meta-modelo (parte 5)

Nº	DistClasse1Group125	DistClasse1Group150	DistClasse1Group175	DistClasse1Group1Max	DistClasse1Group2Mean
1	38.7	55.3	93.85	325.8	167.307936
2	942.683131	1207.046383	1520.585803	2996.10201	1569.287582
3	50	61	70	95	267534.5516
4	102786.4471	125475.1198	152908.0352	402620.8987	1337.995219

Tabela B-19 - Dataset a classificar pelo meta-modelo (parte 6)

Nº	DistClasse1Group2Std	DistClasse1Group2Min	DistClasse1Group225	DistClasse1Group250	DistClasse1Group275
1	31.033513	95.6	145.95	174.3	190.9
2	673.729667	562.611094	1070.558862	1434.57364	1889.725342
3	98368.2417	25272.5	219875.4	263593	302981.2
4	210.052829	845.50646	1200.069344	1329.6404	1465.123895

Tabela B-20 - Dataset a classificar pelo meta-modelo (parte 7)

Nº	DistClasse1Group2Max	DistClasse2Group1Mean	DistClasse2Group1Std	DistClasse2Group1Min	DistClasse2Group125
1	214.2	170.164406	91.245644	49.3	100.75
2	4927.76445	643.803809	162.919089	246.083848	541.125469
3	855728.1	67.475698	15.652766	42	55
4	3099.272878	229150.237	80077.72933	82798.50839	171509.1404

Tabela B-21 - Dataset a classificar pelo meta-modelo (parte 8)

Nº	DistClasse2Group150	DistClasse2Group175	DistClasse2Group1Max	DistClasse2Group2Mean	DistClasse2Group2Std
1	149.3	220.15	403.3	199.884746	22.111801
2	639.201889	751.674668	1263.12979	636.031361	184.421318
3	66	80	100	257292.8077	99923.63722
4	212762.3671	275771.3061	513756.2753	1858.640055	352.704869

Tabela B-22 - Dataset a classificar pelo meta-modelo (parte 9)

Nº	DistClasse2Group2Min	DistClasse2Group225	DistClasse2Group250	DistClasse2Group275	DistClasse2Group2Max
1	158.5	182.4	199.2	214.3	254
2	222.042381	510.01611	621.69286	757.52813	1386.62996
3	47157.6	197813.325	258970.3	311862.4	629327.4
4	1046.846801	1620.336119	1800.520244	2094.471454	3696.837697

Tabela B-23 - Dataset a classificar pelo meta-modelo (parte 10)

Nº	MissgingValuesAllClasses	MissgingValuesPercAllClasses	MissgingValuesClass1	MissgingValuesPercClass1	MissgingValuesClass2
1	0	100	0	100	0
2	0	100	0	100	0
3	0	100	0	100	0
4	0	100	0	100	0

Tabela B-24 - Dataset a classificar pelo meta-modelo (parte 11)

Nº	MissgingValuesPercClass2	CorrelationGroup1	CorrelationGroup2	OutliersAllClasses	OutliersPercAllClasses
1	100	22.461443	15.299921	2.732240437	33.33333333
2	100	169.455588	182.752004	3.561804335	3.333333333
3	100	4.978835	5.214135	2.369007804	58.33333333
4	100	14.701313	12.646981	2.875	0

Tabela B-25 - Dataset a classificar pelo meta-modelo (parte 12)

Nº	OutliersClass1	OutliersPercClass1	OutliersClass2	OutliersPercClass2	ClassBalance
1	2.254098361	58.33333333	1.024590164	58.33333333	93.65079365
2	1.031048623	0	1.874633861	0	59.3837535
3	1.755852843	50	0.863991081	50	47.29064039
4	0.652777778	0	0.763888889	0	100

Tabela B-26 - Dataset a classificar pelo meta-modelo (parte 13)

Nº	FeaturesImportanceGroup1	FeaturesImportanceGroup2
1	0	1
2	0.00100384	0.99899615
3	0.13014603	0.86985398
4	0.10389162	0.89610838

Anexo C: Apresentação das amostras do *dataset*

Em seguida, será possível perceber e observar os dados existentes em cada um dos *datasets* a análise. É possível verificar a distribuição dos valores entre os dois grupos existentes. Esta distribuição faz-se acompanhar pelos valores mínimos e máximos, valores do primeiro, segundo e terceiro quartil assim como o desvio padrão e a média. Podemos através da média ou mediana observar o centro de distribuição dos dados. É possível também observar a dispersão que é representada pela amplitude dos valores, ou seja, os valores máximos e mínimos que representam variações nos dados consoante a sua amplitude. Imagine-se então que a mediana é próxima do primeiro quartil, isto quer dizer que os dados são positivamente assimétricos, no entanto, se a mediana é próxima do terceiro quartil os dados são negativamente assimétricos.

Este processo de extração de informação referente à distribuição dos dados de cada um dos *datasets* repete-se para cada um destes.

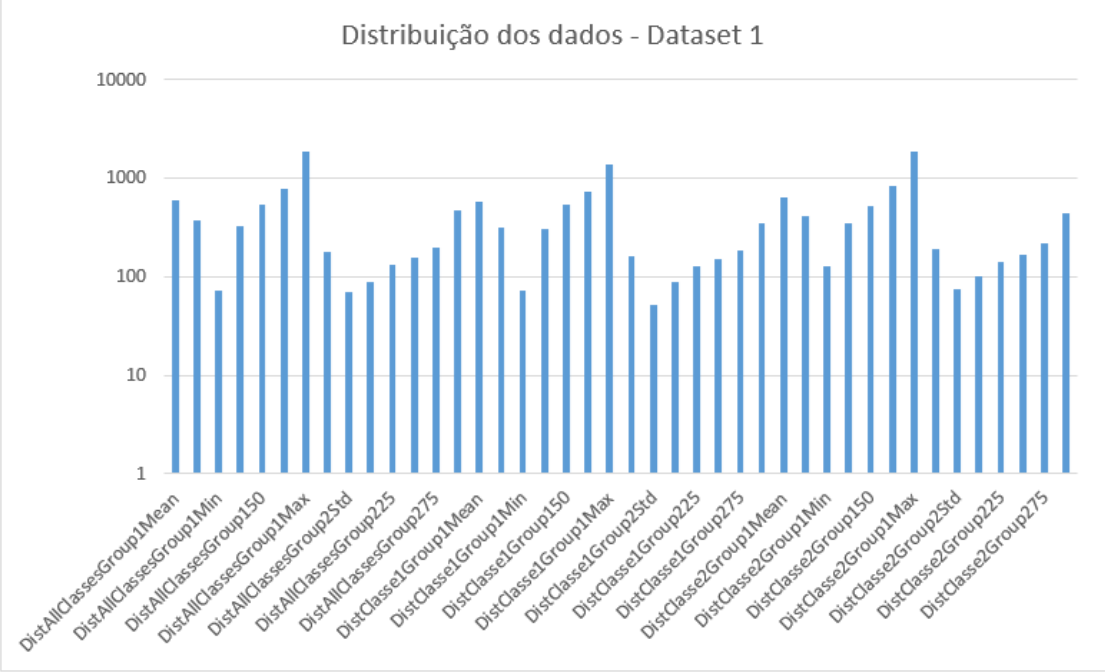

Figura C-1 - Distribuição dos dados do *dataset* 1

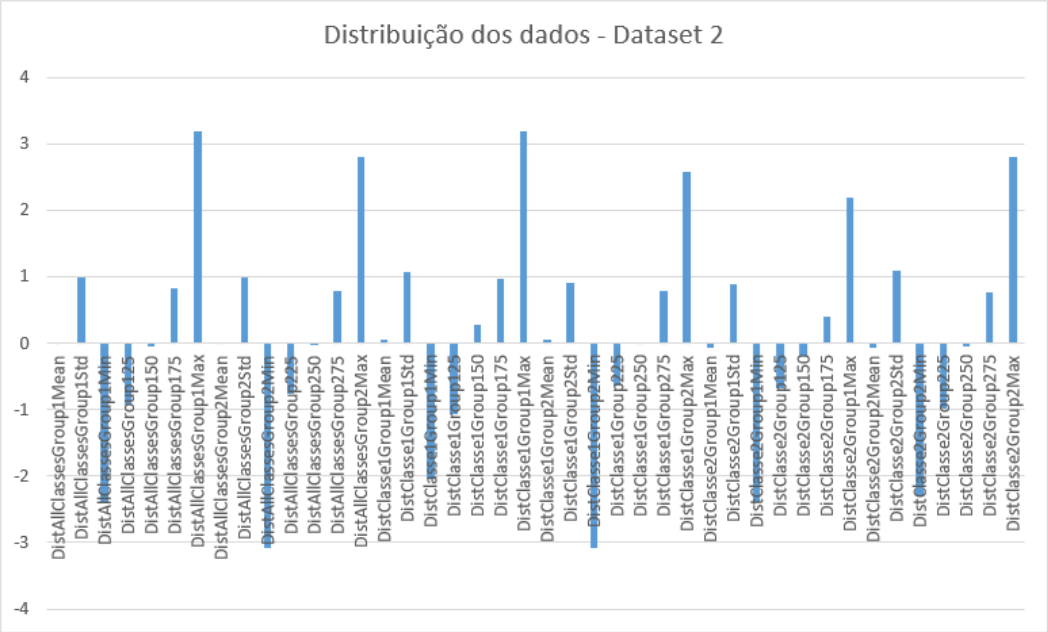

Figura C-2 - Distribuição dos dados do *dataset* 2

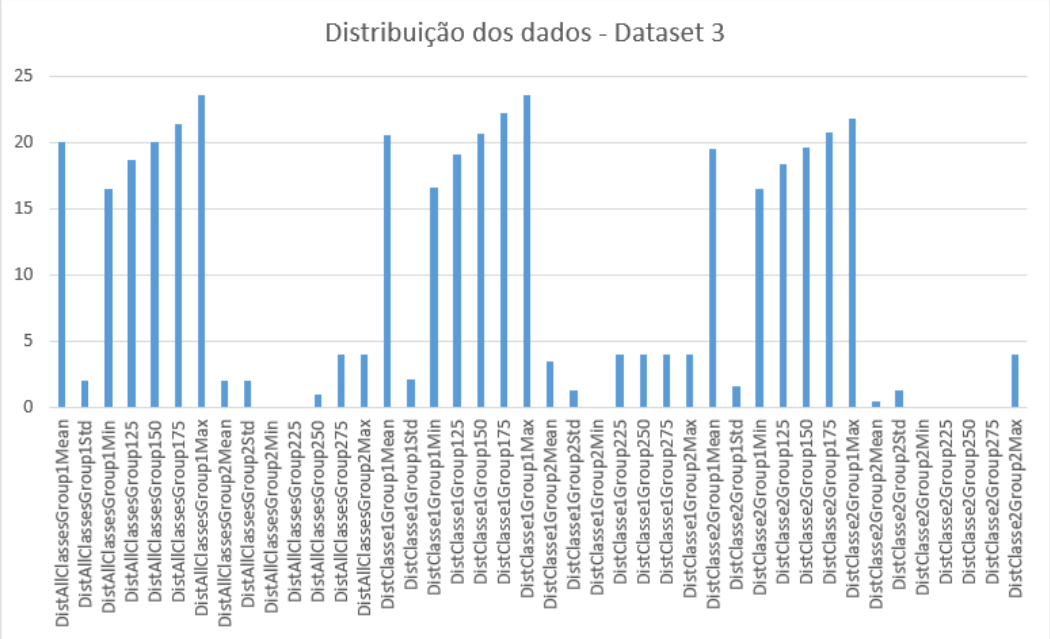

Figura C-3 - Distribuição dos dados do *dataset* 3

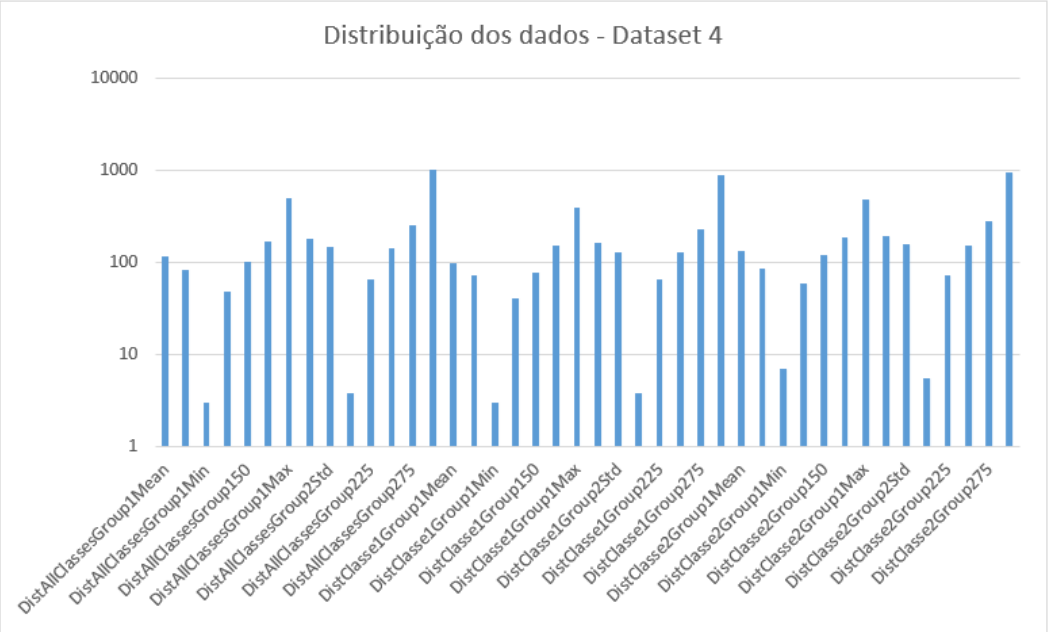

Figura C-4 - Distribuição dos dados do *dataset* 4

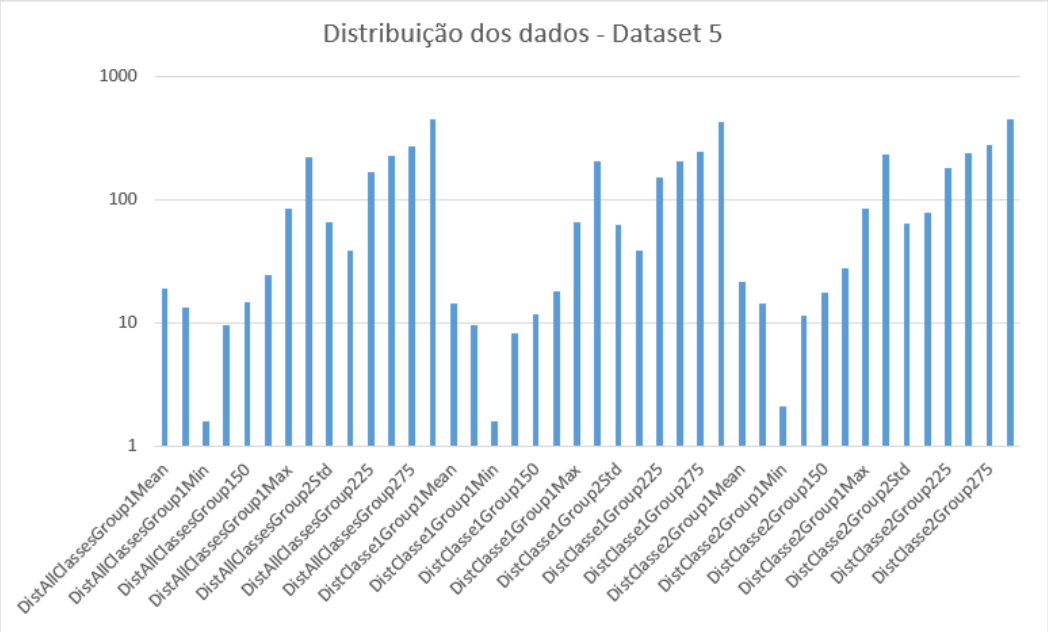

Figura C-5 - Distribuição dos dados do *dataset* 5

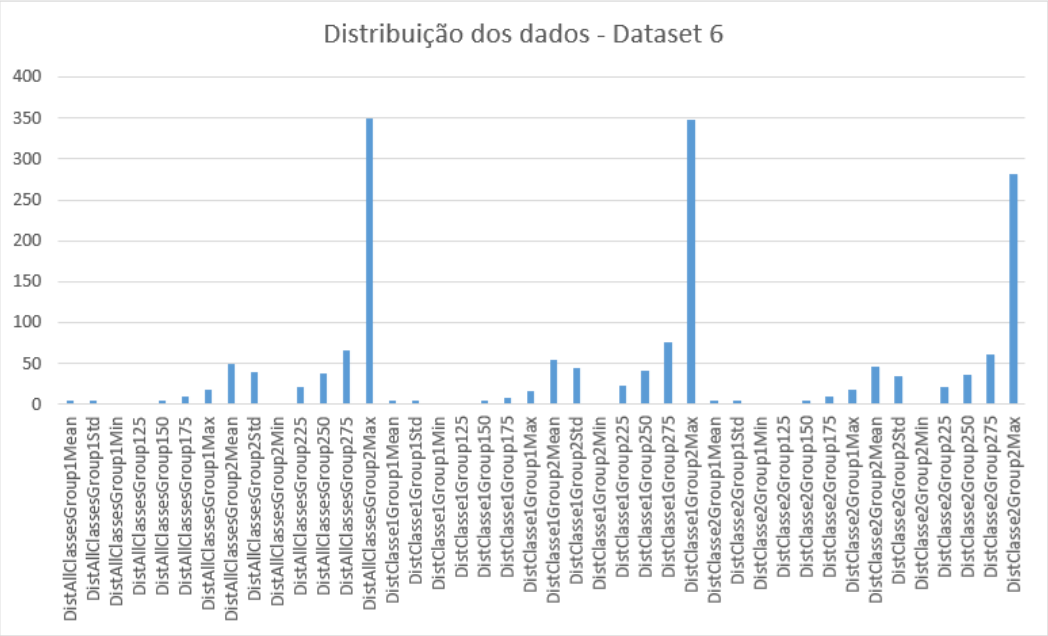

Figura C-6 - Distribuição dos dados do *dataset* 6

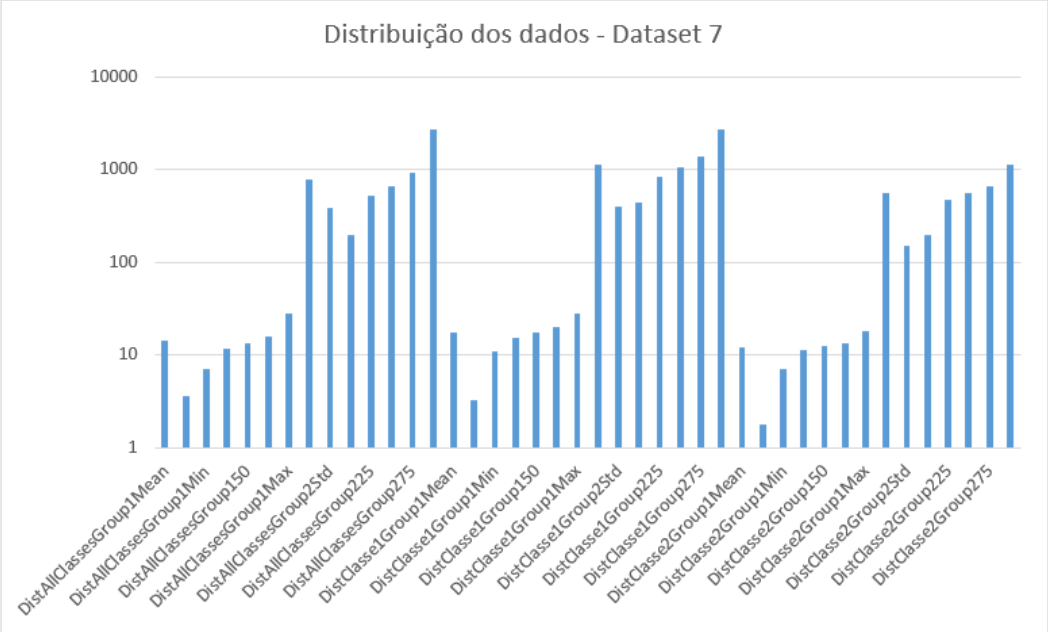

Figura C-7 - Distribuição dos dados do *dataset* 7

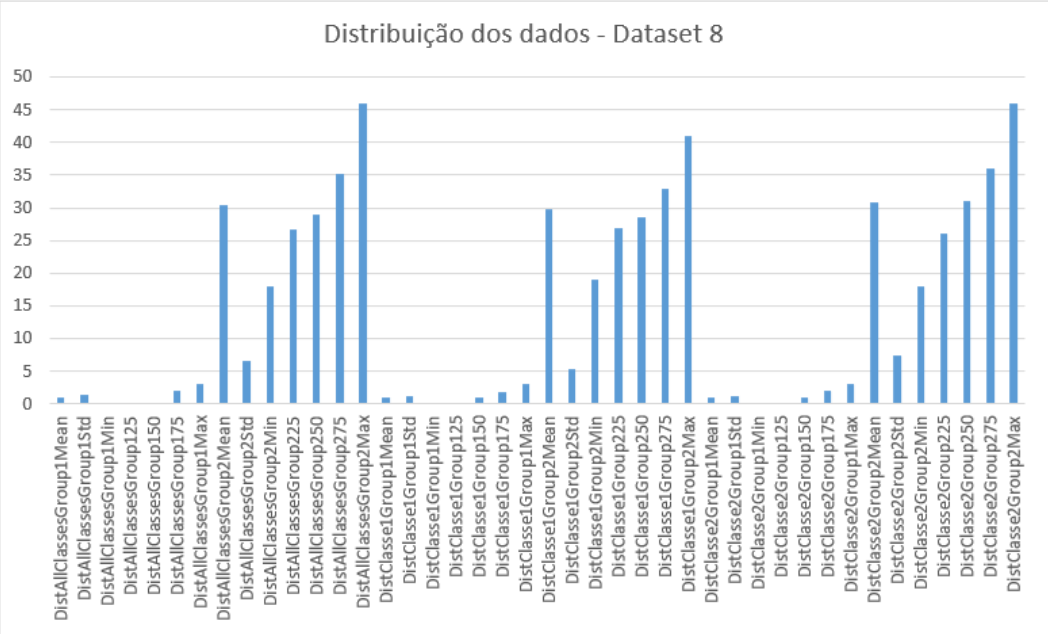

Figura C-8 - Distribuição dos dados do *dataset* 8

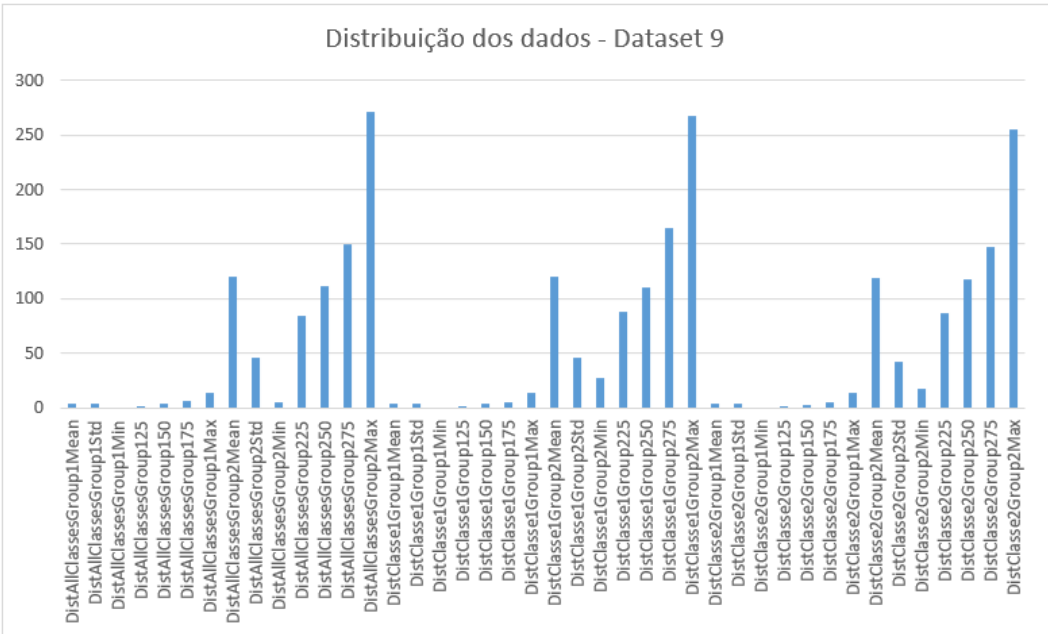

Figura C-9 - Distribuição dos dados do *dataset* 9

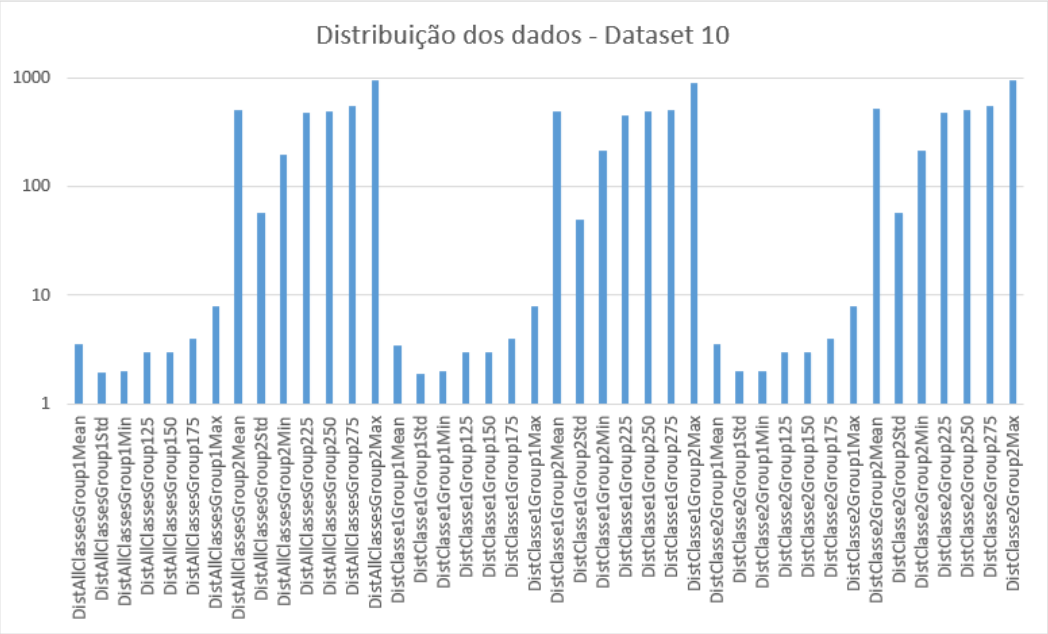

Figura C-10 - Distribuição dos dados do *dataset* 10

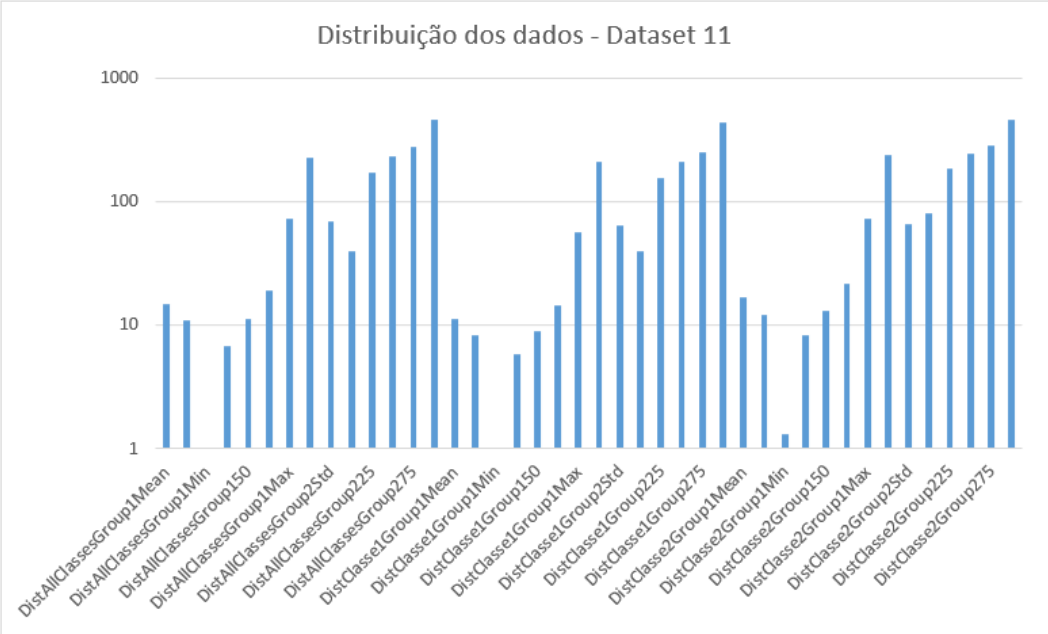

Figura C-11 - Distribuição dos dados do *dataset* 11

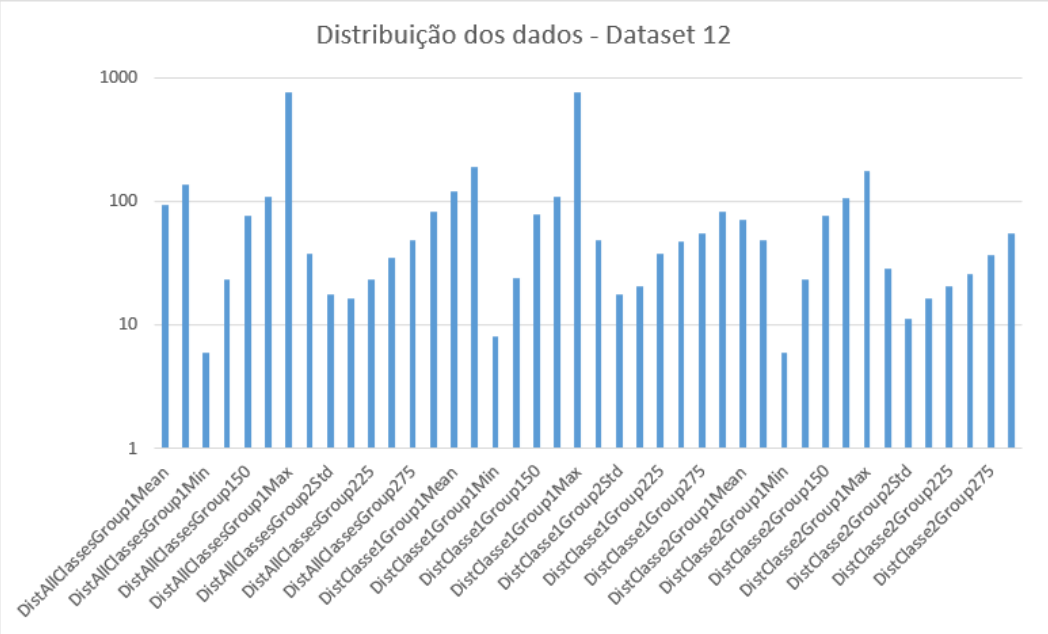

Figura C-12 - Distribuição dos dados do *dataset* 12

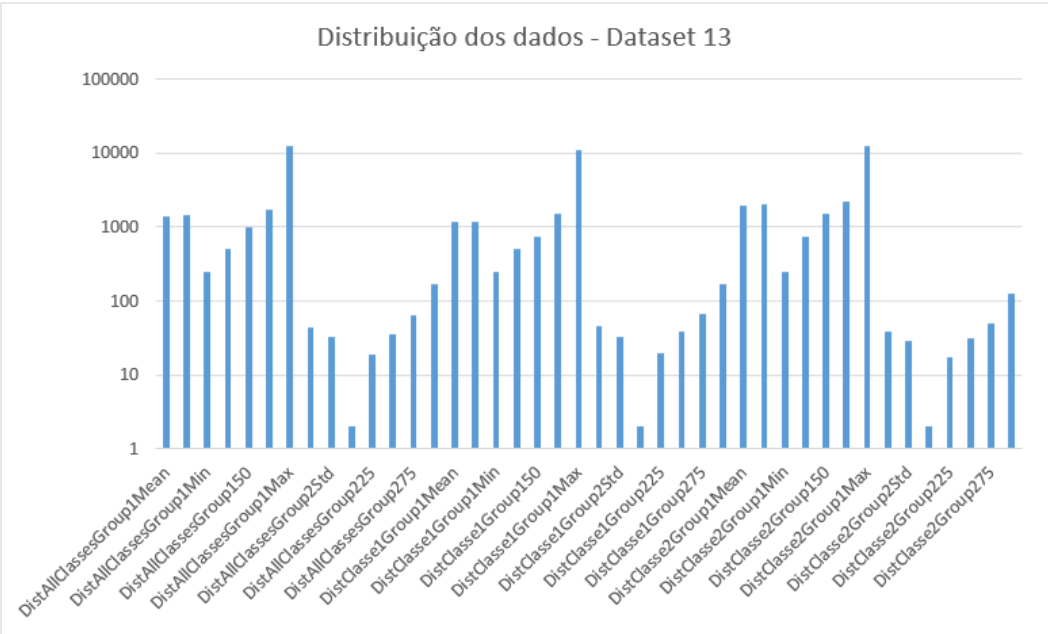

Figura C-13 - Distribuição dos dados do *dataset* 13

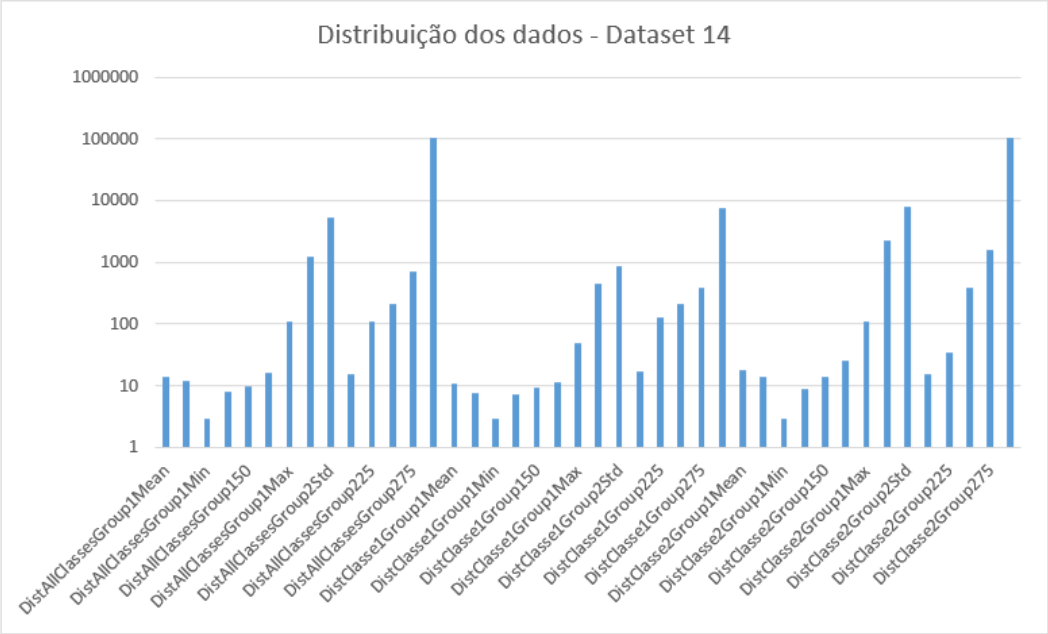

Figura C-14 - Distribuição dos dados do *dataset* 14

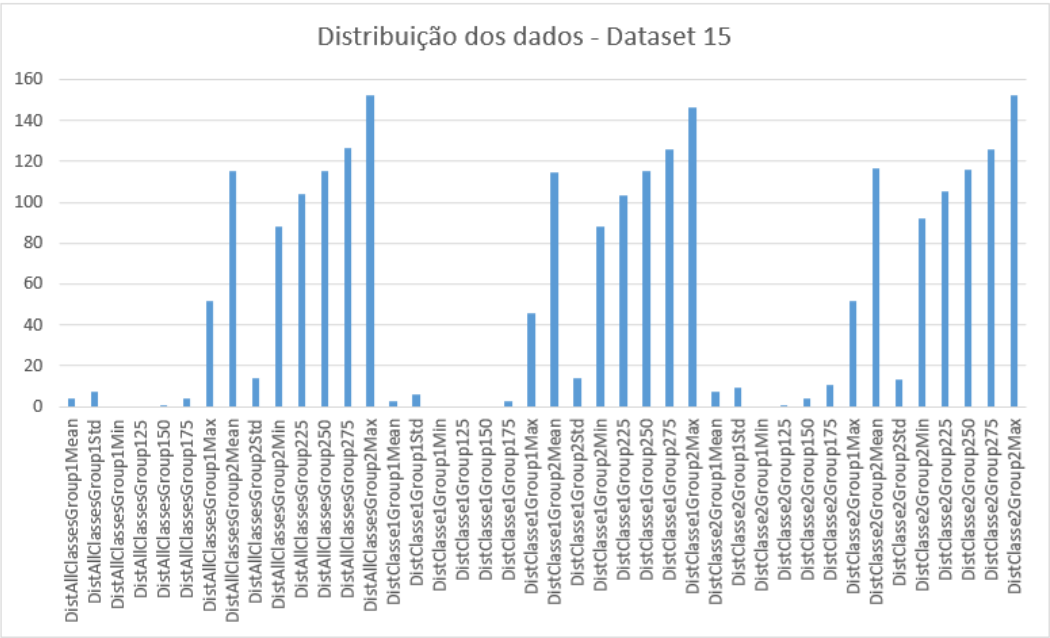

Figura C-15 - Distribuição dos dados do *dataset* 15

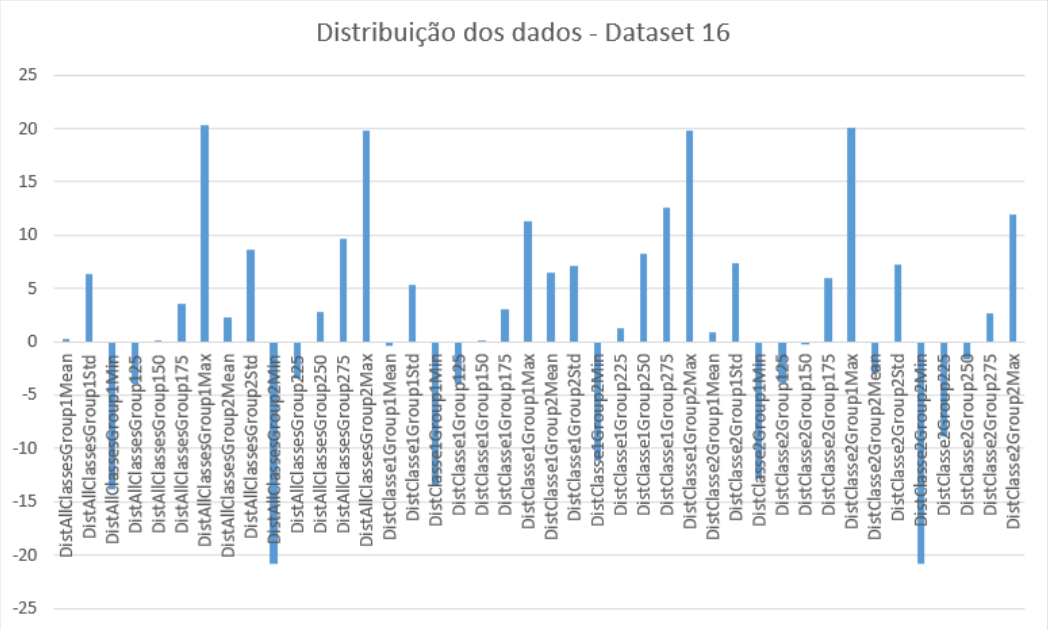

Figura C-16 - Distribuição dos dados do *dataset* 16

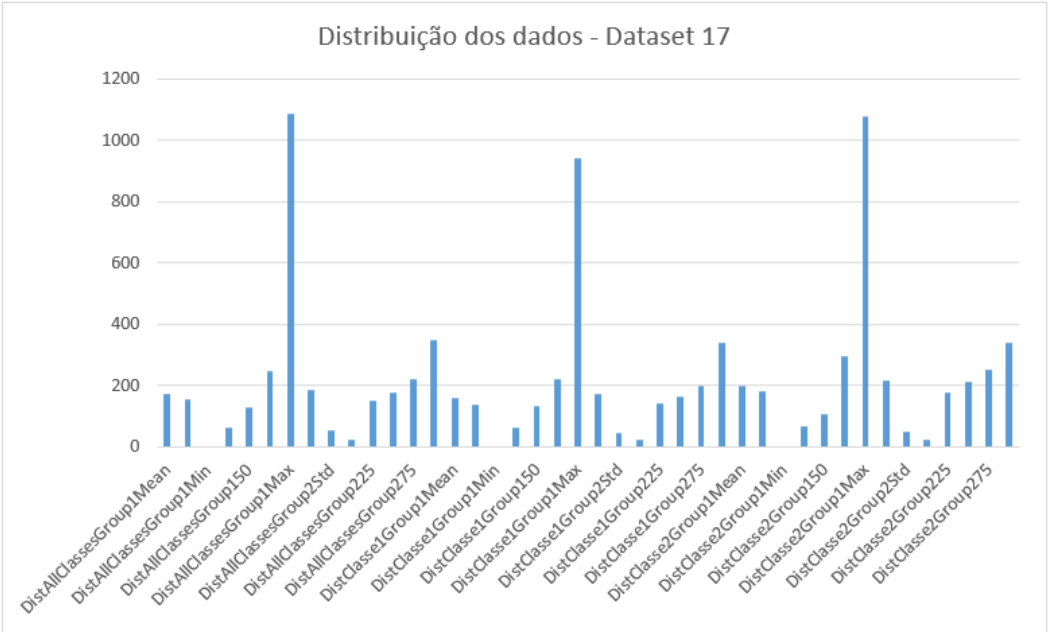

Figura C-17 - Distribuição dos dados do *dataset* 17

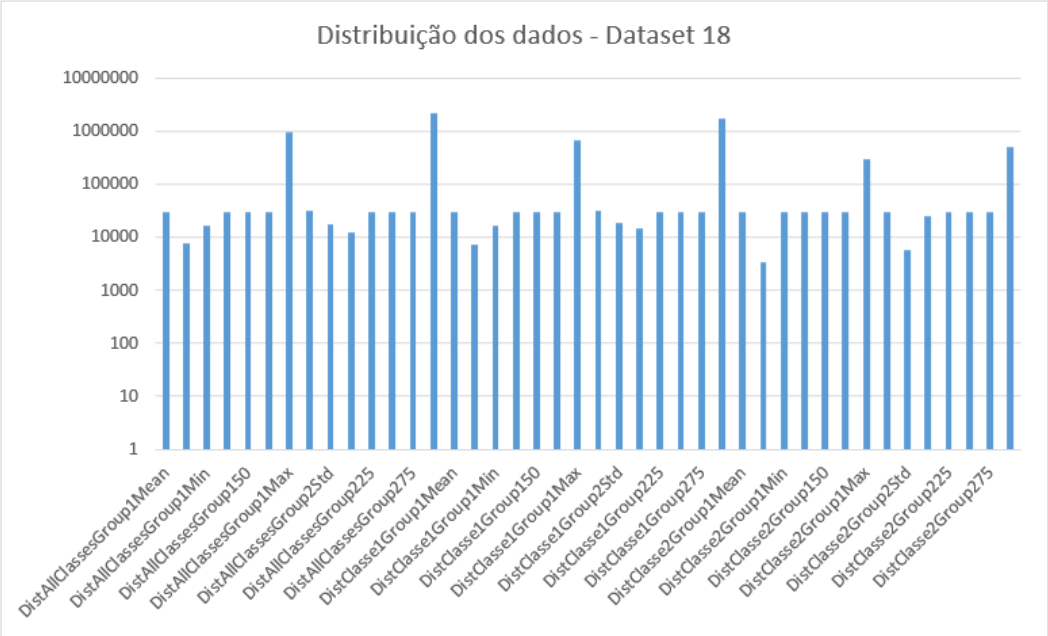

Figura C-18 - Distribuição dos dados do *dataset* 18

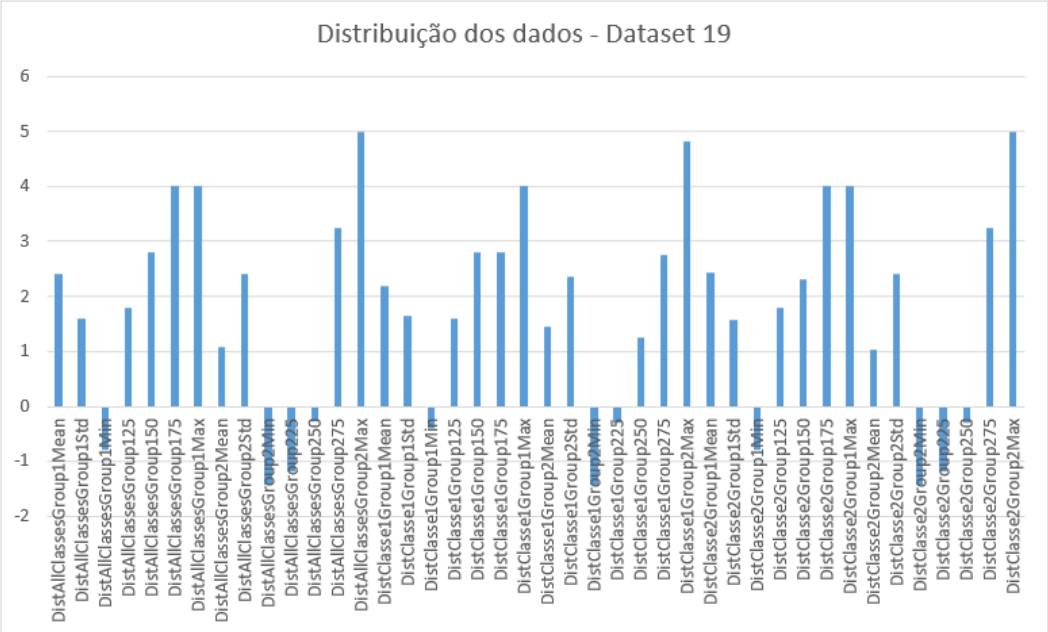

Figura C-19 - Distribuição dos dados do *dataset* 19

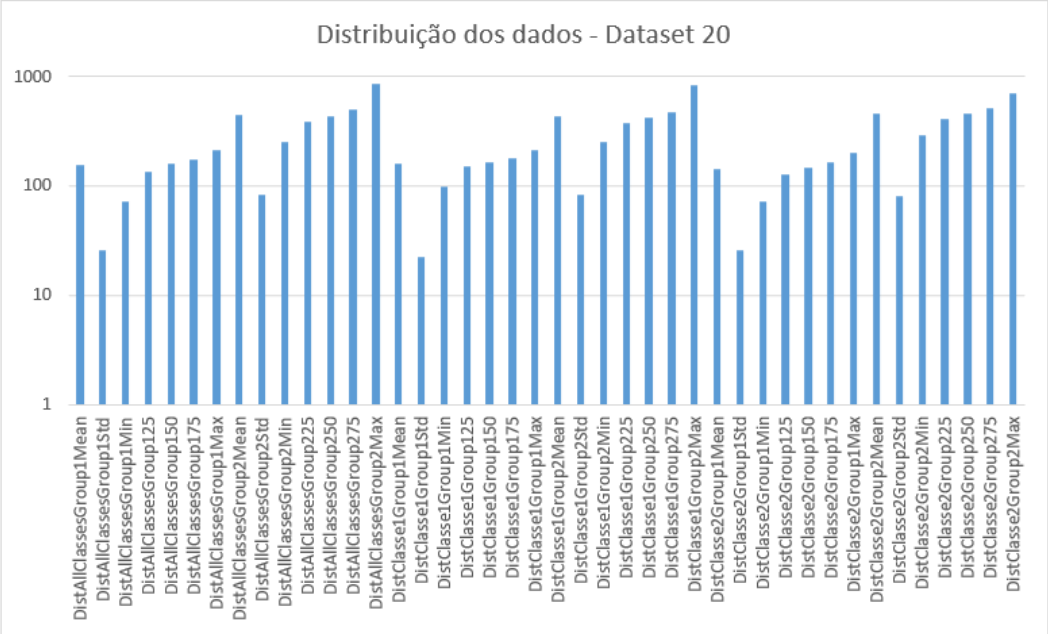

Figura C-20 - Distribuição dos dados do *dataset* 20

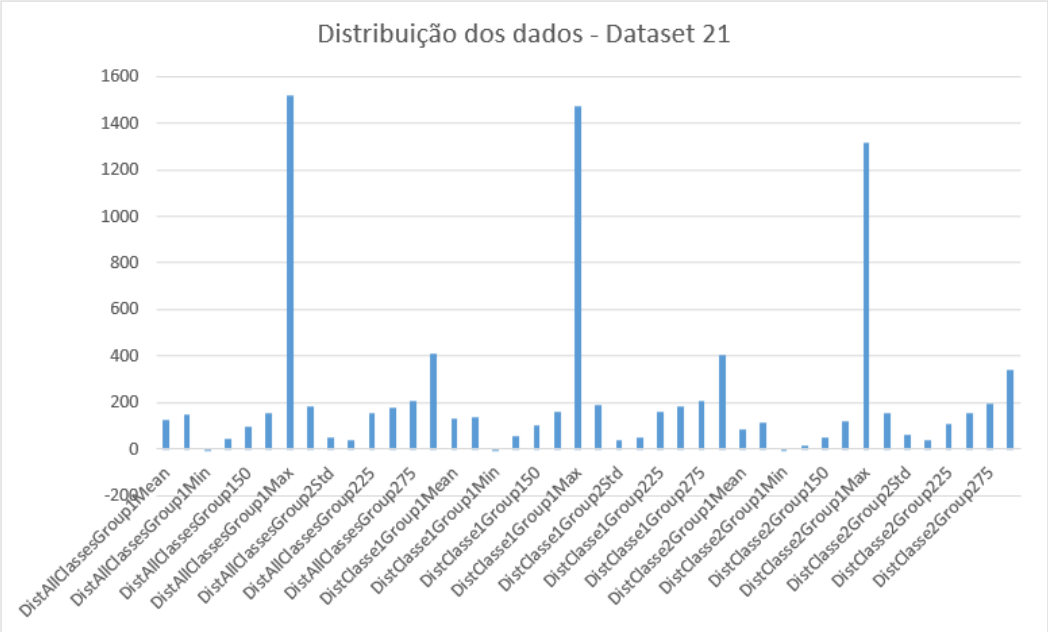

Figura C-21 - Distribuição dos dados do *dataset* 21

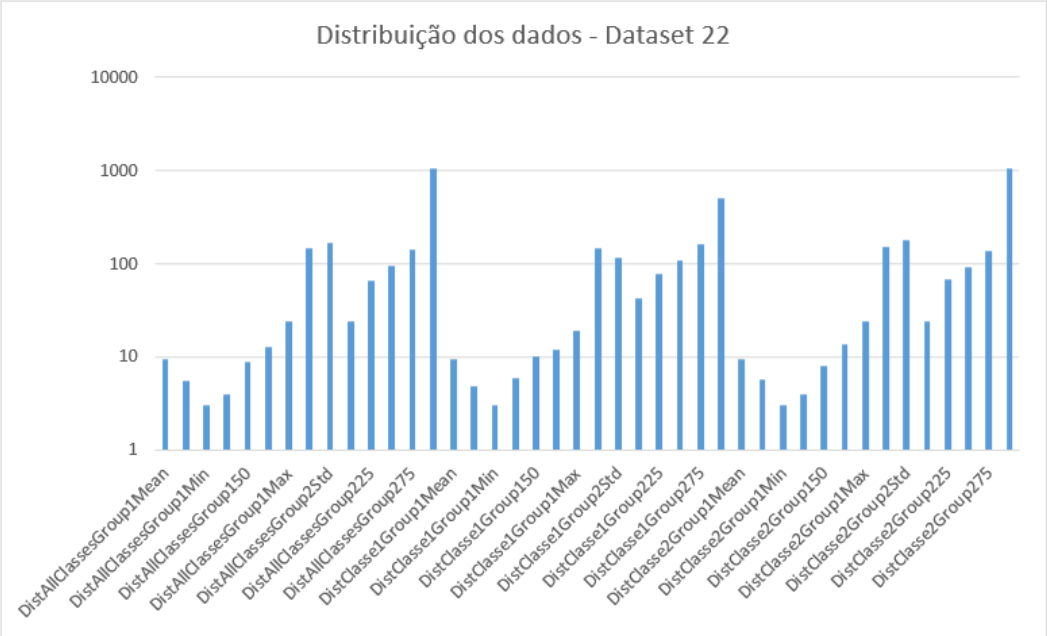

Figura C-22 - Distribuição dos dados do *dataset* 22

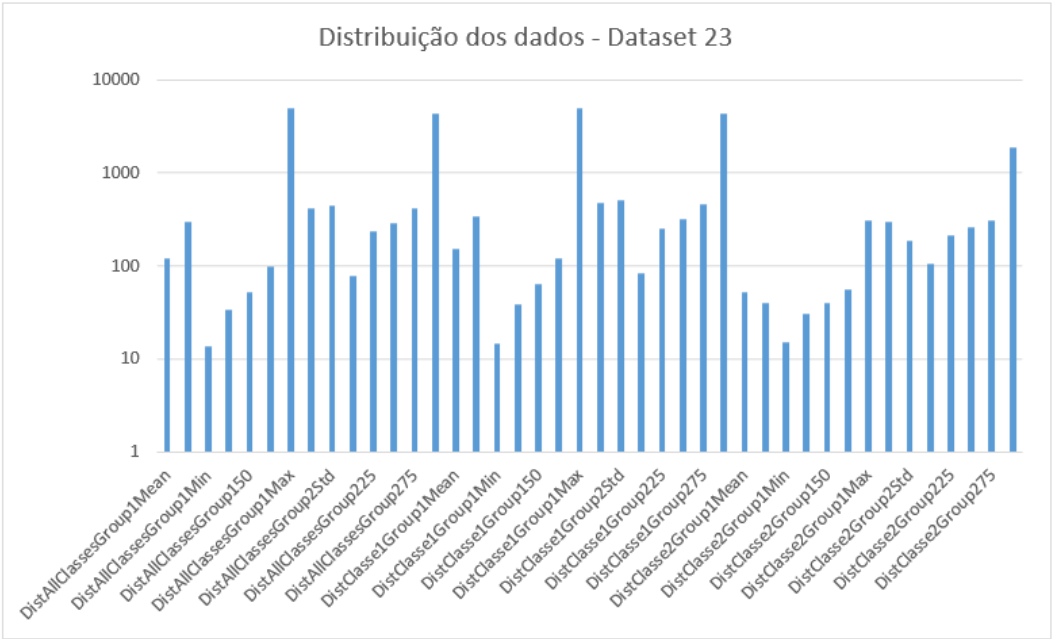

Figura C-23 - Distribuição dos dados do *dataset* 23

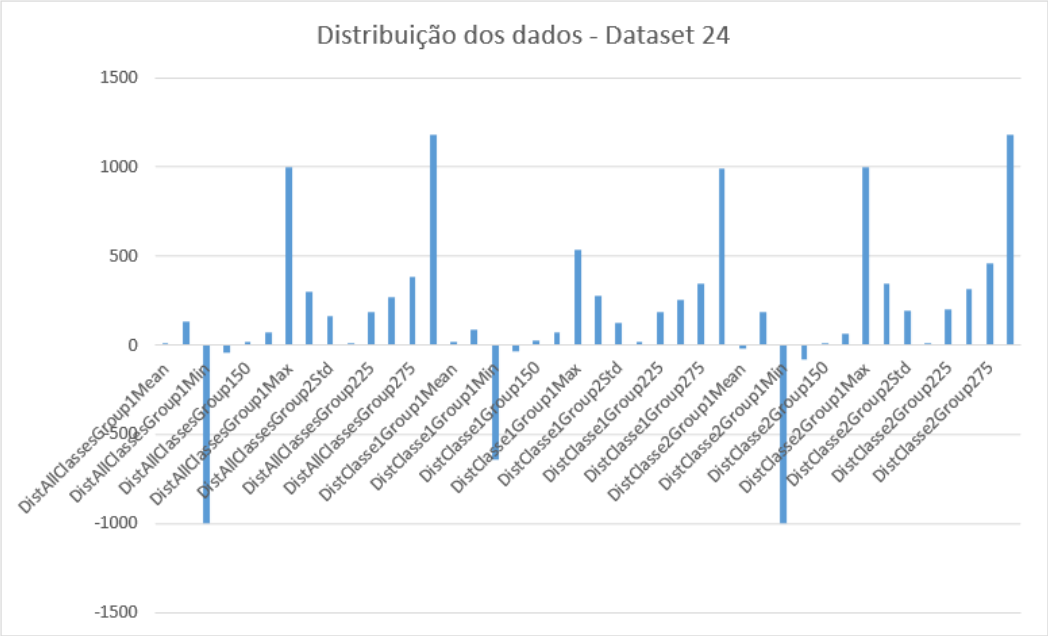

Figura C-24 - Distribuição dos dados do *dataset* 24

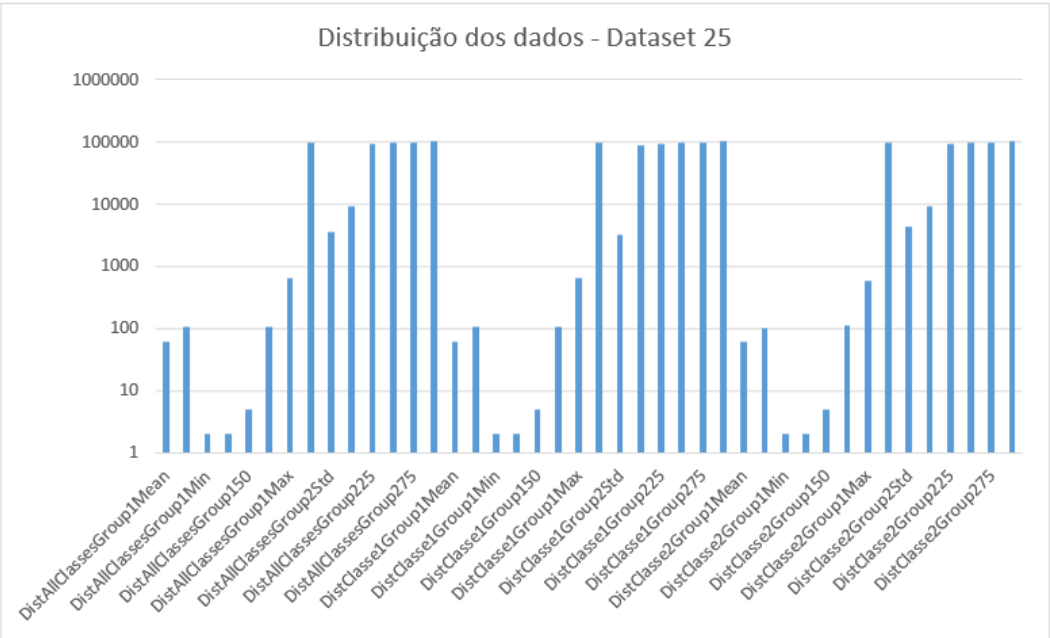

Figura C-25 - Distribuição dos dados do *dataset* 25

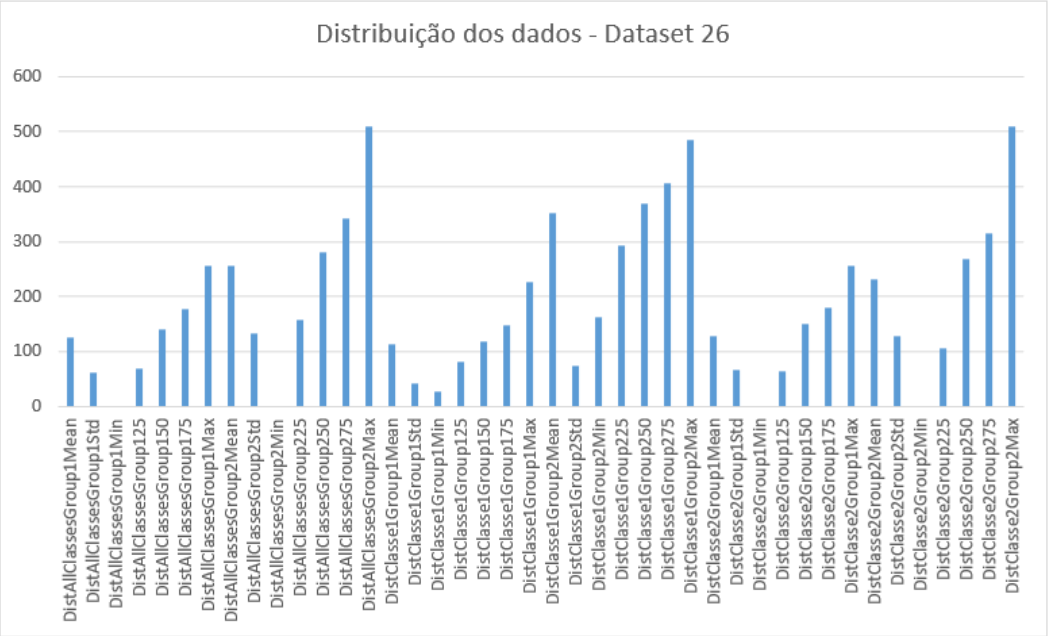

Figura C-26 - Distribuição dos dados do *dataset* 26

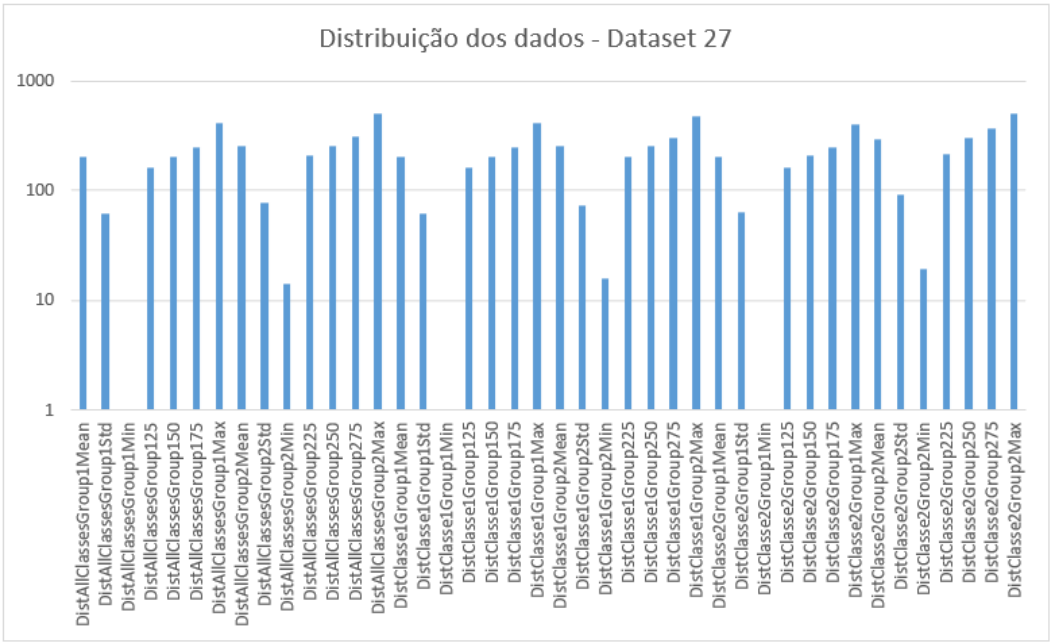

Figura C-27 - Distribuição dos dados do *dataset* 27

Com as figuras seguintes é possível ter uma real perceção do balanceamento entre as classes existentes entre cada um dos *datasets*. Sendo que, e interpretando o gráfico 100% significa que existe um balanceamento total, ou seja, o número de amostras de cada uma das classes é bastante semelhante ou igual.

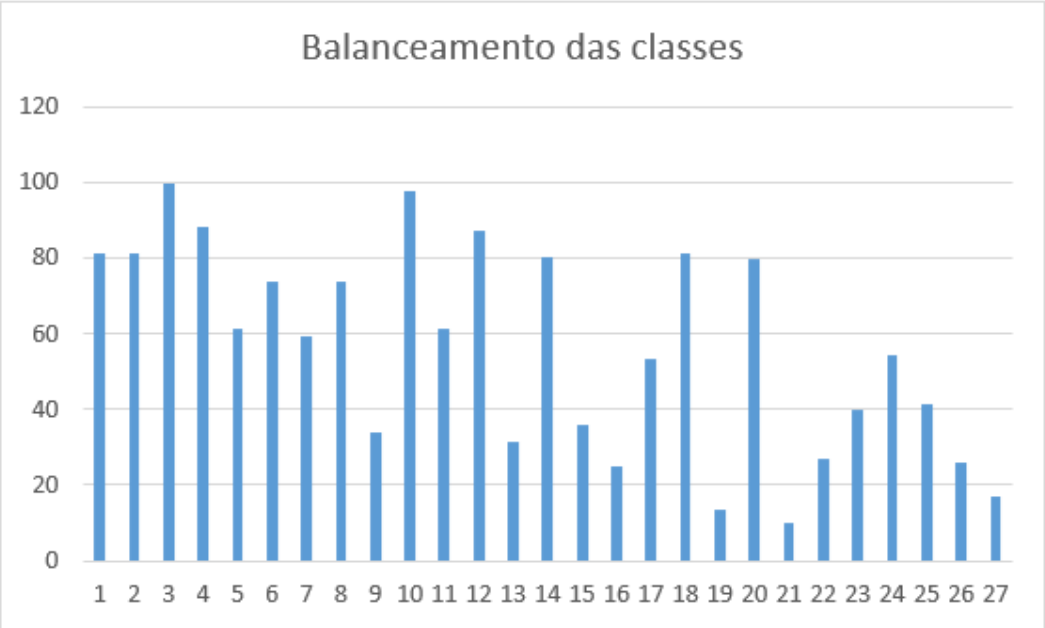

Figura C-28 - Balanceamento entre Classes

Sendo a próxima figura associada à importância das características referentes ao grupo 1 e 2 nos diversos *datasets* a análise.

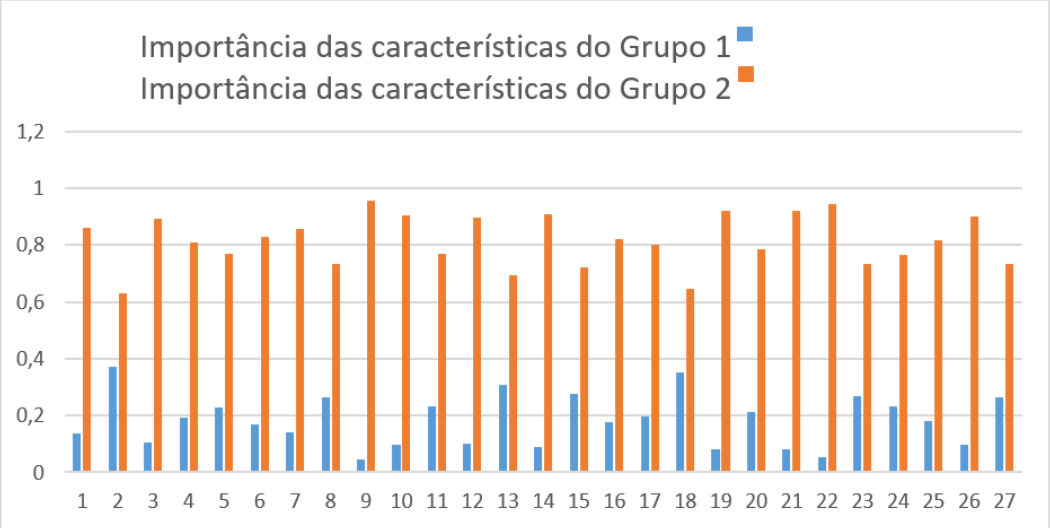

Figura C-29 - Importância das Características

Anexo D: Estudo Inicial / Caso de teste

Introdução

Este estudo tem como objetivo analisar uma grande quantidade de dados e, através destes, conseguir identificar e extrair conhecimento aplicando técnicas e algoritmos de inteligência artificial. Irá ser analisado um *dataset* disponibilizado e armazenado em "*machine learning repository*" sendo este um *dataset* já processado, contendo apenas informação já tratada e consequentemente pronta a ser estudada. O *dataset* contém informação referente a sintomas associados a problemas cardíacos de diversos indivíduos. Toda esta informação vai alimentar um algoritmo de inteligência artificial, que neste caso é a árvore de decisão, onde este vai ser "manipulado" com o objetivo de melhorar a sua performance no que toca ao estudo dos dados.

Dados para estudo – *Dataset*

Tabela D-1 - *Dataset*-Doenças Cardíacas
Fonte: https://archive.ics.uci.edu/ml/datasets/heart+disease

Age	Sex	CP	Trestbps	Chol	Fbs	Restecg	Thalach	Exang	Oldpeak	Slope	Ca	Thal	Target
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0

Para este estudo foram usados dados presentes na "*machine learning repository*", contendo estas 14 propriedades que vão ser usadas para efeitos de análise e estudo, sendo que na tabela D-1 está representado apenas um pequeno exemplo do *dataset* a ser analisado. Cada amostra tem um campo "target" que determina a ausência ou não de sintomas de problemas de coração medidos de 0 a 4 por ordem de intensidade. O *dataset* usado para este estudo, no entanto, já se encontra totalmente processado, contendo apenas os 14 atributos a serem analisados. Cada atributo irá adicionar informação relevante para que o algoritmo seja capaz de construir um modelo inteligente. Este *dataset* conta com 303 amostras, sendo que todas vão entrar para este estudo.

No entanto, este *dataset* contém a ausência de valores em determinadas amostras. Para abordar esta situação existem duas maneiras de lidar com a ausência de valores, a saber:

- Pela substituição dos valores em falta:
  - Substituir pelo valor médio presente na coluna no *dataset*;
  - Substituir pela mediana presente na coluna no *dataset*;
  - Substituir pelo valor mais frequente – moda – da coluna do *dataset*.
- Remover as amostras dos valores em falta:
  - Para este caso é necessário remover as amostras onde os valores são nulos ou ausentes, normalmente usado quando essa coluna em específico contiver

70%-75% dos valores em falta. Este método é recomendado apensa quando existirem amostras suficientes no conjunto de dados. A remoção das amostras pode levar à perda de informação.

Neste caso, foi usada a primeira opção, ou seja, sempre que exista a ausência de valor numa amostra, este valor é substituído pela média de valores nessa mesma coluna tendo em conta os restantes registos do *dataset* e da classe a que pertence.

Características que constituem o *dataset* sendo estes valores numéricos:

- *Age* = Idade do indivíduo;
- *Sex* = Sexo do indivíduo;
- *Cp* = Dor no peito (1-4);
- *Trestbps* = Pressão sanguínea em repouso;
- *Chol* = Colesterol;
- *Fbs* = Açúcar no sangue em jejum. (0-1);
- *Restecg* = Resultados de uma eletrocardiograma em repouso. (0,1,2) ;
- *Thalach* = Frequência cardíaca máxima alcançada;
- *Exang* = Angina (dores peitorais) induzida pelo exercício (0,1);
- *Oldpeak* = Depressão no segmento ST induzida pelo exercício em relação ao repouso;
- Slope = Inclinação do pico de exercício segmento ST (1,2,3);
- Ca = Número de grandes vasos (0,1,2) colorido por fluoroscopia;
- Thal = 3 = normal; 6 = defeito fixo; 7 = defeito reversível;
- Num = Atributo previsto, se tem ou não problemas cardíacos numa escala de 0-4.

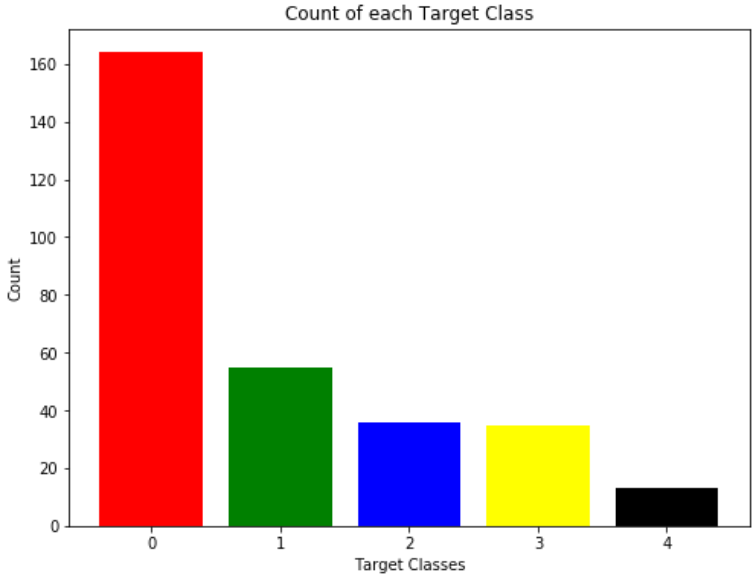

Figura D-1 - Amostras do *Dataset*

Para se ter uma melhor perceção das amostras presentes no *dataset*, é possível através da figura D-1 verificar que no *dataset* existem 164 amostras com "*target*" 0, ou seja, amostras de dados nos quais não existem indícios de problemas cardíacos, 55 amostras com target 1, 36 amostras com target 2, 35 amostras com target 3 e 13 amostra com target 4 no qual significa que existem 13 amostras com indícios muito fortes da existência de problemas cardíacos.

```
# load dataset
dataset = pd.read_csv("C:/Dir/heart-disease.csv", header=None,usecols=(0,1,2,3,4,5,6,7,8,9,10,11,12,13), names=col_names)
y = dataset['target']
X = dataset.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X_imp, y, test_size = 0.33, random_state = 0)
```

Figura D-2 - Processamento dos dados a análise

Na figura D-2 é possível verificar um exemplo de como os dados são carregados na variável *dataset* para que seja possível posteriormente separar dados destinados aos testes e dados destinados ao treino do modelo.

Estratégia usada

Após algum estudo e análise do problema em questão, problema este que passa por analisar uma grande quantidade de dados e usá-los para criar um modelo inteligente capaz de prever se um determinado indivíduo tem ou não problemas cardíacos. Para iniciar este estudo será tomada a decisão de fazer três testes no sentido de apurar aquele que apresentam melhores resultados, e analisar métricas de avaliação sempre que os parâmetros para a construção do modelo forem alterados.

- **Opção 1:** Efetuar testes com os dados existentes.
- **Opção 2:** Balancear os dados, ou seja, aplicar as técnicas de *Under-sampling* às amostras com target 0 e *over-sampling* às amostras com target 1,2,3,4 até encontrar um ponto de equilíbrio.
- **Opção 3:** As amostras com target 0 ficam inalteradas sendo que as restantes amostras com target 1,2,3,4 ficarão com o mesmo target, ou seja, vão passar a ter o target 1.

Métricas de avaliação

Antes de iniciar a análise das métricas de avaliação a serem usadas durante a construção do modelo, é importante apresentar algumas definições, tais como:

- Verdadeiros Positivos (TP) - Amostra que foi predita como positiva e que na realidade era positiva.
- Verdadeiros Negativos (TN) - Amostra que foi predita como negativa e que na realidade era negativa.
- Falsos Positivos (FP) - Amostra predita como positiva mas que na realidade era negativa.
- Falsos Negativos (FN) - Amostra predita como negativa mas que na realidade era positiva.

Em cada opção terá a apresentação de várias métricas de avaliação com o objetivo de se ter uma melhor perceção do modelo gerado como por exemplo:

Matriz de confusão: A matriz de confusão (ver figura D-3) é o resultado de valores previstos em problemas de classificação. Através da matriz de confusão conseguimos medir o sucesso no que diz respeito às previsões do modelo comparando com as amostras atuais e preditas. É possível também utilizar a matriz de confusão para verificar que tipo de classes são mais facilmente confundidas com outras classes.

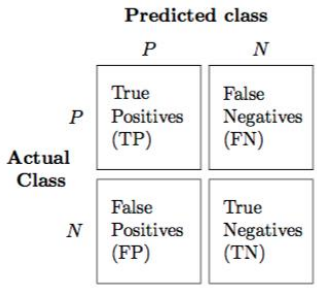

Figura D-3 - Matriz de Confusão exemplo

***Accuracy*:** *Accuracy* permite verificar o número de previsões corretas feitas pelo modelo em todos os tipos de previsões feitas. Esta métrica não responde bem se não existir um bom balanceamento entre as classes. *Accuracy* = (TP+TN) / (total de exemplos).

***Precisão*:** Responde à pergunta: "Quando prevê o resultado positivo, com que frequência está correto?" A precisão é utilizada geralmente quando o objetivo é limitar o número de falsos positivos. No nosso caso seria a métrica a usar se o objetivo fosse focado em filtrar as classes classificadas como 0 mas que na realidade são 1,2,3 ou 4 e assim sucessivamente. *Precision* = TP / (TP+FP).

***Recall*:** *Recall* é a percentagem de exemplos positivos no conjunto de exemplos positivos que o modelo foi capaz de identificar. Este é usado também quando o objetivo é limitar o número de falsos negativos. O objetivo é minimizar o número de classes classificadas como 1, 2, 3, 4, mas que na realidade foram classificadas como 0 e assim sucessivamente. *Recall* = (TP) / (TP+FN).

F1*:** Sinteticamente, é a combinação entre as métricas *precision* e *recall*, e é usado para indicar o balanceamento entre estas duas métricas. *F1*=2*((*precisionrecall*) / (*precision*+*recall*)).

Para estas três últimas métricas de avaliação serão apresentados dois parâmetros diferentes entre cada uma delas no sentido de ter uma melhor perceção dos resultados obtidos, tais como:

> **'*micro*':** Calculado *F1*/*Recall*/*Precision* agregará as contribuições de todas as classes para calcular a média. Numa configuração de classificação de várias classes, a micro-média é preferível num episódio de desequilíbrio nas classes.
>
> **'*macro*':** Calculará a métrica independentemente para cada classe e, em seguida, obterá a média (portanto tratando todas as classes igualmente).

Geração do modelo

Para a construção do modelo inteligente será necessário implementar um algoritmo que seja capaz de responder corretamente às necessidades deste estudo. Assim sendo, será usado neste estudo a biblioteca livre providenciada pela *Scikit-learn* para que seja possível de uma maneira "simples", "rápida" e "eficaz" proceder à construção de um modelo inteligente denominado por árvore de decisão. Esta biblioteca é um módulo *python* baseado no *NumPy* e *SciPy* e, no que respeita à análise de dados, é das melhores bibliotecas criadas até ao momento. Este módulo disponibiliza diversos algoritmos para diversas tarefas-padrão relacionadas com o desenvolvimento de uma máquina de aprendizagem, assim como a mineração de dados, clustering de dados, regressão, classificação, redução de dimensionalidade e seleção de modelo.

Para este estudo é necessário construir um objeto "*DecisionTreeClassifier*", providenciando diversos parâmetros para desta forma manipular a sua construção. Em seguida, serão apresentados os parâmetros usados na construção deste modelo através da figura D-4 e consequente justificação em seguida.

```
DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2,
                       min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None,
                       random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, class_weight=None, presort=False)
```

Figura D-4 - Parametrização da Árvore de decisão

criterion: string, optional (default='gini')

Parâmetro que permite medir a qualidade da separação dos atributos. O critério Gini tem como objetivo separar todas as classes no sentido de diminuir a impureza, considerando que uma população é pura quando pertencem todas à mesma classe. Outro critério possível de aplicar é o critério *entropy*, que tem como objetivo maximizar a pureza dos dados, é usada cada vez que é criado um nó da árvore. Em cada separação o objetivo é diminuir a entropia entre os dados, uma

vez que estes cálculos são feitos antes e depois da separação e, caso a entropia diminua, prossegue-se para a divisão seguinte.

***splitter*: string, optional (default='best')**

Existem duas técnicas de separação, a técnica *best*, que é aquela que oferece melhores valores depois de medir a qualidade da separação. Em cada nó o algoritmo considera todos os atributos e escolhe o melhor a dividir, ou seja, escolhe aquele que uma vez dividido oferece mais informação à criação da árvore. Já o parâmetro definido como *random* irá selecionar o atributo a sofrer a separação de forma aleatória.

***max_depth*: int or None, optional (default=None)**

Define a profundidade máxima que a árvore pode atingir. Se o parâmetro "max_depth" não for especificado a árvore irá crescer até que todas as folhas sejam puras. Quanto mais profunda a árvore for, mais divisões terá, assim como a retenção de mais informação sobre os dados.

***min_samples_split*: int, float, optional (default=2)**

Número mínimo de características necessário para dividir um nó. Sempre que necessário proceder a uma divisão em um determinado nó o presente parâmetro permite definir o número mínimo de características a ter em consideração para proceder a sua separação e definir um critério de divisão.

***min_samples_leaf*: int, float, optional (default=1)**

Número mínimo de amostras necessário para ser considerado um nó folha. Por exemplo, imagine-se um exemplo em que *min_samples_split*=5 e existem 7 amostras num determinado nó, a divisão será permitida resultando em duas folhas sendo que uma delas com 1 amostra e a segunda com 6 amostras por exemplo. Se *min_samples_leaf*=2 a divisão não será permitida mesmo o nó contendo 7 amostras porque uma das folhas resultantes terá menos que o número mínimo de amostras necessário para criar um nó folha.

***min_weight_fraction_leaf*: float, optional (default=0.)**

O mínimo da soma total dos pesos de todas as amostras de entrada necessário para estar num nó-folha. As amostras têm um peso igual quando *sample_weight* não é fornecido. É semelhante ao parâmetro anterior "*min_samples_leaf*" mas ao invés de ser necessário ter um número mínimo de amostras necessário para ser considerado um nó folha é necessário conter uma fração de amostras (pesos) para ser considerado nó folha.

***max_features*: int, float, string or None, optional (default=None)**

Número máximo de características a serem consideradas para efetuar a separação de um nó. Imagine-se um exemplo com 50 atributos com *max_features*=10, cada vez que é necessário

proceder à divisão de um nó, 10 atributos são selecionados aleatoriamente no intuito de encontrar o melhor atributo a ser dividido. Este processo é repetido sempre que necessário dividir um nó.

***random_state*: int, RandomState instance or None, optional (default=None)**

Controla a obrigatoriedade que é aplicado aos dados antes de ser feita a divisão. Sempre que o parâmetro *random_state* não é especificado os dados destinado ao treino e teste do modelo irão conter valores diferente em cada execução ao contrário de uma definição deste parâmetro que uma vez definido os dados dedicados ao treino e ao teste do modelo serão sempre os mesmos.

***max_leaf_nodes*: int or None, optional (default=None)**

A árvore de decisão vai crescer/formar até existir um máximo de nós folha. Ao definir este parâmetro o algoritmo irá gerar primeiramente o nó folha com maior diminuição de impureza. É possível observar numa árvore de decisão com diferentes profundidades, onde é possível priorizar aqueles que diminuem mais a impureza existente nos dados.

***min_impurity_decrease*: float, optional (default=0.)**

O nó é dividido apenas se essa divisão adicionar uma diminuição da impureza maior ou igual a valor providenciado. A divisão de um nó é realizada tendo em conta o ganho de informação gerado ao fazer a divisão, através deste processo é possível fazer a melhor divisão significando isto que quando a impureza é 0 todas as amostras no nó folha estão corretamente classificadas. O que este parâmetro faz é precisamente impedir a divisão de um nó sempre que o valor da diminuição da impureza gerada com essa divisão seja inferior ao valor fornecido.

***class_weight*: dict, list of dicts, “balanced” or None, default=None**

Atribuição de pesos às classes. Por padrão, todas as classes têm o mesmo peso, se necessário é possível atribuir diferentes pesos a cada classe. Existindo duas classes sendo que a classe 1 faz se representar com mais amostras que a classe 2, podemos concluir que a classe 1 terá mais peso e influência uma vez que contém mais amostras e informação associada. Este parâmetro permite atribuir, se necessário, mais peso à classe 2 e igualando a influência que ambas as classes podem conter. Os pesos que cada classe pode conter influenciará a classificação de classes na fase de treino do algoritmo.

Exemplo de aplicação do algoritmo:

```
#Decision Tree Object
dt_classifier = DecisionTreeClassifier(criterion='entropy', splitter='best', random_state = 0)
clf = dt_classifier.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

Figura D-5 - Aplicação do Algoritmo

Na figura D-5 é possível verificar um pequeno exemplo de como criar um objeto relativo à árvore de decisão usando os dados anteriormente processados.

Testes

Opção 1:

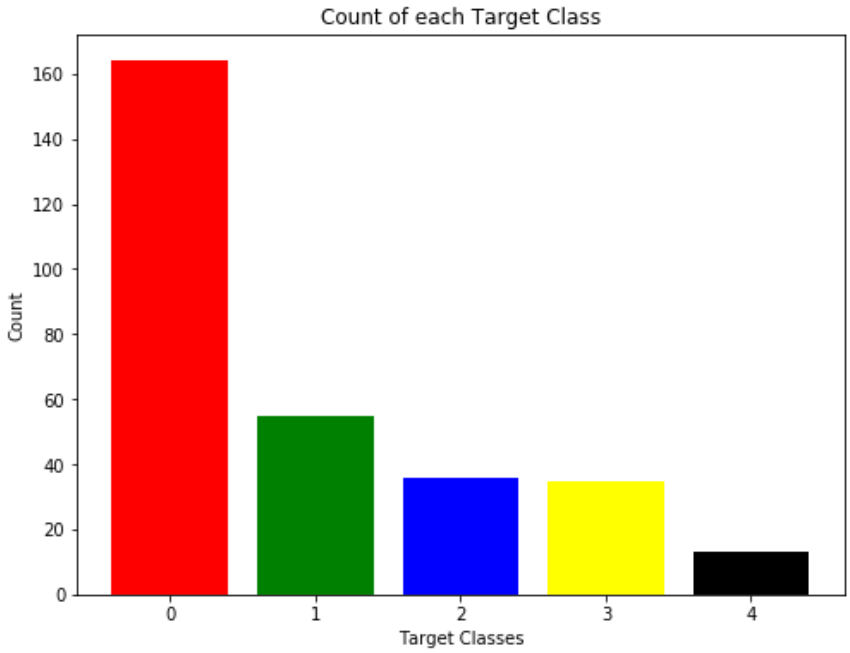

Figura D-6 - Divisão de amostras no teste 1

Neste caso de estudo, foram analisadas 303 amostras, como é possível verificar pela figura D-6, de dados em que 164 têm target 0 (ausência de problemas), 55 com target 1, 31 com target 2, 36 com target 3 e em que 13 com target 4, sendo que para as amostras com target 1,2,3,4 já existem alguns indícios de problemas com ordem crescente de intensidade.

Tabela D-2 - Matriz de confusão do Teste 1

	Classe Prevista				
Classe Real	40	7	2	0	0
	12	4	2	5	0
	6	5	2	1	0
	3	2	2	3	0
	1	0	2	1	0

Na matriz de confusão (ver tabela D-2) existem no total 100 dados analisados que representam 33% dos dados reservados para os testes do modelo. (303 amostras * 0.33% dados reservados para o teste do modelo).

Tabela D-3 - Métricas do Teste 1

Métricas	Valor
Accuracy	48,9%
F1 micro	49,9%
F1 macro	34,5%
Precision micro	48,9%
Precision macro	34,1%
Recall micro	51%
Recall macro	35,8%

Métricas de avaliação do modelo tendo em conta a tabela D-3:

Recall - Esta métrica tem a capacidade de calcular a percentagem de exemplos positivos no conjunto de amostras positivas existentes no modelo. Por outras palavras, esta métrica avalia as amostras pertencentes a classe 0,1,2,3,4 e que realmente pertenciam a estas classes. No caso do cálculo da métrica *Recall-Micro*, para além do que foi mencionado anteriormente, este tem ainda a particularidade de agregar as contribuições de todas as classes e em seguida calcular a média. Posto isto, e em termos práticos, a percentagem determinada foi de 49% após a aplicação da fórmula:

$$\text{Recall} = \frac{tp}{tp + fn} \quad \frac{49}{49 + 51} = 49\%$$

Equação D-1 - Formula de cálculo *Recall* Micro no Teste 1

Podemos então verificar (equação D-1) que o modelo é capaz de prever corretamente as amostras classificadas como verdadeiros positivos com uma taxa de sucesso de 49%.

Na métrica de avaliação *recall macro* (equação D-2) tem uma ligeira diferença, diferença essa que passa por executar exatamente os mesmos cálculos mas para cada classe, e em seguida obter a média, ou seja, trata assim todas as classes igualmente. Para esta métrica foi aplicado a fórmula (t)/(tp+fn) para cada classe em particular:

$$\text{Recall} = \frac{tp}{tp + fn} \quad \frac{40}{40 + 9} = 0.82 \quad \frac{4}{4 + 19} = 0.17 \quad \frac{2}{2 + 12} = 0.14 \quad \frac{3}{3 + 7} = 0.3 \quad \frac{0}{0 + 4} = 0 \quad \frac{1.43}{5} = 30\%$$

Equação D-2 - Formula de cálculo *Recall* Macro no Teste 1

Precision - Esta métrica foca-se essencialmente em identificar as classes classificadas como 0 mas que na realidade são 1,2,3 ou 4, assim como identificar as classes que foram classificadas como pertencentes à classe 1 mas que o modelo identificou como 0,2,3 ou 4 e assim sucessivamente. Para este caso foi calculado *precision-micro* que visa agregar as contribuições de todas as classes, sendo depois calculado a média e *precision-macro* no qual os cálculos são feitos para cada classe, tratando cada de igual forma. Relativamente ao cálculo da métrica *precision-micro* (ver equação D-3), é depois aplicada a fórmula:

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \frac{49}{49 + 51} = 49\%$$

Equação D-3 - Formula de cálculo *Precision* Micro no Teste 1

No cálculo da métrica *precision-macro* (ver equação D-4) teremos então a aplicação da mesma fórmula mas tratando cada classe de igual forma:

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \frac{40}{40 + 22} = 0.65 \qquad \frac{4}{4 + 14} = 0.22 \qquad \frac{2}{2 + 8} = 0.2 \qquad \frac{3}{3 + 7} = 0.3 \qquad \frac{0}{0 + 0} = 0 \qquad \frac{1.37}{5} = 30\%$$

Equação D-4 - Formula de cálculo Precision Macro no Teste 1

F1 - Esta métrica combina as métricas *precision* e *recall* com o objetivo de identificar a qualidade geral do modelo gerado. Esta métrica tem boas respostas no que toca a classes desproporcionais. Pode também ser interpretada como uma média ponderada entre a *precision* e *recall*. A fórmula para calcular esta métrica é: *F1* = 2 * (*precision* * *recall*) / (*precision* + *recall*). À semelhança das métricas anteriores, é possível calcular a métrica *f1-micro* que visa agregar as contribuições de todas as classes e a métrica *f1-macro* na qual os cálculos são feitos para cada classe, tratando cada de igual forma. Para calcular a métrica *f1-micro* (ver equação D-5), usamos a seguinte fórmula:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad 2 * \frac{0.49 * 0.49}{0.49 + 0.49} = 49\%$$

Equação D-5 - Formula de cálculo *F1* Micro no Teste 1

Métrica *f1-macro* (ver equação D-6):

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad 2 * \frac{0.3 * 0.3}{0.3 + 0.3} = 30\%$$

Equação D-6 - Formula de cálculo *F1* Macro no Teste 1

Accuracy – Sinteticamente, esta é a métrica mais simples de calcular visto que permite analisar o número correto de amostras que o sistema foi capaz de calcular corretamente, ou seja, os verdadeiros positivos a dividir pelo número total de amostras existentes.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \qquad \frac{49}{100} = 49\%$$

Equação D-7 - Formula de cálculo *Accuracy* no Teste 1

O sistema foi capaz de calcular corretamente (ver equação D-7) as amostras pertencentes a cada classe com uma percentagem de 49%.

Torna-se então possível, e necessário tecer as seguintes conclusões:

- Classes não balanceadas;
- Quanto menos amostras existem de uma determinada classe menos capacidade o modelo tem de distinguir os verdadeiros positivos;
- Quanto mais amostras de uma classe existirem maior probabilidade existe do modelo conseguir distinguir as amostras considerando o facto de ter mais exemplos para analisar.

Opção 2:

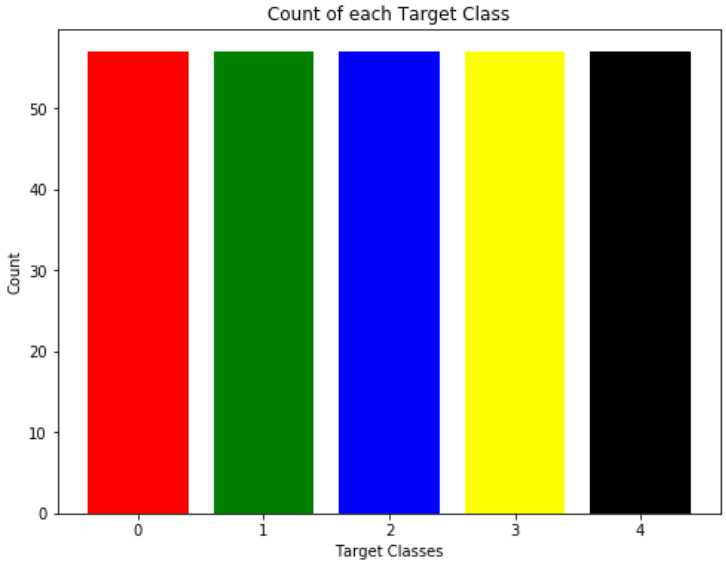

Figura D-7 - Divisão de amostras no teste 2

Nesta opção foi tomada a decisão de balancear os dados existentes como é possível ver na figura D-7. A estratégia usada passou por equilibrar as amostras, ou seja, adicionar novas amostras às classes 1,2,3,4 e remover algumas amostras à classe 0. Na classe 0 foi aplicada a técnica de RandomUnderSampler, que significa remover amostras aleatoriamente, e também foi aplicada a técnica RandomOverSampler às amostras 1,2,3 e 4, que passou por duplicar amostras de uma forma aleatória. É importante realçar que as duas técnicas aplicadas anteriormente apenas foram executadas nas amostras destinadas ao treino sendo que as amostras destinadas aos testes do modelo ficaram inalteradas. Basicamente, num universo de 303 amostras existentes, cerca de 33% são destinadas ao teste do modelo, sendo que os restantes 77% são destinadas ao treino do modelo e posteriormente aplicado as técnicas de RandomUnderSampler e RandomOverSampler.

Tabela D-5 - Matriz de confusão do Teste 2

Classe Real	Classe Prevista				
	34	9	4	2	1
	9	6	3	4	1
	4	3	3	3	1
	2	3	3	2	1
	0	1	1	1	1

Tabela D-4 - Métricas do Teste 2

Métricas	Valor
Accuracy	45,5%
F1 micro	45,6%
F1 macro	30,3%
Precision micro	45,5%
Precision macro	31,3%
Recall micro	45,6%
Recall macro	30,4%

Métricas de avaliação do modelo tendo em conta as tabelas D-5 e D-4:

Recall - Esta métrica tem a capacidade de calcular a percentagem de exemplos positivos no conjunto de amostras positivas existentes no modelo. Por outras palavras, esta métrica avalia as amostras pertencentes a classe 0,1,2,3 e 4 e que realmente pertenciam a estas classes. No caso do cálculo da métrica *Recall-Micro*, para além do que foi mencionado anteriormente, este tem ainda a particularidade que visa agregar as contribuições de todas as classes e em seguida calcular a média. Posto isto, e em termos práticos, a percentagem determinada foi de 0.4599% após a aplicação da fórmula:

$$\text{Recall} = \frac{tp}{tp + fn} \quad \frac{45.6}{45.6 + 54.5} = 50\%$$

Equação D-8 - Formula de cálculo *Recall* Micro no Teste 2

Podemos então verificar (ver equação D-8) que o modelo é capaz de prever corretamente as amostras classificadas como verdadeiros positivos com uma taxa de sucesso de 50%.

Na métrica de avaliação *recall macro* (ver equação D-9) tem uma ligeira diferença, diferença essa que passa por executar exatamente os mesmos cálculos mas para cada classe e em seguida obter a média, ou seja, trata assim todas as classes igualmente. Para esta métrica foi aplicado a fórmula para cada classe em particular:

$$\text{Recall} = \frac{tp}{tp + fn} \quad \frac{34.1}{34.1 + 15} = 0.7 \quad \frac{5.6}{5.6 + 17.4} = 0.2 \quad \frac{3.3}{3.3 + 10.7} = 0.2 \quad \frac{2.1}{2.1 + 7.9} = 0.2 \quad \frac{0.5}{0.5 + 3.5} = 0.1 \quad \frac{1.4}{5} = 30\%$$

Equação D-9 - Formula de cálculo *Recall* Macro no Teste 2

Precision - Esta métrica foca-se essencialmente em identificar as classes classificadas como 0 mas que na realidade são 1,2,3 ou 4, assim como identificar as classes que foram classificadas como pertencentes a classe 1 mas que o modelo identificou como sendo 0,2,3 ou 4 e assim

sucessivamente. Para este caso foi calculado *precision-micro* que visa agregar as contribuições de todas as classes, sendo depois calculada a média e *precision-macro* no qual os cálculos são feitos para cada classe, tratando cada de igual forma. Relativamente ao cálculo da métrica *precision-micro* (ver equação D-10), e depois de aplicado a fórmula:

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \frac{45.6}{45.6 + 54.5} = 50\%$$

Equação D-10 - Formula de cálculo *Precision* Micro no Teste 2

No cálculo da métrica *precision-macro* (ver equação D-11) teremos então a aplicação da mesma fórmula mas tratando cada classe de igual forma.

$$\text{Precision} = \frac{tp}{tp + fp} \quad \frac{34.1}{34.1 + 15.6} = 0.7 \quad \frac{5.6}{5.6 + 16.3} = 0.3 \quad \frac{3.3}{3.3 + 9.8} = 0.3 \quad \frac{2.1}{2.1 + 10.3} = 0.2 \quad \frac{0.5}{0.5 + 2.5} = 0.2 \quad \frac{1.7}{5} = 30\%$$

Equação D-11 - Formula de cálculo *Precision* Macro no Teste 2

F1 - Esta métrica combina as métricas *precision* e *recall* para com o objetivo de identificar a qualidade geral do modelo gerado. Esta métrica tem boas respostas no que toca a classes desproporcionais. Pode também ser interpretada como uma média ponderada entre a *precision* e *recall*. A fórmula para calcular esta métrica é: *F1* = 2 * (*precision* * *recall*) / (*precision* + *recall*). À semelhança das métricas anteriores, é possível calcular a métrica *f1-micro* que visa agregar as contribuições de todas as classes e a métrica *f1-macro* no qual os cálculos são feitos para cada classe, tratando cada de igual forma. Para calcular a métrica *f1-micro* (ver equação D-12):

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad 2 * \frac{0.5 * 0.5}{0.5 + 0.5} = 50\%$$

Equação D-12 - Formula de cálculo *F1* Micro no Teste 2

e a métrica f1-macro (ver equação D-13).

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad 2 * \frac{0.3 * 0.3}{0.3 + 0.3} = 30\%$$

Equação D-13 - Formula de cálculo F1 Macro no Teste 2

Accuracy – Concisamente, esta é a métrica mais simples de calcular, visto que permite analisar o número correto de amostras que o sistema foi capaz de calcular corretamente, ou seja, os verdadeiros positivos a dividir pelo número total de amostras existentes.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \qquad \frac{45.6}{100} \; 50\,\%$$

Equação D-14 - Formula de cálculo Accuracy no Teste 2

O sistema foi capaz de calcular corretamente as amostras pertencentes a cada classe com uma percentagem de 50% (ver equação D-14).

Algumas conclusões possíveis de serem traçadas:

- Classes balanceadas;
- Técnicas de RandomUnderSampler e RandomOverSampler aplicadas apenas ao conjunto destinado ao treino do modelo;
- Apesar das classes terem sidos balanceadas, não existem amostras com valores diferentes suficientes para o sistema "treinar", ou seja, a existência de mais amostras diferentes seria uma mais-valia;
- Exceto a classe 0, o sistema sente dificuldades em distinguir amostras das classes 1,2,3 e 4 devido ao facto de não existirem amostras com valores diferentes para adicionar diversidade às amostras existentes;
- O sistema "confunde" com alguma regularidade uma determinada classe com classes vizinhas. (por exemplo a classe 0 com a classe 1 e a classe 1 com a classe 2 e assim sucessivamente);
- A classe com mais amostras diferentes (classe 0) apresenta melhores resultados pelo facto de o sistema ter identificado a classe 0 com muita regularidade;

Opção 3 A:

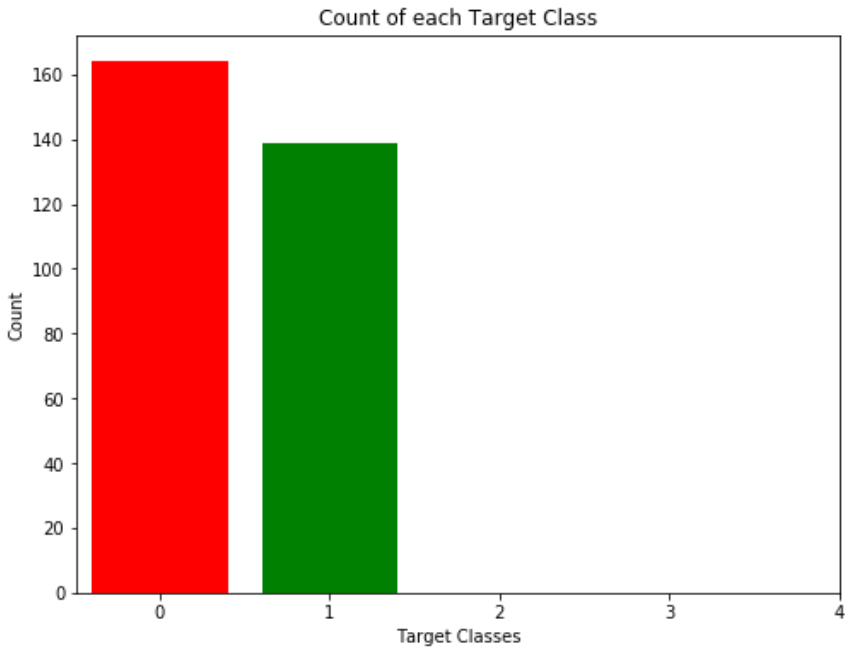

Figura D-8 - Divisão de amostras no teste 3A

Nesta opção, a estratégia passou por manter a classe com mais amostras no qual representa ausência de qualquer valor (classe 0) e somar as restantes classes que representam alguns sintomas de problemas cardíacos como é possível verificar pela figura D-8.

Tabela D-7 - Matriz de confusão do Teste 3A

Classe Real	Classe Prevista	
	39	10
	18	33

Tabela D-6 - Métricas do Teste 3A

Métricas	Valor
Accuracy	71,9%
F1 micro	71,9%
F1 macro	71,8%
Precision micro	71,9%
Precision macro	72,5%
Recall micro	71,9%
Recall macro	72,1%

Tendo como base a figura D-8 e as tabelas D-7 e D-6, num universo de 100 amostras (33% destinado ao treino dos dados de 303 amostras globais), o sistema no que toca à métrica de avaliação **precision** foi capaz de, com uma taxa de 68%, de filtrar as classes classificadas como pertencentes à classe 0 e que efetivamente pertenciam à classe 0, assim como classificou também corretamente com uma percentagem de 77% as classes pertencentes à classe 1 a que realmente pertenciam. Esta métrica permite também identificar dentro de todas as classificações de classes positivas que o modelo fez, quantas estão corretas. Relativamente à métrica **recall**, ou seja, dentro de todas as situações de classes positivas como valor esperado quantas efetivamente estão corretas e o modelo gerado previu uma probabilidade de 80% para a classe 0 e 65% para a classe 1. Já a métrica de avaliação **f1,** que permite combinar a métrica recall e precision de maneira a trazer um número único que indique a qualidade geral do modelo, foi capaz de identificar para o modelo gerado uma percentagem de 74% para a classe 0 e de 70% para a classe 1. Relativamente à métrica de avaliação **accuracy** que permite analisar o número correto de amostras que o sistema foi capaz de calcular corretamente, este modelo conseguiu fazê-lo com uma percentagem de 72%.

Identificou corretamente 39 em 49 como pertencentes à classe 0, sendo que as restantes 9 identificou incorretamente como não pertencentes à classe 0. O sistema identificou também corretamente 33 amostras como pertencentes à classe 1(1+2+3+4 classes), sendo que identificou incorretamente 18 como pertencentes à classe 0 mas que na realidade pertenciam à classe 1.

Algumas conclusões possíveis de serem delineadas:

- Junção da classe 1,2,3 e 4 permitiu balancear as classes;
- Classe 1 identifica amostras de pessoas com possíveis sintomas cardíacos;
- O sistema consegue distinguir com uma taxa acima dos 70% a classe 0 da classe 1;
- A existência de mais amostras da classe 0 do que da classe 1 permite ao modelo mais facilmente distinguir as amostras que realmente são da classe 0 do que da classe 1.

Opção 3 B:

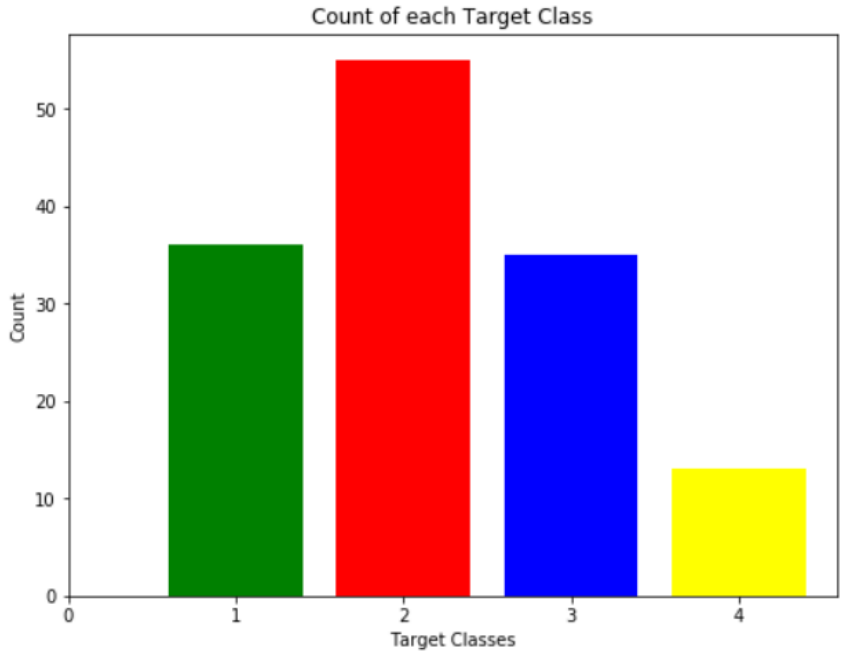

Figura D-9 - Divisão de amostras no teste 3B

Ainda nesta opção de teste foi tomada a decisão de "partir" as amostras, removendo assim a classe com mais amostras. Para esta análise vão ser estudadas as amostras 1,2,3 e 4, amostras estas que representam alguma presença de problemas cardíacos, como é possível verificar pela figura D-9. Este teste tem como grande objetivo analisar o comportamento do modelo e as diferenças entre as classes 1,2,3 e 4. Num universo de 139 amostras, 33% (46 amostras) são destinadas ao treino do algoritmo, como é possível analisar na figura D-9.

Tabela D-9 - Matriz de confusão do Teste 3B

Classe Real	Classe Prevista			
	8	0	7	1
	1	5	4	1
	6	3	5	0
	0	4	0	1

Tabela D-8 - Métricas do Teste 3B

Métricas	Valor
Accuracy	41,3%
F1 micro	41,3%
F1 macro	38,3%
Precision micro	41,3%
Precision macro	39,8%
Recall micro	41,3%
Recall macro	37,7%

Métricas de avaliação do modelo tendo em conta a tabela D-9 e D-8:

Recall - Esta métrica tem a capacidade de calcular a percentagem de exemplos positivos no conjunto de amostras positivas existentes no modelo. Dito de outra forma, esta métrica avalia as amostras pertencentes às classes 1,2,3 e 4 e que realmente pertenciam a estas classes. No caso do cálculo da métrica *Recall-Micro*, para além do que foi mencionado anteriormente, importa referir que esta tem a particularidade de agregar as contribuições de todas as classes e em seguida calcular a média. Posto isto, e em termos práticos, a percentagem determinada foi de 40% após a aplicação da fórmula:

$$\text{Recall} = \frac{tp}{tp + fn} \quad \frac{19}{19 + 27} = 40\%$$

Equação D-15 - Formula de cálculo *Recall* Micro no Teste 3B

Podemos então verificar (ver equação D-15) que o modelo é capaz de prever corretamente as amostras classificadas como verdadeiros positivos com uma taxa de sucesso de 40%.

Na métrica de avaliação *recall macro* (ver equação D-16) tem uma ligeira diferença, diferença essa que passa por executar exatamente os mesmos cálculos mas para cada classe, e em seguida obter a média, ou seja, trata assim todas as classes igualmente. Para esta métrica foi aplicado a fórmula (tp)/(tp+fn) para cada classe em particular:

$$\text{Recall} = \frac{tp}{tp + fn} \quad \frac{8}{8+8} = 0.5 \quad \frac{5}{5+6} = 0.5 \quad \frac{5}{5+9} = 0.4 \quad \frac{1}{1+4} = 0.2 \quad \frac{1.6}{4} = 40\%$$

Equação D-16 - Formula de cálculo *Recall* Macro no Teste 3B

Precision - Esta métrica foca-se essencialmente em identificar as classes classificadas como 1 mas que, na realidade, são 2,3 ou 4, assim como em identificar as classes que foram classificadas como

pertencentes à classe 2 mas que o modelo identificou como sendo 1,3 ou 4, e assim sucessivamente. Para este caso foi calculado um *precision-micro* que visa agregar as contribuições de todas as classes, sendo depois calculada a média e *precision-macro* na qual os cálculos são feitos para cada classe, tratando cada um de igual forma. Relativamente ao cálculo da métrica *precision-micro* (ver equação D-17), e depois de aplicado a fórmula:

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \frac{19}{19 + 27} = 40\%$$

Equação D-17 - Formula de cálculo *Precision* Micro no Teste 3B

No cálculo da métrica *precision-macro* (ver equação D-18), teremos então a aplicação da mesma fórmula mas tratando cada classe de igual forma.

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \frac{8}{8 + 7} = 0.5 \quad \frac{5}{5 + 7} = 0.4 \quad \frac{5}{5 + 11} = 0.3 \quad \frac{1}{1 + 2} = 0.3 \quad \frac{1.5}{4} = 40\%$$

Equação D-18 - Formula de cálculo *Precision* Macro no Teste 3B

F1 = Esta métrica combina as métricas *precision* e *recall* com o objetivo de identificar a qualidade geral do modelo gerado. Esta métrica tem boas respostas no que toca a classes desproporcionais. Pode também ser interpretada como uma média ponderada entre a *precision* e *recall*. A fórmula para calcular esta métrica é: F1 = 2 * (*precision* * *recall*) / (*precision* + *recall*). À semelhança das métricas anteriores, é possível calcular a métrica *f1-micro* que visa agregar as contribuições de todas as classes e a métrica *f1-macro* na qual os cálculos são feitos para cada classe, tratando cada de igual forma. Para calcular a métrica *f1-micro* (ver equação D-19):

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad 2 * \frac{0.4 * 0.4}{0.4 + 0.4} = 40\%$$

Equação D-19 - Formula de cálculo F1 Micro no Teste 3B

e a métrica *f1-macro* (ver equação D-20).

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad 2 * \frac{0.4 * 0.4}{0.4 + 0.4} = 40\%$$

Equação D-20 - Formula de cálculo F1 Macro no Teste 3B

Accuracy – Abreviadamente, esta é a métrica mais simples de calcular, visto que permite analisar o número correto de amostras que o sistema foi capaz de calcular corretamente, ou seja, os verdadeiros positivos a dividir pelo número total de amostras existentes.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \qquad \frac{19}{46} = 40\%$$

Equação D-21 - Formula de cálculo Accuracy no Teste 3B

O sistema foi capaz de calcular corretamente as amostras pertencentes a cada classe com uma percentagem de 41% (ver equação D-21).

Algumas conclusões possíveis de avançar:

- Classes relativamente balanceadas;
- Poucas amostras para retirar valor;
- As classes são facilmente confundíveis entre si.

Algumas Conclusões – Caso de Teste

- Classes não balanceadas não permitem obter grandes resultados, uma vez que não existem amostras suficientes de cada classe para construir um bom modelo;
- Balancear classes pode levar ao erro, uma vez que podem existir muitas amostras duplicadas.
- De uma forma geral, o modelo consegue distinguir mais facilmente a classe 0 das restantes, uma vez que existem mais amostras para treinar o modelo.
- As amostras 1,2,3 e 4 revelam fragilidade e são mais facilmente confundíveis entre elas próprias, eventualmente porque os valores associados a cada amostra são muito semelhantes.