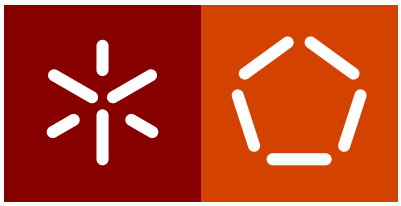
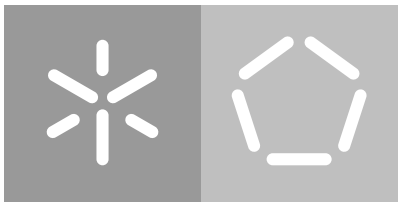


Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Miguel Amorim Noivo

Análise de Padrões Biométricos para Otimização do Desempenho Académico

Outubro 2019



Universidade do Minho
Escola de Engenharia
Departamento de Informática

João Miguel Amorim Noivo

Análise de Padrões Biométricos para Otimização do Desempenho Académico

Tese de Mestrado
Mestrado Integrado em Engenharia Informática

Trabalho realizado sob orientação
Paulo Jorge Novais

Outubro 2019

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

AGRADECIMENTOS

Esta dissertação é o ponto final da minha jornada académica realizada nestes seis anos. Desta forma, há muitas pessoas a quem gostaria de agradecer que disponibilizaram o seu tempo, conhecimento, paciência e apoio, e sem elas eu não teria conseguido terminar esta jornada.

Gostaria de agradecer ao meu orientador, Professor Paulo Jorge Freitas de Oliveira Novais, que demonstrou disponibilidade para ajudar, transmitiu bons conselhos, confiança para realizar este trabalho, e especialmente motivação para o terminar.

Um agradecimento especial ao Professor Filipe Manuel Gonçalves, pela atenção e paciência prestadas, a clarificação imediata das dúvidas, a oportunidade de trabalhar neste grande projeto e o apoio incalsável demonstrado ao longo de todo este processo.

Agradecer também a todos os meus amigos que me apoiaram durante esta fase complicada e sempre se mostraram presentes quando eu mais precisava.

Por último, à minha família que sempre me apoiou em todos os momentos, demonstrando carinho e motivação ao longo destes anos. Sem eles nada disto seria possível, obrigado.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

ABSTRACT

Nowadays the demand for better results both academically and professionally has been increasing, causing people to have to deal with this tension daily. The fact that they leave the zone of tranquility does not mean that it is harmful, but when they are exposed to this type of situations for an extended time usually leads to health degradation. In this way, it can compromise the performance in the accomplishment of the tasks and influence in a negative way the desired productivity.

Although there are already some techniques to help control stress, it is difficult to have an exact idea of when it appears and its influence on the final results. With the study focus on the academic environment, the goal is to collect the most information from the students during the exams. After the collection process, it is necessary to treat the data to later apply machine learning, using the most appropriate algorithm, to establish a pattern that allows to perceive the impact of the stress in the students.

However, linking the different stages of the process can be an adversity to the progress of the study of stress, compromising the quality of the results and the time to achieve them. In order to solve this problem, in this dissertation project was developed a platform, Learning Management System (LMS), where the students can carry out the exams and in parallel the respective biometric data is collected to be analyzed afterwards. After the process is completed, the platform presents the results obtained through simple and intuitive interfaces allowing the user to visualize and draw conclusions.

Keywords - Learning Management System, Stress, Machine Learning

RESUMO

Hoje em dia a exigência de melhores resultados quer a nível académico como profissional tem vindo a aumentar, levando as pessoas a ter que lidar com essa tensão diariamente. O facto de saírem da zona de tranquilidade, não significa que seja prejudicial, mas quando ficam expostas a este tipo de situações por tempo prolongado normalmente leva a um agravamento a nível do estado de saúde. Deste modo, poderá comprometer o desempenho na realização das tarefas e influenciar de forma negativa o rendimento desejado.

Apesar de já existirem algumas técnicas para ajudar a controlar o stress, não se consegue ter uma ideia exata do momento em que este aparece e a sua influência nos resultados finais. Tendo como foco de estudo o meio académico, o objetivo é recolher o maior número de informações dos estudantes durante a realização dos exames. Após o processo de recolha é necessário tratar os dados para posteriormente aplicar algoritmos de *machine learning*, utilizando o algoritmo mais adequado, de forma a estabelecer um padrão que permita perceber o impacto do stress nos estudantes.

Contudo, a ligação das diferentes fases do processo pode ser uma adversidade para o progresso do estudo do stress, comprometendo a qualidade dos resultados e o tempo para a sua realização. De forma a resolver este problema, neste projeto de dissertação propõe-se o desenvolvimento de uma plataforma, Sistema de Gestão da Aprendizagem (SGA), onde os alunos possam realizar as provas e em simultâneo sejam recolhidos os respetivos dados biométricos a analisar posteriormente. Após o processo estar concluído, a plataforma apresenta os resultados obtidos através de interfaces simples e intuitivas permitindo ao utilizador visualizar e tirar conclusões.

Palavras-chave **-** Sistema de Gestão da Aprendizagem, Stress, Machine Learning

CONTEÚDO

NOTAÇÃO E TERMINOLOGIA

NOTAÇÃO

Ao longo do documento são utilizadas siglas representativas dos nomes dos modelos. Para melhor entendê-los, este capítulo foi criado para que o leitor entenda a sua interpretação.

ACRÓNIMOS

AAC - Aprendizagem Assistida em Computador
AD - Árvore de Decisão
DB - *Database*
FA - Floresta de Aleatória
HTTP - *Hypertext Transfer Protocol*
IA - Inteligência Artificial
IAC - Instrução Assistida em Computador
IBC - Instrução Baseada em Computador
IDE - *Integrated Development Environment*
LMS - *Learning Management System*
ML - *Machine Learning*
Moodle - *Modular Object-Oriented Dynamic Learning Environment*
PQ – Programação Quadrática
RAM - *Random Access Memory*
REPL - *Read-Eval-Print Loop*
RL - Regressão Logística
ROSE - *Random Over-Sampling Examples*
SAI - Sistema Aprendizagem Integrado
SGA - Sistemas de Gestão da Aprendizagem
SMOTE - *Synthetic Minority Oversampling Technique*
SVM - *Support Vector Machine*

LISTA DE FIGURAS

LISTA DE TABELAS

1

INTRODUÇÃO

Este documento insere-se no âmbito da dissertação de Mestrado de Engenharia Informática na Universidade do Minho. Tem como título *Análise de Padrões Biométricos para Optimização do Desempenho Académico*.

O trabalho abrange diferentes áreas da engenharia informática como *machine learning*, Sistema de Gestão da Aprendizagem (SGA) e Inteligência Artificial (IA).

Neste capítulo, a motivação e suporte do trabalho serão apresentados, seguidos de uma breve explicação do tema e objetivos a alcançar, o método de pesquisa usado e a estrutura do documento.

1.1 MOTIVAÇÃO

Com o crescimento exponencial da tecnologia, muitas são as vantagens que o ser humano usufrui no seu dia a dia, tanto no trabalho como na vida pessoal. Estas novidades tecnológicas ajudam a abrir novas portas e a permitir que projetos inicialmente ambiciosos comecem a tornar-se possíveis.

Uma plataforma *e-learning* (ou Sistema de Gestão da Aprendizagem - SGA) é uma aplicação software para administração, documentação, monitorização, reportar e disponibilizar cursos educacionais ou programas de treino ou desenvolvimento (Ellis, 2009).

A primeira aparição dos SGA surgiu no âmbito da educação no fim dos anos 90's (Davis et al., 2009), ocupando o maior segmento no que diz respeito ao mercado de sistemas de aprendizagem. Se no início havia algum receio em relação ao uso destas ferramentas, isso facilmente foi ultrapassado pelas inúmeras vantagens que demonstraram ter.

As principais razões que levam a preferir um SGA deve-se à sua interoperabilidade, acessibilidade, reutilização, durabilidade, manutenção e adaptabilidade (Long, 2004), estas são as caraterísticas base de um SGA.

No mundo académico, alguns docentes usam o acesso à tecnologia, nomeadamente os computadores, para a realização de exames, de forma a pouparem trabalho com as correções e permitir aos alunos saberem o resultado, logo após a sua finalização.

Podemos então, tirar proveito da tecnologia para recolher os dados biométricos, sobre as nossas ações e ficarmos a perceber melhor como é que interagimos, isto é, em casos sob pressão por se tratar de momentos de avaliação.

1.2 SGA

A história do uso de computadores na educação é composta com termos genéricos como instrução baseada em computador (IBC), instrução assistida em computador (IAC), e aprendizagem assistida em computador (AAC), geralmente descritos por programas de exercícios e conteúdos práticos da matéria, em que o utilizador recebe opiniões em relação ao exercício, tutoriais mais elaborados, e dicas mais individualizadas, respetivamente. Inicialmente, SGA era conhecido como sistema de aprendizagem integrado (SAI) que oferecia funcionalidades adicionais para além do conteúdo base, tais como gestão do sistema e instruções mais personalizadas de acompanhamento, e integração de todo o sistema. O termo SAI foi originalmente criado por Jostens Learning, e SGA era usado para descrever a parte de gestão do sistema aprendizagem *PLATO K-12*, livre de conteúdo e separado do material didático (Chaubey and Bhattacharya, 2015).

Nos últimos anos tem-se notado um crescimento em relação ao uso de sistemas de gestão da aprendizagem (SGA) nos institutos educacionais, como universidades, providenciando vários níveis de suporte aos estudantes e professores durante a sua implementação. Isto, pelo menos em teoria, oferece mais condições favoráveis para ambientes de trabalho produtivos, tendo como base críticas sociais construtivas e conteúdos disponíveis a todos os estudantes, quer os que conseguem estar presencialmente ou os que estudam à distância (Weaver et al., 2008).

As vantagens do *e-learning* em relação ao sistema tradicional de ensino são inúmeras, visto que o *e-learning* torna a educação independente do tempo e do local. Mais importante, abre um conjunto de possibilidades para implementar inovações pedagógicas num ambiente onde os estudantes sejam ativos, independentes, críticos e participativos. A acrescentar, o *e-learning* acompanha os estudantes na gestão dos cursos online, permitindo-lhes criar, adicionar, modificar, customizar, e reutilizar conteúdos digitais e matérias de estudo (Kakasevski et al., 2008).

Apesar da educação tradicional ainda abranger maior parte dos sítios por todo o mundo, esta diferente e inovadora abordagem ainda em crescimento, poderá ser uma das opções viáveis num futuro próximo para a educação, sem ser preciso tantos meios e recursos (Rodrigues et al., 2005). As vantagens são diversas e novas utilidades podem ser acrescentadas a este tipo de sistemas a qualquer momento, como as que iremos abordar neste projeto.

1.3 MACHINE LEARNING

De uma forma breve, *machine learning* é o conjunto de métodos que conseguem detetar padrões nos dados de uma forma automática, e depois usar esses padrões descobertos para prever dados futuros, ou realizar diferentes tomadas de decisão sobre a incerteza (por exemplo, planear como fazer a recolha dos dados) (Robert, 2014).

É um campo da Inteligência Artificial que se preocupa com os algoritmos que permitem aos sistemas de IA aprender. Apesar da aprendizagem por computação ser tão antiga como IA, a sua importância tem aumentado nos sistemas de IA, especialmente nos agentes autónomos, que estão a operar em domínios progressivamente mais complexos e em constante mudança. Grande parte das técnicas ML são de aprendizagem supervisionada no qual o sistema é instruído através de dados de treino sem supervisão, ou sistemas autónomos. Aprendizagem de reforço, juntamente com recompensas artificiais, é frequentemente utilizado para aprender novas tarefas (Frankish and Ramsey, 2014).

Os algoritmos de ML foram desde o início projetados e usados para analisar bases de dados médicas. Hoje em dia, as técnicas de ML são ferramentas indispensáveis para realizar análises inteligentes de dados. Especialmente nos últimos anos, a revolução digital providenciou meios relativamente baratos e disponíveis para recolher e armazenar os dados. Hospitais modernos estão bem equipados a nível da monitorização e de outros dispositivos de recolha de dados, a partir dos quais os dados são guardados e partilhados em grandes sistemas de informação. A tecnologia dos algoritmos de ML é atualmente adequada para analisar dados médicos, em particular com o desenvolvimento de trabalhos no âmbito do diagnóstico médico em pequenos problemas especializados (Kononenko, 2001).

Atualmente , o campo dos algoritmos de ML organiza-se à volta de três focos de pesquisa:

- Estudos Orientados a Tarefas - o desenvolvimento e análises de sistemas de aprendizagem para melhorar o desempenho num conjunto predeterminado de tarefas;
- Simulação Cognitiva - a investigação e simulação de computador dos processos de aprendizagem dos humanos;
- Análises Teóricas - a exploração teórica do espaço de possíveis métodos de aprendizagem e algoritmos independentes do domínio da aplicação.

Embora muitas pesquisas foquem-se principalmente para um desses objetivos, o progresso de um objetivo muitas vezes influência o progresso de outro objetivo. Por exemplo, a fim de investigar o espaço de possíveis métodos de aprendizagem, um ponto de partida razoável, seria considerar o único exemplo conhecido de comportamento de aprendizagem robusto, conhecido como humanos (e talvez outro sistema biológico). De forma idêntica, as investigações de psicologia de aprendizagem humana, através de análises teóricas, podem sugerir diferentes possibilidades de modelos de aprendizagem. A necessidade de adquirir

de uma determinada forma conhecimento num respetivo estudo orientado à tarefa pode por si só dar origem a novas análises teóricas. Esta forma de desafio mútuo e objetivos similares é a reflexão de um campo da inteligência artificial, onde a pesquisa de sistemas desenvolvidos, a simulação cognitiva, e estudos teóricos ajudam a resolver problemas e desenvolver novas ideias (Michalski et al., 2013).

Este campo também aborda problemas da mesma forma que um médico descobre o diagnóstico ao colocar questões, aprendendo as regras dos dados. Começa com observações a nível do paciente, o algoritmo examina um vasto número de variáveis, procurando por combinações que possam prever os resultados de forma assertiva. De certo modo, esse processo é semelhante ao dos modelos tradicionais de regressão: há um resultado, variáveis e uma função de estatística que liga estes dois. Mas onde os algoritmos de ML aparecem é a lidar com números enormes de precisões e combiná-los de formas não-lineares e altamente interativas. Esta capacidade permite usar novos tipos de dados, cujo o volume ou complexidade anteriormente não permitiria retirar possíveis relações entre eles (Obermeyer and Emanuel, 2016).

1.4 STRESS

A primeira e mais genérica definição de *stress* foi proposta por Hans Selye: *"Stress é uma resposta não específica do corpo a qualquer tipo de exigência"*. Selye enfatizou constantemente que o facto de usar a palavra *stress* como uma resposta não específica a qualquer tipo de exigência era a definição mais apropriada.

Nas ciências comportamentais, o *stress* é considerado como: *"perceção de ameaça, com consequente desconforto de ansiedade, tensão emocional, e dificuldade no ajustamento"* (Pereira, 2002).

Por fim, Richard Lazarus, famoso pelo seu trabalho na psicologia cognitiva e foco nas emoções, alerta a dificuldade em encontrar uma definição precisa que satisfaça todas as áreas, tal como referiu: *"Apesar da confusão existente sobre a definição precisa do termo, stress é reconhecido como um problema central da vida humana*. Cientistas de diferentes áreas têm o conceito de *stress* contextualizado, mas cada área apresenta algo diferente em mente quanto ao seu significado. Para os sociólogos, é um desequilíbrio social, isto é, distúrbios na estrutura social em que as pessoas vivem. Engenheiros associam *stress* a uma força externa que produz tensão nos materiais expostos a ela. Psicólogos lidam com os fatores de pressão que incluem uma ampla gama de condições e de estímulos que são nocivos para o corpo (Fink, 2010).

1.5 INTELIGÊNCIA ARTIFICIAL

É a ciência e a engenharia de fazer máquinas inteligentes, especialmente programas de computadores inteligentes. Está relacionado com tarefas semelhantes de usar computadores para entender a inteligência humana, mas a IA não precisa de se ligar a métodos biologicamente observáveis(McCarthy, 1998).

Surgem diferentes definições em relação à IA, segundo Kurzweil em 1990 defendia que é *"a arte de criar máquinas que exerçam funções que requerem inteligência quando feitas por pessoas"*. Outro ponto de vista é *"o ramo da informática que está interessado pela automatização do comportamento inteligente"* (Luger and Stubblefield, 1993). Já Rich e Knight em 1991, diziam que é *"o estudo de como fazer os computadores resolverem as coisas, que atualmente, as pessoas são melhores"*. Estas definições levam a assentar em quatro pontos fundamentais, que se resumem a sistemas que pensam e agem como humanos e sistemas que pensam e agem racionalmente (Russell et al., 1995).

Esta temática teve início após a 2ª Guerra Mundial, aquando um grupo de pessoas independentes começou a trabalhar nas máquinas inteligentes. O matemático inglês, Alan Turing, foi um dos pioneiros. Ele deu uma palestra sobre o assunto em 1947, por isso é considerado o criador. Ele também decidiu que a IA deveria ser trabalhada através do uso dos computadores, para melhorar a rentabilidade, ao invés de construir máquinas. Pelo fim de 1950, já existiam muitas pesquisas sobre o assunto, e a grande maioria baseava os seus trabalhos nos computadores de programação (McCarthy, 1998).

IA é o ramo da ciência da computação preocupado com o estudo e a criação de um sistema de computador que exiba de alguma forma inteligência: sistemas que aprendem novos conceitos e tarefas. Sistemas que conseguem raciocinar e tirar conclusões importantes sobre o nosso mundo. Sistemas que conseguem entender uma linguagem, ter uma precessão visual e desempenhar outros tipos de funções que requerem inteligência humana (Patterson, 1990).

1.6 TEMA E OBJETIVOS

O tema deste trabalho é Análise de Padrões Biométricos para Optimização do Desempenho Académico. Sendo o foco, melhorar as capacidades dos alunos na realização das tarefas no âmbito universitário, tendo como ponto de partida encontrar uma plataforma que permita a recolha dos dados biométricos dos alunos durante a realização de diferentes provas. Desta forma, teremos que comparar diferentes plataformas existentes e analisar a que cumpre melhor com os requisitos estabelecidos.

As perguntas de investigação que permitiram orientar o percurso do trabalho foram:

- Quais os sistemas de gestão da aprendizagem que são utilizados?

- Quais os padrões biométricos a ter em conta na análise do stress?
- Quais os algoritmos de ML que melhor se adequam?
- Que tipo de informações queremos ter com a análise dos dados?
- Qual a margem de melhoria do desempenho num aluno?

As perguntas de investigação especificadas anteriormente permitem alcançar os seguintes objetivos estabelecidos:

1. Identificar os aspetos importantes num sistema de gestão da aprendizagem;
2. Identificar as caraterísticas relevantes do ser humano para usar na análise dos dados;
3. Identificar situações de stress, de forma a prevenir ou atenuar as consequências;
4. Identificar os algoritmos que têm interesse para tratar os dados;
5. Implementar a recolha dos dados através do uso da plataforma;
6. Disponibilizar os resultados obtidos de um respetivo utilizador ao mesmo.

1.7 METODOLOGIA DE PESQUISA

Em relação à metodologia de pesquisa foi adotada a metodologia de pesquisa-ação (Somekh, 2005). Inicialmente, foi recolhido uma vasta coleção de informações para a construção de um protótipo de uma solução possível. Em seguida, foi feita a pesquisa dos conceitos e projetos relevantes sobre esta temática. A assimilação de conceitos e projetos esteve sempre em constante renovação, à medida que novas ideias e informações iam surgindo. A última parte do trabalho foi o desenvolvimento de um modelo funcional que permitiu a realização do conjunto de metas estabelecidas.
Esta metodologia de pesquisa tem cinco fases de iteração:

1. **Diagnosticar** - Definição do problema e as caraterísticas;
2. **Plano de ação** - Atualização constante do estado da arte e objetivos do trabalho;
3. **Tomada de ação** - Desenvolvimento de um protótipo para atingir os objetivos definidos;
4. **Avaliar** - Análise e correção de protótipos com base nos resultados obtidos;
5. **Aprendizagem específica** - A difusão de conhecimentos e resultados obtidos na comunidade científica.

Para o desenvolvimento de software a metodologia usada foi uma adaptação do SCRUM. Desta forma, todos as etapas anteriormente explicadas também são aplicadas ao desenvolvimento da plataforma. O primeiro passo é diagnosticar o problema e atualizar o estado da arte e respetivos objetivos do trabalho. A seguir, é o desenvolvimento da plataforma para os objetivos propostos. Com estas tarefas concluídas, segue-se a validação do trabalho realizado, cujos resultados estão descritos no trabalho. Através destes resultados, novos problemas irão aparecer o que levará a um novo ciclo de trabalho.

Desenvolvimento SCRUM é uma metodologia simples com o objetivo de solucionar problemas com longos períodos de desenvolvimento, permitindo ao programador concentrar-se num conjunto de metas definidas. Esta metodologia também resolve problemas de incompatibilidade entre os requisitos de negócios de um produto e a implementação real, visto que o cliente também é parte do processo.

1.8 estrutura do documento

Este trabalho foi estruturado em seis capítulos, organizados da seguinte forma:

1. **Introdução** - O primeiro capítulo tem uma breve introdução do ponto de situação, uma introdução aos conceitos chave, uma motivação, tema, objetivos e metodologia de pesquisa e a estruturação do documento;

2. **Estado da Arte** - O segundo capítulo aborda os diferentes tipos de sistemas de gestão da aprendizagem e as respetivas vantagens e desvantagens da sua utilização. O objetivo é comparar os diferentes sistemas, tais como, *Moodle*, *Blackboard Learn*, *LatitudeLearning*,*eFront* e *Dokeos*. No fim do capítulo é feita uma análise dos aspetos chaves dos respetivos sistemas e as suas respetivas limitações.

3. **Arquitetura** - Neste capítulo são abordadas as métricas a serem estudadas na plataforma, a sua arquitetura e os atores que interagem com o sistema.

4. **Machine Learning** - Tem o objetivo de explicar os objetivos dos algoritmos que irão processar os dados recolhidos e fazer a respetiva previsão.

5. **Implementação da Plataforma** - No quinto capítulo, é definida a implementação da plataforma, todas as fases intervenientes no processo, assim como os resultados obtidos ao longo do desenvolvimento.

6. **Conclusão** - O último capítulo resume o trabalho feito até ao momento e as principais conclusões a serem retiradas.

2

ESTADO DA ARTE

Este capítulo pretende descrever os sistemas de aprendizagem de gestão existentes no mercado, e comparar os seus aspetos chaves e as principais lacunas.

Em informática, estas plataformas são consideradas como uma ferramenta de ensino que possibilitam oportunidades às pessoas que têm mais dificuldade em aceder fisicamente aos materiais de ensino ou ajudar escolas/universidades na organização da matéria lecionada. É necessário disponibilizar a aprendizagem às pessoas, permitindo-lhe o fácil acesso à informação, em vez de terem de ser as pessoas a procurar essa aprendizagem (Pingle, 2011). Algumas das mais importantes SGA são:

- *Moodle*
- *Blackboard Learn*
- *LatitudeLearning*
- *Dokeos*
- *eFront*

Nestes próximos subcapítulos, todas as plataformas serão explicadas com mais detalhe.

2.1 MOODLE

Moodle é um acrónimo para "*Modular Object-Oriented Dynamic Learning Environment*", é grátis e é uma SGA de código aberto. Foi desenvolvido pelo Australiano, Martin Dougiamas, para ajudar professores a criar cursos online com conteúdos interativos. Hoje em dia, *Moodle* tem uma vasta lista de programadores dedicados a melhorar a plataforma.

Moodle é um exemplo de uma aplicação "*LAMP*". *LAMP* originalmente ficou para *Linux*, *Apache*, *MySQL* e *Perl*. Com o passar do tempo, vários componentes da sigla mudaram, nomeadamente PHP que se tornou na linguagem predominante para aplicações *LAMP*. Esta sigla refere-se a programas escritos em linguagens *web*, usando uma base de dados *SQL* para armazenar a informação. Com o aumento da popularidade de correr código

aberto em aplicações *web* em ambos *Windows* e *Mac OS X*, dois novos termos apareceram, respetivamente: *WAMP* e *MAMP* (Moore, 2010).

A figura 1 apresenta todos os intervenientes na constituição da arquitetura do *Moodle*.

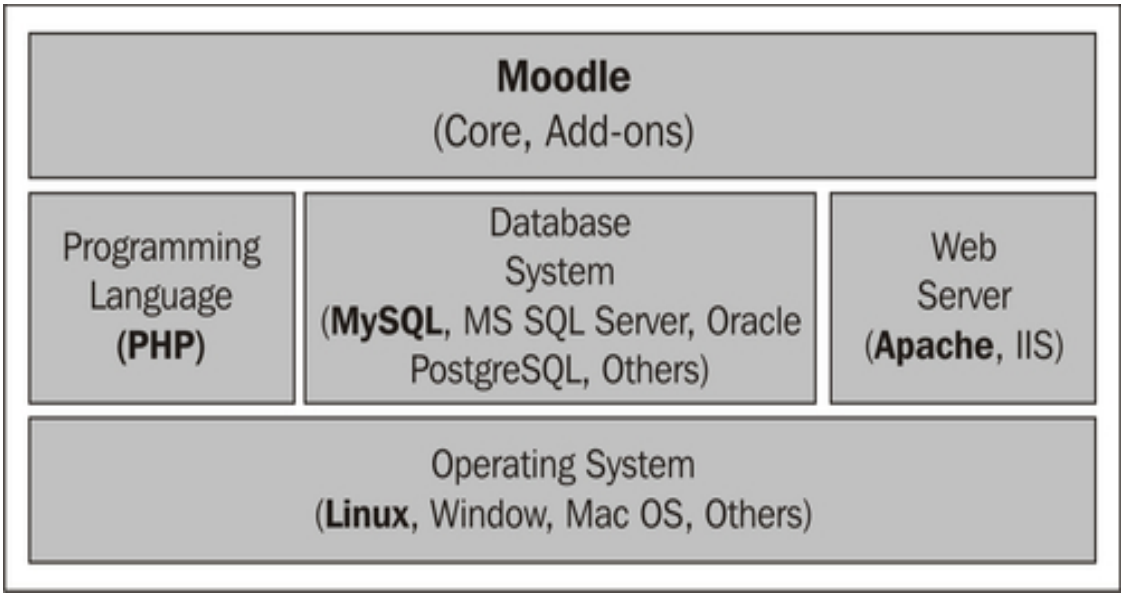


Figura 1.: Arquitetura do *Moodle* (extraída do livro Moodle 2 Administration [1]).

Alguns dos motivos que levam os utilizadores a escolher o *Moodle*:

- É um programa de código aberto, o que significa que a instalação é grátis, podemos usá-lo, modificá-lo e até distribuí-lo sobre os termos da licença GNU;
- É considerado um sistema de gestão do conteúdo e um ambiente virtual de aprendizagem que permite aos professores providenciar e partilhar documentos, atribuir classificações, fóruns de discussão, entre outras funcionalidades de forma a proporcionar aos estudantes uma aprendizagem facilitada, e cursos online de elevada qualidade;
- *Moodle* pode ser usado em quase todos os servidores que usam PHP. Os utilizadores podem descarregar e usar em qualquer computador e facilmente atualizá-lo de uma versão para uma mais recente;
- A chave para o *Moodle* é que é desenvolvido tendo a tecnologia e pedagogia como referências base. Uma das maiores vantagens do *Moodle* em relação aos outros sistemas é a sua forte aposta na pedagogia social e boas ferramentas educacionais;
- O software *Moodle* é usado mundialmente por professores independentes, escolas, universidades e empresas. A credibilidade do *Moodle* é muito elevada. Atualmente,

1 https://subscription.packtpub.com/book/hardware_and_creative/9781849516044/2/ch02lvl1sec01/moodle-architecture

há mais de 35000 sites de 175 países que se registaram com ele, e tem 75 línguas disponíveis;

- *Moodle* funciona sem qualquer modificação em qualquer sistema que suporte PHP assim como *Unix*, *Linux* e *Windows*. Usa *MySQL*, *PostgreSQL* e *Oracle databases*, e outros também suportados;
- Tem muitos recursos úteis para potenciar a aprendizagem dos alunos, como a fácil instalação, personalização de opções e configurações, bom suporte e boas ferramentas educacionais. Além disso, possui excelente documentação e forte apoio à segurança e administração. (Al-Ajlan and Zedan, 2008)

Em contrapartida, alguns utilizadores criticam o *Moodle* pela falta de:

- Capacidade de integração com outros sistemas de recursos humanos;
- Capacidade de integrar sistemas de administração de estudantes e informações sobre os alunos do *Moodle*;
- Suporte para modelos de processos de negócios específicos e complexos;
- Uso de um modelo de administração distribuída para suportar várias escolas e departamentos;
- Capacidades sofisticadas de avaliação e classificação (Martinez and Jagannathan, 2008).

2.2 BLACKBOARD LEARN

A *Blackboard Learn* é um dos principais produtos comerciais de SGA utilizados na América do Norte e na Europa (Munoz & Duzer,2005) e é o sistema de gestão de aprendizagem mais adotado entre as instituições americanas do ensino superior. Fornece um ambiente protegido por palavra-chave e possui ferramentas administrativas que facilitam o ensino online (Lower,2003) (Martin, 2008).

A utilização da *Blackboard Learn* tem algumas vantagens como:

- Maior disponibilidade. *Blackboard* pode ser acedido a partir de qualquer hora e em qualquer lugar. Os estudantes podem recuperar todos os materiais do curso incluindo tarefas, apontamentos, slides, hiperligações de internet e auxiliares de áudio/visual. Os alunos podem enviar as suas tarefas assim que estiverem concluídas. É essa acessibilidade que mais satisfaz os alunos, com base no estudo realizado pela Universidade de Duke, em que os alunos escolheram entre 10 funcionalidades da *Blackboard* qual era a mais útil. A escolha número um para 85% dos alunos foi *"acesso fácil aos materiais e leituras do curso"* (Belanger, 2004).

- Respostas rápidas. Algumas das principais respostas que os alunos receberem através da *Blackboard*: Através da funcionalidade *Blackboard's Test Manager* para *quizzes* e exames, os professores selecionam as perguntas que entendem como adequadas e os alunos podem aceder e realizar. Após a realização, só precisam de submeter e recebem os resultados no momento. O mesmo aplica-se para os trabalhos de casa, após concluírem só precisam de submeter na plataforma e recebem a classificação. Por uma questão de privacidade, a *Blackboard* tem a funcionalidade do *Gradebook* onde as tarefas podem ser resolvidas pelos estudantes e as notas consultadas confidencialmente. Outra forma de receber uma resposta rápida, é através da opção *Survey* que possibilita os alunos de responder de forma imediata e anónima a questões de escolha múltipla e verdadeiros e falsos, sobre a matéria lecionada.

- Melhor comunicação. Existem várias formas na *Blackboard* de comunicar com os alunos. As quatros principais são anúncios, fóruns, salas virtuais e email. Os anúncios estão disponíveis aos alunos após estes entrarem no sistema da *Blackboard*. Isto garante que todos os alunos estejam informados e reduz o trabalho administrativo. Os fóruns de discussão permitem aos alunos interagirem uns com outros alunos, de forma a colocarem questões e responderem aos colegas, tem a vantagem de ser assíncrono, não precisando assim de estar ativos ao mesmo tempo. Por outro lado, temos a sala virtual, que é um ambiente síncrono, e que funciona à base do *chat* e do vídeo, permitindo aos alunos interagirem entre eles. Por fim, temos a opção do email que é muito flexível. Cada estudante pode ter o seu email guardado no perfil. A *Blackboard* possibilita o envio do email de forma individual aos alunos, a grupos de alunos, ou até a todos os alunos.

- Monitorização. A *Blackboard* monitorização os cursos usados pelos alunos e mostra esses resultados na área de estatísticas do curso. Professores podem obter as estatísticas de todos os estudantes ou de algum aluno específico do curso. As tarefas individuais também podem ser monitorizadas. A data e o tempo estão incluídos na última submissão/modificação da tarefa, podendo ser encontrados na respetiva área de submissão de tarefas, permitindo descobrir facilmente as tarefas entregues depois do prazo estabelecido.

- Desenvolvimento de competências. Tais como a organização e a gestão do tempo, são habilidades que ajudam os alunos a lidar com os compromissos eficientemente. A *Blackboard* providencia um calendário para cada curso em que o aluno está inscrito, ajudando assim a otimizar o esforço e conseguir realizar as expectativas traçadas. Todos os documentos do curso têm data, permitindo aos alunos consultar o calendário e fazer uma gestão eficiente do tempo (Bradford et al., 2007).

A figura 2 mostra como é constituída a interface da plataforma *Blackboard Learn*. Através da figura, é possível verificar as fases de processamento da informação quando a plataforma é utilizada.

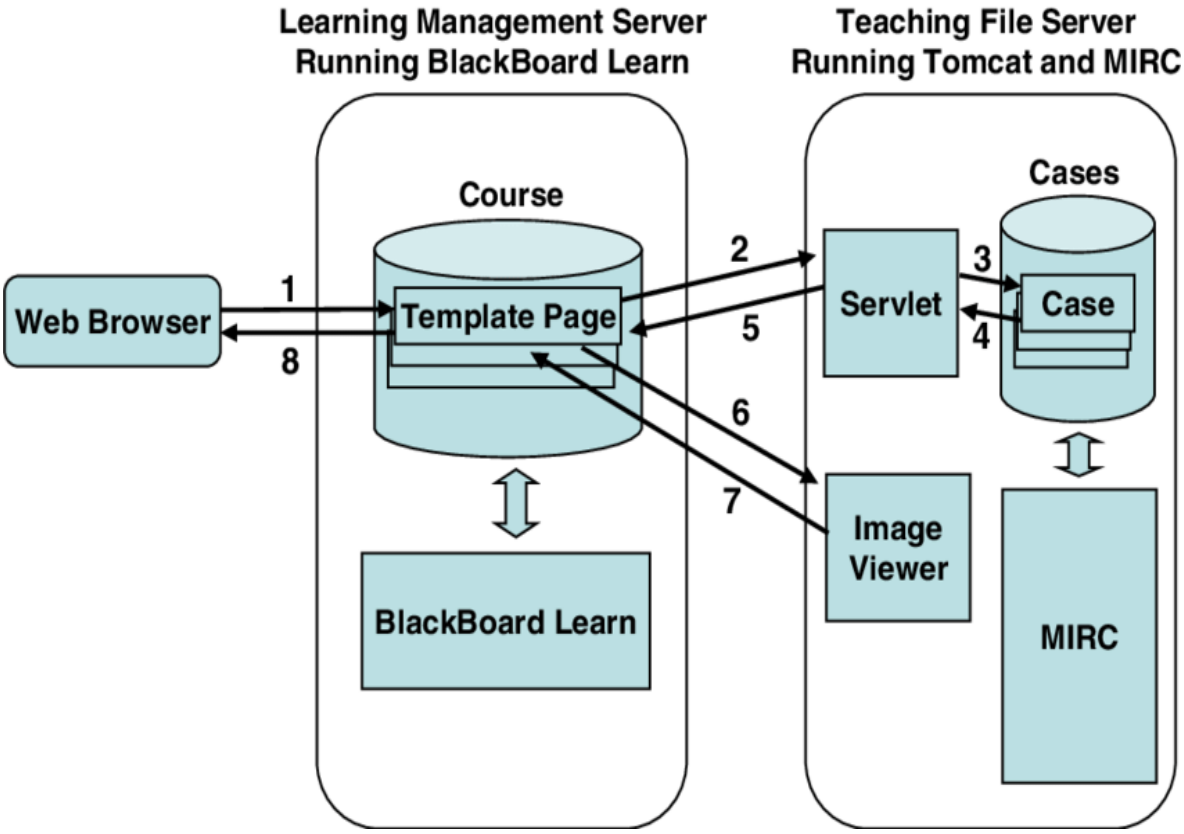


Figura 2.: Interface da Plataforma *Blackboard Learn* (extraída do site *Technews* [2]).

Visão geral da arquitetura e transações na utilização da plataforma *BlackBoard Learn*. 1) O aluno abre uma página, iniciando-se um pedido ao servidor da *BlackBoard*. 2) O código *JavaScript* da página faz uma solicitação JSONP para o *servlet* (classe Java usada para estender as funcionalidades de um servidor) de tradução da execução no servidor de ficheiros de ensino. 3) e 4) O *servlet* executa o pedido solicitado no MIRC. 5) O *servlet* processa o ficheiro e retorna o conteúdo de texto e imagem no formato JSON. 6) e 7) O código *JavaScript* em execução.

2 https://www.researchgate.net/figure/Overview-of-USRC-architecture-and-transactions-1-Student-requests-a-pa
fig3_224820878

2.3 LATITUDELEARNING

Foi idealizado com a ideia de ser um projeto de instrução, associando-se a desafios específicos para manter os seus serviços a par com o tipo de instrução e conhecimento dos seus funcionários. Um dos principais problemas que os SGA enfrentam, é a falta de controlo sobre a adaptação dos alunos.

Por um lado, o SGA mantém os seus funcionários em diferentes zonas do mundo de forma a poderem adaptar-se ao tipo de modelo de negócio requisitado. Desta forma, a adaptação dos clientes à plataforma fica mais fácil e ao mesmo tempo consegue abranger um maior número de localizações.

Um trajeto de desempenho da plataforma permite mapear o desenvolvimento de cada aluno. Cada aluno tem um conjunto de cursos que pode escolher e um respetivo processo de integração de alunos. Pode-se também programar cursos com objetivos claros e métricas de desempenho e desta forma incentivar à conclusão do curso. Um quadro de pontuação ajuda a avaliar os resultados dos alunos e a ajustar as estratégias dos cursos.

Da mesma força, pode-se criar uma comunidade onde os alunos possam partilhar conhecimento e aumentar o seu envolvimento tendo por base os cursos e as estratégias escolhidas.

Alguns dos benefícios do uso da SGA *LatitudeLearning* são:

- Canal de treino específico. Apresenta a sua própria dinâmica e problemas, mais precisamente: 1) Os alunos são desconhecidos; 2) Não há controlo de gestão sobre o ensino; 3) Os alunos estão localizados em diferentes sítios; 4) Os alunos têm diferentes ambientes de aprendizagem. O *LatitudeLearning* procura abordar esses problemas com um auxiliar de desempenho que ajuda a entender o progresso do aluno num respetivo curso. Inclui a integração de parcerias, de forma a ajudar no acompanhamento dos alunos e promover uma melhor realização do curso.

- Suporte *End-to-end*. O fornecedor garante suporte desde a integração até a manutenção. Isto inclui a identificação dos diferentes tipos de programas de ensino que correspondem às suas necessidades e avaliações de metas, fluxos de trabalho e estratégias mensuráveis. O suporte também inclui exames regulares, atualizações e suporte administrativo.

- Desenvolvimento aberto. O fornecedor permite uma participação ativa nas sugestões, desenvolvimentos e introduções de novos recursos da SGA. Usando um processo transparente, a equipa de desenvolvimento de produtos regista todas as ideias pertinentes e de seguida vão a um processo de votação para prioridades, de forma a estabelecer um calendário de desenvolvimento.

- Interesse dos alunos. A SGA permite desenvolver um catálogo de cursos que o administrador e os alunos podem escolher o que pretendem para corresponder às lacunas

de conhecimento. Também se pode construir uma comunidade dentro da plataforma, onde os alunos partilham ideias e conhecimentos. Da mesma forma, as metas podem ser incentivadas para estimular tanto os administradores quanto os alunos a concluírem o seu programa de aprendizagem.

2.4 DOKEOS

É um ambiente de SGA e é desenvolvido em *PHP*, que usa *MySQL* como base de dados, possibilitando atividades de aprendizagem e outras dentro de uma comunidade de ensino: distribuição de materias de ensino, acompanhamento, comunicação em tempo real, etc. Apresenta uma adaptação fácil e é flexível. Tem muitas ferramentas disponíveis, tais como: Documentos (criação de documentos, armazenamento e organização de arquivos), Módulos (criação de módulos), *Dropbox*, Avisos, entre outras (Jovanovic and Jovanovic, 2015).

Dokeos é considerado uma das maiores ferramentas de aprendizagem a nível da *internet*, pois contém a maioria das ferramentas, como fóruns, cursos de ensino e blogs. O que diferencia o *Dokeos* é a possibilidade dos alunos usarem o *Free Campus*, que apresenta algumas vantagens, tais como:

- Oferece uma maneira fácil de criar e controlar um curso *online* para qualquer professor e aluno sem exigir um servidor, conhecimento para instalar e manter o programa, o que é uma vantagem sobre muitas ferramentas de gestão de cursos gratuitos ou comerciais.
- Inclui sessões de conversa que ficam gravadas automaticamente e armazenadas, para poderem usufruir posteriormente durante o curso.
- Permite que os professores adaptem as ferramentas de acordo com as necessidades dos alunos, dando assim um acompanhamento adequado e segurança (Kilickaya, 2009).

Outro aspeto importante usado pelo *Dokeos* é a computação em nuvem tornando a aprendizagem mais eficaz e menos dispendiosa, como se fosse um programa com serviços. As principais vantagens desta abordagem são:

- Aprender por menos. Certas análises mostram que os governos poderiam economizar entre 20 a 25% nos orçamentos de tecnologia se usassem a computação em nuvem. Nas empresas também, o programa como serviço é uma fonte significativa da economia. As soluções de aprendizagem estão se a tornar menos dispendiosas, menos complicadas de implementar, mais rápidas de criar e, por fim, mais adaptáveis às necessidades de ensino dos seus utilizadores. É possível criar cursos completos em apenas alguns dias.

- Ensino à distância é fácil. Aplicações distribuídas de aprendizagem com programa como serviço podem ser controladas através da internet como *Firefox* ou *Internet Explorer*, com os consumidores a não precisarem de atualizar constantemente, adicionar versões de segurança, e assegurarem a disponibilidade do serviço. *DOKEOS* consegue gerir tudo *online*, retirando a pressão do lado do consumidor. Implementação da solução de aprendizagem fica rápida, fácil e mais direta. Isto significa que podemos concentrar no que realmente interessa, ou seja, preparar ou desenvolver os conteúdos de ensino para os cursos.

- A flexibilidade da computação em nuvem. O tempo que de outra forma seria desperdiçado, agora pode ser usado para aprender. A flexibilidade oferecida pela solução de aprendizagem *Dokeos*, simplicidade e usabilidade, faz com que esta ferramenta se adapte com facilidade a qualquer equipa de trabalho.

2.5 EFRONT

eFront é uma plataforma de aprendizagem de código aberto (também conhecido como Sistema de Gestão de Cursos, ou SGA, ou Ambiente de Aprendizagem Virtual). *eFront* foi escrito do zero, fazendo alterações essenciais na estrutura principal do sistema e lançado sob uma licença de código aberto em setembro de 2007.

Tem como objetivo ajudar na criação de comunidades de aprendizagem *online*, oferecendo várias oportunidades de colaboração e interação através de uma interface de usuário baseada em ícones. A plataforma oferece ferramentas para criação de conteúdo, criação de testes, gestão de tarefas, relatórios, avisos internos, fórum, troca de mensagens, pesquisas, calendário e outros. É um modelo compatível com o modelo de referência de objeto de conteúdo compartilhável (SCORM1.2) e sistema compatível com SCORM 2004/4ª edição traduzida em 40 idiomas (Fariha and Zuriyati, 2014).

Apresenta algumas caraterísticas idênticas entre SGA, tais como:

- Conteúdo Amigável. Reutiliza o melhor conteúdo em vários cursos. Importa material útil da internet para melhorar ainda mais.

- Mecanismos de avaliações. Suporte nativo para vários tipos de perguntas, questionários e testes com relatórios extensivos sobre todos os resultados.

- Mecanismos de pesquisas. Pesquisas personalizadas entre os utilizadores. Recolha de dados úteis sobre o envolvimento dos alunos e a eficiência do curso.

- *Scorm & TinCan (xApi)*. Suporte interno para os padrões de interoperabilidade de conteúdo mais usados para enriquecer os seus cursos e aprimorar a experiência dos alunos.

- Tarefas. Permite aos alunos enviar tarefas em diferentes formatos. Definir condições e prazos de acordo com as regras e pré-requisitos do curso.
- *HTML5*. Criar conteúdos interativos na plataforma com o editor *H5P* do *eFront*.
- Conteúdo reutilizável. Os cursos podem incluir novo material de aprendizagem e compartilhado. Extrair conteúdo de uma biblioteca de aulas ou unidades individuais diretamente dos cursos.
- Repositório de ficheiros. Mantém todos os ficheiros armazenados no mesmo sítio. Organiza e reutiliza em diferentes cursos ou partilha com outros colegas.

2.6 DISCUSSÃO

Após a revisão das diferentes plataformas, neste capítulo iremos fazer uma comparação dos aspetos chave, as principais diferenças entre cada SGA e determinar as caraterísticas fundamentais que levaram a escolher a melhor opção para realizar o trabalho futuro.

Desta forma, um conjunto de caraterísticas comparativas foi selecionado como um meio para analisar e avaliar estas plataformas. Estes recursos são alguns dos mais importantes ao analisar a experiência do aluno. Este conjunto consiste em:

1. **Grátis** - Sendo um trabalho realizado por via académica sem recurso a apoios financeiros é importante que a plataforma a escolher possibilite o seu uso sem ser necessário o recurso ao pagamento;
2. **Código aberto** - é um tipo de desenvolvimento que permite à comunidade colaborar e modificar o produto. Desta forma, o projeto tem de ser transparente, para todas as pessoas conseguirem desenvolver, com melhores soluções a surgir por ser um maior número de pessoas a trabalhar;
3. **Sistemas operativos** - Hoje em dia os produtos acabam, pela grande maioria, por poder serem usados em quase todos os sistemas de forma a abranger um maior número de clientes. Contudo convém analisar de forma a prevenir problemas futuros;
4. **Base de dados** - Os dados recolhidos têm de ser armazenados em algum sítio. Cada SGA escolhe normalmente um sítio para redirecionar estes dados, de forma a poder usar quando for necessário;
5. **Servidor Internet** - é o responsável por aceitar os pedidos *HTTP* de clientes, normalmente navegadores da *internet* e responder a esses pedidos;
6. **Suporte móvel** - O sistema em questão é utilizado em diferentes dispositivos, como telemóveis, *tablets* ou outro tipo de aparelho;

Em seguida, é apresentada uma tabela de comparação das plataformas estudadas anteriormente, usando as caraterísiticas comparativas explicadas.

Caraterística/Sistema	*Moodle*	*Blackboard*	*LatitudeL*	*Dokeos*	*eFront*
Grátis	x		x	x	x
Código Aberto	x			x	x
Sistemas Operativos	Todos	Windows, Linux, and Mac OS X	Todos	Windows, Linux, and Mac OS X	Todos
Base de dados	MySQL			MySQL	MySQL
Servidor Internet	Apache	Apache	Apache	Apache	Apache
Suporte móvel	x	x		x	

Tabela 1.: Tabela de comparação dos aspetos importantes para a implementação.

Como podemos observar a tabela de comparação (Tabela 1), permite concluir que outros aspetos também poderiam estar incluídos, contudo estes servem de base para escolher a SGA que melhor se enquadra no âmbito deste projeto.

Através da análise da tabela observamos que a SGA *Blackboard* não é gratuita o que limita logo à partida o trabalho por não haver orçamento para se desenvolver este projeto.

De seguida temos a importância das plataformas se desenvolverem em código aberto, porque permite um maior suporte por parte da comunidade e o respetivo desenvolvimento de caraterísticas futuras. Sendo o *LatitudeLearning* desenvolvido apenas pela empresa responsável e não mostrando o seu código torna o desenvolvimento mais complicado, sendo assim rejeitada.

Em relação aos sistemas operativos, ao servidor da *internet* e à base de dados, todas as plataformas que restam tiveram um bom desempenho, não limitando assim a tarefa de seleção, mostrando grande diversidade e adotando o *MySQL* como preferência para armazenar os dados.

Por fim, sendo o recurso ao telemóvel uma ação cada vez mais frequente, é relevante que o uso da plataforma eleita se consiga realizar neste dispositivo, tendo assim que excluir o *eFront* da lista, uma vez que não fornece este tipo de suporte.

Sendo uma *Moodle* uma plataforma mais completa pela diversidade de ferramentas que dispõe, como serviço de vídeo ou até para os próprios alunos, como grupos de trabalho e o rastreio do seu desenvolvimento, em relação ao *Dokeos*(Al-Ajlan and Zedan, 2008).

Com este estudo completo, terminamos o estado da arte, revelando alguns aspetos importantes na escolha de SGA's e optando por prosseguir o trabalho com o *Moodle*.

3

ARQUITETURA

De forma a perceber melhor a arquitetura deste projeto, neste capítulo vamos abordar todos os intervenientes que entram no processo. Desde a recolha dos dados, o armazenamento numa base de dados, seguido do tratamento dos mesmos e de forma a finalizar o processo, aplicar o algoritmo mais adequado para estabelecer uma relação do *stress* com a os dados biométricos recolhidos.
A primeira secção dá uma ideia do processo completo, onde todas as etapas são descritas com rigor. Também é abordado o tipo de tecnologias escolhidas, de forma a garantir um resultado final otimizado. Na secção das Caraterísticas e Comportamentos é explicado o processo de captura dos dados através das ferramentas indicadas e é especificado o processo de avaliação de recolha do nível de *stress*. A secção dos algoritmos de ML apresenta as diferentes técnicas abordadas, e os resultados obtidos através do uso das mesmas, com o objetivo de avaliar e comparar os resultados obtidos.
De forma a concluir este capítulo, a última secção é dedicada à discussão e análise de todos os temas abordados, fazendo um apanhado dos aspetos mais importantes.

3.1 ARQUITETURA DO PROGRAMA

Esta arquitetura constitui diferentes fases para o processamento da informação recolhida, desde a fase inicial da recolha dos dados até a utilização dos algoritmos para estabelecer uma relação entre os resultados obtidos. Como podemos observar na figura 3 a arquitetura definida para este projeto.

Este programa inicia-se com a recolha de informação através das acções do utilizador com os periféricos, rato e teclado, do computador (Carneiro et al., 2015). Os dados biométricos são então armazenados e o próprio utilizador durante um momento de avaliação/*stress* realiza uma auto-avaliação do seu nível de *stress* (Carneiro et al., 2019). De seguida, os dados são tratados de forma a cumprir com os objetivos estabelecidos, neste caso, as métricas definidas previamente para estudo. Por fim, os dados são aplicados nos algoritmos de forma a fazer uma análise do algoritmo que melhor se adequa para cumprir com os objetivos.

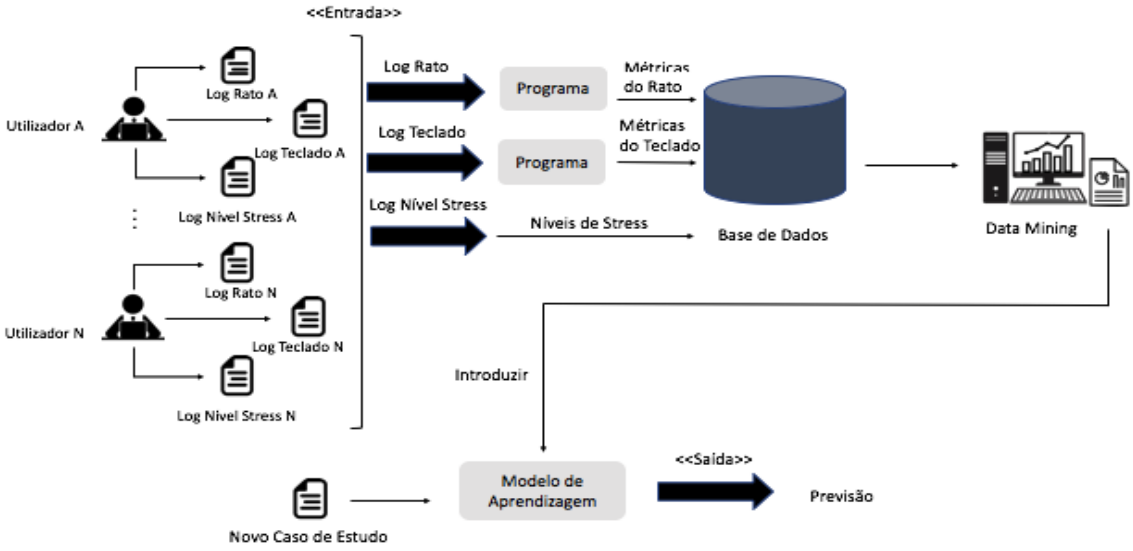


Figura 3.: Arquitetura do Programa

3.1.1 *Rato/Teclado*

Estas ferramentas permitem que seja possível analisar as ações provenientes do utilizador, para posteriormente fazer a análise dos dados e relacioná-los com o nível de *stress* apresentado (Pimenta et al., 2016). Esta forma não intrusiva de saber as ações do utilizador, permite que não haja interferência no resultado final da prova.

3.1.2 *Tipos de Tecnologia*

Em relação ao desenvolvimento do software para recolha dos movimentos feitos com o rato e o uso do teclado, a linguagem escolhida foi o *Python*.

Foi projetado em 1990 por Guido van Rossum, sendo uma linguagem de programação interpretada, interativa e orientada a objetos. Fornece estruturas de dados de alto nível, como listas e matrizes associativas (dicionários), digitação e vinculação dinâmica, módulos, classes, exceções, gestão automática de memória, entre outros. Tem uma sintaxe extremamente simples e elegante e ainda é uma linguagem de programação poderosa e de propósito geral. Como muitas outras linguagens de *script*, ela é gratuita, mesmo para fins comerciais, e pode ser executada praticamente em qualquer computador moderno. Um programa *python* é compilado automaticamente pelo interpretador no código do byte independente da plataforma que é então interpretado (Sanner et al., 1999).

Algumas das vantagens que levaram à escolha desta linguagem foram:

- **Bibliotecas de suporte extensas** - Fornece grandes bibliotecas padrão que incluem as áreas como operações de *string*, *Internet*, ferramentas de serviços *Web*, interfaces de sistema operacionais e protocolos. A maioria das tarefas de programação usadas já estão definidas, o que permite ter código escrito em *Python*.

- **Produtividade** - Fortes recursos de integração de processos, estrutura de testes de unidade e os recursos aprimorados de controlo contribuem para aumentar a velocidade da maioria das aplicações.

- **Recurso de Integração** – O *Python* integra o *Enterprise Application Integration*, que facilita o desenvolvimento de serviços da *Web* invocando componentes. Contém desenvolvidas capacidades de controlo, uma vez que chama diretamente através de C, C++ ou Java via Jython. O *Python* também processa XML e outras linguagens de marcação, pois pode ser executado em todos os sistemas operacionais modernos por meio do mesmo código de *bytes*.

Na base de dados, a tecnologia utilizada foi *MongoDB* por ser poderosa, flexível e escalável. Combina a caraterística de ser dimensionável com as caraterísticas tais como os indexes secundários, consultas por intervalo, ordenamento e agregações.

Os motivos que levaram o *MongoDB* a destacar-se das outras bases de dados foram:

- **Fácil uso** - é uma base de dados orientada a documentos em vez de ser racional. A razão principal para não escolher o modelo relacional tem haver com a facilidade de crescimento que as bases de dados orientadas a documentos apresentam. Este modelo substitui o termo "linha"por um modelo mais flexível, "documento". Por permitir documentos embebidos e matrizes, a abordagem orientada a objetos torna possível representar relações hierárquicas complexas com um simples registo. Também não tem esquemas pré-definidos, o que facilita na remoção e adição de ficheiros.

- **Escalamento** - Sendo um dos principais objetivos, é fácil dividir os dados entre múltiplos servidores. O *MongoDB* automaticamente equilibra o balanceamento dos dados e os carregamentos através do *cluster*, redistribuir documentos automaticamente e direcionar os pedidos dos clientes para as máquinas certas. Isto permite aos programadores ter o foco na aplicação e não no seu escalamento. Quando o *cluster* precisa de mais espaço, novas máquinas podem ser adicionadas e o *MongoDB* calcula como é que a distribuição da informação deverá ser processada.

- **Várias Ferramentas** - O *MongoDB* destina-se a ser uma base de dados de propósito geral, para além de criar, ler, atualizar e eliminar os dados, também providência uma lista com funcionalidades únicas:

  - *Indexar* - Suporta indexes secundários genéricos, permitindo uma variedade de consultas rápidas, e providência capacidades de índex de texto completo também.
  - *Agregação* - Suporta uma agregação sequencial que permite construir agregações complexas a partir de peças simples e permite uma otimização da base de dados.
  - *Tipos de coleção especiais* - Suporta coleções com prazos de tempo para os dados que eventualmente expiram, assim como sessões. Também suporta coleções de tamanho fixo, o que é útil para dados recentes, como *logs*.
  - *Armazenamento de ficheiros* - Suporta um protocolo que permite o uso fácil para armazenamento de grandes ficheiros e ficheiros com metadados.

- **Qualidade no tempo de resposta** - Adiciona um preenchimento dinâmico aos documentos e pré alocações aos ficheiros de dados para trocar o espaço extra de uso por rendimento consistente. Usa o que pode em relação *RAM* (memória de acesso aleatório), assim como na cache e tenta escolher automaticamente os corretos indexes para as consultas. Desta forma, quase todos os aspetos referentes a arquitetura do *MongoDB* foram a pensar no alto desempenho.

Por outro lado, algumas das caraterísticas do modelo relacional não foram adotadas por este, nomeadamente, os *joins* e transações complexas de colunas. Omitir estas funções foi uma decisão arquitetural para permitir um melhor escalamento, pois ambas as funcionalidades dificultam o nível de eficiência em sistemas distribuídos (Chodorow, 2013).

3.2 CARATERÍSTICAS E COMPORTAMENTOS

Através da *script* criada, o sistema permite capturar as informações referentes ao comportamento dos utilizadores com o uso do rato e teclado durante a realização dos exames. A informação recolhida assenta em três diferentes objetivos: as métricas de comportamento do rato (obtidas através da *pynput.mouse*), métricas de comportamento do teclado (obtidas através da *pynput.key*) e o nível de stress (através de uma janela, que permite ao utilizador classificar numa escala de 1 a 5). Os dados obtidos do comportamento do rato têm de ser posteriormente tratados, uma vez que a biblioteca usada apenas se destina à recolha das coordenadas na sua posição. Entre muitas métricas disponíveis a calcular, apenas as que consideramos mais relevantes serão apresentadas:

1. *Métricas de Comportamento do Rato*
   - *Soma absoluta dos ângulos (SAA)*: Esta métrica descobre quanto o rato virou, independentemente da direção que foi (em graus);

- *Diferença da Distância em Linha Reta e Distância Real Entre Cliques*: Esta métrica quantifica a distância percorrida pelo rato entre dois cliques e subtrai com a distância efetuado pelo rato em linha reta definida por dois cliques consecutivos (em *pixels*);
- *Duração do Clique*: Mede o tempo demorado entre cliques. Quanto maior o tempo de duração dos cliques menor é eficiência de interação (em milissegundos);
- *Velocidade do rato*: A distância percorrida pelo rato (em *pixels*) ao longo do tempo (em milissegundos);
- *Número de Total de Cliques*: Guarda o número de cliques efetuados pelo utilizador;
- *Número total de logs*: É o número de registos guardados durante o intervalo de tempo, ou seja, número total de todas as ações efetuadas com o rato;

2. *Métricas de Comportamento do Teclado*
   - *Número de BackSpace*: Indica número de vezes que o utilizador pressionou a tecla *backspace*;
   - *Lado esquerdo do Teclado*: Indica o número de vezes que o lado esquerdo foi usado pelo utilizador;
   - *Lado direito do Teclado*: Indica o número de vezes que o lado direito foi usado pelo utilizador;
3. *Precessão do Nível de Stress*
   - *Nível de stress*: O utilizador indica o nível de *stress* que sente no momento, numa escala de 1 a 5. Esta ação repete-se de X em X tempo;

3.3 ATORES DO SISTEMA

Um ator é um classificador do comportamento que específica um papel desempenhado por uma entidade externa que interage com o utilizador do sistema (por exemplo, troca de dados), outro sistema ou *hardware* usando serviços relacionados com esse sistema.

O termo função é usado informalmente como algum tipo, grupo ou faceta específica de utilizadores que requerem serviços específicos do assunto modelado com casos de uso associados. Quando uma entidade externa interage com o assunto, ela desempenha o papel de um ator específico. Essa única entidade física pode desempenhar várias funções diferentes e uma função específica pode ser desempenhada por uma ou várias instâncias diferentes (Boggs and Boggs, 2002).

Neste projeto, existem dois tipos de utilizadores: Estudantes e Administrador. De um modo geral, os atores são os elementos que irão interagir com o sistema. Esses atores são representados como elementos importantes no mecanismo operacional do sistema.

3.3.1 *Administrador*

Este utilizador é responsável por gerir todos os dados recolhidos no programa. O administrador tem a responsabilidade de garantir o funcionamento adequado do sistema, manter o programa atualizado e verificar o *feedback* dos utilizadores, criando, se necessário, alterações que modifiquem positivamente a utilização do programa. É responsável por fazer a previsão do nível de *stress* com base nos valores recolhidos.

3.3.2 *Professor*

É responsável pela criação dos exames de avaliação para os estudantes realizarem. Estabelece o período de duração dos exames e o tipo de método de avaliação do nível de *stress*, quer intervalado (de X em X tempo) ou só no início e no fim do momento de avaliação.

3.3.3 *Estudante*

Estes utilizadores são responsáveis pela realização dos exames e podem consultar os resultados obitdos do programa. Após iniciarem o programa, os dados referentes à utilização do rato e teclado são recolhidos e o utilizador faz uma avaliação do seu nível de *stress*.

3.4 Análise de Requisitos

Esta parte do desenvolvimento inicial do projeto é fundamental para haver sucesso. A engenharia de requisitos tem um papel determinante na identificação dos objetivos a serem alcançados pelo programa, as suas funcionalidades e atribuição de responsabilidades da criação dos mesmos (Van Lamsweerde, 2000). Este processo abrange as tarefas que determinam as necessidades ou condições a serem prestadas para a realização de um novo produto ou alteração de um produto, tendo em consideração possíveis conflitos de requisitos dos diferentes interessados. Os requisitos devem ser mensuráveis, testáveis, relacionados com as necessidades ou oportunidades de negócio e com um nível de detalhe suficiente para o *design* do sistema (Pohl, 2010).

O objetivo da engenharia de requisitos é desenvolver e manter atualizado um documento especifico de requisitos que descreva as necessidades do sistema. Estes requisitos podem ser funcionais ou não funcionais.

3.4.1 *Requisitos Funcionais*

São as funções do sistema ou componentes, onde uma função descreve um comportamento específico na utilização do programa (Fulton and Vandermolen, 2017).

Os requisitos funcionais descrevem as caraterísticas disponíveis do sistema, explicando de forma completa e consistente. Por outras palavras, requisitos funcionais devem incluir ações de determinados momentos, os procedimentos da linha de trabalho do sistema, e outros assuntos ou requisitos complementares que são importantes para o funcionamento do sistema (Wiegers and Beatty, 2013).

A lista que segue apresenta um conjunto de caraterísticas disponíveis quando o programa é utilizado:

- Permitir ao utilizador aceder a todas as funcionalidades do programa;
- Permitir ao utilizador recolher os dados referentes ao uso do teclado durante a respetiva utilização;
- Permitir ao utilizador recolher os dados referentes ao uso do rato durante a respetiva utilização;
- Permitir ao utilizador recolher o nível de *stress* baseado numa classificação em escala;
- Permitir ao utilizador calcular um conjunto de métricas referentes aos dados recolhidos;
- Permitir ao utilizador prever o nível de *stress* com base nas métricas calculadas;

3.4.2 *Requisitos Não Funcionais*

Requisitos não funcionais (ou qualidades do sistema) descrevem termos relacionados com a segurança, confiabilidade, manutenção, escalabilidade, usabilidade e tecnologias integradas no sistema (Chung et al., 2012).

Estes riquisitos são qualidades e restrições persistentes que asseguram a usabilidade e eficácia de todo sistema. Falhar em algum destes pontos pode comprometer o funcionamento do sistema e não cumprir com as exigências de negócio, dos utilizadores e do mercado (Young, 2001).

A lista que segue apresenta um conjunto de caraterísticas disponíveis quando o programa é utilizado:

- **Acesso Livre**: desenvolvimento e implementação do programa através do uso de tecnologia grátis(*PyCharm, Bibliotecas de Python*);

- **Apresentação**: o programa providência uma forma atraente, apelativa, simples e fácil de ser usado;
- **Tempo de Aprendizagem**: os utilizadores devem saber como utilizar o sistema num curto espaço de tempo;
- **Escalabilidade**: o sistema deve ser capaz de manter o tempo de resposta quando há um aumento de informação (mais dados sobre os utilizadores);
- **Padrão**: A utilização do programa deve seguir uma determinada ordem de forma a ficar simples e de fácil utilização;
- **Desempenho**: O sistema deve processar qualquer evento num determinado período de tempo;
- **Segurança**: todos os dados envolvidos no processo apenas têm utilização nas tarefas do programa, ficando impossibilitada qualquer tipo de partilha de informação;
- **Espaço de Armazenamento Alocado**: todos os dados utilizados durante o processo de recolha e processamento são armazenados na base de dados, que posteriormente são utilizados para os cálculos de forma a otimizar a informação pertinente a ser guardada;
- **Portabilidade**: O programa deve correr na maioria das plataformas de computador (desde que exista uma versão do *PyCharm* disponíveis para ser instaladas nessa plataforma);
- **Comunicação**: a solução deve incluir uma forma de partilhar e aceder por qualquer pessoa;
- **Requisitos Éticos**: o sistema não deve usar a informação do utilizador para propósitos comerciais externos;
- **Organização**: manter todos os ficheiros organizados e estruturados de forma a ser fácil de interpretar pelo utilizador;

3.5 CASOS DE USO

Casos de uso são uma lista de ações ou eventos, geralmente definem as interações entre o papel a desempenhar (conhecido na Linguagem de Modelagem Unificada como ator) e um sistema, para atingir uma meta. O ator pode ser uma pessoa ou um sistema externo (Jacobson et al., 2011). Por outras palavras, um caso de uso é uma série de interações relacionadas entre um utilizador (ou mais geralmente, um ator) e um sistema que permite ao utilizador

atingir uma meta (Bittner, 2002). No entanto, os diagramas de casos de uso nunca descrevem como eles são implementados. Eles podem ser visualizados como uma caixa preta, onde apenas a entrada, a saída e a função da caixa preta são conhecidas.

Os objetivos dos diagramas de casos de uso podem ser os seguintes:

- Usado para reunir requisitos de um sistema;
- Usado para obter uma visão externa de um sistema;
- Identificar fatores externos e internos que influenciam o sistema;
- Mostrar a interação entre os requisitos e os atores;

A análise dos casos de uso é uma técnica de análise de requisitos importante e valiosa que tem sido amplamente utilizada na engenharia de *software* moderna, tornando-a uma das melhores maneiras de capturar requisitos funcionais de um sistema (Cockburn, 2008).

A seguir, serão mostradas as diferentes interações entre os utilizadores e o sistema.

3.5.1 *Diagrama de Casos de Uso*

Existem cinco tipos de relacionamento num diagrama (Bittner, 2002):

- Associação entre um ator e um caso de uso;
- Generalização de um ator;
- Estender o relacionamento entre dois casos de uso;
- Incluir o relacionamento entre dois casos de uso;
- Generalização de um caso de uso;

Neste diagrama estão representadas as interações possíveis pelos diferentes utilizadores. O Professor é responsável pela parte da prova, que o estudante vai realizar, podendo Criar, Editar e Eliminar as provas. O Estudante é o alvo do estudo, que durante a resolução da prova estruturada pelo Professor, os dados biométricos são recolhidos. Por fim, o Administrador fica responsável por garantir o bom funcionamento do programa, caso alguma situação não corra como previsto ou seja preciso atualizar o programa. A responsabilidade de garantir que os dados estão de acordo com o planeado é também da responsabilidade do administrador.

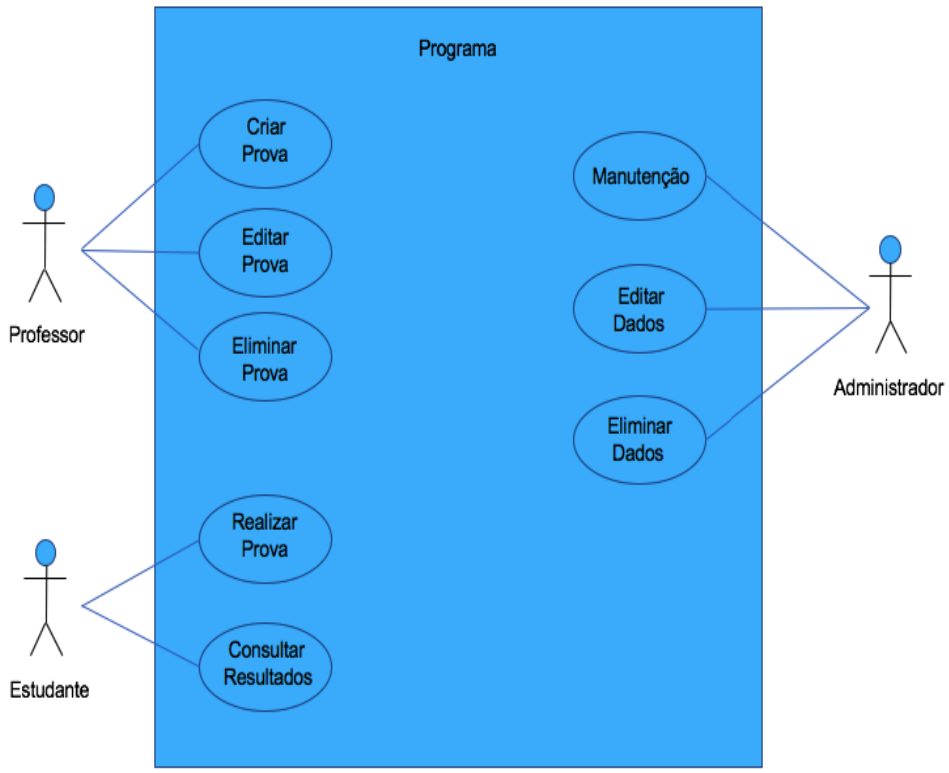


Figura 4.: Diagrama de Casos de Uso do programa

3.5.2 *Descrição dos Casos de Uso*

A Descrição dos Casos de Uso permite saber o que o utilizador final deseja alcançar, identificando primeiro o seu problema. Quando os problemas são conhecidos, as soluções começam a ser definidas. Este processo permite delinear as interações entre o ator e o sistema na solução de um problema, conforme descrito numa situação do sistema.

A Descrição dos Casos de Uso constitui uma conversa de alto nível entre usuário e sistema, que visa descobrir as intenções ou ações dos atores e como o sistema reage a essas intervenções do ator. A descrição deve ser concisa ao decidir o que incluir no fluxo de eventos. Esta análise visa identificar requisitos do ponto de vista do utilizador final.

As tabelas 2, 3, 4, 5, 6, 7, 8 e 9 no Anexo A mostram as descrições dos diagramas de casos de uso.

3.6 DISCUSSÃO E ANÁLISE

Com o estudo da arquitetura, das métricas a calcular quer do teclado e do rato e os dados referentes ao nível de *stress*, é possível clarificar a base de trabalho para desenvolver este programa. A linguagem de programação *Python* apresentou um vasto conjunto de vantagens para a solução, para além de ser de fácil perceção e muito utilizada na comunidade dos programadores, crescendo de uma forma muito acentuada nos últimos anos.

Para o armazenamento das informações retiradas foi escolhido o *mongoDB* como base de dados, a fácil utilização e compreensão do funcionamento foi uma das principais características na escolha. O seu tempo de resposta também é bastante satisfatório e a nível de escalamento de dados apresenta bons resultados.

Por fim, foram apresentados os intervenientes no sistema, que apesar de não estar integrado num SGA, foram traçadas as funcionalidades pretendidas, de forma a dar uma preceção ideal do sistema que teria o programa implementado. Assim é possível observar que existem três atores do sistema, o Estudante, que realiza as provas e pode consultar os seus resultados obtidos, o Professor, realiza as provas e estabelece o tempo de duração das mesmas de forma a alertar o programa do tempo de monitorização, e por fim, o Administrador, que fica responsável por controlar os dados recolhidos e garantir o bom funcionamento do programa, mantendo-o atualizado.

4

MACHINE LEARNING

Com a evolução constante dos meios tecnológicos e a procura de melhores respostas, criou-se a necessidade de desenvolver um sistema que conseguisse através de resultados anteriores fazer previsões. Então começaram a criar um mecanismo que através da análise dos dados fosse capaz de estabelecer uma relação significativa entre eles. Este processo é feito através de um computador programado para que possa "aprender"através das informações que lhe são passadas. Por outras palavras, a aprendizagem é no fundo um processo de conversão da experiência em conhecimento. Através de um conjunto de dados, experiência, consegue-se extrair uma opinião, conhecimento, normalmente direcionada para alguma tarefa específica.

Uma ideia importante é que quanto maior for a aprendizagem inicial do processo, mais fácil vai ser aprender com exemplos futuros. Por outro lado, quantas mais assunções tiver, menos flexível vai ser a aprendizagem.

Normalmente o uso de técnicas de ML ocorre quando os problemas são de elevada complexidade ou precisam de ser adaptados. No caso dos problemas complexos, podem ser (Shalev-Shwartz and Ben-David, 2014):

- **Tarefas realizadas por Humanos**: Existem inúmeras tarefas que fazemos na nossa rotina e que não temos uma perceção total da sua realização para elaborar um programa. Exemplos como conduzir, perceber imagens e reconhecer discursos. Em todas estas tarefas, os programas de ML, programas que "aprendem com a experiência", alcançam resultados satisfatórios, quando submetidos a vários exemplos de treino.

- **Tarefas complexas para os Humanos**: Outro conjunto de tarefas que beneficia do uso das técnicas ML está relacionado com o elevado número e complexidade dos dados: dados astronómicos, tornar arquivos médicos em conhecimento médico, previsões do tempo, motores de busca da *internet*. Com as quantidades de informação arquivada, torna-se para o ser humano difícil de reter toda a informação e analisar os pormenores. Aprender a detetar padrões em grandes conjuntos de dados complexos é um domínio apropriado para a combinação de programas que aprendem com a capaci-

dade de memória quase ilimitada e aumentado a velocidade de processamento dos computadores alcançar novos objetivos.

- **Adaptabilidade**: Uma das limitações que as ferramentas dos programas apresentam é a sua rigidez, quando concluídos normalmente mantém-se inalterados. Contudo, as exigências mudam com o tempo ou de utilizador para utilizador. As ferramentas de ML, programas que adaptam o seu comportamento aos dados recolhidos, apresenta uma solução para este problema. Conseguem por norma, adaptar as mudanças ao ambiente com que estão a interagir.

Partilha conteúdos comuns com outras áreas como os campos da matemática, isto é, estatística, otimização, teoria da informação (estuda a quantificação, armazenamento e comunicação da informação) e teoria do jogo (estuda situações estratégicas onde jogadores escolhem diferentes ações na tentativa de melhorar o seu retorno). É uma área da ciência da computação, que tem como objetivo programar máquinas para que consigam aprender. De certa forma, os algoritmos de ML podem ser visto como um ramo da IA (Inteligência Artificial), pois a habilidade de transformar experiência em conhecimento ou detetar padrões significantes em conjuntos de dados complexos é preocupação da inteligência humana. Apesar destas semelhanças, em contraste com a IA tradicional, ML não tenta construir imitações autónomas de comportamento inteligente, mas em vez disso usar as capacidades dos computadores para complementar a inteligência humana.

Nesta fase do projeto o uso ds algoritmos de ML tem um papel importante para se estabelecer uma relação entre os dados recolhidos. Esta área da inteligência artificial é composta por muitos algoritmos, contudo serão selecionados alguns com base nas suas caraterísticas mais relevantes e posteriormente, através da inserção dos dados, fazer uma avaliação dos resultados obtidos em cada um dos algoritmos.

Na figura 5 está representado o processo de ML, mostrando as diferentes fases e qual a responsabilidade de cada uma.

Este processo é dividido em duas fases principais, a primeira corresponde ao tratamento dos dados e a segunda à aplicação dos algoritmos sobre os mesmos. A fase inicial consiste na recolha dos dados referentes à tarefa a realizar e de seguida averiguar se os dados apresentam os requisitos para avançarem no processo. Na segunda fase, já com os dados tratados, são aplicados diferentes algoritmos para estabelecer um padrão entre os dados, de forma a retirar informação útil sobre a matéria em questão.

4.1 ALGORITMOS

Existem diferentes tipos de algoritmos utilizados em ML com específicas abordagens tendo como base os dados iniciais, o que se quer extrair e o tipo de tarefa ou problema que se

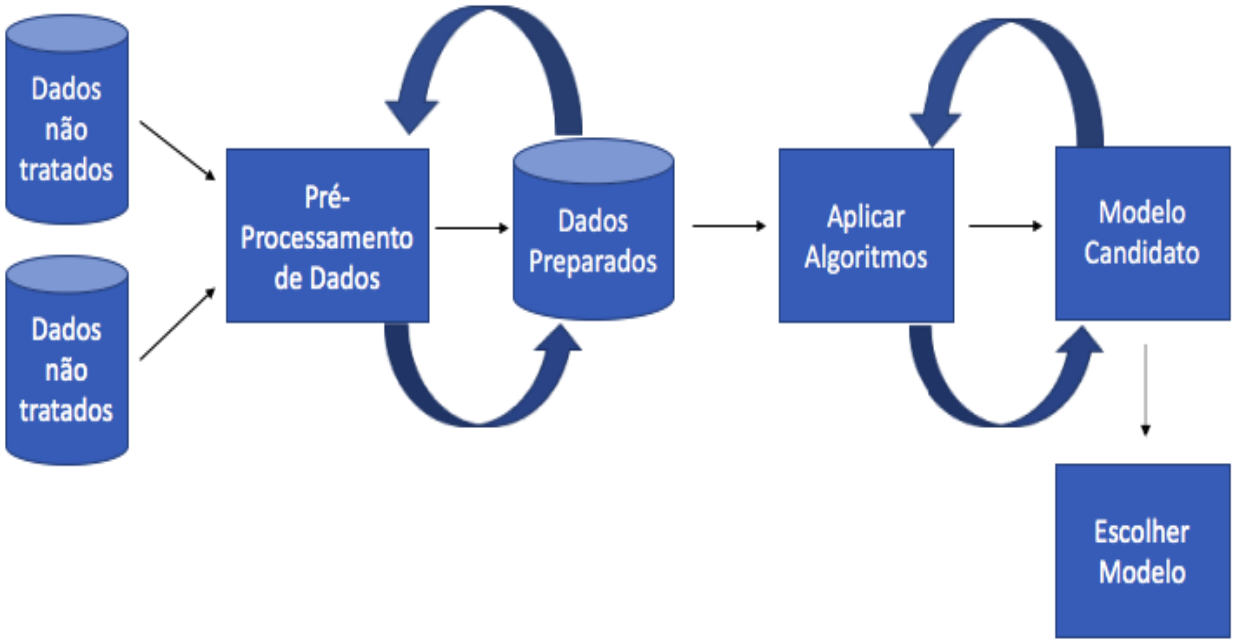


Figura 5.: Processo de ML

pretende resolver. As pessoas costumam cometer erros durante as análises de dados ou, provavelmente, quando tentam estabelecer relações entre as várias caraterísticas apresentadas. Desta forma, fica difícil encontrar soluções para certos problemas. As técnicas de ML conseguem várias vezes ser aplicada com sucesso aos problemas, melhorando a eficiência dos sistemas.

Cada instância na base de dados dos algoritmos de ML é representada usando o mesmo conjunto de caraterísticas. As variáveis podem ser caraterizadas em numéricas (continuas ou discretas) ou categóricas (ordinal ou nominal). Se as instâncias apresentam relações nas suas caraterísticas conhecidas (correspondem ao resultado final) então a aprendizagem é *supervisionada*, por outro lado, a *aprendizagem não supervisionada* apresenta dados em que as caraterísticas não estão identificadas, ou seja, não é conhecido nenhuma relação entre elas. Aplica-se algoritmos *não supervisionados* (*clustering*), com esperança para encontrar relações desconhecidas. Outro tipo de algoritmos de ML é a *aprendizagem por reforço*. A informação de treino providenciada ao sistema de aprendizagem constitui a medida de qualidade de funcionamento, atribuindo uma pontuação negativa ou positiva. A aprendizagem não é feita dizendo as ações a fazer, mas descobrir as ações que dão mais vantagens, através da experimentação de cada uma (Kotsiantis et al., 2007).

Na *aprendizagem não supervisionada*, a máquina simplesmente recebe os dados de entrada, mas não obtém informações de destino supervisionadas nem respostas exteriores. Pode parecer difícil imaginar a máquina a aprender, pois não recebe nenhum *feedback* do ambiente exterior. No entanto, é possível desenvolver uma estrutura formal para *aprendizagem não supervisionada* com base na noção de que o objetivo da máquina é construir representações de entrada que podem ser usadas para tomada de decisões, prever entradas futuras, comunicar eficientemente as entradas a outra máquina etc. De certo modo, a *aprendizagem não supervisionada* pode ser considerada como encontrar padrões nos dados que à primeira vista nem seriam considerados informação útil (Ghahramani, 2003).

No processo de desenvolvimento as técnicas de aprendizagem vão ser no âmbito da *aprendizagem não supervisionada*, tendo como objetivo extrair um ou mais padrões que permitam estabelecer relações entre os dados. Umas das técnicas muito utilizada neste contexto é *clustering*, os padrões dentro de um *cluster* escolhido são mais semelhantes entre si, em relação a um padrão pertencente de outro cluster. Pontos pertencentes ao mesmo *cluster* recebem o mesmo título. A variedade de técnicas para representar dados, medir a proximidade entre elementos de dados e agrupar elementos de dados faz com exista uma variedade enorme de métodos de agrupamento.

4.1.1 *Clustering*

É importante entender a diferença entre *clustering* (não supervisionada) e análise discriminante (supervisionada). Na classificação supervisionada, recebemos uma coleção de padrões atribuídos (pré-classificados). O problema é atribuir um padrão recém-encontrado, ainda sem descrição. Normalmente, os padrões marcados (de treino) são usados para aprender as descrições de classes que, por sua vez, são usadas para atribuir um novo padrão. No caso de *clustering*, o problema é agrupar uma determinada coleção de padrões não atribuídos em *clusters* significativos. Desta forma, as descrições também estão associadas aos *clusters*, mas essas descrições de categoria são orientadas por dados, isto é, elas são obtidas apenas dos dados.

O armazenamento em *cluster* é conveniente em várias situações de exploração de análise de padrões, agrupamento, tomada de decisão e técnicas de ML, incluindo mineração de dados, recuperação de documentos e classificação de padrões. No entanto, em muitos desses problemas, há pouca informação prévia (por exemplo, modelos estatísticos) disponível sobre os dados, e a tomada de decisão deve ter em conta o mínimo de suposições sobre os dados. É com base nessas restrições que a metodologia de agrupamento é particularmente apropriada para a exploração de relações entre as caraterísticas dos dados de forma a ter uma perceção da sua estrutura (Jain et al., 1999).

O processo de agrupamento de padrões normal envolve as seguintes etapas (Jain et al., 1988):

1. Representação de padrão (opcionalmente incluindo extração dos recursos e/ ou seleção);
2. Definição de uma medida de proximidade de padrão apropriada ao domínio dos dados;
3. *Clustering* ou agrupamento;
4. Abstração dos dados;
5. Avaliação do resultado;

A Figura 6 descreve uma sequência típica das três primeiras etapas, incluindo um ciclo *feedback* em que a saída do processo de agrupamento poderá afetar a extração de dados e os processos de similaridade.

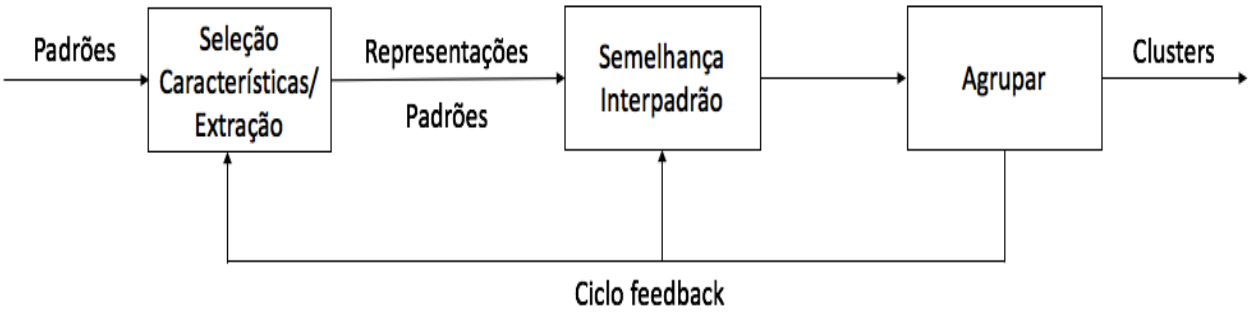


Figura 6.: Fases do *Clustering*

Representação de padrão refere-se ao número de classes, ao número de padrões disponíveis e ao número, tipo e escala dos recursos disponíveis para o algoritmo de agrupamento. Algumas dessas informações podem não ser controláveis. A seleção de dados é o processo de identificação do subconjunto mais eficaz dos dados originais a serem usados no armazenamento em cluster. Extração de dados é o uso de uma ou mais transformações dos dados de entrada para produzir novos dados. Uma ou ambas as técnicas podem ser usadas para obter um conjunto apropriado de dados para usar em *clustering*.

A proximidade do padrão é geralmente medida por uma função de distância definida através de pares de padrões. Uma variedade de medidas de distância é usada em várias comunidades. Uma medida simples de distância, como a distância euclidiana, pode frequentemente ser usada para refletir a diferença entre dois padrões, enquanto outras medidas

de similaridade podem ser usadas para caraterizar a semelhança conceitual entre padrões (Michalski and Stepp, 1983).

A etapa de agrupamento pode ser executada de várias maneiras. O agrupamento de saída pode ser difícil (uma partição dos dados em grupos) ou difusa (em que cada padrão possui um grau variável de associação em cada um dos *clusters* de saída). Algoritmos de *clustering* hierárquicos produzem uma série de partições com base num critério para dividir *clusters* com base nos padrões. Os algoritmos de *clustering* parciais identificam a partição que otimiza um critério de *clustering*. Podem ser utilizadas outras técnicas para a operação de agrupamento que incluem métodos de agrupamento probabilísticos (Brailovsky, 1991).

Abstração de dados é o processo de extrair uma representação simples e compacta de um conjunto de dados. Aqui, a simplicidade é da perspetiva da análise automática (para que uma máquina possa realizar um processamento adicional de forma eficiente) ou é orientada para o homem (para que a representação obtida seja fácil de compreender e intuitiva). No contexto de *clustering*, uma abstração de dados típica é uma descrição compacta de cada *cluster*, geralmente em termos de protótipos de cluster ou padrões representativos (Diday and Simon, 1976).

Todos os algoritmos de *clustering*, quando apresentados com dados, produzirão *clusters* - independentemente de os dados conterem *clusters* ou não. Se os dados contiverem *clusters*, alguns algoritmos podem obter *clusters* mais elaborados. A avaliação da saída de um procedimento de agrupamento, então, tem várias fases. Uma delas é, uma avaliação no domínio dos dados - os dados que não contêm possíveis *clusters* não devem ser processados por um algoritmo de *clustering*.

A análise de validade do *cluster* consiste numa avaliação dos resultados de um agrupamento. Muitas vezes, essa análise usa um critério específico, contudo esses critérios são geralmente subjetivos. Assim, existem poucos padrões determinantes no agrupamento, exceto em subdomínios bem definidos. As avaliações da validade são objetivas e são realizadas para determinar se o resultado é significativo. Uma estrutura de *clustering* é válida se não tiver ocorrido por acaso. Quando abordagens estatísticas para agrupamento são usadas, a validação é realizada aplicando cuidadosamente métodos estatísticos e testando hipóteses. Existem três tipos de estudos de validação. Uma validação externa do *cluster* compara a estrutura recuperada a uma estrutura a priori. Uma validação interna do *cluster* tenta determinar se a estrutura é intrinsecamente apropriada para os dados. Uma validação relativa do *cluster* compara duas estruturas e mede o seu mérito relativo. Desta forma conseguimos garantir um resultado é de qualidade, não comprometendo o trabalho realizado (Kassambara, 2017).

4.1.2 *Árvore de Decisão*

A árvore de decisão é um sistema de suporte à decisão que usa decisões de grafos semelhantes a uma árvore e os seus possíveis efeitos posteriores, como resultados de eventos aleatórios, custos de recursos e utilidade. O método consiste em aprender uma função de classificação que obtém o valor de um atributo dependente (variável) dados os valores dos atributos (variáveis) independentes (Bhargava et al., 2013). Este tipo de modelo é usado frequentemente em áreas como as finanças, marketing, engenharia e medicina.

Podemos descrever as árvores de decisão como diagramas capazes de enumerar todas as possibilidades lógicas de uma sequência de decisões e ocorrências incertas. Elas mostram esquematicamente todo o conjunto de ações alternativas e acontecimentos possíveis ao longo de um projeto.

Este algoritmo consiste em encontrar o melhor atributo de um conjunto de dados e colocá-lo na raiz da árvore, isto é, a primeira tomada de decisão do algoritmo, de forma a ter um impacto maior. De seguida, separar os conjuntos de treino em subconjuntos que devem ser feitos de maneira a que cada subconjunto contenha dados com o mesmo valor para o atributo. Repetir o processo descrito anteriormente até encontrar um nodo em todos os ramos da árvore.

Quando uma árvore de decisão é usada para tarefas de classificação, é mais apropriado referir como árvore de classificação. Sendo que quando é usada para tarefas de regressão, deve ser árvore de regressão. Neste projeto vamos usar árvore de classificação, como o nome indica, para classificar um objeto ou uma instância (exemplo jogar ténis) de um conjunto de classes predefinido (sim/não) com base nos seus atributos (humidade e vento), como podemos observar na figura 7.

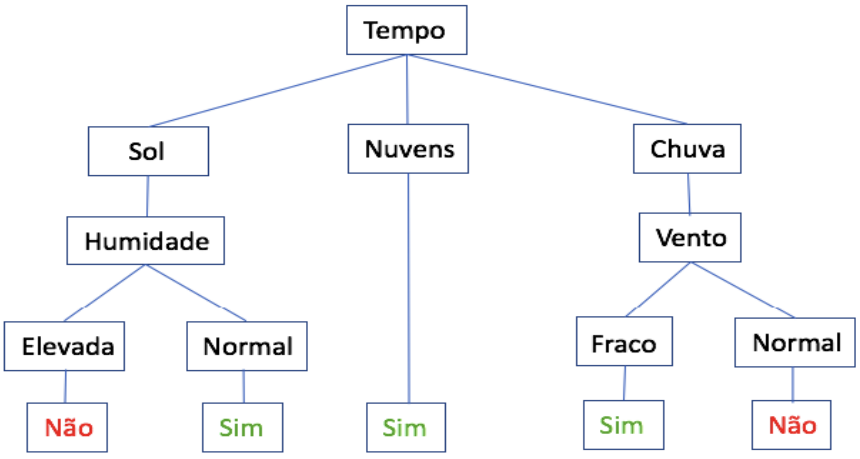


Figura 7.: Exemplo de uma Árvore de Classificação

Esta abordagem apresenta as seguintes vantagens defendidas por Freund e Manson (Freund and Mason, 1999):

- É fácil de entender pelo o utilizador final;
- Consegue assegurar uma variedade de dados à entrada (nominal, numérico e texto);
- Capaz de processar vários conjuntos de dados errados ou valores ausentes;
- Alto desempenho com pequeno esforço computacional;

4.1.3 *Floresta Aleatória*

Floresta Aleatória (*Random Forest*) é uma combinação de árvores de decisão, de tal forma que cada árvore depende dos valores de um vetor aleatório de forma independente e com a mesma distribuição para todas as árvores na floresta. O erro de generalização para florestas converge a um limite quando o número de árvores na floresta se torna grande. O erro de generalização de uma floresta de classificadores de árvores depende da força das árvores individuais na floresta e da correlação entre elas. As estimativas internas monitoram o erro, a força e a correlação e são usadas para mostrar a resposta ao aumento do número de recursos usados na divisão. Estimativas internas também são usadas para medir a importância da variável (Breiman, 2001).

Uma grande vantagem do algoritmo de florestas aleatórias é que ele pode ser utilizado tanto para tarefas de classificação ou para tarefas de regressão, o que representa a maioria dos sistemas com algoritmos de ML atualmente. A floresta aleatória de classificação é a que vamos usar no âmbito deste trabalho, uma vez que também é um dos algoritmos de ML mais conhecidos. Podemos observar no seguinte exemplo uma floresta aleatória na figura 8.

A floresta aleatória ajusta muitas árvores de classificação a um conjunto de dados e combina as previsões de todas as árvores. O algoritmo começa com a seleção de muitas amostras de *bootstrap*, é uma técnica de reamostragem estatística que envolve amostragem aleatória de um conjunto de dados com substituição. É frequentemente usado como um meio de quantificar a incerteza associada a um modelo de ML, dos dados. Numa amostra típica de *bootstrap*, aproximadamente 63% das observações originais ocorrem pelo menos uma vez. Observações no conjunto de dados original que não ocorrem numa amostra de *bootstrap* são chamadas observações *out-of-bag*, método de medir o erro de previsão. Uma árvore de classificação é adequada para cada amostra de *bootstrap*, mas em cada nó, apenas um pequeno número de variáveis selecionadas aleatoriamente está disponível. As árvores são totalmente criadas e cada uma é usada para prever as observações *out-of-bag*. A classe prevista de uma observação é calculada pelo voto maioritário das previsões *out-of-bag* para aquela observação (Cutler et al., 2007).

De uma forma simples podemos descrever o processo do algoritmo da floresta aleatória, como sendo a criação de várias árvores de decisão e combiná-las para obter um resultado com maior precisão e estabilidade.

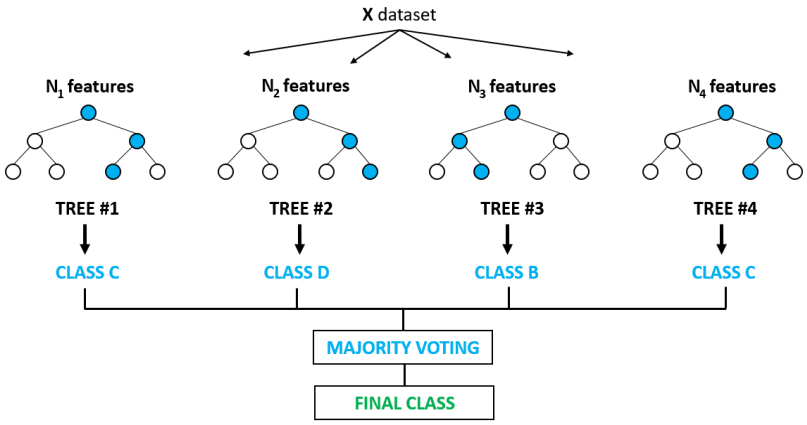


Figura 8.: Etapas da Floresta Aleatória[1]

As principais vantagens apresentadas pela floresta aleatória são:

- Natureza não paramétrica;
- Alta precisão de classificação;
- Capacidade para determinar a importância da variável;
- Fornece um algoritmo para estimar valores em falta;
- Flexibilidade para realizar vários tipos de análise de dados (Rodriguez-Galiano et al., 2012);

4.1.4 *Regressão Logística*

Antes de saber o que é regressão logística, é importante abordar a definição de regressão. Regressão é um método de modelagem de um valor alvo baseado em preditores independentes. Este método é usado principalmente para previsão e descobrir relações de causa e efeito entre variáveis. As técnicas de regressão diferem principalmente com base no número de variáveis independentes e no tipo de relação entre as variáveis independentes e dependentes.

A regressão logística é um algoritmo de classificação usado para atribuir observações a um conjunto discreto de classes. Alguns dos exemplos de problemas de classificação são

1 https://www.globalsoftwaresupport.com/random-forest-classifier-bagging-machine-learning/

spam de e-mail, fraudes online, tumor maligno ou benigno. A regressão logística usa a função sigmóide, o nome vem da forma do gráfico que apresenta um "S", logística para retornar um valor de probabilidade.

As principais diferenças que distinguem os dois tipos de regressão relacionam-se com o facto de as previsões de regressão linear serem contínuas (números num intervalo). As previsões de regressão logística serem discretas (somente valores específicos ou categorias são permitidas). A regressão logística divide-se em dois tipos, a binária (2 hipóteses) e a múltipla.

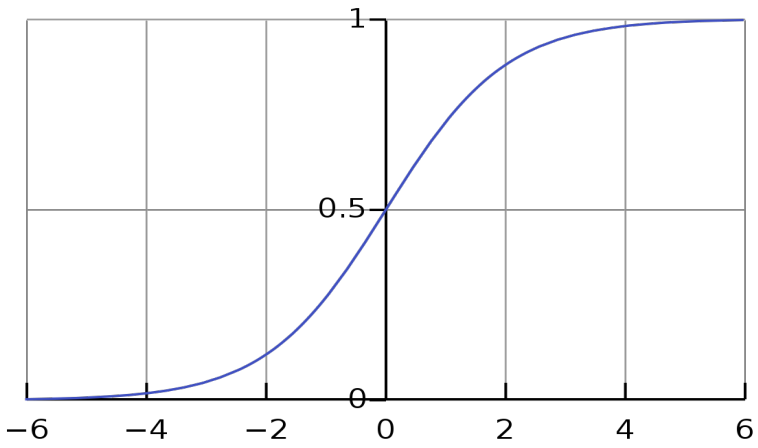


Figura 9.: Gráfico da Função Sigmóide [2]

A função sigmóide permite mapear os valores previstos das probabilidades. A função mapeia qualquer valor real noutro valor entre 0 e 1. É uma função matemática de amplo uso em campos como a economia e a computação. Até há pouco tempo atrás, a função sigmóide era a mais utilizada em ARN (ácido ribonucleico), por ser biologicamente mais plausível. Como os neurónios biológicos funcionam de forma binária (ativado ou não ativado), a função sigmoide é uma boa forma de modelar esse comportamento, já que assume valores apenas entre 0 (não ativação) e 1 (ativação).

A Regressão Logística mede a relação entre a variável dependente e uma ou mais variáveis independentes (caraterísticas), estimando as probabilidades utilizando a sua função logística subjacente. Essas probabilidades devem então ser transformadas em valores binários para se fazer uma previsão. Esta é a tarefa da função logística, também chamada de função sigmóide. Esta função pode selecionar qualquer número real e mapeá-lo num valor entre o intervalo de 0 e 1, mas nunca exatamente nesses limites. Esses valores entre 0 e 1 serão então transformados em 0 ou 1 usando um classificador de limite. Na figura 10 está apresentado este processo, de forma a agregar as etapas descritas anteriormente.

2 https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36
3 https://machinelearning-blog.com/2018/04/23/logistic-regression-101/

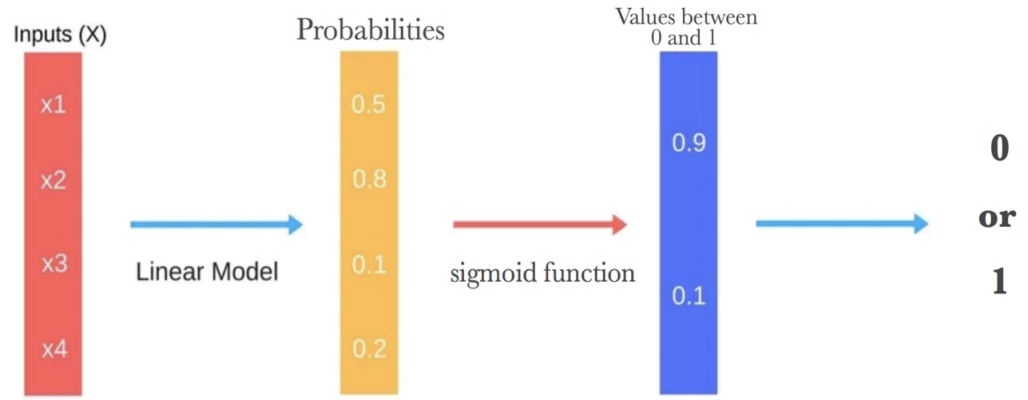


Figura 10.: Fases da Regressão Logística [3].

4.1.5 *Support Vector Machine*

Inicialmente, *support vector machine* (SVM) foi criado para resolver problemas de reconhecimento de padrões. Neste algoritmo ocorre um mapeamento dos dados para um plano dimensional mais elevado e cria-se um hiperplano de separação neste espaço. Este processo de uma forma simples, permite resolver um problema de programação quadrática, enquanto que os métodos de ensinamentos baseados em gradientes são para redes neurais, contudo, sofrem com o facto de existirem muitos mínimos locais.

Os parâmetros e as funções do *kernel* são escolhidos de maneira a que um limite na dimensão seja minimizado. Com o passar do tempo, o método do SVM foi acrescentando novas funções de forma a conseguir resolver problemas de estimativa de funções. Para tal ser possível, a função de perda de *Vapnik* e a função de perda de *Huber* foram adicionadas. Baseando-se no princípio de minimização de risco estrutural e no conceito de capacidade com definições combinatórias puras, a qualidade e a complexidade da solução SVM não dependem diretamente da dimensionalidade do espaço de entrada (Suykens and Vandewalle, 1999).

A ideia principal deste algoritmo é encontrar o hiperplano otimizado, que separa duas classes à maior distância possível. A margem é a distância entre o hiperplano e o ponto de dados mais próximo, vetor de suporte, de forma a maximizar a margem entre duas classes. No conjunto de dados, seleciona-se os dois hiperplanos que separam os dados sem pontos entre eles e maximizam a distância entre esses dois hiperplanos. Como podemos observar na figura 11 um exemplo de uma SVM.

4 https://en.proft.me/2014/04/22/how-simulate-support-vector-machine-svm-r/

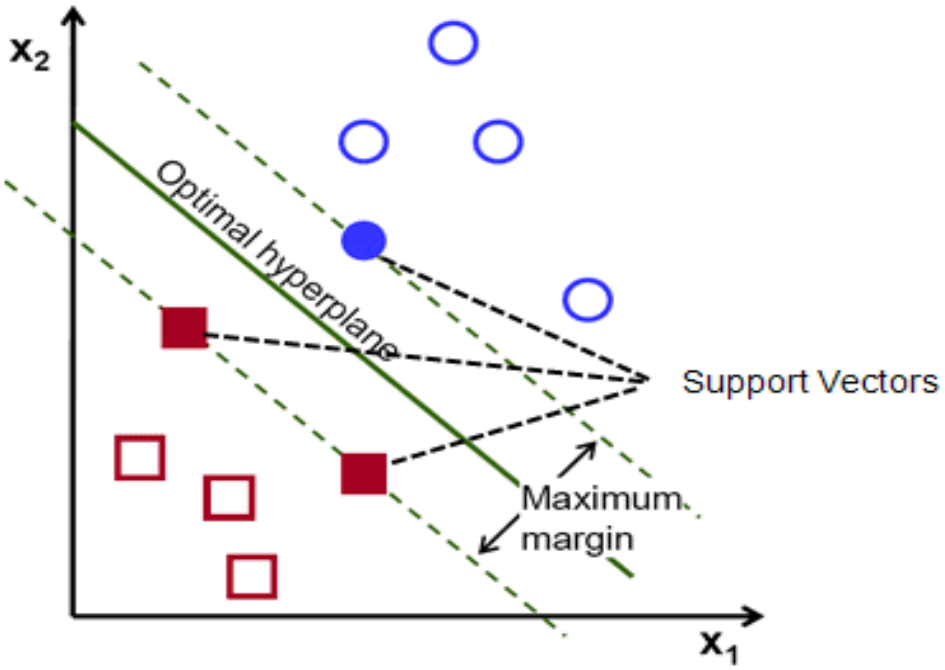


Figura 11.: Processo de decisão do *Support Vector Machine* [4]

Treinar uma SVM requer resolver um problema de programação quadrática (PQ) num número de coeficientes igual ao número de exemplos de treino. Para conjuntos de dados muito grandes, técnicas numéricas de padrão para PQ tornam-se inviáveis. Técnicas práticas decompõem o problema em subproblemas de forma a permitir melhor desempenho sobre parte dos dados ou, no limite, realizam otimização iterativa por pares ou componentes. Uma desvantagem dessas técnicas é que elas podem fornecer uma solução aproximada e podem exigir muitas passagens pelo conjunto de dados para alcançar um nível razoável de convergência (Cauwenberghs and Poggio, 2001).

Um benefício importante da abordagem do SVM é que a complexidade do classificador resultante é caraterizada pelo número de vetores de suporte, em vez da dimensionalidade do espaço transformado. Como resultado, as máquinas de vetores de suporte tendem a ser menos propensas a ter problemas de ajuste excessivo em comparação com outros métodos.

Há três vantagens principais: em primeiro lugar, tem um parâmetro de regularização, que faz com que o usuário pense em evitar o ajuste excessivo. Em segundo lugar, usa o truque do *kernel*, para que possa desenvolver um conhecimento especializado sobre o problema por meio da tecnologia do *kernel*. Por último, um SVM é definido por um problema de otimização convexa (sem mínimos locais) para o qual existem métodos eficientes.

As desvantagens são que a teoria só cobre a determinação dos parâmetros para um dado valor da regularização e parâmetros do *kernel* e a respetiva escolha do *kernel*. De certa forma,

o SVM move o problema do ajuste excessivo da otimização dos parâmetros para a seleção do modelo. Infelizmente, os modelos do *kernel* podem ser bastante sensíveis em relação ao ajuste excessivo do critério de seleção do modelo.

4.1.6 *K-Vizinhos mais próximos*

Suponhamos que temos um conjunto de dados todos eles devidamente classificados. Este conjunto será usado como o conjunto de treino. Quando aparece um novo registo, sem classificação, vamos comparar este registo com todos os registos do conjunto de treino. No fim da comparação, escolhemos os k-vizinhos mais próximos e observamos a sua classificação. Aqui, k é um número inteiro, normalmente menor do que 20. Deste conjunto extraído vemos qual a classificação que mais se repete e atribuímos essa classificação ao novo registo.

Este algoritmo é baseado na semelhança dos elementos: como elementos quase fora da amostra assemelham-se ao nosso conjunto de treino determina como classificamos um determinado conjunto de dados: K-Vizinhos pode ser usado para classificação - a saída é uma associação da classe (prevê uma classe - um valor discreto). Um objeto é classificado por uma votação maioritária dos vizinhos, com o elemento a ser atribuído à classe mais comum entre seus k vizinhos mais próximos. Também pode ser usado para regressão - o valor de saída é para o objeto (prevê valores contínuos). Este valor é a média (ou mediana) dos valores de seus k vizinhos mais próximos.

De uma forma mais simples, o processo consiste num inteiro positivo k que é especificado, juntamente com uma nova amostra. De seguida selecionamos as entradas k que temos armazenadas que estão mais próximas da nova amostra. Encontramos a classificação mais comum dessas entradas e por fim, essa é a classificação que atribuímos à nova amostra.

Na figura 12 observamos que a instância de treino pertence ao K=3, resta agora saber qual a classe que vai ser atribuída. Através da imagem vemos que existem dois elementos da classe 2 e apenas um elemento da classe 1, sendo a classe 2 maioritária, atribuímos esta classe à instância de treino. Caso a instância de treino estivesse no K=5, já seria da classe 1 por haver mais elementos desta classe em relação à classe 2.

De seguida são apresentadas as principais vantagens da utilização deste algoritmo:

- Não toma pressupostos sobre os dados, é útil para dados não lineares;
- Algoritmo simples, torna-se fácil de explicar e interpretar;
- Elevada precisão relativa, porque não chega a ser competitivo com outros algoritmos;
- Versátil, é utilizado para classificação e regressão;

Por outro lado, este algoritmo também apresenta algumas debilidades, tais como:

5 https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

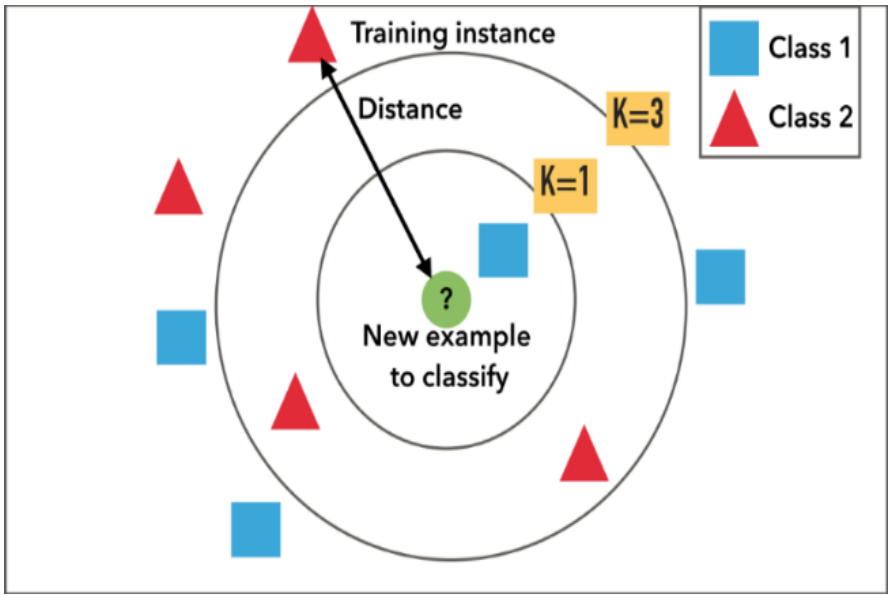


Figura 12.: Demonstração do k-Vizinhos mais próximos [5].

- Dispendioso computacionalmente, o algoritmo guarda todos os dados de treino;
- Requer um número elevado de memória;
- Previsões demoraram algum tempo, em comparação com outros algoritmos;
- Sensível a caraterísticas irrelevantes e à escala dos dados;

4.2 DISCUSSÃO E ANÁLISE

As técnicas de ML procuram ajudar a resolver problemas simples, do quotidiano como problemas complexos que necessitam da ajuda do computador. Tarefas simples relacionas com a conhecida expressão "aprender com a experiência", por exemplo a robótica, condução, perceber imagens, até tarefas mais elaboradas como decisões médicas com base nos diagnósticos e previsões da meteorologia. Outra característica fundamental, é a sua adaptabilidade, que permite ao programa adaptar-se com relativa facilidade com o ambiente que está a interagir.

O processo de ML engloba várias fases de processamento, o primeiro passo é a recolha dos dados que varia de acordo com a questão a ser tratada e o ambiente a interagir. Após a recolha, os dados têm de ser tratados para uniformizar a informação recolhida e certificar que satisfazem os requisitos para as fases seguintes. Com a base de dados preparada, é preciso aplicar os algortimos aos dados tratados e ver qual é que apresenta o resultado mais satisfatório, de forma a ser o modelo de eleição e o mais indicado para a questão em

causa. No momento da decisão é preciso ter em conta outros aspetos, como a escalabilidade e o tempo de resposta, para garantir um bom desempenho do programa.

Por fim, o estudo dos algoritmos, uma das partes mais importantes neste projeto, visto que desenvolvem um papel fundamental na previsão dos níveis de *stress* do utilizador. Apesar da maioria dos algoritmos serem supervisionados, isto é, existe um conhecimento dos dados que temos e qual a variável que queremos prever, o mesmo não acontece nos não supervisionados, como é o caso do *clustering*, que agrupa os dados com base nas semelhanças apresentadas. Apesar deste algoritmo se destacar dos restantes, é provável que o resultado seja menos satisfatório porque os dados não apresentam grande diferenciação entre eles.

5

IMPLEMENTAÇÃO DA PLATAFORMA

Neste capítulo é descrito o conjunto das diferentes fases para o desenvolvimento do programa, para ser possível traçar uma relação do *stress* com as diferentes ações do utilizador de uma forma simples e rápida. Assim como, uma explicação do aparecimento desta ideia, as diferentes bibliotecas de *Python* usadas, as suas limitações e vantagens, e os diferentes tipos de abordagens usados consoante as dificuldades no processo de desenvolvimento.

Este desenvolvimento pode-se dividir em duas partes importantes, uma primeira de recolha e tratamento dos dados e uma final que recai sobre o uso dos dados para aplicar os algoritmos, procurando estabelecer uma relação. Os próximos subcapítulos correspondem a recolha de dados, o processo adotado e as ferramentas utilizadas, o armazenamento destes dados, qual a estrutura escolhida e as respetivas métricas a guardar.

Por fim, aplicar os algoritmos aos dados armazenados e fazer a respetiva avaliação crítica dos resultados obtidos.

5.1 TECNOLOGIAS E FERRAMENTAS

A ideia de criar este programa é estudar de uma forma não intrusiva, a relação dos resultados obtidos nas provas pelos alunos no âmbito académico com o nível de *stress* apresentado durante a realização dos mesmos. Para desenvolver o código, foi utilizado a IDE do *Python*, *PyCharm*, permitindo assim otimizar o tempo de desenvolvimento. Tem uma enorme variedade de *plugins* e permite personalizar o editor de texto, ficando do agrado do utilizador.

Para perceber melhor o potencial desta ferramenta, são apresentadas de seguida algumas das caraterísticas de maior importância que auxiliam o utilizador no desenvolvimento com recurso a liguagem *Python*, (Islam, 2015):

- **Editor de código Inteligente**: *Pycharm* fornece suporte para diferentes linguagens populares, como *Javascript*, *CSS*, *Typescript* entre outros. Tem reconhecimento de idioma através da conclusão de código, deteção de erros e correções de código em tempo real;

- **Navegação de código Inteligente**: permite navegar para qualquer classe, arquivo ou símbolo, inclusive qualquer ação IDE ou janela de ferramentas, através da pesquisa inteligente;
- **Refatoração**: Através de renomear e excluir de forma segura, extração de métodos, introduzir varáveis e variáveis embutidas, entre outros métodos de refatoração. As refatorações específicas de linguagem e estrutura ajudam a executar alterações em todo o projeto;
- **Consola Interativa *Python***: Executar uma consola REPL(*read-eval-print loop*) *Python*, que oferece muitas vantagens sobre a consola normal, como a verificação de sintaxe em tempo real com inspeções, verifica a correspondência das chavetas e pontos e ajuda a completar código por terminar;
- **Integração CONDA**: Permite manter as dependências isoladas ao ter diferentes ambientes CONDA (gere os pacotes, dependências e ambiente) para cada projeto, o *PyCharm* facilita a criação e a seleção do ambiente indicado;
- **Suporte Científico**: O *PyCharm* possui suporte inter para bibliotecas científicas. Oferece um conjuntos de ferramentas como gráficos, visualizador de matrizes, entre outros. Esta funcionalidade só é suportada na versão paga.

5.2 METODOLOGIA DE DESENVOLVIMENTO

A metodologia adotada para o desenvolvimento deste projeto foi a ágil, conhecido como *agile*, por ser mais adaptada às mudanças, pois não há um planeamento muito detalhado no início do projeto, apesar de haver requisitos estabelecidos que ficam sujeitos a mudanças durante o processo de desenvolvimento. Este tipo de planeamento opta por uma abordagem iterativa e incremental. A prioridade das tarefas a realizar é determinada consoante as preferências mencionadas pelas partes interessadas.

Dentro da metodologia ágil existe um enorme número de forma de desenvolver um projeto, contudo as ideologias da metodologia *Scrum* foram de encontro às preferências traçadas, estabelecendo *sprints* de uma semana, isto é, a cada 7 dias reuníamos para avaliar o trabalho desenvolvido e estabelecer as próximas tarefas a realizar. O trabalho elaborado estava sujeito a mudanças consoante as alterações necessárias para realizar os objetivos.

Na figura 13 é representado um esquema da abordagem adotada para o desenvolvimento do projeto. Podemos observar um conjunto de tarefas a terem de ser realizadas, e desse conjunto eram selecionadas as com maior prioridade. Depois das tarefas serem realizadas num espaço temporal de 7 dias, eram adicionadas ao projeto. Este processo de desenvolvi-

mento era repetido até todas as tarefas estarem concluídas e consequentemente, o projeto finalizado.

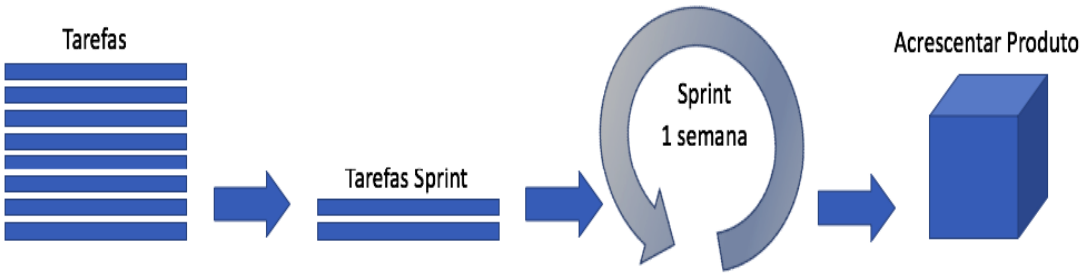


Figura 13.: Metodologia *Agile Scrum* utilizada no projeto

5.3 FASES DE DESENVOLVIMENTO

Este programa é dividido em diferentes fases de execução. De forma a entender melhor o seu funcionamento, irei abordar cada fase separadamente e explicar a função de cada uma no desenvolvimento do programa.

As diferentes fases de todo o processo de desenvolvimento são ilustradas nas figuras 14, 16 e 17, desde a fase da recolha, onde são guardadas as informações respetivas à utilização do rato, do teclado e do nível de *stress* por parte do utilizador, até à fase dos algoritmos, onde os dados depois de tratados na fase de tratamento, ficam prontos para serem utilizados nos algoritmos de forma a testar os que melhor se adequam ao caso de estudo. Todas estas fases vão ser abordadas detalhadamente nas subsecções seguintes.

5.3.1 *Fase de Recolha*

Nesta fase os dados são recolhidos do utilizador através do registo da monitorização do uso do rato e do teclado. Inicia-se o programa e este cria subprocessos para fazer a recolha de todas as ações executadas. Com o auxílio da biblioteca de *python pynput* é possível criar uma *script* de monitorização.

A *script* responsável pelo controlo do rato armazena as movimentações realizadas num ficheiro de texto, identificando as coordenadas da sua localização, o X e Y, e sempre que houver um clique, tanto com o botão direito ou esquerdo. Todas estas informações são armazenadas com o respetivo *timestamp*. Este ficheiro de texto armazena os dados com o seguinte formato:

```
2019-10-01 10:26:52,185: Mouse moved to (1011.515625, 420.2265625)
```

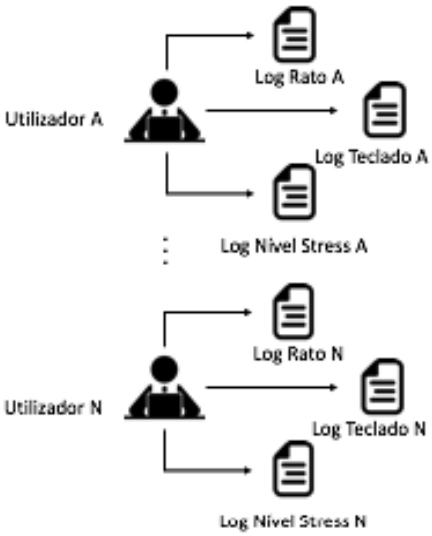


Figura 14.: Fase de Recolha

```
2019-10-01 10:26:52,635: Mouse clicked at (1011.515625, 420.2265625) with
    Button.left
```

Para a recolha das ações do teclado, sempre que o utilizador pressionar uma tecla esta fica registada, com o respetivo *timestamp*. Estes registos ficam guardados num ficheiro de texto com o seguinte formato:

```
2019-10-01 10:26:39,320: 'b'
2019-10-01 10:26:53,534: Key.shift
```

De forma a armazenar o nível de *stress*, assim que se inicia o programa, aparece uma interface para o utilizador selecionar o seu nível de *stress* numa de escala de 1 (Nenhum) a 5 (Muito). Esta interface aparece de X em X tempo, sendo o X uma variável predefinida conforme o intervalo de tempo estabelecido. Para a realização o intervalo de tempo considerado foi de 1 minuto. A seguinte figura 15 mostra a interface de classificação do nível de *stress*, permitindo ao utilizador selecionar o respetivo nível que se enquadra.

Após a realização da avaliação do nível de *stress*, a informação é guardada num ficheiro de texto, onde ficam armazenadas todos as avaliações realizadas pelo utilizador. O seguinte formato é referente à forma de armazenamento do *stress* com o respetivo *timestamp*

```
2019-10-01 17:41:45,265: 2
```

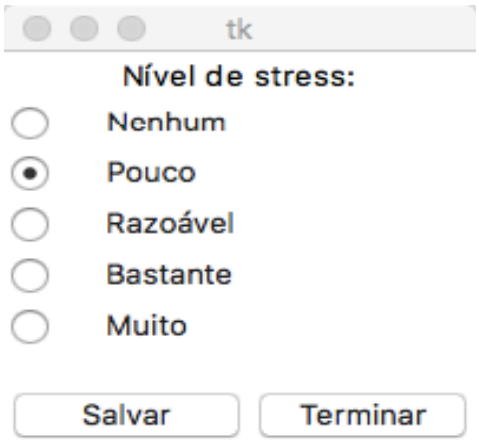


Figura 15.: Interface da Avaliação do Nível de *Stress*

5.3.2 *Fase de Tratamento dos Dados*

Nesta fase, os dados recolhidos são usados para calcular as métricas definidas do rato e do teclado. O objetivo é utilizar a informação guardada e gerar informação pertinente para estabelecer relações com o nível de *stress*.

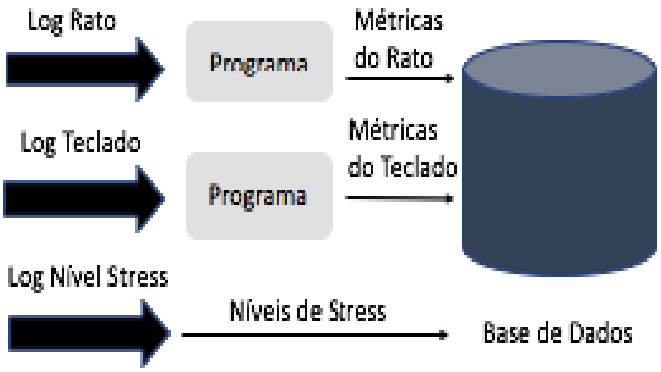


Figura 16.: Fase de Tratamento dos Dados

A *script* vai buscar os dados guardados com recurso das expressões regulares, de forma a agrupar os dados que pertencem ao mesmo campo, por exemplo as coordenadas do eixo *X* ocupam a mesma posição em cada *array*, para facilitar os cálculos a efetuar. Algumas das bibliotecas foram usadas, tais como:

1. *math*: de forma a permitir determinar os ângulos formados pelo rato, a velocidade e transformar de radianos para graus, recorrendo a funções como *math.sqrt*, *math.tan*, *math.fabs* e *math.degrees*;

2. *re*: permitir capturar os diferentes campos do ficheiro de texto, recorrendo a expressões regulares. Como por exemplo: *\d+[-]\d+[-]\d+|\d+[:]\d+[:]\d+[,]\d+* para extrair a parte referente ao *datetime* do ficheiro de texto;

3. *datetime*: conseguir converter o tempo exato de captura em segundos ou minutos consoante o intervalo de tempo estabelecido. Desta maneira fica possível agrupar os dados e determinar as métricas definidas.

Por fim, as métricas calculadas são guardadas na base de dados, *mongoDB*, de forma a serem usadas na próxima fase.

5.3.3 *Fase dos Algoritmos*

A última etapa do programa e mais complexa, que consiste na análise dos resultados obtidos com base nos algoritmos utilizados. Os resultados são comparados, de forma a observar o que tem um resultado mais satisfatório na previsão do nível de *stress*.

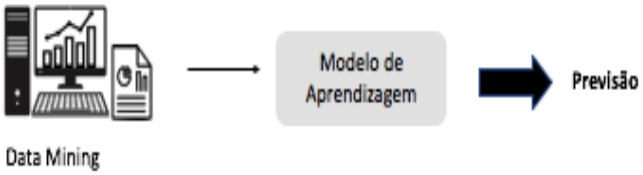


Figura 17.: Fase dos Algoritmos

O primeiro passo é transferir os dados para um ficheiro onde seja possível aceder a partir do *spyder*, o ambiente de desenvolvimento para programação científica na linguagem *Python*. De forma a proceder à extração da base de dados *mongodb* utilizou-se o seguinte comando:

```
mongoexport --db DB --collection AllMetrics --type=csv --fields velocity,
    logCount,distDiff,clickDuration,clickTotal,backSpace,stressLevel --out
    ~/Desktop/CSV/allMetrics.csv
```

Neste comando podemos observar os diferentes campos a serem preenchidos:

1. *DB* - Este campo é referente ao nome da base de dados;
2. *AllMetrics* - Este campo indica o nome da coleção onde as métricas ficaram armazenadas;
3. *Métricas* - Selecionar os campos pertinentes, neste caso foram as métricas calculadas anteriormente;
4. *Diretoria final* - Local onde vai ser guardado o ficheiro final com extensão *csv*.

De seguida, foi preciso carregar o ficheiro para a plataforma de desenvolvimento, para preparar o *dataset* de forma a ser utilizado nos algoritmos pretendidos. O objetivo foi construir um modelo de ML que, ao fornecer um conjunto de dados, este indicasse corretamente o respetivo nível de *stress* a que os dados pertencem. Durante este processo foi observado uma distribuição não equitativa do número de dados referentes aos níveis de *stress*. Na seguinte tabela apresentam-se as percentagens de amostras atribuídas a cada nível respetivamente, num total de 156 amostras recolhidas, de 6 indivíduos durante a realização de um teste com duração de 26 minutos.

Nível de Stress	Amostras	Percentagem
1	41	26%
2	50	32%
3	32	20%
4	12	8%
5	21	14%

Figura 18.: Número de amostras referentes ao nível de *stress*

Como resultado, um número de técnicas de pré processamento foram necessárias para resolver o balanceamento do *dataset*, permitindo análises mais apropriadas dos casos de estudo.

Oversampling foi uma das técnicas adotas para ajustar a distribuição da base de dados. O objetivo principal é ajustar a discrepância encontrada no conjunto de dados inicial. De maneira a equilibrar a base de dados, este método replica aleatoriamente observações das classes em minoria (todas exceto a do nível de *stress* 3) o que leva a um número igual

de dados em cada nível de *stress*. O método aplicado foi o ROSE (*Random Over-Sampling Examples*), que ajuda a atenuar os efeitos de uma distribuição desequilibrada das classes, auxiliando as fases de estimativa e avaliação do modelo. São geradas amostras balanceadas artificias de acordo com uma abordagem *bootstrap* (técnica de reamostragem usada para estimar estatísticas numa população, sugerindo um conjunto de dados com substituição) e permite auxiliar ambas as fases de estimativa e avaliação da precisão de um classificador binário na presença de uma classe fraca.

O método SMOTE (*Synthetic Minority Oversampling Technique*) é um algoritmo que cria amostras de dados artificiais com base nas semelhanças entre as amostras minoritárias. Por outras palavras, através das amostras da classe minoritária, introduz exemplos artificiais ao longo dos segmentos de linha que juntam as classes minoritárias, 'k' vizinhos mais próximos. Dependendo da quantidade da amostra necessária, os vizinhos dos 'k' vizinhos mais próximos são escolhidos aleatoriamente (Chawla et al., 2002).

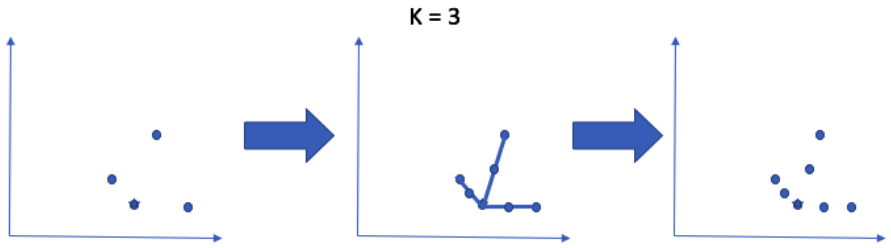


Figura 19.: Diferentes fases do Método *SMOTE*

Através destes métodos de balanceamento de base de dados apresentados conseguimos equilibrar a distribuição entre as classes e assegurar um resultado final mais preciso. A escolha de testar com dois métodos diferentes, tem como objetivo, avaliar os diferentes resultados finais e ver qual o método que melhor se adequada ao caso de estudo. Como maior parte dos dados são variáveis com as respetivas medições, isto é, não estão uniformizadas à mesma escala, o próximo passo foi normalizar os conjuntos de dados. Isso foi alcançado recorrendo ao método de escala *min-max*.

A escala *Min-Max* é um processo que converge todos os valores das variáveis para um intervalo fixo. Este método é muito sensível à presença de outliers (Al Shalabi et al., 2006). Mantém a distribuição original do conjunto de dados, exceto o fator de escala, e transforma todas as variáveis num intervalo comum entre 0 e 1. Os valores de distância podem ser transformados em valores de similaridade subtraindo a pontuação normalizada *min-max* de 1.

Esta abordagem de normalização foi realizada no conjunto de dados obtidos anteriormente, resultante em dois grupos de dados, os normalizados através da escala *Min-Max* e os originais da base de dados. O desempenho foi observado através do uso de técnicas

de ML (Árvore de Decisão, *Clustering*, Floresta Aleatória, K-vizinhos Mais Próximos, Regressão Logística e *Support Vector Machine*) e calcular a média dos resultados obtidos das diferentes formas de balanceamento da base de dados, mostrando que a aplicação da escala *Min-Max* teve um aumento ligeiro de 4,03%, quando comparado com a base de dados original.

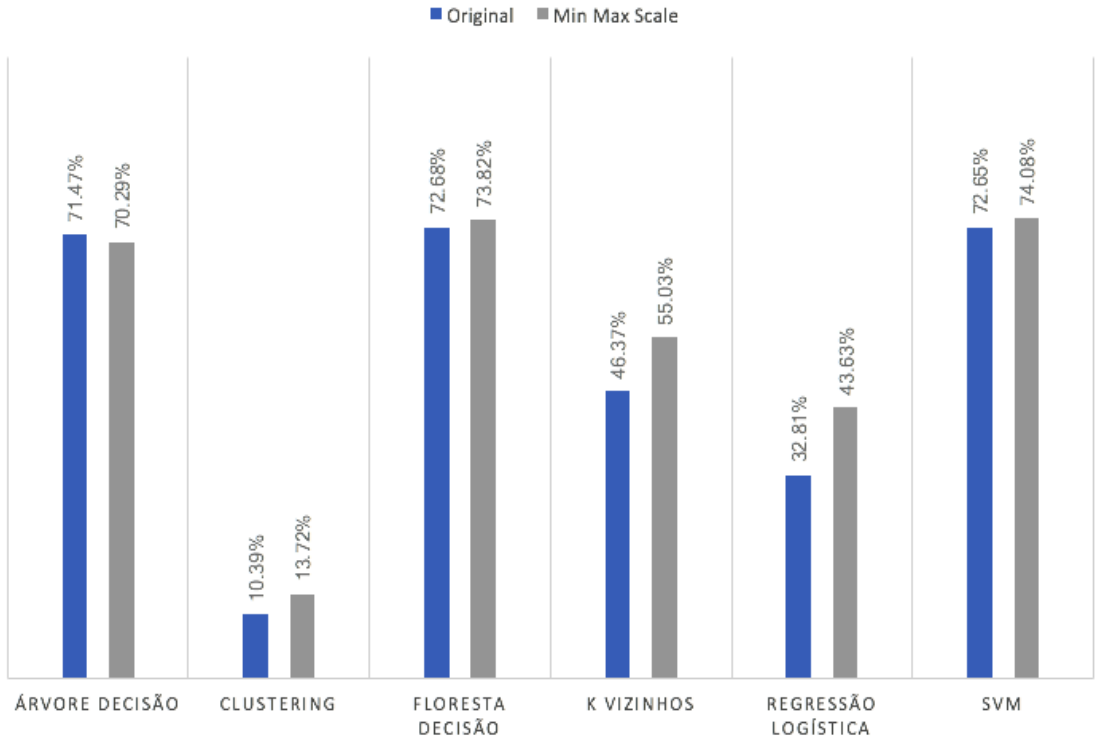


Figura 20.: Comparação do desempenho Base Original vs *MinMax Scale*

Durante a seleção das métricas a definir para o caso de estudo, verificou-se na parte final do desenvolvimento, que existia correlação entre certas variáveis, e como tal, o resultado final da precisão do algoritmo estava influenciado. Foi preciso redefinir as métricas a serem alvo de estudo, e analisar a sua dependência. De forma a garantir uma seleção correta das variáveis de maior importância, aplicou-se o coeficiente de correlação de *Pearson* que determina a correlação entre duas variáveis. As métricas escolhidas para o caso de estudo foram velocidade e ângulos do rato, diferença entre a distância real e em linha reta de cliques consecutivos, total de cliques e respetiva duração. Em relação ao teclado temos o número de vezes que o *backspace* foi usado, a utilização do lado esquerdo e direito do teclado. Por fim, a variável que indica o nível de *stress* resultante da avaliação do próprio utilizador.

5.4 TESTES E RESULTADOS

Após a conclusão dos processos de preparação de dados e seleção das métricas, a fase seguinte foi: desenvolvimento e seleção das técnicas de ML mais apropriadas para prever os níveis de *stress* de um indivíduo que realize um teste proposto. A análise dos dados começou com a seleção de um conjunto de técnicas de ML que melhor se adequam ao caso de estudo. As técnicas selecionadas foram: Árvore de Decisão, *Clustering*, Floresta Aleatória, K-Vizinhos Mais Próximos, Regressão Logística e SVM. Esses modelos foram treinados através do conjunto de dados preparado nas etapas anteriores. Além disso, para fornecer um resultado mais fiável, dois métodos diferentes foram usados para treinar / testar os modelos de ML: dividir o conjunto de dados em subconjuntos de treino e teste (divisão de 70% e 30%, respetivamente), através do uso da validação cruzada, aplicada 10 vezes.

O método ROSE quando selecionado para fazer o balanceamento do conjunto de dados apresentou uma precisão média de 50,67%, por outro lado, o método SMOTE quando aplicado, verificou-se uma precisão média de 51,69%. Estes valores foram influenciados de forma negativa, devido ao algoritmo *Clustering* que apresentou uma precisão muito baixa quando comparado com os restantes algoritmos. Na determinação destes valores de precisão não foi aplicada a técnica de normalização, *Min-Max Scale*, que acrescenta um aumento na precisão de cerca de 4,03% como visto anteriormente. O facto dos valores de precisão do *Clustering* serem tão baixos deve-se ao facto de ser um algoritmo não supervisionado, isto é, tenta selecionar grupos com base nas métricas definidas estabelecendo padrões entre elas, tendo escolhido cinco clusters (o mesmo número que os níveis de *stress*, para este estudo).

Os algoritmos que melhor se comportaram em termos de precisão foram o SVM e a Floresta Aleatória. Com uma precisão média de 73,49% a Floresta Aleatória foi o melhor algoritmo, ficando logo atrás com 72,41% o SVM.

Levando em consideração esses resultados, algumas técnicas de ajuste foram usadas para melhorar o desempenho da precisão dos modelos SVM e FA. O modelo Floresta Aleatória demonstrou os melhores resultados ao usar o número padrão de 500 árvores, enquanto o modelo SVM apresentou melhoria significativa após o ajuste dos parâmetros, com aumento de média de 6%. Para o ajuste, foi utilizada uma pesquisa exaustiva na grade de parâmetros, permitindo que o modelo ajuste os seus parâmetros para melhorar a categorização de futuros estudos de caso. As respetivas matrizes de confusão dos modelos SVM e FA sintonizados são mostradas na Figura21.

Atual\Previsão	1	2	3	4	5	Precisão
1	10	8	0	0	0	78,67%
2	0	17	0	0	0	
3	0	5	7	0	0	
4	0	0	0	13	0	
5	0	3	0	0	12	

(a) Matriz Confusão SVM.

Atual\Previsão	1	2	3	4	5	Precisão
1	12	1	3	1	1	81,33%
2	1	13	2	1	0	
3	0	2	9	1	0	
4	0	0	0	13	0	
5	0	1	0	0	14	

(b) Matriz Confusão FA.

Figura 21.: Resultados da Matriz de Confusão após ajustar os modelos SVM e FA

5.5 DISCUSSÃO E ANÁLISE DA SOLUÇÃO

Neste capítulo, é explicado todos as fases referentes ao processo de desenvolvimento criado para análise do *stress*, desde a fase de recolha até a fase da precisão do nível de *stress* com base nas métricas. Durante o desenvolvimento foram tomadas decisões com vista a otimizar o processo e estudar as vantagens e desvantagens de cada interveniente.

As métricas foram sujeitas a alterações durante o processo, uma vez que existia correlação nas previamente escolhidas, afetando desta forma a precisão dos algoritmos. Foram então estudadas possíveis variáveis e através do coeficiente de correlação de *Pearson* selecionaram-se as que melhor se adequavam ao caso de estudo.

A decisão mais importante no processo de desenvolvimento foi a escolha do algoritmo que melhor se adequa ao tratamento dos dados. Para testar a capacidade de precisão de cada algoritmo preparou-se um conjunto de dados uniforme e utilizados diferentes algoritmos, no qual se destacaram o SVM e FA. Após esta escolha, o próximo passo foi aumentar a eficiência dos algoritmos. Para esse objetivo implementou-se um método que testa diferentes combinações dos parâmetros dos algoritmos e vê qual é que é mais eficiente. Com este método houve uma melhoria média de 6%, ambos os algoritmos ficando mais eficientes na precisão do nível de *stress*.

De salientar que na tomada de escolha do algoritmo a implementar é importante ter noção que o algoritmo Floresta Aleatória apesar de ser mais preciso, tem um custo mais elevado a nível de tempo e processamento, necessitando de mais recursos. Por outro lado, o SVM apresenta um resultado ligeiramente mais inferior, mas o tempo de resposta e uso de recursos é menor quando comparado com o FA.

6

CONCLUSÃO

Neste capítulo, é descrito brevemente as conclusões obtidas neste trabalho, assim como, o conjunto de objetivos concluídos, as limitações da solução proposta e é apresentado perspetivas para trabalhos futuros.

Este trabalho foca a importância do *stress* na vida das pessoas e a influência que este tem sobre as nossas ações no dia a dia, principalmente em situações sob pressão, como realização de exames escolares, fazer projetos complexos, pouco tempo para a realização de tarefas. Os resultados podem ser afetados negativamente devido ao *stress*, não demonstrando o verdadeiro potencial da pessoa e quando exposto a longos períodos de tempo pode trazer inúmeras complicações de saúde.

O foco passa por analisar o nível de *stress* para saber se uma determinada pessoa está com *stress*, através da interação com o teclado e rato e fazendo uma autoavaliação do seu estado de *stress*, de uma forma não intrusiva e não invasiva. Através destas informações, é possível estabelecer relações e prever o respetivo nível de *stress* que a pessoa indica.

A implementação deste projeto no âmbito académico era a prioridade inicial, por isso foi feita uma análise detalhada das diferentes plataformas que auxiliam e permitem a realização de exames aos alunos. A plataforma que apresentou melhores caraterísticas foi o *Moodle*, contudo não foi possível elaborar um *plugin* para incorporar a monitorização de *stress*.

Através da monotorização do *stress*, o foco pode ser adaptado consoante a situação, como a produtividade de um trabalhador ser muito boa quando apresenta um nível e ser menos eficiente quando o nível é diferente, permitindo estabelecer um padrão e entender melhor o comportamento da pessoa, acabando todos por conseguir diversas vantagens.

Ainda é necessário um trabalho significativo nesta área. Especificamente, o uso de um sistema semelhante em ambientes stressantes, como locais de trabalho. Além disso, uma análise baseada nos resultados estatísticos dos testes deve ser elaborada para melhor compreensão da variação dos níveis de stress. O principal objetivo é fornecer uma resposta mais rápida e eficaz aos utilizadores cujo o desempenho pode ser melhorado através do uso de técnicas de controlo de *stress*, que melhoram o desempenho do utilizador, e respetivamente a sua qualidade de vida.

6.1 OBJETIVOS ALCANÇADOS

Em relação aos objetivos definidos previamente, pode-se considerar que eles foram quase todos alcançados satisfatoriamente, ficando em falta a implementação numa plataforma de ensino. Levando em consideração que foi preparada uma extensa análise do problema. A implementação de uma plataforma com este nível de complexidade tornou-se complicado. O cumprimento dos objetivos e contribuições era uma tarefa difícil. No entanto, no final, quase todos os requisitos do projeto foram implementados e, como resultado, a solução final atende aos objetivos impostos no início.

Para primeiro objetivo, identificar os aspetos importantes num sistema de gestão da aprendizagem, foi preciso uma pesquisa detalhada das diferentes plataformas existentes no mercado e comparar as respetivas caraterísticas de maior importância. Após elaborada uma lista com essas caraterísticas, verificou-se de plataforma em plataforma quais eram cumpridas e quais não estavam implementadas. Por fim, foi criada uma tabela 1 para ver a plataforma que mais se adequaria para o âmbito deste projeto, sendo o *Moodle* a plataforma mais satisfatória.

No segundo e terceiro objetivo, identificar as caraterísticas relevantes do ser humano para usar na análise dos dados e identificar situações de *stress*, de forma a prevenir ou atenuar as consequências, com base nos recursos disponíveis e de forma a realizar uma monotorização não invasiva e não intrusiva, as caraterísticas escolhidas foram o uso do teclado e do rato por parte do utilizador. Desta maneira, o utilizador não sentiria a presença da respetiva monitorização, e não seria influenciado a mudar o seu comportamento. A situação de *stress* foi a resolução de um exame de código, em que durante a sua realização, teriam de copiar as perguntas para um ficheiro de texto, de forma a simular parte escrita de um exame com a utilização do rato e do teclado.

Para o quarto objetivo, identificar os algoritmos que têm interesse para tratar os dados, a pesquisa foi elaborada ao longo do capítulo 3, onde foram abordados os diferentes tipos de algoritmos candidatos a serem implementados na plataforma para posterior previsão do nível de *stress*. A análise do desempenho de cada algoritmo foi realizada no capítulo 5, onde dos vários algoritmos previamente selecionados, os algoritmos SVM e FA foram os que apresentaram os melhores resultados para este tipo de tarefa, tendo noção que apesar do FA ter melhores resultados tem um peso maior a nível de processamento e tempo de resposta ao contrário do SVM que apresentou um resultado de precisão ligeiramente inferior.

No quinto objetivo, implementar a recolha dos dados através do uso do SGA, apesar da recolha de dados ter sido realizada através de subprocessos a correr no computador utilizado para a realização do estudo, não foi possível implementar a arquitetura elaborada numa SGA. Esta apresentou alguns problemas por causa criação do *plugin*.

Por fim, o sexto objetivo, disponibilizar os resultados obtidos de um respetivo utilizador ao mesmo, foi cumprido no capítulo 5, onde os utilizadores podem ver os resultados obtidos consultando os gráficos disponíveis na plataforma para melhor perceção dos resultados finais.

6.2 LIMITAÇÕES E TRABALHO FUTURO

Uma das maiores limitações no caso de estudo foi o facto da SGA *Moodle* ter descontinuado a monitorização do rato e teclado, considerando que inicialmente se tinha definido o uso dessas ferramentas para captura dos dados. Como tal, foi preciso essas *scripts* de raiz, impossibilitando a integração do programa nesta plataforma. Outra solução que ocorreu foi a criação de um *plugin*, mas este método era demasiado complexo para ser implementado.

Outro obstáculo encontrado foi o número de utilizadores para recolha dos dados, visto que para tal seria preciso disponibilizar o programa em diferentes dispositivos. Como tal, a recolha dos dados fez-se localmente, tendo 6 utilizadores realizado o teste para captura dos dados. Para uma posterior recolha significativa, é preciso ter atenção à permissão da captura dos dados, tendo os utilizadores que concordar com esta intervenção.

Com base neste trabalho, os próximos passos seriam implementar este programa numa SGA de forma a permitir um número elevado de dados e ao mesmo tempo num ambiente académico. Desta forma teríamos uma perceção mais realista dos resultados a obter. Contudo, este programa também poderia ser implementado num ambiente profissional de forma a monitorizar o nível de *stress* dos funcionários, com vista a otimizar o desempenho, um pouco ao que já existe referente ao cansaço que aconselha o utilizador a fazer pausas.

Um ponto que teria interesse abordar, seria recorrer a outras métricas para uma previsão mais assertiva, que permitissem, com base nos resultados dos exames realizados ou num ambiente profissional, atribuir um grau de dificuldade à tarefa proposta ao funcionário, desta forma teríamos uma noção das pessoas mais aptas a realizar tarefas mais complexas.

Numa frente mais avançada, o ideal seria aplicar este programa noutros dispositivos que não usem o rato e teclado, de forma a monitorizar o nível do *stress* através do toque, por exemplo, telemóveis ou *tablets*.

BIBLIOGRAFIA

Al-Ajlan, A. and Zedan, H. (2008). Why moodle. In *2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems*, pages 58–64. IEEE.

Al Shalabi, L., Shaaban, Z., and Kasasbeh, B. (2006). Data mining: A preprocessing engine. *Journal of Computer Science*, 2(9):735–739.

Belanger, Y. (2004). Summary of fall 2003 blackboard survey results. *Blackboard. duke. edu/pdf/Bb_ survey_report_f2003. pdf*.

Bhargava, N., Sharma, G., Bhargava, R., and Mathuria, M. (2013). Decision tree analysis on j48 algorithm for data mining. *Proceedings of International Journal of Advanced Research in Computer Science and Software Engineering*, 3(6).

Bittner, K. (2002). *Use case modeling*. Addison-Wesley Longman Publishing Co., Inc.

Boggs, W. and Boggs, M. (2002). *Mastering UML with rational rose 2002*. Sybex.

Bradford, P., Porciello, M., Balkon, N., and Backus, D. (2007). The blackboard learning system: The be all and end all in educational instruction? *Journal of Educational Technology Systems*, 35(3):301–314.

Brailovsky, V. L. (1991). A probabilistic approach to clustering. *Pattern Recognition Letters*, 12(4):193–198.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Carneiro, D., Novais, P., Durães, D., Pego, J. M., and Sousa, N. (2019). Predicting completion time in high-stakes exams. *Future Generation Computer Systems*, 92:549–559.

Carneiro, D., Novais, P., Pêgo, J. M., Sousa, N., and Neves, J. (2015). Using mouse dynamics to assess stress during online exams. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 345–356. Springer.

Cauwenberghs, G. and Poggio, T. (2001). Incremental and decremental support vector machine learning. In *Advances in neural information processing systems*, pages 409–415.

Chaubey, A. and Bhattacharya, B. (2015). Learning management system in higher education. *International Journal of Science Technology & Engineering*, 2(3):158–162.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chodorow, K. (2013). *MongoDB: the definitive guide: powerful and scalable data storage*. "O'Reilly Media, Inc.".

Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J. (2012). *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media.

Cockburn, A. (2008). Why i still use use cases. *alistair. cockburn. us*.

Cutler, D. R., Edwards Jr, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., and Lawler, J. J. (2007). Random forests for classification in ecology. *Ecology*, 88(11):2783–2792.

Davis, B., Carmean, C., and Wagner, E. D. (2009). The evolution of the lms: From management to learning. *Santa Rosa, CA: e-Learning Guild*.

Diday, E. and Simon, J. (1976). Clustering analysis. In *Digital pattern recognition*, pages 47–94. Springer.

Ellis, R. K. (2009). Field guide to learning management systems. *ASTD Learning Circuits*.

Fariha, Z. and Zuriyati, A. (2014). Comparing moodle and efront software for learning management system. *Australian Basic and Applied Sciences*, 8(4):158–162.

Fink, G. (2010). *Stress science: neuroendocrinology*. Academic Press.

Frankish, K. and Ramsey, W. M. (2014). *The Cambridge handbook of artificial intelligence*. Cambridge University Press.

Freund, Y. and Mason, L. (1999). The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133.

Fulton, R. and Vandermolen, R. (2017). *Airborne Electronic Hardware Design Assurance: A Practitioner's Guide to RTCA/DO-254*. CRC Press.

Ghahramani, Z. (2003). Unsupervised learning. In *Summer School on Machine Learning*, pages 72–112. Springer.

Islam, Q. N. (2015). *Mastering PyCharm*. Packt Publishing Ltd.

Jacobson, I., Spence, I., and Bittner, K. (2011). Use case 2.0: The definite guide. *Ivar Jacobson International*.

Jain, A. K., Dubes, R. C., et al. (1988). *Algorithms for clustering data*, volume 6. Prentice hall Englewood Cliffs.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

Jovanovic, D. and Jovanovic, S. (2015). An adaptive e-learning system for java programming course, based on dokeos le. *Computer Applications in Engineering Education*, 23(3):337–343.

Kakasevski, G., Mihajlov, M., Arsenovski, S., and Chungurski, S. (2008). Evaluating usability in learning management system moodle. In *ITI*, volume 2008, page 30th.

Kassambara, A. (2017). *Practical guide to cluster analysis in R: unsupervised machine learning*, volume 1. STHDA.

Kilickaya, F. (2009). Another powerful e-learning and course management tool for web-based learning: Dokeos. *CALL-EJ Online*, 11(1):1–5.

Kononenko, I. (2001). Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109.

Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24.

Long, P. D. (2004). Encyclopedia of distributed learning. *SAGE Publications*.

Martin, F. (2008). Blackboard as the learning management system of a computer literacy course. *Journal of Online Learning and Teaching*, 4(2):138–145.

Martinez, M. and Jagannathan, S. (2008). Moodle: A low-cost solution for successful e-learning. *Learning Solutions Magazine [Online]*.

McCarthy, J. (1998). What is artificial intelligence?

Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.

Michalski, R. S. and Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In *Machine learning*, pages 331–363. Springer.

Moore, J. (2010). *Moodle 1. 9 Multimedia Extension Development: Customize and Extend Moodle by Using Its Robust Plugin Systems*. Packt Publishing Ltd.

Obermeyer, Z. and Emanuel, E. J. (2016). Predicting the future—big data, machine learning, and clinical medicine. *The New England journal of medicine*, 375(13):1216.

Patterson, D. W. (1990). *Introduction to artificial intelligence and expert systems*. Prentice-hall of India.

Pereira, A. M. T. B. (2002). *Burnout: quando o trabalho ameaça o bem-estar do trabalhador*. Casa do Psicólogo.

Pimenta, A., Carneiro, D., Neves, J., and Novais, P. (2016). A neural network to classify fatigue from human–computer interaction. *Neurocomputing*, 172:413–426.

Pingle, S. S. (2011). Higher education students readiness for e-learning. *TechnoLearn: An International Journal of Educational Technology*, 1(1):155–165.

Pohl, K. (2010). *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated.

Robert, C. (2014). Machine learning, a probabilistic perspective.

Rodrigues, M., Novais, P., and Santos, M. F. (2005). Future challenges in intelligent tutoring systems: a framework.

Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., and Rigol-Sanchez, J. P. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:93–104.

Russell, S., Norvig, P., and Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25(27):79–80.

Sanner, M. F. et al. (1999). Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1):57–61.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Somekh, B. (2005). *Action research: a methodology for change and development: a methodology for change and development*. McGraw-Hill Education (UK).

Suykens, J. A. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Van Lamsweerde, A. (2000). Requirements engineering in the year 00: a research perspective. In *Proceedings of the 22nd international conference on Software engineering*, pages 5–19. ACM.

Weaver, D., Spratt, C., and Nair, C. S. (2008). Academic and student use of a learning management system: Implications for quality. *Australasian journal of educational technology*, 24(1).

Wiegers, K. and Beatty, J. (2013). *Software requirements*. Pearson Education.

Young, R. R. (2001). *Effective requirements practices*. Addison-Wesley Longman Publishing Co., Inc.

A

DESCRIÇÃO DOS CASOS DE USO

Este Anexo contém a descrição dos diferentes casos de uso existentes no sistema, sob a forma de tabelas 2-9. A descrição do texto do caso de uso referente a criar, editar e eliminar provas está relacionado com os exames a que os alunos vão ser submetidos para a realização da recolha dos dados biométricos. A descrição dos casos de uso referentes aos dados, tem o intuito de esclarecer o tratamento que o administrador pode realizar com os mesmo, desde editar caso alguma situação não esteja de acordo ou eliminar se os mesmos não forem pertinentes.

Legendas:

- **UD** - *User Defined* (funcionalidade acedida pelo utilizador);
- **OUT** - *Output* (Dados de saída pelo sistema);
- **INP** - *Input* (Dados de entrada pelo utilizador);

Tabela 2.: Criar novas provas

Nome			**Criar novas provas**
Objetivo			Criar novos métodos de recolha da informação, recorrendo ao uso das provas
Pré-Condição			Ter efetuado o *login* como professor.
Pós-Condição			Nenhuma.
Super Caso de uso			Nenhum.
Fluxo Eventos			**Comportamento normal.**
	UD	1.	Professor entra na página inicial;
	INP	2.	Professor carrega no botão de "Carregar Provar";
	OUT	3.	O sistema permite ao professor carregar o ficheiro;
	INP	4.	Professor seleciona o ficheiro;
		5.	Professor carrega no botão "Guardar";
	OUT	6.	O sistema guarda o ficheiro;
		7.	Retorna ao ponto 1.

Tabela 3.: Editar provas

Nome			**Editar provas do sistema**
Objetivo			Editar provas existentes no sistema
Pré-Condição			Ter efetuado o *login* como professor e existerem provas criadas.
Pós-Condição			Nenhuma.
Super Caso de uso			Nenhum.
Fluxo Eventos			**Comportamento normal.**
	UD	1.	Professor entra na página inicial;
	INP	2.	Professor acede à opção de editar a prova;
	OUT	3.	O sistema mostra as provas que o professor já carregou;
	INP	4.	Professor seleciona a prova que pretende alterar;
	OUT	5.	O Sistema mostra a prova selecionada;
	INP	6.	Professor carrega no botão "Guardar";
	OUT	7.	O sistema salva a alteração;
		8.	Retorna ao ponto 1.

Tabela 4.: Eliminar provas

Nome			**Eliminar provas do sistema**
Objetivo			Eliminar provas existentes no sistema
Pré-Condição			Ter efetuado o *login* como professor e existerem provas criadas.
Pós-Condição			Nenhuma.
Super Caso de uso			Nenhum.
Fluxo Eventos			**Comportamento normal.**
	UD	1.	Professor entra na página inicial;
	INP	2.	Professor acede à opção de eliminar a prova;
	OUT	3.	O sistema mostra as provas que o professor já carregou;
	INP	4.	Professor seleciona a prova que pretende eliminar;
	OUT	5.	O Sistema mostra a prova selecionada;
	INP	6.	Professor carrega no botão "Eliminar";
	OUT	7.	O sistema elimina a prova;
		8.	Retorna ao ponto 1.

Tabela 5.: Resolver provas

Nome			**Resolver provas**
Objetivo			Resolver provas para recolher os dados biométricos dos estudantes
Pré-Condição			Ter efetuado o *login* como estudante e existerem provas.
Pós-Condição			Nenhuma.
Super Caso de uso			Nenhum.
Fluxo Eventos			**Comportamento normal.**
	UD	1.	Estudante entra na página inicial;
	INP	2.	Estudante acede à opção de resolver a prova;
	OUT	3.	O sistema mostra as provas disponíveis;
	INP	4.	Estudante seleciona a prova que pretende realizar;
	OUT	5.	O sistema disponibiliza a prova;
	INP	6.	Estudante realiza a prova;
		7.	Estudante carrega no botão "Terminar";
	OUT	8.	Retorna ao ponto 1.

Tabela 6.: Consultar resultados

Nome			**Consultar resultados**
Objetivo			Visualizar os resultados obtidos
Pré-Condição			Ter efetuado o *login* como estudante e ter os resultados das provas realizadas.
Pós-Condição			Nenhuma.
Super Caso de uso			Nenhum.
Fluxo Eventos			**Comportamento normal.**
	UD	1.	Estudante acede à página inicial;
	INP	2.	Estudante acede à opção de consultar resultados;
	OUT	3.	O sistema mostra os resultados disponíveis;
	INP	4.	Estudante visualiza os resultados.

Tabela 7.: Manutenção

Nome			**Manutenção do programa**
Objetivo			Permitir o fluxo normal do programa
Pré-Condição			Ter efetuado o *login* como administrador e ter a notificação dos erros.
Pós-Condição			Nenhuma.
Super Caso de uso			Nenhum.
Fluxo Eventos			**Comportamento normal.**
	UD	1.	Administrador acede à página inicial;
	OUT	2.	O sistema mostra a notificação dos erros;
	UD	3.	Administrador carrega na notificação dos erros;
	OUT	4.	O sistema mostra o erro que precisa de ser corrigido;
	INP	5.	Administrador resolve o erro;
	OUT	6.	Retorna ao ponto 1.

Tabela 8.: Editar Dados

Nome			**Editar Dados**
Objetivo			Permitir selecionar os dados de interesse para a situação em causa
Pré-Condição			Ter efetuado o *login* como administrador e o sistema ter dados recolhidos.
Pós-Condição			Nenhuma.
Super Caso de uso			Nenhum.
Fluxo Eventos			**Comportamento normal.**
	UD	1.	Administrador acede à página inicial;
	INP	2.	Administrador carrega na botão "Editar Dados";
	OUT	3.	O sistema mostra os dados disponíveis;
	INP	4.	Administrador faz as respetivas alterações;
		5.	Administrador carrega no botão "Guardar";
	OUT	6.	O sistema guarda as alterações;
		7.	Retorna ao ponto 1;

Tabela 9.: Eliminar Dados

Nome			**Eliminar Dados**
Objetivo			Permitir eliminar os dados de interesse para a situação em causa
Pré-Condição			Ter efetuado o *login* como administrador e o sistema ter dados recolhidos.
Pós-Condição			Nenhuma.
Super Caso de uso			Nenhum.
Fluxo Eventos			**Comportamento normal.**
	UD	1.	Administrador acede à página inicial;
	INP	2.	Administrador carrega na botão "Eliminar Dados";
	OUT	3.	O sistema mostra os dados disponíveis;
	INP	4.	Administrador elimina os dados;
		5.	Administrador carrega no botão "Guardar";
	OUT	6.	O sistema guarda as alteraçoes;
		7.	Retorna ao ponto 1;