ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

M
MESTRADO
ENGENHARIA INFORMÁTICA

Deteção de ataques em aplicações Web através de algoritmos de Machine Learning

Pedro Miguel Ribeiro Sampaio

12/2020

Deteção de ataques em aplicações Web através de algoritmos de Machine Learning
Pedro Miguel Ribeiro Sampaio

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

M
MESTRADO
ENGENHARIA INFORMÁTICA

Pedro Miguel Ribeiro Sampaio. Deteção de ataques em aplicações Web através de algoritmos de Machine Learning

12/2020

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

M

MESTRADO
ENGENHARIA INFORMÁTICA

Deteção de ataques em aplicações Web através de algoritmos de Machine Learning

Pedro Miguel Ribeiro Sampaio

Orientador: João Paulo Ferreira de Magalhães

Co-Orientador: Davide Rua Carneiro

Agradecimentos

Gostaria de agradecer a todas as pessoas que tornaram este momento possível e que ajudaram e contribuíram para conseguir obter esta etapa. Gostaria de agradecer também ao professor João Paulo Magalhães pela sua orientação e trabalho prestado durante este percurso. Também agradecer ao professor Davide Carneiro pela sua disponibilidade, trabalho prestado e ajuda em decisões que me levaram a chegar a esta etapa. Muito obrigado a estas duas pessoas que sem a sua preciosa ajuda não conseguia chegar tão longe e alcançar um objetivo de vida.

Agradecer também aos meus amigos por ajudarem e contribuírem para a concretização deste trabalho.

Gostava também de deixar a minha gratidão a todos os professores de Mestrado de Engenharia Informatica pela motivação e ajuda que permitiram aumentar o meu conhecimento/crescimento na área aumentando assim as minhas capacidades.

Um especial agradecimento à minha família pela motivação prestada nos momentos mais difíceis e que me incentivaram sempre a dar o melhor e a não desistir.

Muito obrigado a todos pela contribuição!

Resumo

As aplicações Web têm um papel fulcral em muitas organizações, suportando e potenciando os seus negócios. A sua popularidade e criticidade é também apelativa aos atores maliciosos que exploram possíveis vulnerabilidades para executar os seus ataques, levando a prejuízos significativos de reputação e financeiros por partes das organizações e pessoas afetadas.

Neste projeto explorámos a utilização de algoritmos de *Machine Learning* para detetar anomalias e possíveis ciberataques através da análise dos logs que a aplicação produz, com base em dados reais e dados gerados a partir de testes de intrusão foi criado um dataset para análise através de algoritmos *Machine Learning* de modo a determinar ciberataques em curso. O dataset é composto por 36 parâmetros resultantes da agregação de dados por janela temporal e do enriquecimento de dados. Os datasets foram tratados e analisados através de diferentes algoritmos de *Machine Learning* considerando a atualização de hiperparâmetros verificando-se a existência de algoritmos capazes de detetar anomalias e possíveis ciberataques com uma taxa de acerto superior a 90%. A eficácia verificada contribui para a criação de uma solução independente focada no desempenho e na rapidez de resposta que permite detetar e/ou bloquear ataques numa fase inicial mitigando as suas consequências negativas junto das organizações.

Palavras-chaves: *Machine Learning*, ciberataques, algoritmos, atividade maliciosa

Abstract

Web applications play a central role in many organizations, supporting and enhancing their business. Its popularity and criticality is also appealing to malicious actors who exploit possible vulnerabilities to carry out their attacks, leading to significant reputational and financial losses by affected organizations and people.

In this project we explored the use of *Machine Learning* algorithms to detect anomalies and possible cyberattacks through the analysis of the logs that the application produces, based on real data and data generated from intrusion tests, a dataset was created for analysis using *Machine Learning* algorithms in order to determine ongoing cyberattacks. The dataset is composed of 36 parameters resulting from the aggregation of data by time window and the enrichment of data. The datasets were treated and analysed using different *Machine Learning* algorithms considering the update of hyperparameters verifying the existence of algorithms capable of detecting anomalies and possible cyberattacks with a hit rate greater than 90%. The verified effectiveness contributes to the creation of an independent solution focused on the performance and speed of response that allows detecting and/or blocking attacks at an early stage, mitigating their negative consequences for organizations.

Keywords: *Machine Learning*, cyberattacks, algorithms, malicious activity

Conteúdo

Lista de Tabelas

Lista de Figuras

Glossário

API - Interface de programação de aplicações, *Application Programming Interface*

Dataset - Corresponde a um conjunto de dados ou instâncias com um significado próprio.

ESTG - Escola Superior de Tecnologia e Gestão

ETL - Extrair Transformar Carregar, *Extract Transform Load*

IA - Inteligência Artificial

Logs - Denomina-se por ser informação referente à atividade do utilizador na aplicação ou de um serviço num sistema informático.

ML - Machine Learning

OLAP - Processamento analítico online, *Online Analytical Processing*

SDN - É uma arquitetura dinâmica, ideal para alcançar elevada largura de banda e ajustar às necessidades dinâmicas de comunicação por parte das aplicações. Permite ter um maior controlo de rede e das funções de encaminhamento, permitindo gerir e controlar estas funcionalidades sendo a infraestrutura subjacente abstraída para as aplicações e serviços de rede [35].

SQL - *Structured Query Language*

SGBD - É um programa que permite criar e manipular base de dados, em que os dados estão guardados de forma independente à aplicação onde é usada. Desta forma, o SGBD serve de interface para abstrair os diferentes utilizadores.

Variação (Variance) - É a variação da previsão do modelo para um dado ponto de dados. Um modelo que apresente grande variação apresenta uma grande relação e aprendizagem nos dados de treino e não generaliza novos dados que ainda não tenha conhecimento. Assim, o desempenho é muito bom para os dados de treino, mas apresenta altas taxas de erro nos dados de teste [26].

Viés (Bias) - Denomina-se por ser a diferença entre a previsão esperada e o valor correto que se está a prever, ou seja, mede se os *outputs* dos modelos estão corretos tendo em conta o *input*. Quando o viés é alto o modelo não relaciona corretamente os dados de treino e simplifica demais o modelo, levando a altos erros nos dados de treino e teste [26].

WWW ou Web - World Wide Web

Capítulo 1

Introdução

O crescimento do número de aplicações que consomem e produzem grandes quantidades de dados, desempenham um grande papel na nossa rotina diária e continuam a aumentar. As aplicações são cada vez mais imprescindíveis e assistem a processos importantes, tais como, aceder a dados pessoais, aceder a contas bancárias e armazenar dados. A digitalização a que assistimos e o número de dispositivos capazes de comunicar entre si, cresce igualmente a bom ritmo, trazendo novas funcionalidades para as organizações. Se por um lado temos a componente de evolução tecnológica, por outra verifica-se o crescimento no número de ameaças cibernéticas. Estas ameaças levam a inúmeros ataques com consequências económicas avultadas, prejuízos de reputação e em casos recentes colocando vidas humanas em causa. A deteção de atividades maliciosas é um problema difícil de colmatar e, por vezes, não é suficiente a utilização de métodos e abordagens tradicionais de segurança tais como a utilização de *firewalls*, sistemas de deteção de intrusos, antivírus e mecanismos criptográficos. É um facto que as aplicações proporcionam um meio de acesso a conteúdo alheio [20].

Os ciberataques assumem várias formas. Os que envolvam qualquer atividade ou prática ilícita na rede são designados por cibercrime. Os que consistem em espiar organizações denominam-se por ciberespionagem. Ataques que tiram partido dos sistemas e redes para difundir uma mensagem/lema são designados por hacktivismo. Ataques que visam enfraquecer a capacidade de um determinado país, afetando normalmente as infraestruturas críticas são denominados por ciberguerra. Proteger as organizações e mesmo os estados dos ciberataques não é uma tarefa fácil. Se por um lado as organizações têm necessidade de reforçar as suas ferramentas e procedimentos de defesa por outro os atores maliciosos procuram de

forma contínua novas ferramentas, técnicas e procedimentos de ataques.

Ainda que a adoção de uma abordagem holística que envolve pessoas, processos e tecnologia seja entendida como fundamental para melhorar a postura das organizações relativamente à cibersegurança, na prática, continua-se a verificar inúmeros ataques. Estes ataques de forma geral derivam pela falha na priorização para endereçar o problema, pela descoberta de novas vulnerabilidades (*zero-day*) e por descuido generalizado com as questões da segurança informática. Por exemplo, a alta competitividade a que as organizações estão sujeitas, coloca mais pressão no desenvolvimento das aplicações para lançar o produto mais cedo para o mercado deixando para segundo plano as questões de segurança. Isto tal como referido em [53], acarreta riscos no descuido do desenvolvimento e a incorreta implementação da arquitetura de segurança da aplicação. Este descuido com a segurança faz com que formas comuns utilizadas para obter informação de terceiros como o caso de *SQL injection* sejam ainda hoje em dia uma dor de cabeça para as organizações [17]. A exploração de ataques de *Cross-Site Scripting (XSS)* que é utilizado para executar código malicioso no navegador do utilizador ainda são exploradas. Tentativas para tornar a aplicação *World Wide Web (Web)* lenta ou mesmo parar o funcionamento através do recurso a ataques do tipo *DDoS* também são frequentes.

O número de ataques reportados em locais como [55] demonstra que os ataques a aplicações *Web* ao longo dos anos têm aumentado substancialmente. De acordo com as estatísticas recolhidas e apresentadas no *Website Hacking Statistics*, a cada 39 segundos ocorre um ataque na Web. 73% dos atores maliciosos dizem que os sistemas tradicionais de firewall a antivírus são irrelevantes ou obsoletos. Por dia são atacados 30 mil novos sites Web. Para mitigar este problema tem-se assistido a um aumento da necessidade de proteger a informação que flui nas aplicações e manter os dados do utilizador seguros contra qualquer ataque ou manipulação de dados. As organizações têm procurado encontrar soluções fiáveis para diminuírem substancialmente os ataques. Uma das abordagens mais recentes contempla o recurso a processos de *Machine Learning* (ML) para melhorar a capacidade de deteção de ciberataques e dessa forma acrescentar uma camada extra de segurança nas aplicações [61]. A Microsoft, por exemplo, tira partido de algoritmos de ML na sua plataforma de segurança, o Windows Defender Advanced Threat (ATP) para prever violações ou falhas que possam indicar um ataque em curso e, deste modo, optar por uma resposta adequada à ameaça [1].

Os vários trabalhos analisados apresentaram diversas ideias distintas que demonstraram se-

rem bastantes eficientes no uso contra ciberataques. Permitiram adquirir conhecimento e ideias para a implementação desta solução e serviram como apoio para o foco do projeto. Contudo, para a implementação destas abordagens foram utilizados conjuntos de dados já processados e tratados que servem de apoio para a parte de criar modelos para o uso de *Machine Learning*, ou seja, isto torna a solução dependente do tipo e formatados dos dados.

Neste trabalho consideramos uma abordagem diferente para colmatar diferentes tipos de ataques que podem ocorrer numa aplicação, com diferentes conjuntos de dados de entrada. Para tal, foi criado um processo ETL (Extrair Transformar Carregar) genérico para fazer o tratamento de diferentes padrões que possam existir nos dados de entrada e depois, através dos mesmos treinar um modelo de aprendizagem supervisionada para utilizar na classificação de padrões que escapem ao comportamento normal, podendo os mesmos derivar de um mau funcionamento ou de um ataque à aplicação.

Para o estudo base, foram utilizados dados do Moodle de uma instituição de ensino superior que conta com cerca de 1500 utilizadores entre funcionários docentes, funcionários não docentes e discentes. Os dados foram gerados durante os meses de dezembro 2019 e janeiro de 2020. Todos os logs provenientes do Moodle foram analisados e classificados com sendo normais. Para criar logs relacionados com padrões de ataque foi criado um ambiente controlado onde se simularam ataques à aplicação em estudo. Desta forma, obteve-se um grande volume de dados para analisar e treinar. Para efeito de análise os dados foram agrupados por janelas temporais (frames), contribuindo desta forma para um melhor desempenho da solução. Cada instância do dataset foi criada sobre a definição de frames construidos a partir do espaço temporal de 30 segundos (tempo ótimo encontrado para esta solução após uma análise empírica). Em todo o caso o valor poderá ser ajustado com a simples alteração de um parâmetro. Em resultado foi obtido um conjunto de 185 592 frames de dados classificados como normais que derivaram de 5 793 574 pedidos ao servidor e 1 454 frames classificadas como anormais obtidas a partir de 1 828 169 pedidos ao servidor. Para fazer face à diferença no número de frames classificadas à priori como normais e anormais (*unbalance*) foram aplicadas técnicas para validar a veracidade das respostas de previsão e verificar se não ocorria over-fitting. O dataset criado foi submetido a vários algoritmos de regressão linear para obtermos a melhor solução e após treinar o modelo com o melhor algoritmo determinado (XGBoost) obtivemos resultados bastante promissores com uma taxa de positivo verdadeiro de 98%. Esta percentagem de eficácia na classificação de frames normais e anormais verificou-se mesmo após várias análises aos dados e na utilização de diferentes abordagens para validar os mesmos e

garantir que o modelo foi criado de forma coerente.

1.1 Solução Proposta

Da análise efetuada às abordagens existentes e da leitura de vários artigos (e.g. [62, 51, 16, 63, 22, 12, 24], nenhum correlacionava os diferentes tipos de ataques feitos às aplicações, ou seja, não mostravam abranger os ataques com maior ocorrência nas aplicações e que são prejudiciais numa organização caso sejam bem-sucedidos. Estas apenas se focavam em soluções e aplicações distintas focados num único ponto a seguir descrito, deteção de DDos, correlacionar os dados dos pedidos feitos ao servidor, a interação e fluxo de dados na aplicação, entropias no URL, detetar ataques por similaridade, entre outros.

Por outro lado, nenhuma delas demonstra ter soluções *end-to-end* que sejam totalmente autónomas e independentes de configurações intermédias. Por isso, consideramos desde logo juntar as soluções distintas apresentadas anteriormente que derivaram de diferentes estudos apenas numa única aplicação, focada no desempenho, no tempo de resposta e da sua abstração para ser utilizada em diferentes aplicações. Isto tudo através de vários processos contínuos interligados que dão origem a uma arquitetura baseada no desempenho e no menor erro de previsão. Como temos diferentes situações de ataques tudo junto numa única solução isto vai necessitar de mais tempo de processamento para fazer mais validações e tratamento aos dados, contudo, a solução em janela temporal vai permitir melhorar o tempo de análise e deteção de ataques uma vez que a solução baseada em *Machine Learning* em vez de analisar milhares de pedidos ao servidor 7 621 743 apenas analisa os dados agregados em janelas temporais 187 046.

Desta forma, a solução apresentada tem como foco detetar os seguintes ataques:

- Tentativas de redirecionamento para outros sites.
- Injeção de código (uso de caracteres estranhos).
- Detetar DDOS.
- Relacionar ataques com localização.
- Detetar ataques por similaridade.

- Detetar padrões nos dados que indiquem ataques.

Todos os processos intermédios necessários para chegar a um resultado são baseados no desempenho e em obter tempos de execução baixo. Ao longo deste relatório é descrito todo o processo utilizado na criação dos modelos de previsão tendo em consideração aspetos que os tornam eficientes para um conjunto de dados grande.

1.2 Objetivos

Neste trabalho foi proposto, a partir dos logs que uma aplicação produz criar um sistema em streaming ou batch e detetar alterações nos logs evidenciando anomalias ou ciberataques em curso. Os dados recolhidos são alvo de tratamento e enriquecimento, como por exemplo, a determinação da geolocalização do endereço IP origem, a entropia do URL, as características contidas no próprio pedido da página Web e o tempo entre pedidos. O objetivo é identificar parâmetros que uma vez tratados e analisados possam indicar a presença de anomalias na aplicação. Tal como referido anteriomente, os dados dos logs serão tratados, enriquecidos e agrupados em períodos temporais/frames com vista à posterior análise através de algoritmos de *Machine Learning*. Deste modo, neste trabalho foi abordado o método de aprendizagem supervisionada [15] em que as entradas são conhecidos e o algoritmo através das mesmas faz a previsão do resultado, ou seja, inicialmente todos os logs provenientes da aplicação Web são validados e considerados como acessos normais e depois através de um ambiente controlado com uma versão igual à aplicação são gerados pedidos típicos de um teste de intrusão, ou seja, semelhantes aos realizados durante um ataque informático. Os registos de log gerados deste modo serão analisados e considerados como anormais, ou seja, classificados como potenciais ciberataques. Uma vez obtidos os dados, será criado um dataset e realizadas as tarefas de análise exploratória, tratamento de dados e análise de dados através de vários algoritmos de *Machine Learning*. Esta análise visa determinar qual o melhor modelo, ou seja, aquele que em termos de eficácia e métricas (accuracy, precision, error rate, time detection) é o mais adequado para detetar de forma eficiente e simultaneamente o tempo que levam a detetar por forma a avaliar se a deteção é possível em tempo útil.

Cada fase deste projeto encontra-se dividida por várias etapas, entre as quais:

- Recolha dos dados: Nesta etapa é efetuada a recolha e anomização dos dados do logs;

- Processo de tratamento de logs: Nesta etapa é feita a extração dos dados que cada log contém e é guardada a informação dos mesmos numa base de dados;
- Análise dos dados retirados dos logs: Neste etapa é feita uma análise à informação que os logs dispõem e que outra informação se pode retirar dos mesmos para enriquecer os dados.
- Processo para enriquecer os dados: Nesta etapa os dados são enriquecidos e guardados numa base de dados;
- Processo para distinguir dados antigos de novos logs: Esta etapa acaba por definir uma forma de distinguir os dados que já foram utilizados e analisados pelos algoritmos dos novos logs que vão surgindo. Para isto existe um campo na base de dados que permite fazer a distinção dos dados;
- Processo para criar dataset: Este processo consiste em criar um dataset com os dados criados pelo processo anterior e definir as variáveis relevantes para posterior análise. O dataset será criado por janelas temporais;
- Processo de análise de modelos e métricas: Neste processo são definidos os algoritmos de *Machine Learning* a usar para análise. Desta forma, após análise dos resultados, pode-se definir quais os melhores algoritmos para treino dos modelos e classificação de novos dados.
- Processo para treinar modelo: Este processo define-se por treinar o(s) modelo(s) a utilizar para classificação. No âmbito deste processo o dataset é dividido entre dados de treino 70%, 15% para validação e 15% para testar as previsões de modo a efetuar a análise através de algoritmos. Uma alternativa ao split normal é a validação cruzada;
- Processo automático para treinar modelo: Este processo consiste em criar um processo automático para treinar e gerar um modelo ajustado para poder fazer previsões de dados futuros.

Os passos anteriores são realizados recorrendo a ferramentas de *Machine Learning*. Após análise de várias soluções (e.g. H2O, Python pandas e scikit learn, TensorFlow) a escolha recai sobre o H2O. O fato de ser de código aberto, uso livre e dispor de ferramentas que automatizam o processo de tratamento e análise (e.g. AutoML) de dados pesaram na escolha da solução.

1.3 Estrutura do documento

Esta dissertação está organizada em capítulos. No capítulo 2 é apresentado o estado da arte. O capítulo 3 apresenta a obtenção, preparação de dados e enriquecimento de dados. No capítulo 4 é descrita a análise exploratória, tratamento de dados, análise de dados recorrendo a diferentes algoritmos de *Machine Learning*, análise detalhada sobre o impato dos hiper-parâmetros no modelo e uma análise critica sobre os resultados obtidos. Por fim, no capítulo 5 é apresentada a conclusão do trabalho realizado.

Capítulo 2

Estado da arte

Neste capítulo é apresentado o estado da arte relacionado com o projeto. Em suma, são apresentadas as aplicações Web e as diferentes vulnerabilidades mais comuns nas aplicações e são apresentadas as metodologias e estudos que envolvem a componente de cibersegurança deste tipo de aplicações com enfoque nas soluções baseadas em *Machine Learning*.

2.1 Aplicações Web e ciberataques

As aplicações Web constituem um meio pelo qual os clientes, normalmente através de um navegador ou uma aplicação acedem e interagem com uma lógica de negócio. Numa vertente organizacional estas aplicações são importantes, pois permitem realizar e prosperar negócios *online*, ou seja, abrem novos caminhos para a expansão do negócio [38]. Infelizmente a expansão das aplicações Web tem um lado menos positivo. As aplicações Web apresentam por vezes falhas de segurança, expondo vulnerabilidades que permitem a atores mal-intencionados explorar as falhas afetando o negócio. Estas vulnerabilidades podem, por exemplo, facilitar a obtenção de informação importante e confidencial, tal como dados pessoais ou detalhes de contas bancárias. Considerando o número de aplicações Web, a sua exposição na Internet e a possibilidade de existência de vulnerabilidade torna-as um alvo apetecível para os piratas informáticos (ator malicioso), que procuram nesta um meio para obter informação privada que poderá ser usada, por exemplo para vender a terceiros [14].

As vulnerabilidades das aplicações têm aumentado drasticamente uma vez que o seu uso

para expandir negócio também têm aumentado. As vulnerabilidades não vêm apenas da sua expansão, a pressão que existe para construir aplicações num curto espaço de tempo, o menor tempo dedicado a análise pormenorizada dos requisitos e o lançamento rápido para produção para atender os requisitos do negócio, são aspetos que permitem falhas de segurança [64]. Estas falhas se ocorreram acarretam grandes prejuízos a nível financeiro e de reputação para as empresas e para os clientes lesados. Em certos casos podem levar a inquéritos legais e sanções financeiras, como o caso do Centro Hospitalar Barreiro Montijo que foi multado em 400 000 euros por violar o Regulamento Geral de Proteção de Dados [54].

Com o aumento das tecnologias e meios de construção da arquitetura das aplicações vão surgindo estratégias inovadoras e criativas para aceder a dados pessoais o que nos leva a concluir que as abordagens de segurança que utilizávamos já não são suficientes para proteger as aplicações. Como referido em [47] os métodos de segurança tradicionais como firewall, deteção de intrusões ou utilização de mecanismos criptográficos não são suficientes para a segurança das aplicações o que aumenta a hipótese de terceiros aceder a informação privilegiada.

2.2 Tipos de ataques a aplicações Web

Como referido, a utilização das aplicações Web e o seu papel é cada vez maior e mais importante. Porém, estas aplicações não estão isentas de vulnerabilidades tornando-se um meio comum para ataques mal-intencionados. As organizações têm investido para encontrar soluções fiáveis para proteger os seus negócios, mas apesar dos esforços existem brechas difíceis de proteger e que propiciam ataques informáticos que surgem de vários formas e por vezes representam um grande risco para a organização [57]. Os ataques mais frequentes nas aplicações Web, de acordo com [60], são:

- **DDoS**: Baseia-se em utilizar vários dispositivos geralmente de pontos de acesso diferentes para realizar pedidos ao servidor Web de maneira a sobrecarregar o mesmo. Com tal sobrecarga o serviço fica inacessível para clientes normais. Este tipo de ataque é frequentemente utilizado para interromper o serviço, e por vezes é usado como manobra de diversão para encapsular outros ataques mais elaborados.
- **SQL Injection**: Esta técnica baseia-se em utilizar a linguagem SQL no payload ou URL

da página Web na tentativa de aceder a dados privados, ou fazer alterações na base de dados. Isto ocorre porque não existe uma validação ou sanitização antes de a base de dados interpretar a operação solicitada.

- **Cross-Site Scripting (XSS)**: O cross-site scripting consiste em inserir código ou *scripts* (e.g. javascript) no código do lado do cliente. O XSS pode ser refletido e neste caso apenas o cliente que recebe o URL com XSS embebido é afetado e pode ainda ser armazenado, tornando-se mais abrangente no sentido em que todos os que visitarem a página irão despoletar o XSS. O XSS permite alterar por completo o aspeto e funcionamento das páginas Web para os clientes, facilitando a execução de operações maliciosas. Obter as *cookies* ou variáveis de sessão fazem normalmente parte dos objetivos dos atores maliciosos que exploram esta vulnerabilidade.
- **Manipulação de caminhos**: São utilizados para, normalmente a partir do URL, tentar aceder a diretórios físicos na máquina e dessa forma encontrar informação importante, como arquivos de configuração de base de dados, credenciais, entre outros.
- **Redirecionar para sites maliciosos**: Normalmente é utilizado para, a partir de um site fidedigno, redirecionar o utilizador para sites Web sob controlo dos atores maliciosos.

Além dos ataques acima descritos é de referir que os atores maliciosos exploram ainda vulnerabilidades em componentes que as aplicações usam, exploram também os sistemas de controlo de acesso e de gestão de sessões e casos em que a configuração da aplicação Web e o servidor que a aloja não esteja devidamente feita e validada.

2.3 Machine Learning e a Cibersegurança

A preocupação para proteger as aplicações de ciberataques não é um tópico novo. Porém apesar do aumento de novas abordagens de segurança existem sempre falhas que podem levar a ataques que, tendem a ser cada vez mais sofisticados e difíceis de detetar [57]. Para lidar com o problema dos ataques, procura-se desenvolver novas técnicas de deteção e prevenção. Neste âmbito surge a inteligência artificial (IA) que traz novas formas e abordagens de responder aos ciberataques. Através da IA procura-se analisar grandes volumes de dados, dar respostas rápidas às tentativas de ataques e mesmo antecipar situações de ataque.

Machine learning assume-se como uma parte de inteligência artificial que permite a um sistema aprender padrões nos dados e não através de instruções programadas e explicita em código. Contudo, para chegar à etapa de prever padrões nos dados e dar resultados coerentes dos mesmos é necessário efetuar várias validações e utilizar procedimentos para garantir que o conjunto de dados utilizado é robusto. Quanto maior for o nível de confiança e o volume dos dados melhor se torna para obter um modelo mais preciso. Após treinar um modelo espera-se fazer previsões sobre os novos dados de entrada para obter respostas aos mesmos, ou seja, as previsões realizadas sobre os novos dados de entrada são realizadas através dos dados treinados em que o algoritmo deteta padrões a que consegue dar uma resposta [23].

Existem técnicas utilizadas para melhorar as previsões dos modelos que podem variar consoante o contexto onde são aplicadas ou o tipo de dados onde é aplicado. De certa forma, os algoritmos utilizados dependem da técnica usadas nos dados [23]:

- **Aprendizagem supervisionada** (Supervised learning)

  A aprendizagem supervisionada passa por um processo de análise para determinar a melhor maneira para classificar os dados e os valores utilizados de classificação. Esta assenta em atribuir rótulos/classes aos dados de maneira a dar um significado aos mesmos. Por exemplo, um conjunto de dados que se referem a características das frutas, atribuir a cada dado a devida categorização da fruta a que se refere (maça, pera, entre outros). Desta forma, o algoritmo aprende a classificar novos dados tendo em conta o conjunto de dados de treino e a classificação atribuída aos mesmos.

- **Aprendizagem não supervisionada** (Unsupervised learning)

  Esta aprendizagem é utilizada quando os dados não estão classificados, ou seja, parte do princípio de analisar e processar grandes quantidades de dados, de forma, a compreender o significado dos mesmos para atribuir um rotulo nos mesmos sem intervenção humana. Esta técnica é utilizada em aplicações de média social como Facebook que produz grandes quantidades de dados que não estão classificados.

- **Aprendizagem por reforço** (Reinforcement learning)

  Esta aprendizagem é uma técnica baseada no comportamento, ou seja, o algoritmo recebe o *feedback* da análise dos dados levando o utilizador a receber o melhor resultado. Desta forma, o sistema aprende por tentativa erro, uma vez que não é treinado por um conjunto de dados definidos. Por isso, o resultado do *feedback* torna-se importante para obter uma sequência de decisões certeiras o que leva à melhor solução do problema.

- **Aprendizagem profunda** (Deep learning)

  Parte do princípio de a máquina conter redes neuronais para desta forma aprender através dos dados de maneira iterativa. Desta forma, a máquina consegue melhorar a tarefa destacada através dos dados, sem ser diretamente programada.

No âmbito da cibersegurança o recurso ao *Machine Learning* tem vindo a crescer. A utilização de *Machine Learning* é vista como uma forma a encontrar uma solução capaz de se auto treinar, ou seja, aprender padrões nos dados para dar respostas certeiras aos novos dados de entrada. Isto torna-se uma mais-valia, pois de forma eficiente e mais rápida se responde a ataques não os deixando atingir um estado critico de segurança. Um sistema de segurança gerido de forma automática pode reduzir o tempo gasto em tarefas de gestão e modelação de novos modelos de previsão o que permite que as organizações usem os seus recursos para outras tarefas, ou seja, tornando a cibersegurança mais simples, mais barata e mais eficaz. Claro que, como já referido isto só se consegue atingir através de um árduo trabalho de pesquisa e análise sobre os dados utilizados para prever padrões de ataques de maneira a estes dados representarem a maior gama de possíveis cenários de ataque. Para isto é necessário recolher um conjunto de possíveis cenários e obter dados de fontes fidedignas de forma a abranger o maior número de casos de ataques que possam ocorrer. Contudo, não nos referimos especificamente à quantidade de dados, mas sim à qualidade dos mesmos. Posteriormente estes têm de ser tratados e limpar o ruído dos mesmos se aplicável [5]. Tendo um conjunto de dados bem definido pode-se usar *Machine Learning* para treinar os dados e encontrar padrões sobre possíveis ataques. Ou seja, o objetivo passa por ter um sistema capaz de alertar sobre ataques através da semelhança com casos conhecidos e que seja capaz de aprender a detetar novos padrões sem ter prévio conhecimento sobre o mesmo. É este ponto que tem cativado investigadores e empresas de cibersegurança a optarem por *Machine Learning*. A título de exemplo, a Microsoft tira partido de *Machine Learning* para construir o antivírus Windows Defender Advanced Threat Protection (ATP) [18, 1].

Relativamente à adoção de *Machine Learning* para endereçar problemas de cibersegurança em aplicações Web, a abordagem apresentada em [62] oferece uma solução para software-defined networking (*SDN*) para serviços que estejam alojados na nuvem que tenham grande ocorrência de tráfego e pedidos Web. A solução oferece uma abordagem baseada em *Machine Learning*, escalável em big data capaz de prevenir tentativas de ataques de *SQL Injection*. Para treinar o modelo foram utilizados os algoritmos de TCM SVMTCM SVM (Two-Class Support Vector Machine) e Regressão logística de duas classes (TC LR). No caso em concreto, os

autores exploram a estrutura do SQL. O SQL é constituído por palavras-chave (*tokens*) escritas em inglês. Estes *tokens* dão origem a um conjunto de lógica denominado de *query* utilizada para fazer pedidos à base de dados e desta forma, consoante a implementação retornar o que foi solicitado, por exemplo:

"**SELECT** name **FROM** user **WHERE** id = 1 **OR** 1=1".

Como os autores referem uma tentativa de SQL Injection pode ser feita da seguinte forma através do URL solicitado. Veja-se o exemplo apresentado na Figura 1 onde o ator malicioso força uma tentativa de obtenção de dados na querie no sentido de fazer com que o *SGBD* retorne sempre verdadeiro.

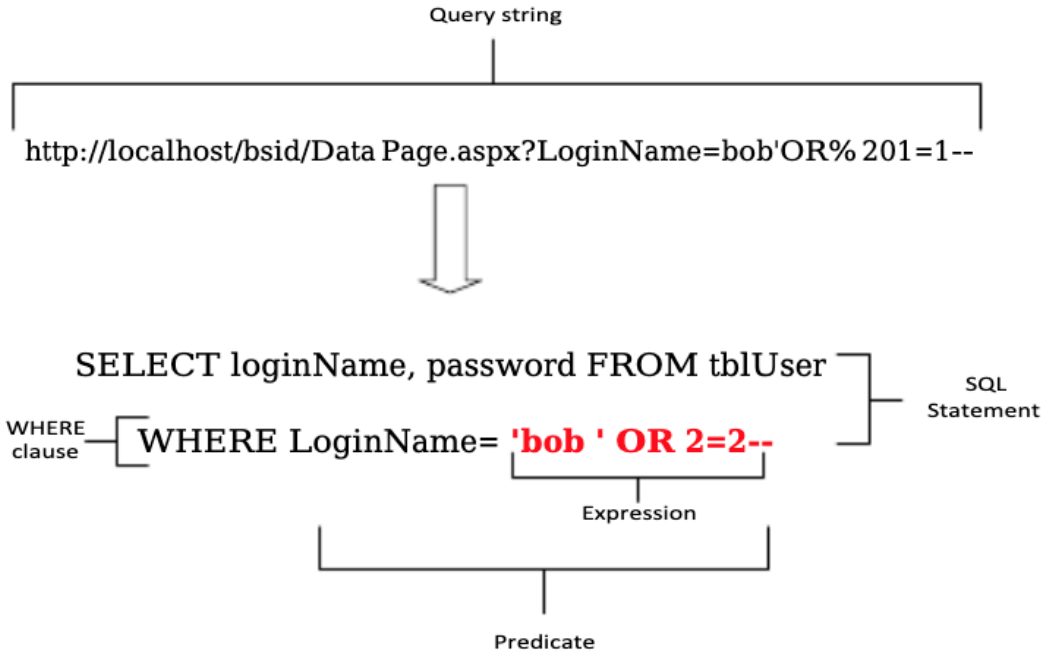


Figura 1: Tentativa de SQL Injection a partir do url. [62]

Fase à estrutura do SQL o autor aplicou autómatos finitos não determinísticos (NFA) implementados em expressões regulares (RegEx) para detetar tentativas de injeção de SQL através dos pedidos Web. O conceito de autómatos finitos é aplicado como uma estratégia para tentar encontrar todas as possíveis combinações que sejam possíveis realizar através de uma palavra para tentar bloquear qualquer técnica de extração de dados maligna. Na Figura 2 é ilustrado como se comporta o autómato finito determinístico.

Alphabet ∑= {b,o}	Transitions			Accepted member strings
	0,1	1,2	2,3	
	b	o	b	bob
	b	o	o	boo
	o	o	b	oob
	o	o	o	ooo
	o	b	o	obo
	b	b	b	bbb
	b	b	o	bbo
	o	b	b	obb

Figura 2: Interpretação de uma string. [62]

Como base nas combinações possíveis, são derivados padrões de deteção de injeção de SQL e foi criado um modelo de aprendizagem supervisionada de forma a prever palavras que coincidam com os tokens padrão de SQL. Os modelos foram treinadas e avaliadas na curva ROC com o TC SVM com resultados bastante promissores. De acordo com os resultados a curva AUC (derivada da curva ROC) obteve um valor 0,986 em que a presença de *SQL Injection* era classificado com a classe binária de um e os pedidos Web normais eram classificados com zero. No caso em estudo, os autores depararam-se com um desequilíbrio entre classes (número de casos em cada classe). Para ultrapassar esse desequilíbrio utilizaram a técnica SMOTE tornando o dataset balanceado. O SMOTE teve como objetivo melhorar a *accuracy* e o F1 score. Tendo em conta o passo anterior para tornar o dataset balanceado o autor efetuou a divisão dos dados atribuindo 80% para treino e 20% para validação. Submeteu os dados aos algoritmos TC LR e TC SVM. Ambos os algoritmos foram rápidos ao treinar o modelo e devolveram valores altos de *accuracy* (cerca de 0,984 e 0,986 para TC LR e TC SVM perante a curva ROC).

Há autores que se focam na deteção automática de sites vulneráveis antes de se tornarem maliciosos. Tal foca-se em identificar Websites que possam tornar-se maliciosos ou involuntariamente hospedar conteúdo malicioso. Como os próprios autores mencionam no estudo efetuado em 2012 [51], 80% dos sites que alojam conteúdo malicioso eram servidores Web que pertenciam a terceiros que não tinham conhecimento do malware lá alojado. Os autores referem ainda que atualmente as abordagens conhecidas focam-se em detetar no presente *software* malicioso que o servidor Web esteja a hospedar. Nesse sentido, eles propõem uma

metodologia para identificar servidores Web que possam vir a tornar-se maliciosos. Os autores focam-se num algoritmo de classificação que se foca em dois pontos:

1. Detetar se um servidor tem probabilidade de se tornar malicioso ou caso seja malicioso este é analisado exaustivamente para detetar se o site hospedado é de cariz malicioso.
2. Adaptar-se às ameaças emergentes, ou seja, o classificador determina se o site analisado tem presente um conjunto de métricas de análise como a utilização de um determinado CMS, estrutura das páginas da Web e presença de determinadas palavra-chaves em páginas para determinar alguma anomalia, fazendo a comparação com sites que foram previamente determinados como sendo maliciosos.

Para construir o modelo os autores treinaram 444 519 sites que continham no total 4 916 203 páginas Web. O estudo conseguiu prever que os sites ao fim de um ano ficavam comprometidos com uma taxa de verdadeiro positivo de 66% e a taxa de falsos positivos de 17%. Tendo em conta que este estudo se baseia em prever o futuro comprometimento do Web site estas taxas de acerto são bastante encorajadoras. Para a construção do dataset foram utilizadas várias fontes de dados disponíveis com várias listas negras de sites maliciosos e com o registo de quando o site foi registado como sendo maliciosos. Assim foi possível determinar o espaço temporal para um site se tornar um potencial risco. A Figura 3 mostra a título de exemplo que os sites maliciosos têm normalmente poucas páginas associadas.

Figura 3: Função distributiva do número de páginas analisadas [51].

Ao longo do estudo do artigo os autores fazem referência a várias abordagens para analisar profundamente um site para detetar anomalias. Uma das abordagens referidas é analisar a

estrutura da página Web denominada por DOM (Figura 4). Tal análise permite obter a hierarquia e disposição do conteúdo da página como as “tags” utilizadas para fornecer conteúdo como imagens, anúncios, ligações (*links*), entre muitos outros.

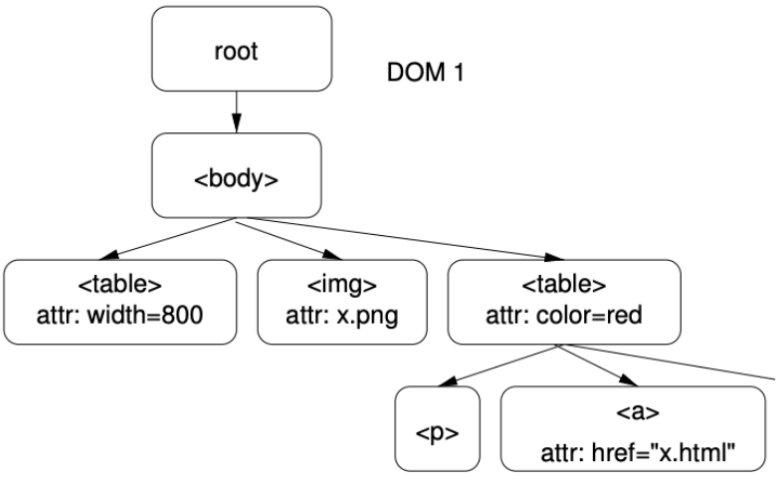


Figura 4: Composição de uma página Web em termos de arquitetura. [51]

Determinar se um site Web poderá vir a tornar-se malicioso é muito interessante, pois permite tirar conclusões sobre o site e qual o seu intuito, ou seja, como os próprios autores descrevem um site que mencione muitas vezes a palavra “investimento” tem grandes hipóteses de pertencer ao setor financeiro. Isto torna-se muito enriquecedor para o modelo utilizado no classificador, pois permite aumentar a dinâmica e a quantidade de variáveis para classificar os potenciais sites maliciosos. Em suma, como os próprios autores referem apesar de existirem algumas lacunas como, por exemplo, um ator malicioso autenticar-se com a conta de um terceiro e o classificador não conseguir adaptar-se a esses casos, o estudo em causa obteve resultados bastante positivos.

2.4 Ciência de dados

A Ciência de dados é uma área transversal a diversos sectores, o seu valor foca-se nas conclusões retiradas da análise aos dados que propiciam vantagens e ilações importantes. Do ponto de vista de *Machine Learning* o objetivo desta análise aos dados é permitir selecionar o melhor algoritmo, determinar os dados mais importantes (precisos e limpos) e os modelos mais apropriados com melhor desempenho. Estes processos juntos permitem dar origem a um modelo robusto, para determinar resultados e também aprender continuamente com os

mesmos.

Os dados são a parte fundamental de ciência de dados. Hoje em dia temos capacidade de obter e processar grandes quantidades de dados e tal é benéfico por permitir aprender e melhorar continuamente o conhecimento que se retira dos dados. Contudo, é necessário que estes sejam de qualidade e tratados para que os modelos de previsão sejam eficientes e robustos. A análise exploratória e preparação dos dados é uma tarefa sensível e demorada, mas se for bem executada leva a obter resultados melhores.

De forma a explicar o conjunto de dados (dataset) e posteriormente algumas referências feitas ao longo deste capítulo é necessário explicar a que se refere uma única linha num grupo de dados. Assim sendo, uma única linha passa por ser uma instância. Os modelos utilizados em *Machine Learning* requerem um conjunto de dados para através dos mesmos aprender padrões e fazer previsões futuras. Para isto, é necessário ter um conjunto de dados de treino robusto para que o algoritmo possa aprender e outro conjunto de dados de validação para testar os resultados obtidos de maneira a garantir que o modelo é eficiente e interpreta corretamente os dados. Torna-se importante existir uma análise e tratamento, pois a qualidade dos dados é muito importante para a previsão de resultados certeiros [7].

Como mostra a Figura 5 existem diferentes tipos de dados.

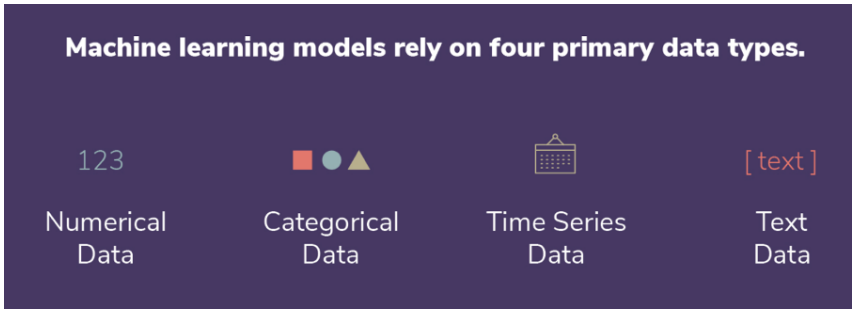


Figura 5: Tipos de forma de dados aceites nos modelos [7]

- **Dados numéricos** (Numerical Data)

  Dados numéricos ou dados quantitativos, são qualquer forma de dados mensuráveis, como medidas, peso, preços entre outros. Os dados numéricos não estão associados a nenhum momento específico, são simplesmente números brutos.

- **Dados categóricos** (Categorical Data)

  Dados categóricos têm um significado de definir ou categorizar algo específico, por exemplo, género, tipo de fruta entre outros. Estes dados são utilizados para fazer agregações,

como classe social.

- **Dados de série temporal** (Time Series Data)

  Dados de série são dados que estão associados a momentos específicos no tempo. Estes dados são utilizados para fazer analise sobre determinado período.

- **Dados de texto** (Text Data)

  Dados de texto dizem respeito a palavras ou frases que podem fornecer informação relevante para os modelos de aprendizagem.

2.4.1 Análise exploratória

A análise exploratória de dados refere-se ao processo de realização de uma análise de investigação critica sobre os dados de forma a descobrir padrões, detetar incoerências, extração de informação importante, análise estatística e representações gráficas. A utilização de gráficos como boxPlot torna a análise dos dados muito mais fácil uma vez que permite de forma simplificada comparar as variáveis, ou seja, é uma forma padronizada de exibir a distribuição dos dados. Desta forma, torna-se relevante entender primeiro os dados e perceber a informação que podemos extrair deles antes de começar a executar qualquer processo que se baseia nos mesmos [40].

2.4.2 Preparação dos dados

A preparação dos dados é um dos processos mais importantes de qualquer projeto de *Machine Learning*. É esta etapa que nos vai ajudar a atingir bons resultados e confiáveis. Contudo, existem etapas posteriormente referidas que podem ser diferentes entre projetos. Em projetos de classificação ou regressão os dados raramente são utilizados em bruto sem qualquer tratamento, isto porque pode acarretar os seguintes problemas:

- Os algoritmos de *Machine Learning* exigem dados de valores numéricos.
- Alguns algoritmos têm regras associadas aos dados.
- Pode ser necessário corrigir o ruído ou erros de cálculo nos dados.

Desta forma, todos os dados devem ser pré-processados e tratados antes de serem utilizados em algum modelo de previsão. Contudo, o tipo de tratamento pode diferenciar consoante o projeto em causa.

Esse tratamento inclui as seguintes tarefas:

- Limpeza de dados para identificar e corrigir erros de cálculos.
- Selecionar e identificar as variáveis relevantes para o modelo.
- Alterar a escala ou transformar os dados.
- Identificar novas variáveis a partir das existentes.
- Redução das dimensões dos dados, por exemplo, anonimizar os dados para uma certa escala ([0,1]).

Estas tarefas implicam tempo e estudo árduo para conseguir obter de forma clara toda a informação importante e precisa dos dados. O objetivo concreto deste processo é descobrir como expor de forma suscita a estrutura subjacente do problema dos algoritmos de aprendizagem, ou seja, o foco é tentar preparar os dados e enriquecer com toda a informação importante de análise efetuada para depois com a utilização de *Machine learning* usufruir dessa informação e atingir resultados fiáveis. Isto claro, para conseguir recolher esta informação por vezes é necessário utilizar diferentes métodos e abordagens para conseguir extrair os dados fulcrais nas instâncias. Por vezes, derivado do grande número de métodos e abordagens temos de fazer um grande trabalho de investigação para cada variável para escolher o melhor método e parâmetros de configurações essenciais. Também devemos sempre tomar decisões em prol do projeto e do algoritmo a utilizar, por exemplo, dependendo do projeto e da informação que os dados acarretam podemos querer manter valores vazios ou a null, uma vez que dependendo do algoritmo este pode lidar facilmente com esses valores. Em suma, esta tarefa requer bastante tempo de análise minuciosa e preparação dos dados uma vez que decisões ou cálculos efetuados erradamente pode levar a resultados péssimos dos modelos de previsão, derivado à ocorrência de under-fitting ou over-fitting.

2.4.3 UNDER-FITTING VS OVER-FITTING

Antes de explicar estes conceitos é extremamente relevante explicar o que é um modelo que simplesmente baseia-se no resultado gerado quando um conjunto de dados de entrada é treinado, ou seja, mapeia dados de entrada para saída [23]. Por exemplo, um algoritmo cria um modelo de previsão e todas as previsões feitas para novas instâncias são baseadas no conjunto de dados de treino do modelo, ou seja, para prever um preço de um carro, poderíamos criar um modelo que através da potência do motor preveja um preço. De certa forma, um modelo representa uma teoria sobre um problema que existe e permite fazer a relação entre os dados, neste caso específico entre a potência do motor e o preço do carro. Por outras palavras, durante o treino o modelo recebe os recursos necessários para mapear automaticamente a informação, e após essa tarefa prever novas instâncias e dar respostas a esses dados [65], ou seja, generalizar bem os dados para obter resultados fidedignos.

A utilização de *Machine Learning* torna-se útil, pois programas normais/tradicionais apenas conseguem dar uma resposta para um conjunto de dados de entrada que conhecem e foram programadas para tal [34]. Por isso, se torna relevante construir um modelo generalizado e à medida que melhoramos o conhecimento surgem questões sobre os modelos relacionadas com Over-fitting, Under-fitting e bias-variance trade-off. São problemas que estão na base da criação de modelos e que é imprescindível referir os mesmos.

Pode-se afirmar que um modelo bem generalizado é um modelo em que não ocorre over-fitting nem under-fitting, isto passa por explicar estes conceitos e subjacente tirar as conclusões sobre esta afirmação. Para explicar estes conceitos foi utilizado como exemplo o seguinte conjunto de dados.

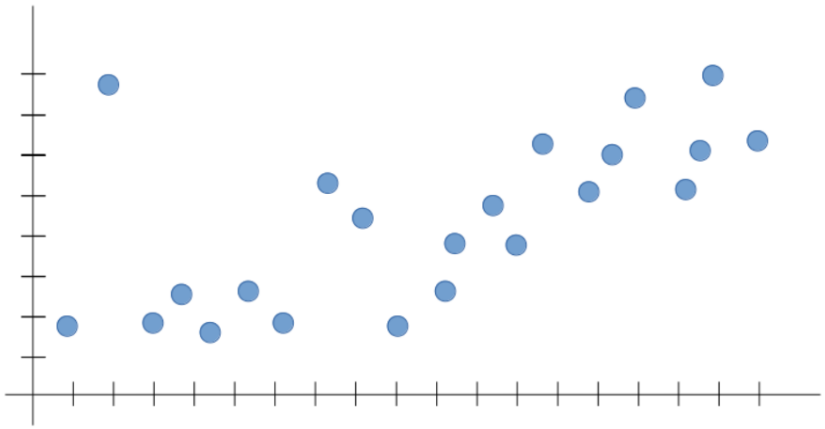


Figura 6: Conjunto de dados utilizado para explicar conceito de under-fitting vs over-fitting [6]

É de referir que o eixo do X representa os dados de entrada e o eixo do y representa os dados de saída/resposta. Assim, será utilizado o conjunto de dados ilustrado na Figura 6 para explicar e enquadrar o conceito de under-fitting e over-fitting. Compreender as configurações ótimas do modelo é importante para entender o problema de precisão baixa do modelo. Desta forma, é possível determinar se num modelo de previsão ocorre under-fitting ou over-fitting analisando o erro de previsão nos dados de treino e nos dados de validação, como referido em [8] e na Figura seguinte 7.

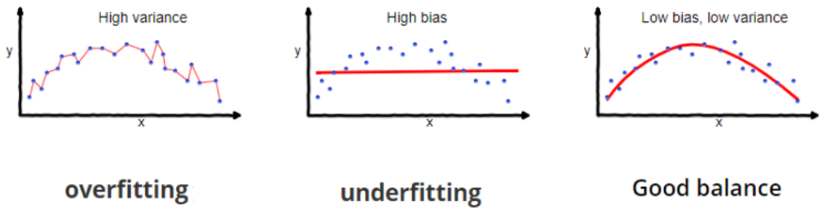


Figura 7: Problemas que podem estar associados aos dados [49]

Torna-se relevante encontrar uma configuração que seja relativamente ótima na qual a linha produza uma distância menor nos pontos de dados, como podemos observar na linha produzida para o conjunto de dados analisados.

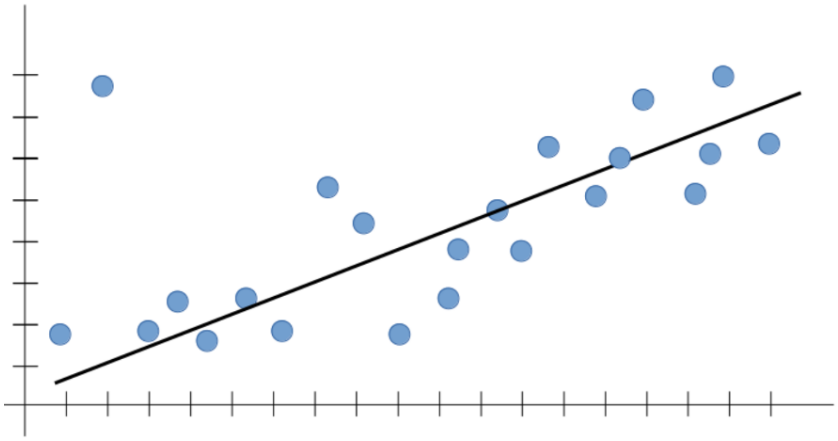


Figura 8: Resultado esperado num conjunto de dados considerado normal [6]

O objetivo é prever uma resposta para novos dados de entrada que não estejam no conjunto de dados de treino, através da Figura 8 é possível visualizar através da linha que é possível fazer previsão para novos dados de entrada.

Contudo, antes de referir os pontos comuns que acontece nos modelos é importante referir que ao construir o modelo este pode ser constituído por dados que podem ser de sinal e ruído.

Sinal é a informação com que nos preocupamos e que permite que ao modelo generalizar para novos dados de entrada. Pelo contrário o ruído é tudo que não importa para o modelo. São erros de cálculos, formulas, leituras de dados que causam variações na resposta do modelo [41].

2.4.3.1 OVER-FITTING

Na execução do algoritmo de treino o objetivo é diminuir a distância de cada ponto a cada iteração. Por vezes, se o número determinado de iterações for elevado faz com que a distância de cada ponto à linha seja mínimo, isto significa que a linha fica ajustada a todos os pontos incluindo ruído e que contenha padrões desnecessários para tornar o modelo balanceado, como mostra na Figura 8. Caso o número de iterações do algoritmo seja elevado podemos ocorrer no seguinte problema em que a linha fica ajustada a todos os pontos.

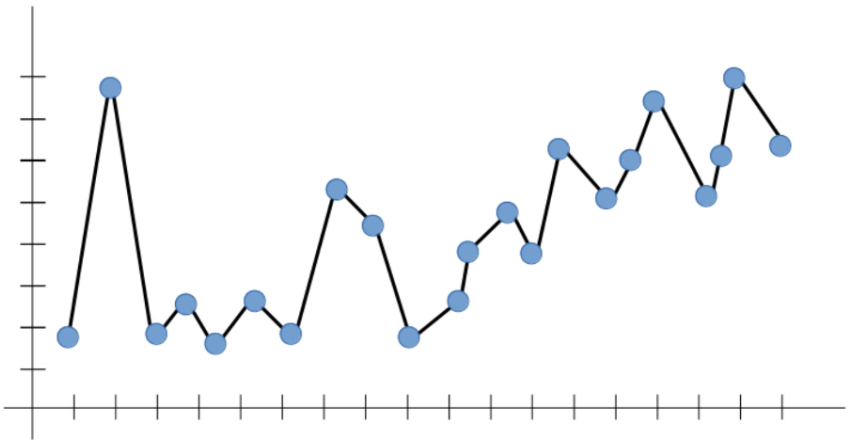


Figura 9: Resultado de ocorrência de over-fitting [6]

A utilização de algoritmos de regressão linear passa por ter uma linha que se ajuste à tendência dominante. Como é óbvio se o algoritmo não conseguir adaptar-se à tendência dominante não poderá prever uma saída provável para novos dados de entrada que não estejam presentes nos dados de treino. Podemos assim chegar à conclusão que para novos dados de entrada que estejam para além dos limites da linha da Figura 9 o modelo não seria capaz de prever um resultado fiável. Isto ocorre devido à aprendizagem do modelo que teve um elevado número de iterações no processo de treino, apenas conseguindo "prever" dados conhecidos, sendo que para dados que desconheça a resposta dada não seja confiável. Por outro lado, também pode ocorrer se a qualidade dos dados for fraca e não existirem diferentes cenários de aprendizagem. Isto ocorre, uma vez que o modelo captura o ruído com o padrão subjacente nos

dados, criando um modelo com baixo *viés* e alta variação [49]. Desta forma, este modelo não serve para efetuar previsões uma vez que para dados desconhecidos este não consegue dar uma resposta fiável, ou seja, ter uma linha ajustada às tendências dominantes [6].

2.4.3.2 UNDER-FITTING

Ao contrário do que foi explicado para o problema de over-fitting que pode ocorrer quando o modelo é treinado em demasia o under-fitting pode ocorrer quando o modelo é treinado com poucas iterações e não conseguiu absorver os padrões suficientes a partir dos dados de treino e possivelmente nem encontrar a tendência dominante, isto denomina-se de under-fitting.

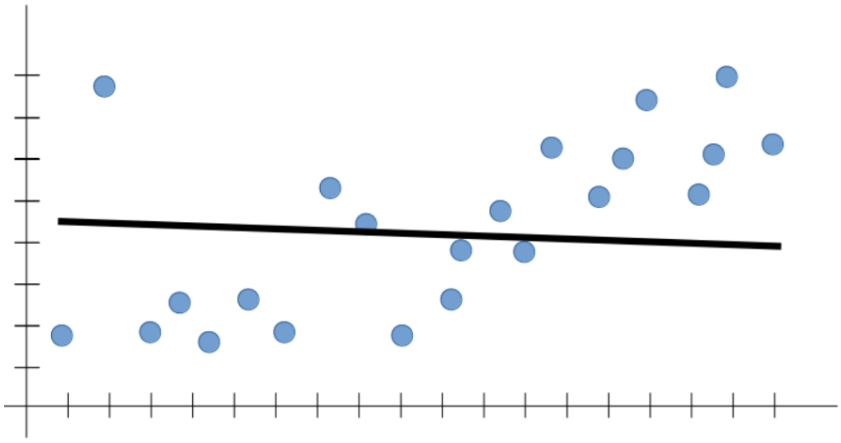


Figura 10: Resultado de ocorrência de under-fitting [6]

De certa forma, under-fiting é derivado da má configuração do modelo e este não aprendeu o suficiente com os dados de treino e não conseguiu estudar os padrões necessários para efetuar as previsões, isto faz com que ocorra a baixa generalização do modelo e o resultado das previsões não são confiáveis. Como podemos visualizar na Figura 10 o resultado é uma linha simples em que não existe praticamente nenhuma relação entre os dados[4]. Este problema ocorre quando um modelo não consegue aprender ou relacionar os padrões nos dados o que normalmente produz modelos com alto *viés* e baixa *variação*. Normalmente ocorre quando temos pouca quantidade de dados para construir o modelo ou tentamos construir um modelo linear com dados não lineares, tornando o modelo simples para tentar encontrar padrões complexos [49].

2.4.3.3 Bias-variance trade-off

De forma simplificada passa por encontrar uma medida certa entre over-fitting e under-fitting encontrando o equilíbrio entre estes dois problemas e o equilíbrio entre viés e variação, ou seja, o objetivo é ter um modelo bem generalizado e conseguir um baixo viés e baixa variação. Por isso, se o modelo for simples e tiver poucos parâmetros poderá ter alto viés e baixa variação. Por outro lado, se tiver um elevado número de parâmetros, terá alta variação e baixo viés, por isso, torna-se importante encontrar o equilíbrio entre viés e variação de maneira a minimizar o erro total.

Total Error = Bias^2 + Variance + Irreducible Error

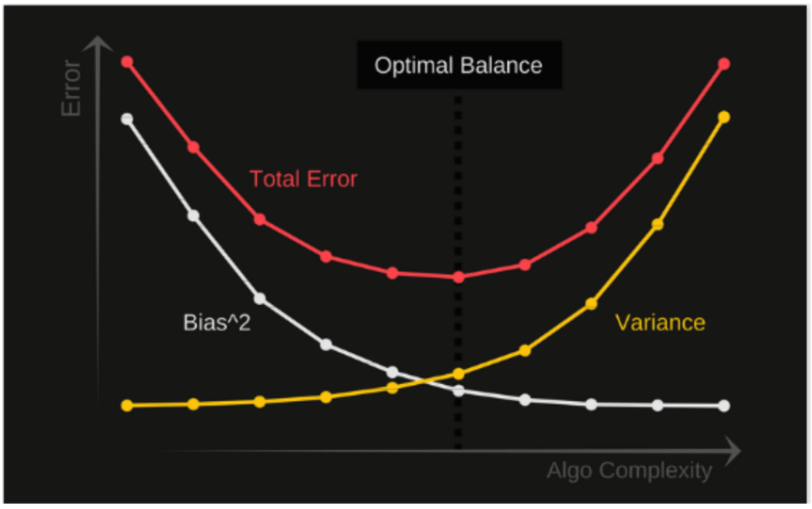


Figura 11: Demonstração da relação entre Bias e variance para ter um dataset balanceado. [6]

Como podemos observar existe uma relação entre viés e variação:

- Aumentando o viés diminui a variação
- Aumentando a variação diminui o viés

Por isso torna-se importante escolher a melhor configuração do algoritmo e a ter um conjunto de dados robusto e com qualidade de forma a encontrar uma relação balanceada entre estas duas variantes de forma a diminuir o erro e evitar os problemas descritos anteriormente.

2.4.4 Análise de dados

Avaliar o resultado dos algoritmos é uma tarefa importante e essencial para perceber o resultado e a confiança do mesmo. O modelo pode retornar resultados satisfatórios quando estes são avaliados com métricas e sob uma análise minuciosa para perceber se esses resultados não estão manipulados (ocorrer over-fitting ou under-fitting). Muitas das vezes o resultado dos algoritmos são avaliados apenas com uma métrica, normalmente pela precisão. Geralmente para ter a certeza que o resultado do modelo é robusta a melhor decisão é utilizar outras métricas para também ajudar a medir o desempenho do mesmo e julgar verdadeiramente o modelo utilizado. Dependendo dos resultados destas métricas por vezes pode levar a escolher outro algoritmo para além do estipulado antes desta análise, desta forma, torna-se importante perceber e analisar as métricas de seguida explicadas para tomar decisões antes de prosseguir com o projeto e a escolha do algoritmo certo [32].

- **Precisão** (Accuracy)

  A precisão define-se por ser a divisão entre o número de casos corretos e o número total de casos de entrada.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Figura 12: Formula de cálculo da precisão [32].

  Funciona bem se existir um equilíbrio mutuo entre classes. Por exemplo, se existir 98% de uma classe e 2% de outra classe, facilmente o modelo poderia obter 98% de precisão. Enquanto, num ambiente equilibrado de 60% de amostra de uma classe e 40% da outra classe a precisão seria de 60%. Esta métrica é otima, mas dá uma falsa sensação de alcançar uma alta precisão.

- **Matriz de confusão** (Confusion Matrix)

  Esta métrica é ótima para descrever por completo o desempenho do modelo, ou seja, descreve diferentes previsões e resultados de teste e os compara com os valores do mundo real.

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figura 13: Tabela de matriz de confusão [42].

  - Positivos verdadeiros (True Positives): os casos em que previmos SIM e a saída real também foi SIM.
  - Negativos verdadeiros (True Negatives): os casos em que previmos NÃO e a saída real foi NÃO.
  - Falsos positivos (False Positives): os casos em que previmos SIM e a saída real foi NÃO.
  - Falsos negativos (False Negatives): os casos em que previmos NÃO e a saída real foi SIM.

- **Pontuaçao F1** (F1 Score)

  A pontuação F1 é a média harmónica determinada pela precisão e a recuperação. O intervalo de resposta é entre [0, 1]. Esta métrica informa quanto preciso é o classificador (quantas instâncias são classificadas corretamente), bem como quão robusto ele é (não perde um número significativo de instâncias). Quanto maior a pontuação desta métrica, melhor é o desempenho do modelo.

$$F_1 = \left(\frac{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}}{2} \right)^{-1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

Figura 14: Formula para obter a pontuação F1 [59].

2.4.4.1 MAE e RMSE

O erro médio absoluto (MAE) e o erro médio quadrático da raiz (RMSE) são duas das métricas mais utilizadas para análise de modelos para medir a qualidade de variáveis contínua como os resultados obtidos pelo modelo de previsão [26].

Estas métricas servem para medir o erro da qualidade dos dados para previsão de resultados e ver o impacto de outliers nos dados e isto pode diferir na escolha da métrica a utilizar para medir a qualidade dos dados. É de referir que quando obtemos valores baixos nestas métricas significa que a qualidade dos dados é boa [39].

Média do erro absoluto (MAE)

O MAE mede a magnitude média dos erros num conjunto de previsões com os dados de input, ou seja, faz o somatório dos valores absoluto do erro do cálculo da diferença entre valor de entrada e de saída (previsão). De uma forma simplificada realiza as etapas de encontrar os valores absolutos do erro entre o valor de entrada e valor de saída, como mostra a seguinte fórmula:

$$\mid valordeentrada - valordesaida \mid \tag{2.1}$$

O erro é calculado através da média dos valores obtidos a partir da fórmula discriminada anteriormente, como mostra na seguinte tabela:

Valor de entrada	Valor de saída	diferença	erro absoluto
20	30	-10	10
100	126	-26	26
50	65	-10	10
			Média = 12

Tabela 2.1: Cálculo para obter o valor de MAE.

Como podemos constatar na Tabela 2.1 não importa o sinal dos dados, ou seja, se é positivo ou negativo para efetuar o cálculo de MAE [26].

Simplificando a lógica descrita na Tabela 2.1 obtemos a seguinte fórmula para calcular o MAE.

$$MAE(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^{n} |y_i - \hat{y_i}| \text{ (1)}$$

Raíz quadrada do erro (RMSE)

Também mede a magnitude média do erro. Uma explicação mais básica é raiz quadrada da diferença ao quadrado dos valores de entrada e de saída, tendo a seguinte equação matemática [26].

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{d_i - f_i}{\sigma_i}\right)^2} \quad (2)$$

Semelhanças

Ambas as expressões calculam o erro médio de previsão do modelo. Quanto mais baixo for o valor obtido destas duas equações apresentadas melhor, significa que tem um erro baixo de previsão no nosso contexto. Por isso, destas fórmulas o expectável é obter sempre valores baixos [26].

Diferenças

O RMSE pode torna-se mais útil quando queremos dar mais relevância a valores indesejáveis, uma vez que os erros são elevados ao quadrado antes da média, o que significa que o RMSE atribui um peso relativamente alto a erros grandes. O MSE tem a virtude de ser robusto para com outliers [26].

Nome da Equação	Operador utilizado	Robusto para com outliers?
MAE	Erro absoluto	Sim
RMSE	Raíz quadrada	Não

Tabela 2.2: Análise das equações descritas [39]

Conclusão

Torna-se importante selecionar a melhor métrica para utilizar nos dados tendo em conta a sua natureza. Tendo em conta o conjunto de dados da solução foi decidido dar ênfase à métrica MAE uma vez que esta lida facilmente com outliers, um dos problemas que é possível constatar no conjunto de dados utilizado. Como já referido o objetivo é conseguir obter resultados baixos para a métrica MAE.

2.5 Análise de dados - *Machine Learning*

Com o conjunto de dados preparado e devidamente analisado coloca-se a questão sobre o melhor algoritmo para o caso em estudo. O objetivo é obter uma resposta rápida a novos dados de entrada e por isso o desempenho e a taxa de precisão são ambos relevantes. Numa fase inicial da análise optou-se por fazer estudo comparativo entre Gradient Boosting Machine (GBM) e Random Forest (RF). Tanto RF como GBM são baseados em aprendizagem por conjunto de modelos (ensemble learning) que combina vários modelos de forma a produzir um único modelo de modo a obter a melhor previsão, ou seja, combina os resultados das várias árvores treinadas para obter o melhor modelo em termos de desempenho e previsão. O processo de divisão e agregação é esquematizado na Figura 15.

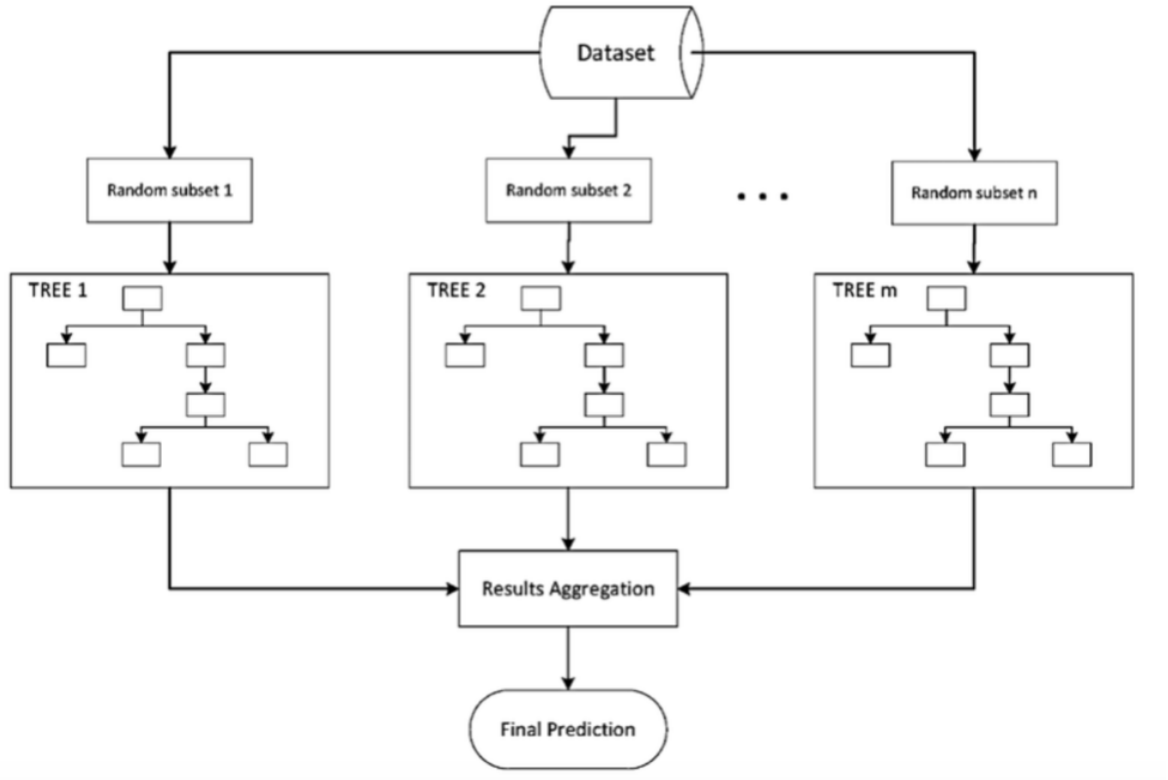


Figura 15: Como as arvores são formadas para dar origem a um resultado [30]

A diferença entre GBM e RF reside na forma como as árvores são construídas, na ordem e como os resultados são agregados. Estudos como [44] e [52] mostram que o desempenho dos algoritmos GBM tende a ser melhor que as RF desde que os parâmetros sejam bem definidos [13].

O GBM difere na forma como constrói as árvores, ou seja, construir uma árvore de cada vez e cada árvore nova ajuda a corrigir a árvore anterior de forma a diminuir o erro. Os algoritmos GDM:

- São aplicados em aprendizagem supervisionada.
- Aplicados em conjunto de dados que são *unbalanced*, como é o caso em estudo.
- Apesar de ser dificil encontrar os parâmetros de configuração ao se obter estes dados corretamente a performance e a taxa de previsão são bastantes promissoras.
- Lida facilmente com *outliers* comparativamente ao de RF.
- Lida facilmente com valores em falta.

As Random Forest [29] treinam cada amostra de dados em árvores independentes, o que ajuda a tornar o modelo mais robusto do que uma única árvore de decisão. Os algoritmos RF :

- São usados em grandes volumes de dados, com tempos de resposta bastante bons.
- Lidam facilmente com valores em falta.
- Encontram os parâmetros certos de configuração.
- Pode levar à ocorrerencia de *overfitting* quando existem outliers no conjunto de dados [28].

Assim, considerando as vantagens e desvantagens entre o GDM e RF optou-se por escolher algoritmos da categoria gradient boosting uma vez que o foco é o desempenho, lidar com valores vazios e os valores limite [3] que fazem parte dos dados. Para suportar a escolha foi ainda realizado uma análise com RF, sendo os resultados apresentados na Figura 16.

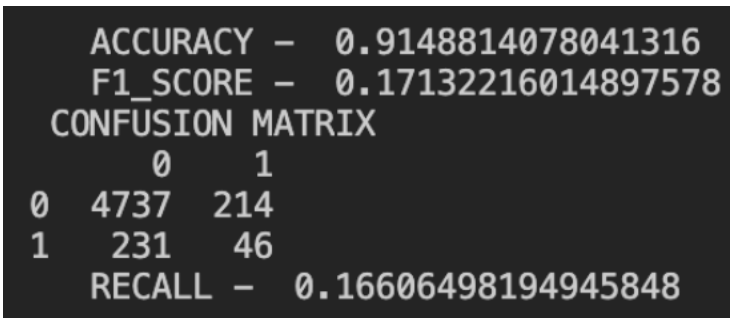


Figura 16: Resultado obtido para a Random Forest

Os resultados não são nada satisfatórios, na medida em que o recall e o F1_score são baixos evidenciando problemas de classificação. Ao longo das próximas subsecções apresenta-se a análise que visa encontrar o melhor algoritmo a usar dentro dos algoritmos GBM.

2.5.1 Análise do melhor Gradient Boosting Machine (GBM) a utilizar

Pelas pesquisas verificou-se que existe uma gama muito abrangente de algoritmos de vários tipos e de configurações diferentes, que se forem bem definidas propiciam melhor desempenho e a melhores resultados. De forma, a otimizar o tempo recorreu-se a uma abordagem automática que assiste ao processo de escolha de parâmetros a usar para os modelos e quais os modelos que produzem melhores resultados. A abordagem envolve a utilização do AutoML incluído no H2O.

2.5.2 Explicação do AutoML

Surgiu da necessidade de ajudar a selecionar de uma vasta gama de algoritmos qual o melhor a utilizar para um conjunto de dados e quais os melhores parâmetros para atingir bons resultados de previsão e de desempenho. Desta forma, o AutoML é uma função desenhada para treinar vários modelos abstraindo o utilizador de criar várias linhas de código ou ações para encontrar o melhor algoritmo para os seus dados de estudo, poupando assim o mesmo de perder muito tempo a procurar a melhor abordagem e configurações, permitindo o mesmo focar-se em processar e tratar os dados para criar um dataset rigoroso. Assim, o AutoML ajuda a melhorar o tempo de trabalho que seria dispensado nesta tarefa árdua e de forma automatizada treina vários modelos de maneira a dar o resultado com o melhor modelo de cada gama de algoritmos, normalmente são os modelos que mais se utilizada no mundo de *Machine Learning* [19].

2.5.3 AutoML

De forma, a determinar o melhor algoritmo da gama GBM, fez-se uso do AutoML. Através do AutoML realizaram-se alguns passos de configuração prévia, tais como:

- Importar os dados de treino (70%), validação (15%) e de teste (15%).

- Definir a variável de output da previsão.
- Selecionar a opção de balancear as classes.

Após o AutoML finalizar foram obtidos os seguintes resultados apresentados na Figura 38.

▾ MODELS

models sorted in order of auc, best first

	model_id	auc	logloss	mean_per_class_error	rmse	mse
0	XGBoost_3_AutoML_20200430_123550	0.9999897879705701	0.0018570301433298768	0.0008789254540751311	0.019287416482910356	0.00037200443458524214
1	XGBoost_1_AutoML_20200430_123550	0.9999864479924981	0.00224233262219216	0.0011317413452818208	0.021236023325998974	0.0004509686867023725
2	XGBoost_2_AutoML_20200430_123550	0.9999739370171226	0.003165600570044567	0.0019974091312955603	0.024778305564296572	0.0006139644266376504
3	XGBoost_grid_1_AutoML_20200430_123550_model_3	0.9999670990471737	0.00426766115155645	0.002031831354282721	0.027759833532640 25	0.000770608357759898
4	XGBoost_grid_1_AutoML_20200430_123550_model_2	0.9999586871357956	0.004428191826028727	0.00217742713309574	0.028777127685054 96	0.0008281230778019567
5	XGBoost_grid_1_AutoML_20200430_123550_model_1	0.9998819512945063	0.004174945873539225	0.001401768347982091	0.02284427641312618	0.0005218609648393131
6	StackedEnsemble_AllModels_AutoML_20200430_123550	0.9998000381773746	0.002851944390284661	0.0011317413452818208	0.021173025272116237	0.0004482969991736728 3
7	StackedEnsemble_BestOfFamily_AutoML_20200430_123550	0.9996895183649376	0.00276556084394082	0.0010245212328881506	0.020575192549538813	0.0004233385484505975 4
8	GLM_grid_1_AutoML_20200430_123550_model_1	0.9996417579178317	0.008203166534578142	0.0031821811488229475	0.03872912020597479	0.001499944751928845
9	XGBoost_grid_1_AutoML_20200430_123550_model_4	0.9985869011327143	0.06566381416388901	0.0035368663712262932	0.07143183020911946	0.005102506367024471

Figura 17: Resultado obtido pelo AutoML

Com base nos resultados, o algoritmo que teve melhor desempenho nas métricas analisadas de acordo com o AutoML é o XGBoost.

2.5.4 XGBoost

Tendo em conta as métricas anteriormente referidas e os seus resultados que serão posteriormente descritos o algoritmo escolhido para este projeto foi o XGBoost. De seguida será feita uma análise ao mesmo como os seus parâmetros de configuração para atingir melhores resultados.

XGBoost é uma implementação que segue o princípio de *gradient boosted* baseado em árvores, que foi desenhado para melhorar o desempenho e a rapidez de resposta. Como já referido existem diferenças na modelação e na criação das árvores de forma a impedir a ocorrência de over-fitting através de uma formalização de modelo mais regularizada desempenhando um melhor desempenho. Como o próprio nome indica é desenhado para utilizar ao máximo todos os recursos de computação para executar rapidamente e otimizar o desempenho na modelação, como o próprio autor Tianqi Chen refere neste artigo [58].

O XGBoost implementa a livraria do “gradient Boosting decision tree algoritm” [50].

Boosting é uma técnica utilizada para adicionar novos modelos para corrigir os erros cometidos pelos anteriores, de forma, a minimizar os erros à medida que são adicionados novos modelos. Assim, estes são criados sequencialmente até que atinja um estado em que não

seja necessário fazer mais melhorias no mesmo. Especificamente o XGBoost apresenta as seguintes características e otimizações a diferentes níveis, como demonstra a figura seguinte.

Funcionamento do XGBoost

Figura 18: Como XGBoost otimiza padrões do algoritmo GBM [33]

- **Paralelização** (Parallelization)

  Para a construção de árvores sequenciais o XGBoost utiliza computação paralela [33].

- **Limpeza de árvore** (Tree Pruning)

  O XGBoost utiliza o parâmetro "max_depth" como valor que especifica a profundidade máxima de uma árvore, ou seja, iterações utilizadas para encontrar padrões nos dados. Valores altos no max_depth podem levar a over-fitting. Este algoritmo utiliza este parâmetro para remover as últimas árvores criadas o que permite melhorar o desempenho computacional [33].

- **Otimização de Equipamento físico** (Hardware Optimization)

  Este algoritmo foi projetado para fazer uso eficiente dos recursos de *hardware*. Desta forma, em cada encadeamento realizado internamente ele armazena buffers com estatísticas necessárias. Outro ajuste importante é que utiliza o espaço em disco disponível para alocar grandes quantidades de dados que não podem ser alocados em memória [33].

Melhorias no algoritmo

- **Regularização** (Regularization)

  Penaliza modelos complexos pela regularização de Lasso (L1) and Ridge(L2) para evitar over-fitting [9].

- **Sensibilização para valores de entrada** (Sparsity Awareness)

  Consegue lidar com grande diversidade de valores de entrada, como valores vazios ou que foram processados pelo método one-Hot encoding utilizado para converter valores do tipo texto para numérico. O XGBoost incorpora um algoritmo de "sparsity-aware split finding" responsável por reconhecer esta dispersão nos dados e encontrar os padrões nos mesmos [33].

- **Divisão dos dados por quantil** (Weighted Quantile Sketch)

  Utiliza o algoritmo de "Quantile Sketch" para encontrar de forma eficiente os melhores pontos de divisão entre diversos conjuntos de dados [33].

- **Validação cruzada** (Cross-validation)

  A cada iteração é feito validações cruzadas automaticamente, não sendo necessário especificar o número de iterações necessárias em cada execução ou fazer a devida análise para encontrar o valor correto de iterações necessárias [33].

Estas melhorias de desempenho e configurações que fazem parte do XGBoost demonstram tornar este algoritmo muito eficiente e rigoroso ao nível de previsão, sendo um dos algoritmos mais utilizados no mundo de *Machine Learning*, pelas seguintes razões:

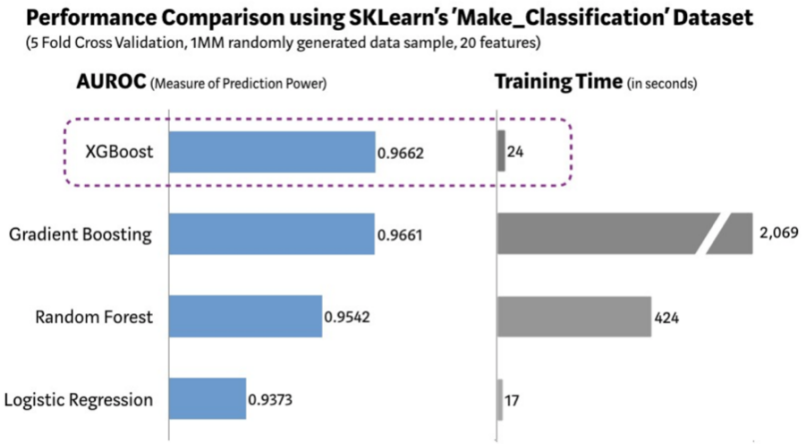


Figura 19: Comparação do XGBoost com outros algoritmos usando o dataset SKLearns Make_Classification [33]

Como já referido a solução apresentada tem como objetivo o desempenho, como podemos ver na imagem o XGBoost demonstra ser um grande candidato à solução.

Parâmetros configuráveis no XGBoost

Existem vários parâmetros que se forem bem configurados podem ser usados para melhorar os resultados de um conjunto de dados . Apenas serão referidos os principais e aqueles que tiveram impacto no desempenho da solução apresentada, que foram os seguintes:

- **eta** [valor padrão=0.3]

  Torna o modelo mais robusto e preciso diminuindo os pesos em cada iteração. Os valores típicos a utilizar são os seguintes]0.01, 0.2[[2].

- **min_child_weight** [valor padrão=1]

  Define a soma mínima dos pesos de todas as observações necessárias para criar um novo nó na árvore [2].

- **max_depth** [valor padrão=6]

  Define-se por ser a profundidade máxima de uma árvore no número de nós permitidos desde a raiz até à folha mais distante da árvore. Utilizado para controlar o over-fitting uma vez que à medida que a profundidade das árvores aumenta o modelo aprende relações mais pormenorizadas para amostras especificas [2].

- **subsample** [valor padrão=1]

  Corresponde à fração de instâncias dos dados para serem amostras aleatórias para cada árvore. O valor a 1 significa que utilizamos todas as instâncias [2].

- **colsample_bytree** [valor padrão=1]

  Define-se por ser a fração de colunas dos dados a serem usadas. O valor a 1 significa que serão utilizadas todas as colunas [2].

Estes parâmetros são utilizados para controlar a complexidade das árvores com o objetivo de melhorar o desempenho na execução do algoritmo de XGBoost. Torna-se importante utilizar as devidas configurações para existir uma boa relação entre Bias-variance trade-off referido na Figura 2.4.3.3.

2.5.5 Ferramentas de Machine Learning

Existem várias ferramentas que podem ser utilizadas para fazer a análise e treino de dados, de forma, a utilizar algoritmos de *Machine Learning* para fazer previsões sobre os dados de entrada, tais como:

- **H2O**

  É uma framework de *Machine Learning* de código-aberto usado principalmente para executar modelos de previsão predefinidos. Ainda assim, foi desenvolvido para garantir a escalabilidade do modelo com algoritmos avançados. De forma, a simplificar o processo de escolha do melhor algoritmo para um conjunto de dados este oferece uma funcionalidade denominada de "AutoML" [36].

- **TENSORFLOW**

  É uma framework de *Machine Learning* de código-aberto usado principalmente como mecanismo computacional que facilita a implementação e utilização de *Machine Learning*, de certa forma, é a base de qualquer modelo. Também é utilizado como biblioteca para escrever gráficos de computação para cálculos vetoriais [36].

- **PYTHON PANDAS**

  É uma biblioteca de código aberto que oferece um vasto conjunto de ferramentas úteis para fazer a análise de dados. Este é utilizado para fazer análise de dados de grande volume de dados, ciência de dados, tratar os dados, agrupar, entre muito outros [45].

- **SCIKIT LEARN**

  É uma biblioteca Python especifica para *Machine Learning* isto inclui várias ferramendas desde manipulação de dados a processamento de métricas. Permite utilizar facilmente algoritmos para experimentar em dados com apenas alguns passos de configuraçoes [45].

Das ferramentas referidas o H2O foi a abordagem escolhida uma vez que permite facilmente integrar o modelo de previsão e é utilizado para fazer sistemas de escalabilidade. Contudo, uma das razões para utilizar esta ferramenta também foi a utilização do "AutoML" para ajudar a determinar o melhor algoritmo e as possíveis configurações.

2.6 Análise Crítica

Após uma análise de soluções existentes verificou-se que estas apenas se focam num ponto distinto de segurança, deixando exposto um vasto leque de outros pontos de acesso à informação privilegiada a possíveis ataques, ou seja, apenas se focavam num tipo específico de ataques. Ora, detetar ataques do tipo SQL injection ou DDoS é pouco para o conjunto de possíveis ataques a uma aplicação Web. Neste âmbito, ser capaz de detetar numa primeira fase a ocorrência de um ataque e numa segunda fase o tipo de ataque tem mais-valias, nomeadamente a deteção antecipada ou mais célere do ataque mitigando as suas consequências.

Outras soluções, como o SIEM [10] já incluem ferramentas que permitem detetar ciberataques. Porém, na sua maioria baseiam a sua capacidade de deteção em base dados sobre acontecimentos e ataques que ocorreram, podendo não estar preparada para novos ataques e análise de padrões diferentes nos dados que ainda não estejam atualizados na base de dados *Open Threat Exchange* [16].

A abordagem proposta consiste numa solução capaz de detetar comportamentos anómalos e potências ataques. Recorrendo à utilização de *Machine Learning* para encontrar padrões nos dados a partir dos logs classificando-os como normais ou anormais. Trata-se de uma aplicação configurável, automatizada e independente dispensando alterações no *hardware* ou da arquitetura da aplicação onde é usada. É rápida e fácil de usar, abstraindo o utilizador da lógica implementada.

Capítulo 3

Análise de Deteção de Ciberataques - Arquitetura

Neste capítulo é apresentada a arquitetura subjacente ao projeto de deteção de ciberataques em aplicações Web. Tal como referido, este projeto faz uso dos logs das aplicações Web, modelando-os e treinando através de algoritmos de *Machine Learning*. Para que tal seja possível é necessário consumir os logs aplicacionais, fazer o seu tratamento e análise. Cada uma destas etapas é apresentada ao longo do capítulo e é feita uma descrição sobre a integração da solução com aplicações de terceiros.

A arquitetura foi programada integrando uma aplicação. A aplicação é totalmente configurável e independente da aplicação onde é utilizada. Desta forma, o objetivo deste projeto foi de criar uma aplicação que consome todos os log proveniente da interação do utilizador com a aplicação que são todos filtrados na camada de middleware da aplicação hospedeira, só depois chegam à parte de backend para serem tratados. Desta forma, a aplicação apresentada não interfere na arquitetura funcional das aplicações comuns adquirindo os pedidos efetuados de forma contínua e efetuando a análise dos mesmos independentemente da aplicação Web a monitorizar. Na Figura 20 é ilustrado o diagrama funcional da arquitetura.

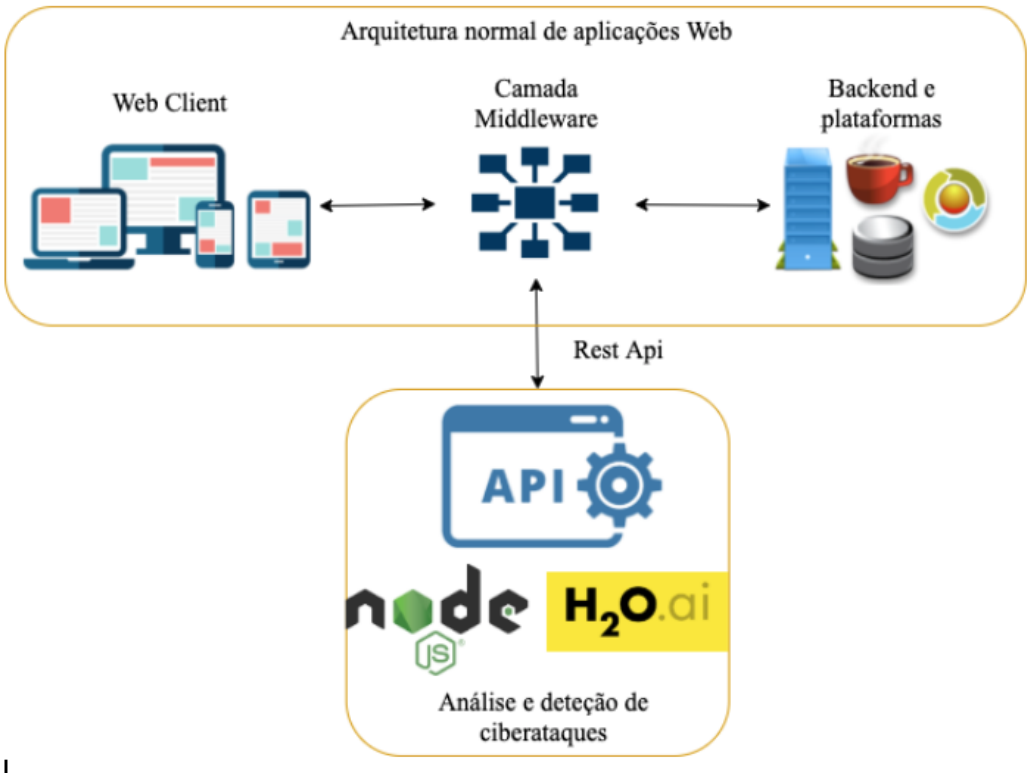


Figura 20: Componente de inserção da solução com aplicações de terceiros

Como ilustrado na Figura 20 o funcionamento da aplicação é totalmente independente e não interfere em nenhum componente ou camada das aplicações comuns. Passando a explicar a parte técnica da aplicação esta é feita em Node.js e a parte que incorpora *Machine Learning* utiliza o H20. A aplicação foi desenvolvida para aceitar diferentes pontos de entrada de dados (streaming ou bash). Assim sendo, os dados posteriormente serão tratados e persistidos numa base dados (MongoDB) responsável por armazenar toda a informação para ser analisada. Os dados armazenados na base de dados são tratados e enriquecidos constituindo um dataset para treino e previsão de novas instâncias para determinar anomalias ou ciberataques em curso.

Nas próximas secções serão descritos pormenorizadamente os processos responsáveis pela aquisição dos dados, secção 3.1, do processo ETL, secção 3.2 e o enriquecimento dos dados, secção 3.3 com informação relevante para depois guardar na base de dados em MongoDB.

3.1 Aquisição de dados

A aplicação está feita em *REST* e preparada para receber continuamente todos os pedidos que passem pela camada *middleware*. Desta forma efetua as validações dos dados até chegar ao estado final do tratamento dos dados. O serviço responsável por adquirir e tratar novas instância é apresentado na Figura 21.

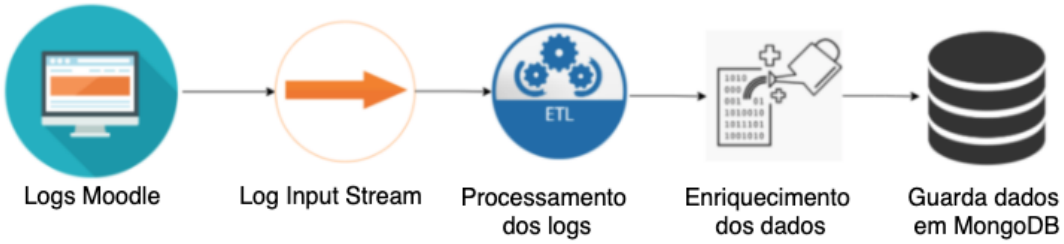


Figura 21: Serviço responsável por fazer a extração e tratamento de logs

Como se pode ver pela Figura 21 o processo de aquisição de dados passa por várias etapas das quais processamentos de novos pedidos, posteriormente enriquecimento dos dados e só depois é que é finalizado o processo guardando a informação extraída na base de dados. É de referir que os logs das aplicações Web adotam normalmente o formato de *Common Log Format (CLF)* [11]. Contudo, a solução apresentada está preparada para receber qualquer tipo de formato de logs.

3.2 Processamento dos logs

Para esta fase considerou-se a adoção de uma técnica escalável e genérica para processar e decompor o formato dos logs. Para a decomposição dos logs utiliza-se regex's, separando o conteúdo de forma a se utilizar facilmente cada variável. Na Figura 22 é ilustrado o formato normal dos pedidos feitos ao servidor provenientes da interação do utilizador com a aplicação.

```
172.20.129.121 - - [12/Jan/2020:05:32:36 +0000] "GET /2019/ HTTP/1.1" 303 467
172.20.129.121 - - [12/Jan/2020:05:32:36 +0000] "GET /2019/login/index.php HTTP/1.1" 200 28991
```

Figura 22: Composição dos logs da aplicação Web

O processo de processamento de logs extraí a informação existente nos dados através do uso de regex, que dão origem às seguintes variáveis:

- **Ip** (e.g. 172.20.129.121): Endereço IP do utilizador que fez um pedido ao servidor proveniente da interação do mesmo com a aplicação.
- **Data do pedido** (e.g. 12/Jan/2020:05:32:36 +0000): Data e a hora em que o servidor recebeu o pedido.
- **URL solicitado** (e.g. "GET /2019/ HTTP/1.1"): Informação referente à ação realizada pelo utilizador, tais como o método utilizado para fazer o pedido ao servidor, neste caso foi um "GET", consegue-se também obter a página ou o recurso solicitado "/2019/" bem como o protocolo utilizado "HTTP/1.1".
- **Resposta do servidor ao pedido do cliente** (e.g. 303, 200): Representa o código HTTP da resposta ao pedido feito pelo utilizador. É de referir que através desta informação podemos retirar outras como, por exemplo:
  - Códigos que comecem por 2 significa que o pedido feito pelo cliente foi bem-sucedido.
  - Código que comecem por 3 foi efetuado um redirecionamento.
  - Código que comecem por 4 ocorreu um erro causado pelo cliente.
  - Código que comecem por 5 ocorreu um erro ao processar o pedido no servidor.
- **Tamanho do objeto** (e.g. 467): Indica o tamanho em bytes do objeto retornado pelo servidor ao cliente.

Todo o processo de extração de dados é referente ao passo processamento de logs como mostra a Figura 21. O próximo passo no processo passa por enriquecer os dados com novas variáveis.

3.3 Enriquecimento de variáveis

Nesta secção são descritos alguns parâmetros que são obtidos por cálculos internos ou regras aplicados na solução e por Interface de programação de aplicações (API's) externas. Nesta secção descrevem-se alguns dos parâmetros acrescentados ao conjunto de dados.

Entropia da página ou recurso solicitado

A entropia do URL solicitado diz respeito a desvios ou padrões que são diferentes dos normais. Por exemplo, em aplicações comuns no URL são utilizados caracteres que estão estipulados como sendo comuns de utilizar pelas aplicações [43], e por vezes de caracteres estranhos ao padrão normal ou o uso de combinações de caracteres pode indicar uma tentativa de ataque à aplicação. O que torna importante esta análise e usufruir desta vertente para incluir no dataset.

Através desta análise consegue-se obter conhecimento importante para o dataset e extrair informação referente às seguintes variáveis:

- **Número de caracteres do URL solicitado** (url_length): As tentativas de obter informação privilegiada pode levar a ter um URL mais extenso. Pelo que faz sentido existir uma variável com o número de caracteres verificado no URL.
- **Existência de "@"no URL** (having_at_symbol): A utilização de "@" leva o navegador a ignorar tudo que venha a seguir a este símbolo.
- **Número de caracteres estranhos** (number_of_strange_characters): Através de regex's verifica-se se existem caracteres estranhos no URL e que podem ser usados para efetuar operações de aquisição de dados, injeção de código ou outras operações mal-intencionadas. Entre os caracteres estranhos a verificar incluem-se:
  - Caracteres utilizados para SQL Injection [31].
  - Caracteres utilizados para injeção de código [37].
  - Caracteres que normalmente não são utilizados em URL's [43].
- **Verifica se o URL é válido** (valid_url): Determina se o URL solicitado é válido e não contém informação que normalmente não é utilizada em URL's como o uso de "//" para fazer redirecionamentos ou "-".

Enriquecimento de dados a partir do endereço IP

Através do IP do cliente é possível obter informação relevante como a posição geográfica do cliente e a partir desta calcular a distância a que este está do servidor que lhe deu a resposta. Para calcular a distância é utilizado uma API de terceiros.

Data e hora a que o servidor recebe pedido

A data e hora permte fazer análises, como, por exemplo, qual o período do dia que tem mais afluências de pedidos.

Tipo de resposta ao pedido efetuado ao servidor.

A resposta dada permite tirar conclusões sobre os pedidos feitos pelo cliente ao servidor. Por exemplo, um acréscimo de respostas com o status code 400 pode significar um comportamento anómalo, face ao período de analisado.

Tamanho do objeto retornado pelo servidor

O tamanho em byte retornado pelo servidor para os pedidos é também interessante, na medida em que as variações podem ser sinais, entre outros, de exflitração indevida de dados.

Na próxima secção 3.4 apresenta-se o dataset final a que se chegou pela extração e enriquecimento de dados.

3.4 Estrutura do dataset criado

O dataset final foi obtido a partir de todas as variáveis e cálculos descritos anteriormente, tudo possível através da análise feita aos dados em bruto de entrada e de processos de tratamento e extração de informação importante. Para a criação do dataset foram utilizadas as variáveis que através de estudos realizados se demonstraram importantes utilizar para o uso posterior no modelo criado. Para a abordagem de construir os dados para o modelo através da técnica de janela temporal tornou-se importante em certas variáveis adicionar o mínimo e o máximo para desta forma determinar a variância da média dessa variável no intervalo de dados analisado. Desta forma, obtivemos as seguintes variaveis que dizem respeito aos respetivos valores da janela temporal analisada:

Variável	Descrição

number_of_logs	Representa o número de pedidos feitos ao servidor
number_of_distinct_ips	Representa o número de IP's distintos que fizeram pedidos ao servidor.
number_of_distinct_urls	Representa o número de URL's distintos que foram solicitados.
number_of_distinct_ips_and_urls	Representa o número IP's e URL's distintos em conjunto.
average_object_size_returned_to_client	Representa o número médio do objeto da resposta do servidor ao pedido do cliente.
min_object_size_returned_to_client	Representa o número mínimo do objeto da resposta do servidor ao pedido do cliente.
max_object_size_returned_to_client	Representa o número máximo do objeto da resposta do servidor ao pedido do cliente.
average_distance_between_client_and_server	Representa o número médio da distância entre a localização do utilizador e o servidor.
min_distance_between_client_and_server	Representa o número mínimo da distância entre a localização do utilizador e o servidor.
max_distance_between_client_and_server	Representa o número máximo da distância entre a localização do utilizador e o servidor.
average_url_length	Representa o número médio do número de caracteres que compõem o URL's.
min_url_length	Representa o número mínimo do número de caracteres que compõem o URL's.

max_url_length	Representa o número máximo do número de caracteres que compõem o URL's.
average_url_length_different_normal	Representa o número médio do número de caracteres que compõem o URL's que diferem do normal (obtido pela média de todos os URL's da aplicação).
min_url_length_different_normal	Representa o número mínimo do número de caracteres que compõem o URL's que diferem do normal (obtido pela média de todos os URL's da aplicação).
max_url_length_different_normal	Representa o número máximo do número de caracteres que compõem o URL's que diferem do normal (obtido pela média de todos os URL's da aplicação).
average_number_at_symbol	Representa o número médio do número de símbolos "@"que compõem o URL's.
min_number_at_symbol	Representa o número mínimo do número de símbolos "@"que compõem o URL's.
max_number_at_symbol	Representa o número máximo do número de símbolos "@"que compõem o URL's.
average_number_double_slash_redirect	Representa o número médio do número de vezes que aparece a combinaçao de símbolos "//" que compõem o URL's.
min_number_double_slash_redirect	Representa o número mínimo do número de vezes que aparece a combinaçao de símbolos "//" que compõem o URL's.

max_number_double_slash_redirect	Representa o número máximo do número de vezes que aparece a combinaçao de símbolos “//” que compõem o URL’s.
average_number_hyphen	Representa o número médio de vezes que aparece o símbolo -”nos URL’s.
min_number_hyphen	Representa o número mínimo de vezes que aparece o símbolo -”nos URL’s.
max_number_hyphen	Representa o número máximo de vezes que aparece o símbolo -”nos URL’s.
average_number_of_strange_characters	Representa o número médio de vezes aparecem símbolos estranhos (que diferem dos padrões comuns definidos) nos URL’s.
min_number_of_strange_characters	Representa o número mínimo de vezes aparecem símbolos estranhos (que diferem dos padrões comuns definidos) nos URL’s.
max_number_of_strange_characters	Representa o número máximo de vezes aparecem símbolos estranhos (que diferem dos padrões comuns definidos) nos URL’s.
average_time_difference_between_logs	Representa o número médio do tempo entre pedidos feitos ao servidor.
min_time_difference_between_logs	Representa o número mínimo do tempo entre pedidos feitos ao servidor.
max_time_difference_between_logs	Representa o número máximo do tempo entre pedidos feitos ao servidor.
number_of_error_logs	Representa o número de pedidos feitos ao servidor com o estado ”4XX”ou ”5XX”de resposta ao pedido.

number_of_logs_with_status_different_error	Representa o número de pedidos feitos ao servidor com o estado diferente de ”4XX”ou ”5XX”.
number_of_valid_urls	Representa o número de pedidos feitos ao servidor que foram determinados como normal, ou seja, nao continham caracteres estranhos nem outras anomalias.
has_attacks	Representa a variável de resposta aos frames analisados, ou seja, se é um frame normal ou representa um ataque.

Tabela 3.1: Descrição das variáveis que compõem o dataset.

3.5 Arquitetura funcional

Como já referido o objetivo é construir uma solução independente e escalável. Na Figura 23 é ilustrada a solução final capaz de efetuar previsões *end-to-end*.

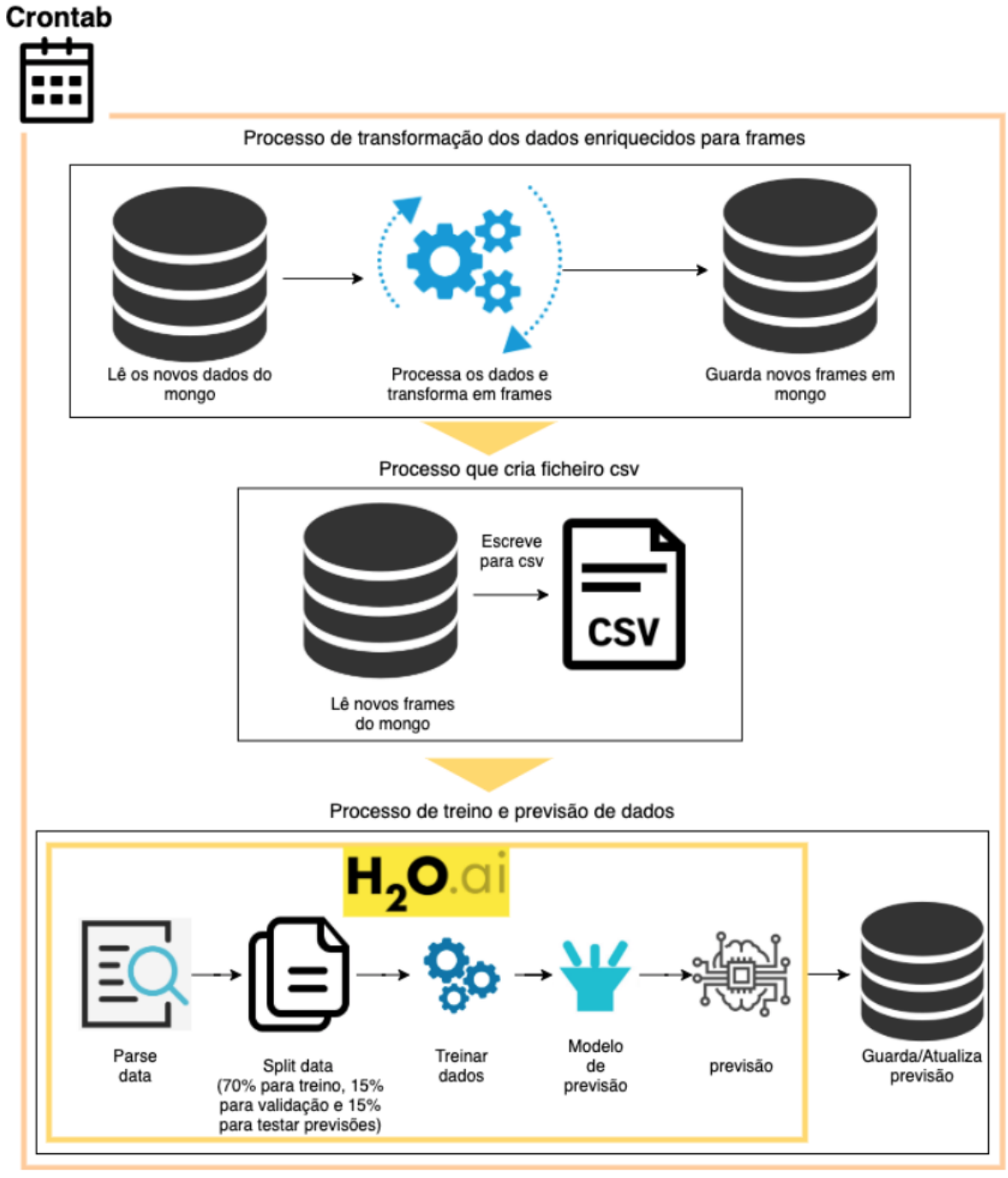


Figura 23: Arquitetura funcional da aplicação

Como se pode observar pela Figura 23 a solução está dividida em três grandes processos ou serviços, que são os seguintes:

- **Processo de transformação dos dados enriquecidos para frames**

  Este processo passa por receber os novos dados de entrada em bruto e fazer o devido tratamento aos mesmos para fazer a extração de toda a informação importante a utilizar para os passos seguintes. É neste processo que temos um extrator genérico de decomposição da informação que contem os pedidos feitos ao servidor. De seguida estes dados serviram para dar suporte à obtenção de nova informação retirada dos dados em bruto para complementar o dataset. Nesta etapa são utilizadas API's externas para obter informação extra do utilizador através do IP e para calcular a distância entre o utilizador e o servidor da Escola Superior de Tecnologia e Gestão (ESTG). Toda a informação obtida através dos dados de entrada é guardada na base de dados em Mongo para processos futuros.

- **Processo que cria ficheiro CSV**

  Este processo é responsável por criar os ficheiros CSV com os novos dados gerados pelo processo anterior para proceder no fluxo da aplicação.

- **Processo de treino e previsão de dados**

  Este processo é responsável por toda a lógica e a implementação relacionada com *Machine learning*. Por isso, como podemos observar na imagem este é um processo sequencial, com os seguintes passos:

  - **Parse data**

    É feito o tratamento dos dados para poder utilizar no H2O.

  - **Split data**

    É feito a separação dos dados em 70% para treino, 15% para validação e 15% para testar as previsões, numa primeira fase de treino dos dados foi decidido esta abordagem porque desta forma os resultados obtidos pela classificação dos frames são mais robustos uma vez que estamos a utilizar dados de validação que o algoritmo desconhece e que não esteve em contacto. Desta forma, ao fazer esta abordagem sabemos que o modelo está a classificar os dados corretamente uma fez que os dados de treino são diferentes dos dados de validação, ou seja, não foram utilizados no "mundo"de dados de aprendizagem. Após a fase de treino e com a aplicação a correr todos os novos dados de entrada são utilizados para fazer a previsão, ou seja, não existe **split dos dados**.

  - **Treinar dados**

Numa primeira fase de treinar dados, este processo serve para criar um modelo de previsão através dos padrões encontrados nos dados utilizados para treino. Após a fase de treino e com a aplicação a correr este passo não ocorre.

– **Modelo de previsão**

Nesta etapa é carregado o modelo de previsão gerado pelo processo ”Treinar dados”ou também pode ser carregado outro modelo anteriormente treinado.

– **Previsão**

Esta etapa é responsável por fazer a previsão dos novos dados de entrada que cheguem à aplicação e classificar os mesmos como sendo normais ou ataques à aplicação.

– **Guarda/Atualiza previsões**

Neste passo é guardado todas as classificações feitas e os seus resultados na base de dados para utilizar no futuro para análise ou cálculos estatísticos.

3.5.1 Definição de frame

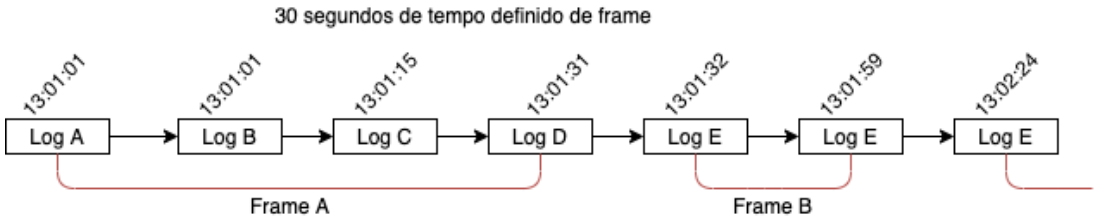


Figura 24: Demonstração de como são construídos os frames por espaço temporal

Como podemos visualizar na imagem a construção dos frames define-se a partir do primeiro log de entrada na aplicação, ou seja, o tempo de entrada do primeiro pedido foi às 13:01:01 então o frame será constituído por todos os pedidos que estão entre o intervalo de tempo [13:01:01; 13:01:31], (30 segundos de espaço temporal). Esta lógica aplicada aos dados apesar de acrescentar algum tempo de processamento para construir os frames (efetuar média, mínimo e máximo sobre as variáveis) torna-se bastante eficiente na previsão dos resultados uma vez que existem menos dados para treinar. Como podemos observador na seguinte imagem o tempo que demora a criar frames é proporcional ao número de pedidos que constitui cada um.

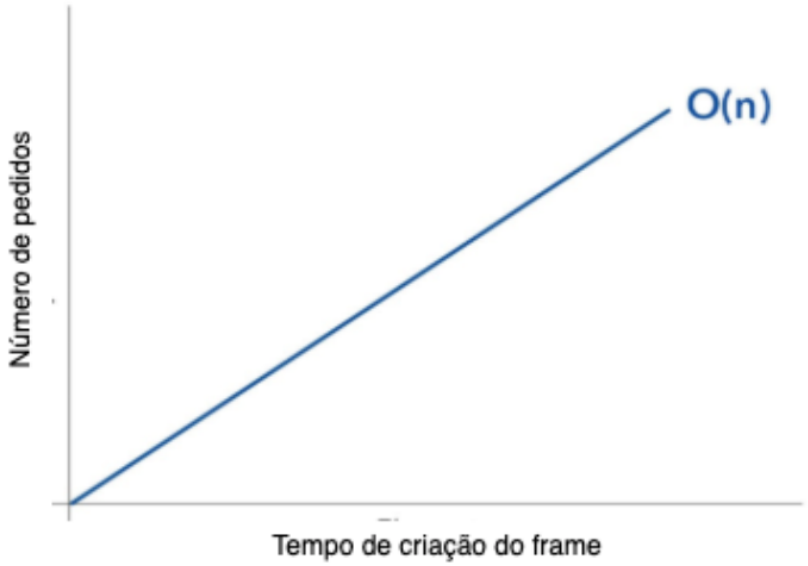


Figura 25: Tempo proporcional ao número de pedidos que dão origem a cada frame.

Como podemos observar quanto maior for o número de pedidos que dão origem a um frame maior é o tempo de execução necessários para fazer o processamento dos dados.

3.5.2 Atualização do dataset

O dataset consiste num ficheiro de forma CSV (Comma-Separated Values) gerado a partir dos frames existente na base de dados. No processo que antecede a análise do dataset é feito o tratamento de dados. Em concreto e tendo em consideração uma análise exploratória prévia, é aplicado o seguinte tratamento aos dados:

- Remover dados duplicados.
- Limpeza de dados que não tinham informação importante para o dataset, por exemplo, por causa de alguma anomalia todos os valores referentes a uma instância estavam vazios.

O dataset resultante do tratamento segue para o fluxo de previsão.

3.5.3 Fluxo de previsão

O fluxo de previsão é uma parte importante. Trata-se do processo responsável por classificar os novos dados de entrada. É um processo contínuo incumbido por classificar as instâncias de frame como normais ou anomalias. O algoritmo de *Machine Learning* para a classificação pode ser predefinido de acordo com os resultados obitos na fase de treino. Existe ainda a possibilidade de executar um conjunto de algoritmos à medida que o dataset cresce e escolher entre os que executaram o que melhor se adapta aos dados e oferece melhor resultado de classificação.

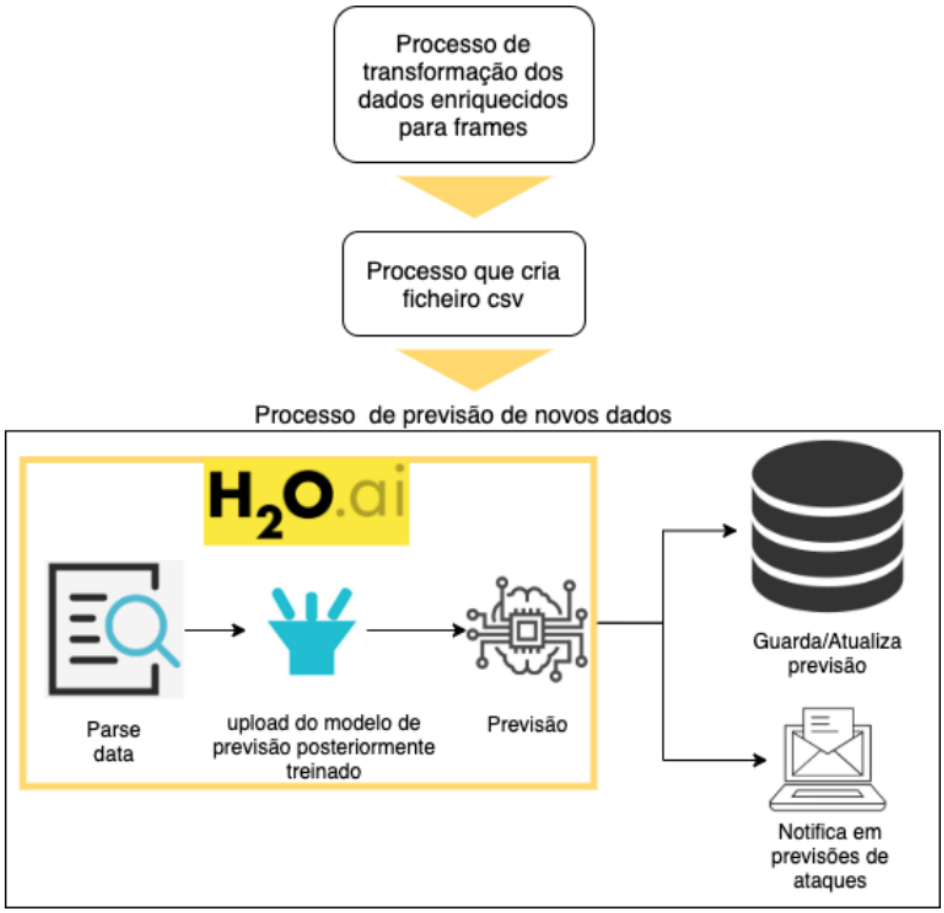


Figura 26: Fluxo de previsão de novos dados de entrada

Como se pode ver pela Figura 26 nesta etapa existe o processo de previsão de novos dados que advém de processos anteriores responsáveis por processar e transformar os dados de entrada em frames. O algoritmo de *Machine Learning* executa a cada 30 segundos se tiver dados para analisar. Os passos realizados são:

- Parser dos dados do csv.
- Faz upload do modelo treinado.
- Faz previsão dos novos dados.

Após efetuar a previsão os resultados são guardados na base de dados e caso exista alguma anomalia ou ciberataque em curso o administrador da plataforma é notificado.

A solução criada oferece um processo de deteção de ataques em aplicações Web *end-to-end* independente e evolutivo. O processo dispensa de qualquer intervenção humana durante o período em que está ativo.

3.5.4 API

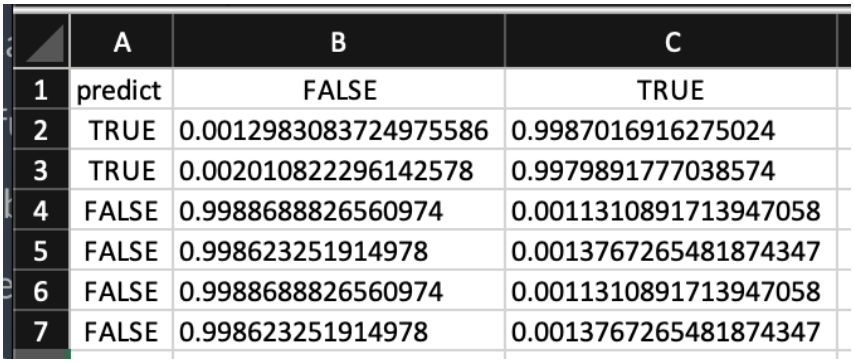
A arquitetura da aplicação está feita em Node.js e utiliza para armazenamento de dados um SGBD MongoDB. Foi criada uma API que está dividida em processos distintos com funções e lógica distintas entre eles, ou seja, está construida sobre a arquitetura de serviços apresentados ao longo deste documento. A API permite receber novas instâncias ou conjuntos de dados dependendo do endpoint (e.g. Figura 27) que seja chamado, para posteriormente prever esses dados e distinguir o mesmos de pedidos normais ou com intenção maliciosa. A API também apresenta um processo independente e automático para inicialmente criar um modelo de previsão através dos dados existentes.


```
{"raw_log": "[15/Dec/2019:03:39:02 +0000] 66.249.70.61 TLSv1.
    2 ECDHE-RSA-AES128-GCM-SHA256 \"GET /horarios/
    turma_LSG3T1_538_19112018.html?6367606178506116 95 HTTP/1.
    1\" 237"
}
```


Figura 27: Inserção de uma instância do pedido feito ao servidor.

Mantendo-se o modelo de treino para previões futuras, o modelo é guardado num ficheiro e carregado quando necessário. Desta forma acelera-se o processo de classificação com um modelo já treinado. É ainda guardado na base de dados informação referente a esse modelo, como o dia e a hora a que foi treinado, número de instâncias de cada classe e o resultado das métricas obtidas. Desta forma mantem-se um histórico de trabalho facilitando futuras

intervenções sobre o modelo. Na Figura 28 é ilustrado um exemplo do output gerado pelo processo de classificação.



	A	B	C
1	predict	FALSE	TRUE
2	TRUE	0.0012983083724975586	0.9987016916275024
3	TRUE	0.002010822296142578	0.9979891777038574
4	FALSE	0.9988688826560974	0.0011310891713947058
5	FALSE	0.998623251914978	0.0013767265481874347
6	FALSE	0.9988688826560974	0.0011310891713947058
7	FALSE	0.998623251914978	0.0013767265481874347

Figura 28: Exemplo de um ficheiro com o resultado de previsão.

De seguida serão explicados os métodos disponíveis na aplicação:

Endpoint: api/load_file
Método: POST
Descrição: Permite a inserção de um ficheiro com uma ou várias instâncias de pedidos feitos ao servidor.
Input: Ficheiro de pedidos feitos ao servidor. Exemplo Figura 22.
Output: Success/Error

Endpoint: api/raw_log
Método: POST
Descrição: Permite a inserção de um pedido feito ao servidor.
Input: Uma instância de dados a ser adicionada. Exemplo Figura 27.
Output: Success/Error

Endpoint: api/create_dataset
Método: POST
Descrição: Processo para criar um ficheiro CSV normalizado com todos as instâncias.
Input: Opcional ou Nome do ficheiro.
Output: Caminho do ficheiro/Error

Endpoint: api/proccess_dataset
Método: POST
Descrição: Processo para fazer a previsão de um conjunto de dados do ficheiro CSV. Exemplo do formato dos resultados na Figura 28.
Input:

- caminho do ficheiro CSV para fazer previsão.
- Nome do modelo.
- Caminho do modelo a utilizar
- Caminho para guardar resultados.

Output: Ficheiro com as previsões/Error

Endpoint: api/create_predict_model
Método: POST
Descrição: Processo para fazer o treino de um modelo de previsão.
Input:

- caminho do ficheiro CSV para fazer previsão.
- Nome do modelo.
- Caminho para guardar resultados.

Output: Modelo para fazer previsões/Error

Capítulo 4

Estudo Experimental

Neste capítulo é apresentado um estudo experimental feito a partir da arquitetura descrita no capítulo anterior.

4.1 Ambiente de testes

Uma vez que não era possível utilizar o Moodle como ambiente real para a organização em estudo para testar a solução, tivemos de criar outra abordagem à semelhança do Moodle para fazer testes. Desta forma, foi criado um ambiente controlado com os mesmos requisitos do Moodle utilizando o "VirtualBox", que permite criar máquinas virtuais para ter diversos ambientes a correr. Para isto foi criado uma máquina virtual com o CentOS6 em que foi instalada a versão "2019052001.1" do Moodle referente à release "3.7.1".

A máquina real que tinha o ambiente virtualizado instalado era um Mac com os seguintes requisitos:

- Processador - 2,7 GHz Dual-Core Intel Core i5
- Memoria RAM - 8 GB 1867 MHz DDR3
- Placa gráfica - Intel Iris Graphics 6100 1536 MB
- Disco - 121 GB

Para a máquina virtual centOS foram dados os seguintes requisitos:

- Disco - 20 GB
- Memoria RAM - 6GB

Para guardar a informação produzida pelo Moodle foi configurado e instalado uma base de dados em MariaDB. Como o Moodle é compatível com vários servidores Web decidimos utilizar o Apache's httpd.

4.2 Aquisição de dados

Para o estudo experimental foi usado um conjunto de dados reais, obtido a partir dos logs de acesso à aplicação Moodle da ESTG do Politécnico do Porto. Os dados começaram por ser anonimizados, ofuscando-se todas as referências a números de estudantes, siglas de docentes e funcionários. A ofuscação consistiu em substituir tais referências pelo valor hash. Os dados provenientes do Moodle foram tratados como sendo normais, ou seja, isentos de cenários de ciberataque.

De forma a obter dados representativos de um cenário de ataque, foi criado um ambiente composto por uma versão do Moodle igual ao da ESTG e foram gerado tráfego para a aplicação utilizando ferramentas de intrusão. Em concreto, usou-se uma aplicação denominada por Arachni que basicamente permite fazer testes de intrusão ou tentativas de aquisição de dados [27]. Com o Arachni foi possível simular vários tipos de ataque como, por exemplo:

- Injeção de SQL.
- Injeção de XSS.
- Injeção de NoSQL.
- Tentativas de redireccionamento para outros sites.
- Injeção de código.

De forma a enriquecer os dados com cenários de ciberataque, usou-se também uma ferramenta de *Denial-of-Service (DoS)*. A ferramenta usada foi o GoldenEye [48], e trata-se de uma

aplicação Python que faz *(D)DoS* ao servidor selecionado. A aplicação permite gerar uma afluência maior de pedidos ao servidor de forma a sobrecarregar o mesmo com a falta de resposta a tantos pedidos.

Os logs do Moodle da ESTG e os logs do Moodle submetido a testes de intrusão e ataque de DoS foram recolhidos para ser analisados e tratados.

4.3 Criação do dataset

Para a criação do dataset contamos com 5 793 574 entradas no logs do pedidos ditos normais e 1 828 169 entradas nos logs do Moodle submetido a testes de intrusão. Considerando o grande volume de dados foi equacionado a criação do dataset agrupado por diferentes paramêtros. Inicialmente foram ponderadas as seguintes abordagens:

- Agrupar dados no dataset por IP.
- Agrupar dados no dataset por dias da semana.

Ambas as abordagens foram analisadas, mas chegou-se à conclusão que não eram adequadas. Por exemplo, como garantir que o IP usado pelo cliente X num dado instante não muda ao fim de algum tempo. Agrupar por dia faria com que muito dos valores contidos nos dados ficasse agrupado, por exemplo pelo valor médio, perdendo valor de análise mais granular. Para evitar estes problemas optou-se por criar um único modelo baseado em frames, ou seja, agrupado por janelas temporais às quais se dá o nome de frames. O tempo do espaço temporal definido foi de 30 segundos, valor a que se chegou após uma análise empiríca, que é sumariamente apresentada na Tabela 4.1.

Tempo definido do frame	Frames obtidos	Tempo de processamento
15 seg	250 701	12min
30 seg	130 125	11min
40 seg	26 523	14min

Tabela 4.1: Análise do melhor tempo para construir os frames

Como se pode observar o melhor tempo foi de 30 segundos porque apresentava menor quantidade de frames e melhor tempo de processamento. Este processo particular engloba os

passos de processamento dos logs, enriquecimento de dados de acordo com o que foi apresentado no capítulo anterior, e agrupa os dados em frames.

É de referir que como o conjunto de dados apenas apresenta valores numéricos facilitou a tarefa de criar frames. Para agrupar os valores dos parâmetros em cada frame foram aplicados cálculos matemáticos, como a média, contagem, mínimo e máximo. Ou seja, num conjunto de entradas nos logs como mostra a Tabela 4.2, consegue-se determinar o valor no frame.

Cálculo	Número de caracteres estranhos	Tamanho do URL
valor de entrada	2	80
valor de entrada	20	50
valor de entrada	7	30
Frame obtido		
Média	9,67	53,3
Mínimo	2	30
Máximo	20	80

Tabela 4.2: Cálculos efetuados para obter um frame

Clarificando os valores apresentados na Tabela 4.2, considerando um URL1, URL2 e URL3 ocorrem num espaço de 30 segundos e que tem 2, 20 e 7 caracteres estranhos respetivamente e tem 80, 50 e 30 bytes de comprimento, no frame os valores serão agrupados sendo registado no dataset para esses 30 segundos uma entrada que contém 9,67 caracteres estranhos e 53,3 como valor médio do tamanhos dos URLs (da mesma forma é guardado o valor mínimo observado no intervalo e o valor máximo).

Na Figura 24 é ilustrada a forma como o dataset é criado. Cada entrada no dataset corresponde um frame e cada frame é o resultado do agrupamento de dados considerando um tempo de 30 segundos (valor este que poderá ser ajustada).

4.4 Análise exploratória e preparação do dataset

Após ter o conjunto de dados preparado a fase seguinte foi a análise do dataset para evitar erros de cálculos ou outros problemas derivados da discrepância dos dados. Numa análise

feita ao nível de cálculos todos os procedimentos e operações matemáticas foram validadas para garantir que não existiam erros que dessem origem a valores falsos e consequentemente a discrepância nos dados.

Na Figura 29 é apresentada a distribuição de dados (frames) pelas classes normal e ataques.

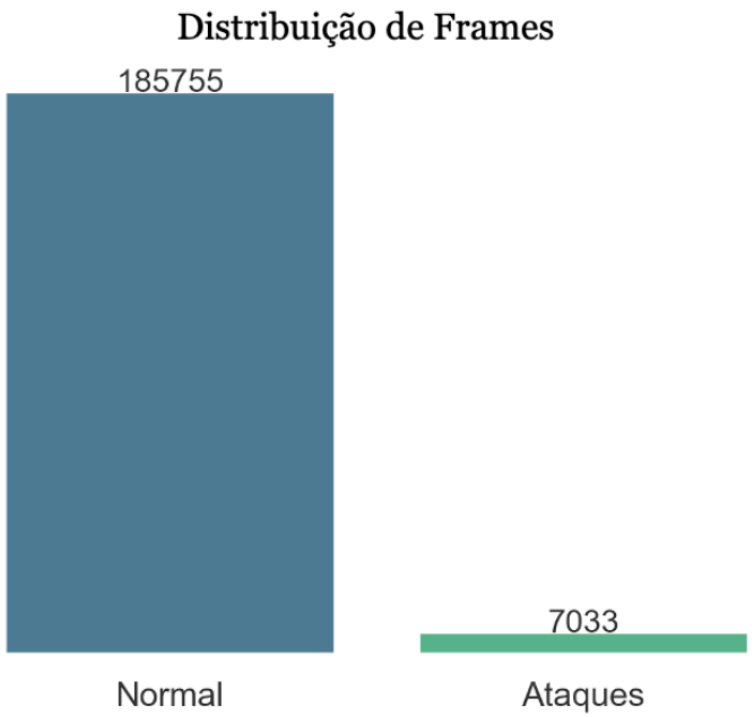


Figura 29: Distribuição de frames pelas classes existentes

Como se pode observar pela Figura 29, a discrepância entre classes é elevada sendo que a relação é de 3,78% de pedidos de ataques em relação ao total de pedidos durante o tempo de estudo. Tal revela que estamos perante um dataset *unbalance*. Tal situação pode fazer com que os algoritmos de *Machine Learning* tendam a dar sempre como resultado de previsão a classe maioritária, uma vez que não conseguiu detetar padrões nos dados para distinguir as classes e se sobre ajustou à classe maioritária. Para evitar tal situação foram aplicadas técnicas e métodos para tornar o dataset mais equilibrado entre as classes.

Um dataset perfeitamente balanceado é quando existe um equilíbrio entre classes, por exemplo, num dataset que a variável de previsão é binária isso significaria que teríamos cerca de 50% da classe de fraude e 50% da classe normais. Para equilibrar o dataset a que chegamos podem ser aplicadas técnicas como o *undersampling*, *oversampling* ou SMOTE.

O *undersampling* consiste em remover dados da classe maioritária para equilibrar as classes

do dataset. A Figura 30 exemplifica o processo de remoção de dados para equilibrar o número de registos por classe.

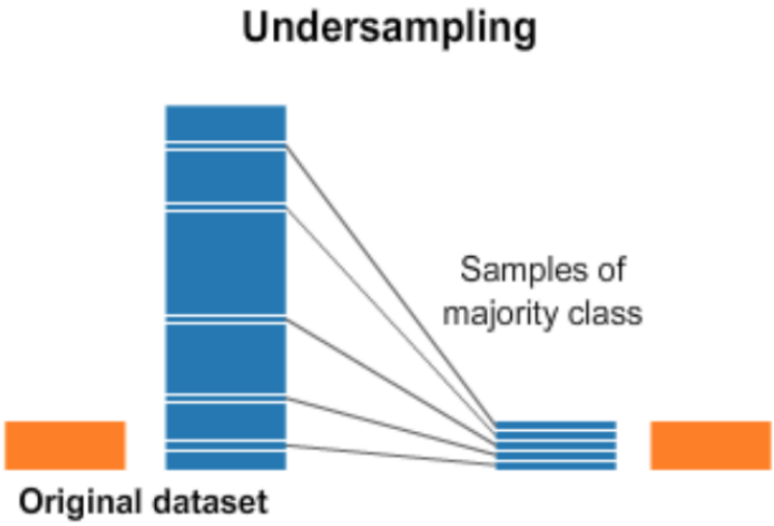


Figura 30: Técnica Undersampling para balancear o dataset [4]

Assim, e de forma a não descartar registos que sejam importantes para previsão de resultados foi utilizado a técnica de agrupamento de dados de k-NN (algoritmo de k vizinhos mais próximos) de forma a obter um equilíbrio entre classes [56]. Desta forma obtemos um conjunto de dados de todas as observações de todos os grupos da classe maioritária, conseguindo-se manter a informação importante de todos os grupos de dados contrariando a perda de informação por remoção de registos de forma aleatória. Aplicando esta técnica obtiveram-se os resultados de classificação como os apresentados na Figura 31.

```
   ACCURACY -  0.9404591104734576
   F1_SCORE -  0.6311111111111112
 CONFUSION MATRIX -
      0    1
0  3720  248
1     1  213
   RECALL -  0.9953271028037384
```

Figura 31: Resultados obtidos pela técnica Undersampling

Outra forma de balancear o dataset consiste no *oversampling*. Esta técnica consiste em adicionar novas estâncias à classe minoritária através da duplicação da classe minoritária. Tal não acrescenta valor ao dataset ou informação servindo apenas para balancear as classes. Na Figura 32 é ilustrado o processo de *oversampling*.

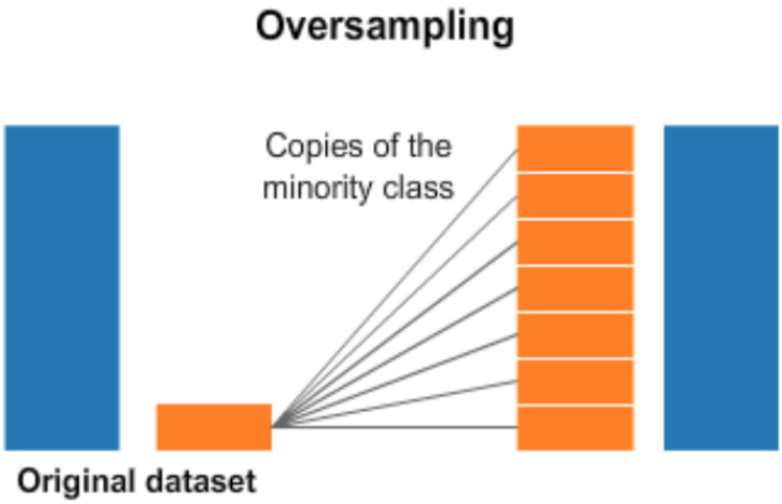


Figura 32: Técnica Oversampling para balancear o dataset [4]

Com a aplicação da técnica de *oversampling* obtiveram-se resultados de classificação tais como os ilustrados na Figura 33.

```
   ACCURACY -  0.9402199904351984
   F1_SCORE -  0.6301775147928994
 CONFUSION MATRIX -
      0    1
0  3719  249
1     1  213
   RECALL -  0.9953271028037384
```

Figura 33: Resultados obtidos pela técnica Oversampling

O SMOTE foi outras das técnicas de balanceamento exploradas. Trata-se de uma técnica utilizada para adicionar novas instâncias criadas sinteticamente através da utilização de um algoritmo do vizinho mais próximo. Esta técnica agrupa os dados da classe minoritária de forma a criar linhas entre os pontos das estâncias como mostra a Figura 34.

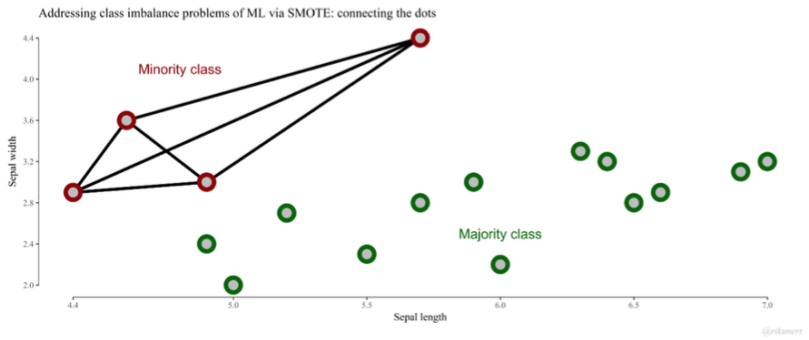


Figura 34: Conexão entre os pontos pela técnica SMOTE [46]

Com esta técnica são definidos novos pontos de dados, ou seja novos dados, tal como ilustra a Figura 35.

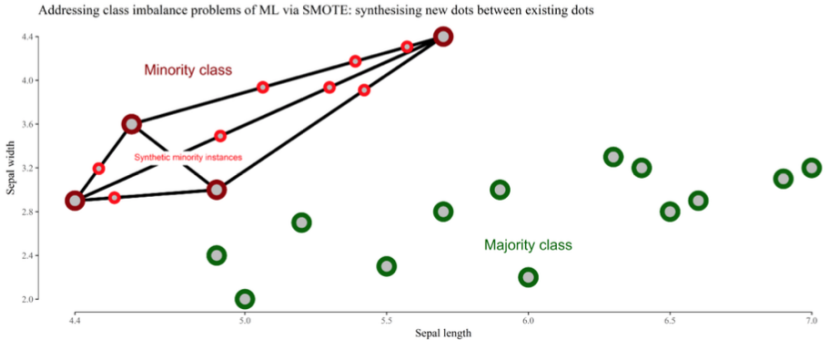


Figura 35: Criação de novas estâncias pela técnica SMOTE [46]

Através do SMOTE facilmente se adicionam novos dados à classe minoritária para equilibrar o dataset. Com esta técnica obtiveram-se resultados de classificação como os ilustrados na Figura 36.

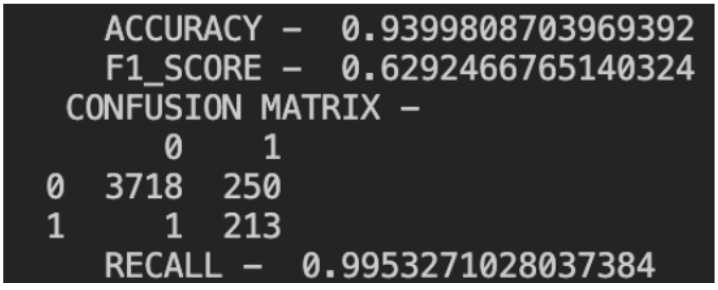


Figura 36: Resultados obtidos pela técnica SMOTE

Além do processo de balanceamento dos dados, foi efetuada uma análise exploratória que consistiu na verificação de ruído nos dados, tais como *outliers* e de valores em falta. Esta análise é fundamental para evitar desvios por parte dos algoritmos de *Machine Learning*.

4.5 Ruído nos dados e valores em falta

Da análise aos diferentes parâmetros que constituem o dataset verificou-se a existência de *outliers*, tal como se verifica pela Figura 37.

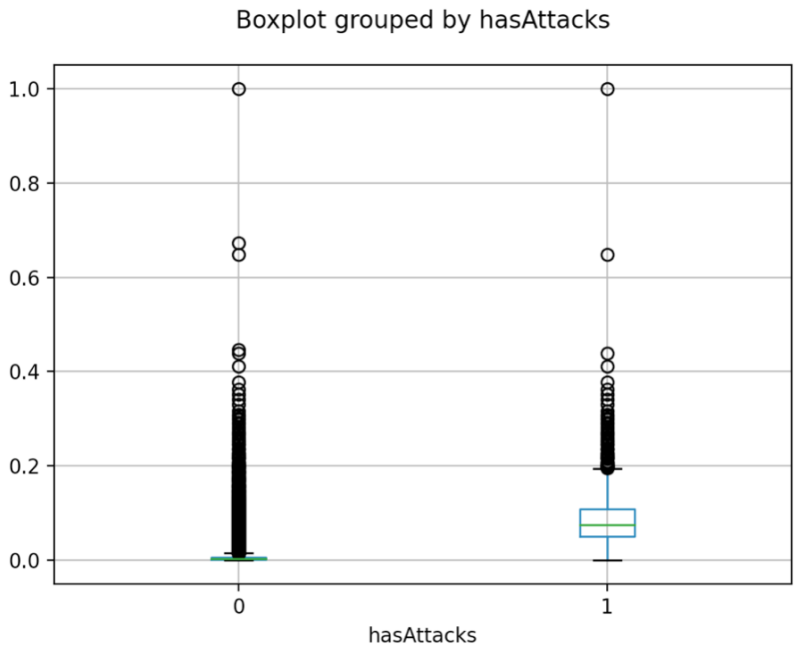


Figura 37: Resultado da distruibuiçao da variàvel ”numero_de_logs”pelas classes em análise

Normalmente a existência de *outliers* significa que esses valores são diferentes e que estão distantes dos pontos considerados normais. É comum proceder à eliminação dos *outliers*, pois têm impacto no desempenho, na previsão do modelo e para análise estatística na média no desvio padrão entre outros [25]. Ao invés de eliminar os *outliers*, e tal como defendido em [21], optou-se por fazer uma análise mais detalhada e verificou-se que esses valores fazem parte dos dados (e.g. horas antes dos exames ou quando sai a pauta de notas existe uma maior afluência de pedidos e a interação com a aplicação é maior). Por isso, não sendo erros de recolha ou medição, assumiu-se manter os valores no dataset.

Relativamente à existência de valores em falta, verificou-se que no dataset existiam vários casos em que tal acontecia. Tal acontece porque se recorreu a API's externas para enriquecer informação sobre o IP que por vezes as seguintes situações acontecem:

- As APIs utilizadas não têm informação sobre o IP solicitado.
- As APIs utilizadas tem limite de pedidos que podemos fazer diariamente para obter

informação sobre IPs.

Mais uma vez optou-se por manter estes dados, pois, apesar de ser relevante a informação relativa à posição geográfica que falhou em obter, os mesmo contêm outras informações relativas ao URL solicitado que são importantes para a análise.

Outro problema que surgiu foi a preparação dos dados para aplicar algoritmos de aprendizagem de forma a tornar os mesmos uniformes. Ou seja, como os dados que pertencem a este domínio apresentavam valores dispare optou-se por normalizar o dataset de maneira a redimensionar os dados numéricos para valores entre o intervalo 0 e 1 (transformação logarítmica para aproximação à distribuição normal). Torna-se útil fazer esta preparação para atributos de entrada, pois podem apresentar uma grande discrepância entre valores e certos algoritmos não reagem bem a grandes intervalos nos dados, por exemplo, o caso particular das distâncias entre o utilizar e o servidor. Em certos casos os métodos de aprendizagem tornam-se mais eficientes quando cada atributo respeita um padrão e estão dentro da mesma escala.

Uma vez analisado e tratado o dataset, procedeu-se à análise de dados, tirando partido dos algoritmos de *Machine Learning*.

4.6 Determinar o melhor modelo - AutoML

O AutoML permitiu averiguar que o XGBoost é o melhor algoritmo a utilizar no nosso conjunto de dados, como mostra os seguintes resultados apresentados na Figura 38.

▾ MODELS

models sorted in order of auc, best first

	model_id	auc	logloss	mean_per_class_error	rmse	mse
0	XGBoost_3_AutoML_20200430_123550	0.9999897879705701	0.0018570301433298768	0.0008789254540751311	0.019287416482910356	0.00037200443458524214
1	XGBoost_1_AutoML_20200430_123550	0.9999864479924981	0.00224233262219216	0.0011317413452818208	0.021236023325998974	0.0004509686867023725
2	XGBoost_2_AutoML_20200430_123550	0.9999739370171226	0.003165600570044567	0.0019974091312955603	0.024778305564296572	0.0006139644266376504
3	XGBoost_grid_1_AutoML_20200430_123550_model_3	0.9999670990471737	0.00426766115155645	0.002031831354282721	0.02775983353264025	0.000770608357759898
4	XGBoost_grid_1_AutoML_20200430_123550_model_2	0.9999586871357956	0.0044281918260287 27	0.00217742713309574	0.02877712768505496	0.0008281230778019567
5	XGBoost_grid_1_AutoML_20200430_123550_model_1	0.9998819512945063	0.004174945873539225	0.001401768347982091	0.02284427641312618	0.0005218609648393131
6	StackedEnsemble_AllModels_AutoML_20200430_123550	0.9998000381773746	0.002851944390284661	0.0011317413452818208	0.021173025272116237	0.0004482699917367283
7	StackedEnsemble_BestOfFamily_AutoML_20200430_123550	0.9996895183649376	0.00276556084394082	0.0010245212328881506	0.020575192549538813	0.0004233385484505975 4
8	GLM_grid_1_AutoML_20200430_123550_model_1	0.9996417579178317	0.008203166534578142	0.0031821811488229475	0.03872912020597479	0.001499944751928845
9	XGBoost_grid_1_AutoML_20200430_123550_model_4	0.9985869011327143	0.06566381416388901	0.0035368663712262932	0.07143183020911946	0.005102506367024471

Figura 38: Resultado obtido pelo AutoML

Com base nos resultados, o algoritmo que teve melhor desempenho nas métricas analisadas de acordo com o AutoML é o XGBoost. O AutoML fornece os melhores modelos tendo em conta as métricas em discrepância do tempo, ou seja, o tempo nao é medido.

Para analisar o comportamento dos algoritmos perante uma escolha dos parâmetros a usar foi feito um procedimento prévio de escolha de paramêtros. Através do AutoML procedeu-se à análise/contributo de cada variavél para a classe e chegou-se aos níveis de importância/significado ilustrados na Figura 39. O objetivo deste passo é verificar se com menos dados os algoritmos continuam a ter bons resultados de classificação e se o tempo de classificação reduz.

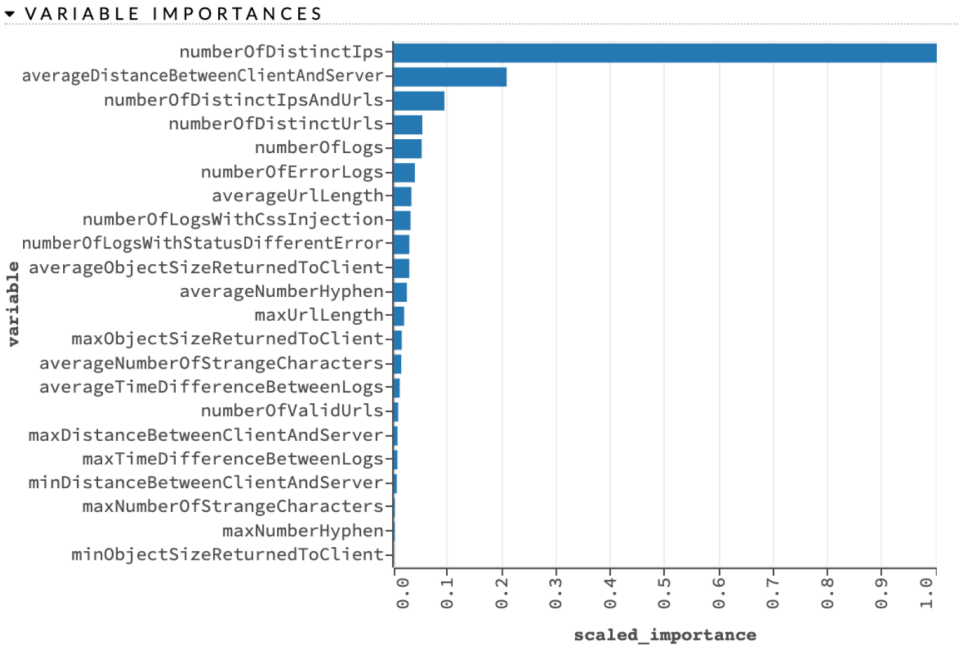


Figura 39: Importância das variàveis no conjunto de dados

De acordo com a Figura 39 as variáveis dominantes são numberOfDistinctIps, averageDistanceBetweenClientAndServer, numberOfDistintIpsAndUrls. Os resultados da previsão realizada com os parâmetros mais significativos e fornecidos pelo AutoML é ilustrado na Figura 40.

▾ PREDICTION

model	xgboost-b786d5a3-2b06-4a4d-bde2-82d25ec59920
model_checksum	1840556713862961676
frame	validation
frame_checksum	7123169215930386006
description	·
model_category	Regression
scoring_time	1595486641668
predictions	prediction-5755fce3-3b05-4444-b91b-e109f11c1164
MSE	0.010063
RMSE	0.100315
nobs	17526
custom_metric_name	·
custom_metric_value	0
r2	0.659964
mean_residual_deviance	0.010063
mae	0.016368
rmsle	0.069033

Figura 40: Resultados obtidos pela previsão efetuada pelo algoritmo XGBoost com os paramêtros definidos pelo AutoML

A imagem anterior mostra os resultados das métricas referentes à taxa de previsão e ao erro associado à utilização do melhor modelo obtido pelo AutoML com os parâmetros sugeridos pelo mesmo. Desta forma, os resultados servem de base para o estudo para alcançar melhores desempenhos do algoritmo. Também de referir que os parâmetros ótimos sugeridos pelo AutoML também serviram como apoio e ponto de partida para começar a descobrir e a investigar os melhor parâmetros para utilizar no modelo de previsão. Contudo, apesar dos resultados obtidos serem bons com uma taxa de MAE relativamente baixa, estes resultados apenas servem de justificação para o uso do algoritmo de XGBoost, como escolha de algoritmo para esta solução. O objetivo foi descobrir com o AutoML o ponto de partida para conseguir atingir os parâmetros ótimos de configuração para este algoritmo.

O aspeto que se seguiu aos resultados obtidos passou pelo ajuste dos paramêtros do algoritmo XGBoost no sentido de melhorar os resultados nomeadamente reduzir a taxa de erro (MAE e RMSE).

4.7 Tunning do XGBoost

Apesar de se utilizar o AutoML para, de forma automática, definir os parâmetros a analisar e escolher o algoritmo a usar torna-se fundamental analisar os hiperparâmetros dos modelos de forma a otimizar os resultados. Para o efeito foi criado um programa em PYTHON que define os hiperparâmetros a usar pelo algoritmo XGBoost. Uma alternativa ao programa PYTHON criado seria utilizar a definição dos paramêtros incluída no H2O, porém este dispensava de mais recursos computacionais para avaliar todas as combinações possíveis para chegar a uma configuração ótima para o modelo. O ambiente em uso não permitiu tirar partido desta funcionalidade do H2O.

Os parâmetros por defeito são os apresentados na Tabela 4.3.

Parâmetros configurados	valor
max_depth	6
min_child_weight	1
eta	0.3
subsample	1
colsample_bytree	1

Tabela 4.3: Paramêtros definidos por padrão que vamos utilizamos inicialmente

Iniciou-se este processo avaliando inicialmente as configurações por defeito e depois estas foram sendo alteradas e avaliadas para aferir a optimização conseguida. Fez-se uso de outros parâmetros para limitar o número de iterações efetuadas para chegar a resultados concretos para os parâmetros, como, por exemplo:

- “num_boost_round” define o número de iterações necessárias para otimizar árvores e outro parâmetro
- ”early_stopping_rounds”define o número de iterações máximas sem existir qualquer melhoria nas árvores criadas em que o algoritmo de pesquisa usa o melhor valor de MAE como critério de paragem para a melhor solução.

Com as configurações por defeitos foram obtidos os valores de MAE apresentados na Figura 41.

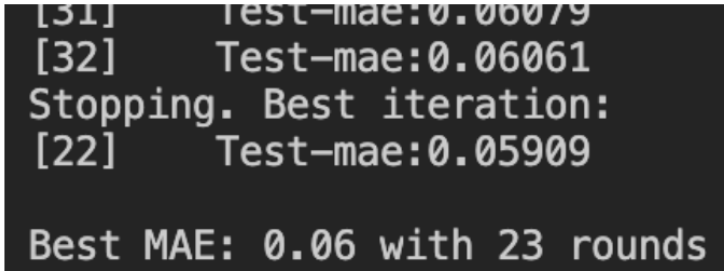


Figura 41: Resultados obtidos para o MAE com paramêtros por defeito

Como se verifica pela Figura 41, na iteração 22 o valor do MAE foi de 0.05909 e mesmo com mais iterações não se verificou melhoria no valor. O algoritmo parou assim na iteração 33. Este resultado é para os parâmetros por defeito. O mesmo processo foi efetuado considerando outros hiperparametros.

4.7.1 Parâmetros max_depth e min_child_weight

Dois dos hiperparametros alterados foram o ”max_depth”e ”min_child_weight”. Estes paramêtros estão relacionados com o melhor trade-off-bias-variance. Para testar diferentes hiperparâmetros foi criado um array com possíveis combinações para estas duas variáveis (código na Figura 42). O resultado é um array de tuplos com as possíveis combinações.

```
gridsearch_params = [
    (max_depth, min_child_weight)
    for max_depth in range(3,12)
    for min_child_weight in range(2,12)
]
```

Figura 42: Gerar valores aleatórios para as variáveis ”max_depth”e ”min_child_weight”

Para determinar a melhor combinação possível destas duas variáveis utilizou-se a função `CV` do XGBoost. Esta função aplica cross-validation no dataset de treino, aplicando os valores dos hiperparâmetros e tenta desta forma encontrar a melhor combinação possível das variáveis, considerando o melhor valor de MAE. O resultado obtido é ilustrado na Figura 43.

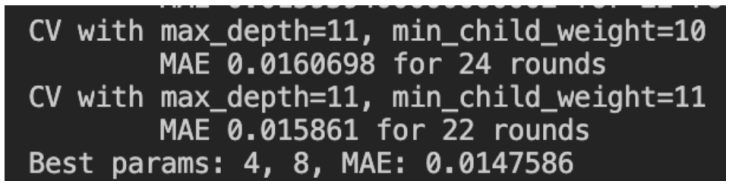


Figura 43: Valor otimizados para as variáveis ”max_depth”e ”min_child_weight”

Como se pode constatar pela Figura 43 o melhor resultado de todas as combinações geradas para o max_depth é 4 e para min_child_weight é 8. É de salientar que outras combinações com resultados melhores podem ainda ser encontradas. É uma questão de tempo de computação para testar mais alternativas.

4.7.2 Parâmetros subsample e colsample_bytree

Os hiperparâmetros subsample e colsample_bytree dizem respeito ao conjunto de dados que é analisado a cada iteração, ou seja, em todas as árvores apenas se analisam certas quantidades limitadas para não ocorrer *over-fitting*. O procedimento para obter o array de tuplos com diferentes combinações é o mesmo que o explicado no processo anterior. Contudo, nesta etapa é feito uma atualização dos parâmetros com os resultados obtidos anteriormente. O resultado obtido com o ajuste nos hiperparâmetros é ilustrado na Figura 44.

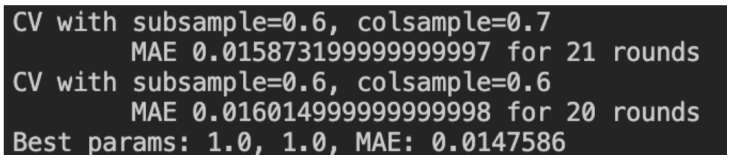


Figura 44: Valor otimizados para as variáveis ”subsample”e ”colsample_bytree”

Como se verifica pela Figura 44, o melhor resultado obtido para subsample é 1 e para colsample_bytree é também 1.

4.7.3 Parâmetro ETA

O parâmetro ”ETA”controla a taxa de aprendizagem, ou seja, define a quantidade de correções que são feitas em cada etapa. Nos algoritmos de XGBoost cada iteração ou nova árvore criada corrige erros da anterior. O objetivo é conseguir encontrar valores baixos para o ETA para tornar o modelo robusto e evitar o *over-fitting*.

Os resultados obtidos são ilustrados na Figura 45.

```
CV with eta=0.005
--- 51.77551198005676 seconds ---
        MAE 0.061407 for 998 rounds

Best params: 0.3, MAE: 0.0589178
```

Figura 45: Valor otimizados para as variável ”eta”

O resultados obtidos através de CV (Cross-Validation) com o ajuste dos hiperparâmetros determinou que 0.3 é o valor ETA que minimiza o valor de MAE.

4.7.4 Análise final após ajuste dos hiperparâmetros

Considerando o ajuste dos hiperparâmetros, determinados anteriormente, o dataset foi submetido a nova análise através do algoritmo XGBoost. Tal como anteriormente o dataset foi dividido em três: 70% treino, 15% validação e 15% teste de previções. O resultado é apresentado na Figura 46 sob a forma de matriz de confusão, evidenciando de forma clara as falhas do modelo. Por exemplo, em 28221 frames, 294 foram classificados como sendo maliciosos quando na verdade não o eram. Em 907 frames maliciosos, 135 foram classificados como não sendo maliciosos (15%) e 772 foram corretamente classificados como maliciosos (85%).

	Predicted No	Predicted Yes	Total
Actual No	27927	294	28221
Actual Yes	135	772	907
Total	28062	1066	29128

Figura 46: Matriz de confusão com os resultados obtidos para o conjunto de dados da solução

Com a definição dos parâmetros verificaram-se melhorias no MAE como se pode ver na Figura 47.

▾ PREDICTION

model	xgboost-c49cbcb4-ffb0-4dcb-a3b8-d9e6482953a1
model_checksum	-474297913906397828
frame	validation
frame_checksum	7123169215930386006
description	·
model_category	Regression
scoring_time	1595486485681
predictions	prediction-caf25d36-3109-4e2a-a280-4f873b8f6026
MSE	0.008353
RMSE	0.091396
nobs	17526
custom_metric_name	·
custom_metric_value	0
r2	0.717740
mean_residual_deviance	0.008353
mae	0.015318
rmsle	0.063447

Figura 47: Resultados obtidos com os melhores parâmetros encontrados

De forma geral conseguimos dar um improve ao MAE de 0.016 para 0.015, isto devido à configuração dos parâmetros de otimização do XGBoost.

De forma resumida, o valores ótimos dos hiperparâmetros determinados são os apresentados na Tabela 4.4.

Parâmetros ótimos	valor
max_depth	4
min_child_weight	8
eta	0.3
subsample	1
colsample_bytree	1

Tabela 4.4: Parâmetros ótimos encontrados para o algoritmo de XGBoost

O resultado das métricas obtido com a aplicação dos hiperparâmetros ótimos encontrados, descrito na Tabela 4.4 levou a atingir os seguintes resultados 4.5.

Accuracy	Precision	Recall	F1 Score
0.9852	0.9896	0.9951	0.9923

Tabela 4.5: Resultados para as métricas de calculo de desempenho do modelo

Para chegar a estes resultados foi criado um script em Python que usufrui da biblioteca desenvolvida para o algoritmo XGBoost que possui a função “XGBClassifier”, em que a mesma foi configurada com os parâmetros ótimos encontrados e descritos na Tabela 2.1, em que a sua execuçao permitiu chegar aos resultados descritos na Tabela 4.5.

Tempo total de processamento em segundos	Tempo que engloba o processo de ML	Número de estâncias analisadas
14.82 s	247 ms	2000
9.67 s	131 ms	1000
1.38 s	82 ms	15
0.79 s	75ms	5

Tabela 4.6: Resultados de desempenho de N estâncias

Como se pode averiguar na Tabela 4.6 é descriminado os tempos que a solução apresenta para prever algumas estâncias. Assim sendo, os valores apresentados dizem respeito a todo o fluxo da aplicação que foi descrito ao longo deste artigo. Dos processos que compõem a aplicação o que apresenta mais impacto é o de enriquecimento de variáveis que é certa de 50% do tempo de processamento. É de referir que estes testes foram realizados no ambiente descrito na secção 4.1 e que dependem de aplicações externas para enriquecer os dados referentes à localização do utilizador.

4.8 Análise Crítica

Ao longo deste capítulo foram apresentados os resultados obtidos pela análise de dados obtidos a partir de uma aplicação Web, modelados para integrar um dataset e enriquecidos para acrescentar valor e significado aos mesmos. O dataset foi criado, tratado e submetido a diferentes abordagens para análise. Os resultados iniciais revelaram-se logo interessantes, na medida em que através da utilização de algoritmos de *Machine Learning* é viável classificar dados de uma aplicação Web no sentido de verificar se a aplicação está a ser alvo de um ciberataque ou com comportamento anómalo.

A utilização de *Machine Learning* é hoje em dia facilitada pela adoção de tecnologias como o H2O e mecanismos capazes de automatizar o processo de escolha dos parâmetros mais adequados a modelar e ainda o algoritmo que melhor se ajusta aos dados. É certo que os resultados obtidos foram desde logo interessantes, porém a optimização de hiperparâmetros, tal como se pode confirmar pelos resultados, é também relevante permitindo melhorar os modelos.

Considerando o próposito do sistema a criar, os resultandos são bastante animadores. Agrupando os dados de uma aplicação Web em frames (no nosso caso 30 segundos) e com base num dataset prévio devidamente classificado, torna-se possível prever anomalias, que po-

dem por exemplo derivar de um ciberataque, com mais de 98% de eficácia e certeza. Será muito interessante e importante ver como o modelo se comporta perante a análise de outras aplicações, ainda que a expetativa seja que o mesmo se adapte de forma fácil às mesmas.

Capítulo 5

Conclusão

A digitalização a que assistimos e o número de dispositivos capazes de comunicar entre si cresce igualmente a bom ritmo, trazendo novas funcionalidades para as organizações. Se por um lado temos a componente de evolução tecnológica, por outra verifica-se o crescimento no número de ameaças cibernética. Estas ameças tem levado a inúmeros ataques com consequências económicas avultadas, prejuízos de reputação e em casos recentes pondo vidas humanas em causa. A deteção de atividade maliciosa é um problema difícil de colmatar e, por vezes, não é suficiente a utilização de métodos e abordagens tradicionais de segurança tais como a utilização de firewalls, sistemas de deteção de intrusos, antivírus e mecanismos criptograficos.

Ao longo deste capítulo foram apresentados os resultados obtidos pela análise de dados a partir de uma aplicação Web, modelados para integrar um dataset e enriquecidos para acrescentar valor e significado aos mesmos. O dataset foi criado, tratado e submetido a análise. Os resultados iniciais revelaram-se logo interessantes, na medida em que através da utilização de algoritmos de *Machine Learning* é viável classificar dados de uma aplicação Web no sentido de verificar se a aplicação está a ser alvo de um ciberataque ou com comportamento anómalo.

Considerando o próposito do sistema a criar, os resultandos são bastante animadores. Agrupando os dados de uma aplicação Web em frames (no nosso caso 30 segundos) e com base num dataset prévio devidamente classificado, torna-se possível prever anomalias, que podem por exemplo derivar de um ciberataque, com mais de 98% de eficácia e certeza.

Neste trabalho abordou-se uma solução para a deteção de ataques feitos às aplicações de

forma a prevenir e bloquear antes que estes se tornem prejudiciais. O foco foi construir uma aplicação robusta baseada no desempenho na capacidade de deteção de ataques sem interferir no funcionamento e desempenho das aplicações.

Com isto, a captação de dados e de informação para análise e consumo da aplicação devesse ao facto da interação que existe entre o utilizador e a aplicação. Torna-se inovador no sentido que abrange mais ataques distintos que podem ser feitos à aplicação e pela utilização de *Machine Learning* para detetar novos padrões que possam indiciar um novo ataque desconhecido.

Como analisado as abordagens existentes focam-se em ataques específicos que abrange pontos específicos de um vasta gama de possíveis contornos que podem levar a obtenção de informação ou prejuízo para as organizações.

Desta forma, a solução apresenta os seguintes fatores:

- Independente da aplicação onde é utilizada.
- Pouco impacto nos recursos da maquina onde é utilizada.
- Não afeta a interação e funcionamento da aplicação hospedeira.
- Criação de modelos de dados e de informação que podem ser utilizadas para análise académica ou da comunidade.

Uma das suas grandes vantagem é como é feita sobre a arquitetura de *REST* API, simplesmente necessita de receber os pedidos feitos ao servidor, dispensando de informação ou configurações extras na aplicação usada, ou seja, apenas necessita de ser instalada e de recolher os logs para fazer a análise, durante esse processo todos os dados importantes e os resultados gerados são todos guardados como histórico.

O trabalho desenvolvido serviu para criar uma aplicação robusta para a deteção de ataques em aplicações.

5.1 Trabalho futuro

Nesta etapa, temos como objetivo testar a aplicação em ambientes reais, para através do *feedback* fazer melhorias no produto. Temos também como objetivo analisar o comportamento da aplicações em outras aplicações sem ser o Moodle da ESTG e analisar o seu desempenho em ambientes com grande carga de utilização para avaliar com mais pormenor o tempo de resposta.

Um dos grandes objetivos que temos é fazer a integração desta abordagem com ELK Stack (Elasticsearch, Logstash, and Kibana) para monitorização e análise dos pedidos realizados pela interação do utilizador para ter uma visualização gráfica de todos os acontecimentos da aplicação em *real-time* e para análise pormenorizada de pedidos que são categorizados como ataques.

Bibliografia

[1] Microsoft . Microsoft Defender Advanced Threat Protection, December 2019.

[2] Jain Aarshay. XGBoost Parameters | XGBoost Parameter Tuning, March 2016.

[3] Ravanshad Abolfazl. Gradient Boosting vs Random Forest | by Abolfazl Ravanshad | Medium, April 2018.

[4] Rahul Agarwal. The 5 Most Useful Techniques to Handle Imbalanced Datasets, 2020.

[5] Perlman Al. The Growing Role of Machine Learning in Cybersecurity, June 2019.

[6] Anas Al-Masri. What Are Overfitting and Underfitting in Machine Learning?, June 2019.

[7] Algorithmia. The importance of machine learning data, March 2020.

[8] AWS Amazon. Model Fit: Underfitting vs. Overfitting - Amazon Machine Learning.

[9] Nagpal Anuja. L1 and L2 Regularization Methods. Machine Learning | by Anuja Nagpal | Towards Data Science, October 2017.

[10] ET Anumol. Use of machine learning algorithms with siem for attack prediction. In *Intelligent Computing, Communication and Devices*, pages 231–235. Springer, 2015.

[11] Apache. Log Files - Apache HTTP Server Version 2.4.

[12] Vinícius Batistela, Marco Antônio, and Marco Trentin. Identificação e análise de tráfego malicioso através do uso de honeypots. *Revista Brasileira de Computação Aplicada*, 1, 01 2009.

[13] Sharma Bhuvan. What are the advantages/disadvantages of using Gradient Boosting over Random Forests? - Quora, November 2015.

[14] Bitdefender. Delivering Security and Performance in the Continuous Data Center.

[15] Jason Brownlee. Supervised and Unsupervised Machine Learning Algorithms. *Machine Learning Mastery*, 16(03), March 2016.

[16] AT&T Business. Open Threat Exchange (OTX) | AlienVault | ATT Cybersecurity.

[17] N DuPaul. Application security vulnerability: Code flaws insecure code.

[18] Gottsegen Gordon. Machine Learning Cybersecurity: How It Works and Companies to Know, June 2019.

[19] H20. AutoML: Automatic Machine Learning — H2O 3.30.1.1 documentation.

[20] Cynthia Harvey. Application Security: How to Secure the Many Apps in Your Enterprise, 2018.

[21] Robert W Hayden. A dataset that is 44% outliers. *Journal of Statistics Education*, 13(1), 2005.

[22] Martin Husák, Jana Komarkova, Elias Bou-Harb, and Pavel Celeda. Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Communications Surveys and Tutorials*, PP, 09 2018.

[23] IBM. Machine Learning - United Kingdom | IBM.

[24] Bowling Jeramiah. AlienVault: the Future of Security Information Management | Linux Journal, May 2010.

[25] Frost Jim. 5 Ways to Find Outliers in Your Data, October 2019.

[26] JJ. MAE and RMSE — Which Metric is Better?, March 2016.

[27] Tasos Zapotek Laskos. Arachni—web application security scanner framework. *Retrieved at¡¿, Retrieved Date*, page 10, 2011.

[28] Liquisearch. Random Forest - Disadvantages.

[29] Liquisearch. Random Forest - Features and Advantages | Features Advantages.

[30] Evan Lutins. Ensemble Methods in Machine Learning: What are They and Why Use Them? *Available in: https://towardsdatascience. com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f*, August 2019.

[31] Ferruh Mavituna. Sql injection cheat sheet. *Document Version*, 1, 2009.

[32] Aditya Mishra. Metrics to evaluate your machine learning algorithm. *Towards Data Science*, 2018.

[33] Vishal Morde. XGBoost Algorithm: Long May She Reign!, April 2019.

[34] Kharkovyna Oleksii. Machine Learning vs Traditional Programming | by Oleksii Kharkovyna | Towards Data Science, August 2020.

[35] ONF. Software-Defined Networking (SDN) Definition.

[36] AI Oodles. TensorFlow Vs H2O: The Best Enterprise-grade Machine Learning Tool | by Oodles AI | Medium, April 2020.

[37] Owasp Org. Code Injection | OWASP.

[38] Marlin Ouverson. Cost-effective Web Applications for Business and Non-Profits.

[39] Christian Pascual. Tutorial: Understanding Linear Regression and Regression Error Metrics, September 2018.

[40] Prasad Patil. What is exploratory data analysis? *Toward Data Science*, 2018.

[41] VV Preetham. Mathematical foundation for Noise, Bias and Variance in #NeuralNetworks | by Preetham V V | Autonomous Agents—#AI | Medium, September 2016.

[42] Draelos Rachel. Measuring Performance: The Confusion Matrix, February 2019.

[43] Singh Rajeev. What is URL Encoding and How does it work?

[44] Abolfazl Ravanshad. Gradient Boosting vs Random Forest, August 2019.

[45] Nina Reunes. Numpy, Pandas and SciKit Learn Explained., December 2019.

[46] Rikunert. SMOTE explained for noobs - Synthetic Minority Over-sampling TEchnique line by line · Rich Data, November 2017.

[47] Nilaykumar Kiran Sangani and Haroot Zarger. Machine learning in application security. In *Advances in Security in Computing and Communications*. IntechOpen, 2017.

[48] Jan Seidl. Goldeneye, June 2020.

[49] Harshdeep Singh. Understanding gradient boosting machines. *Towards Data Science*, 3, 2018.

[50] Harshdeep Singh. Understanding Gradient Boosting Machines, November 2018.

[51] Kyle Soska and Nicolas Christin. Automatically detecting vulnerable websites before they turn malicious. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 625–640, 2014.

[52] Glen Stephanie. Decision Tree vs Random Forest vs Gradient Boosting Machines: Explained Simply, June 2019.

[53] Synopsys Editorial Team. Agile application security vs. traditional application security | Synopsys, September 2015.

[54] Hugo Séneca. Visão | CNPD: Hospital do Barreiro multado em 400 mil euros por permitir acessos indevidos a processos clínicos, September 2018.

[55] Agnes Talalaev. Website Hacking Statistics in 2020, April 2020.

[56] Boyle Tara. Dealing with Imbalanced Data. Imbalanced classes are a common problem... | by Tara Boyle | Towards Data Science, February 2019.

[57] TheEconomyJournal.com. Cybersecurity: necessary or just a business?

[58] Chen Tianqi. Tianqi Chen's answer to What is the difference between the R gbm (gradient boosting machine) and xgboost (extreme gradient boosting)?, September 2015.

[59] Shashwat Tiwari. Complete Guide to Machine Learning Evaluation Metrics, May 2020.

[60] Trustwave. 5 Most Common Web Application Attacks (And 3 Security Recommendations), July 2018.

[61] Emmanuel Urias. Applying Machine Learning and AI to Improve Cyber Security - INVID.

[62] Solomon Ogbomon Uwagbole, William J. Buchanan, and Lu Fan. An applied pattern-driven corpus to predictive analytics in mitigating SQL injection attack. In *2017 Seventh International Conference on Emerging Security Technologies (EST)*, pages 12–17, Canterbury, September 2017. IEEE.

[63] Armando Vieira. Predicting online user behaviour using deep learning algorithms. *arXiv preprint arXiv:1511.06247*, 11 2015.

[64] VMware. What is Application Security? | VMware Glossary.

[65] Koehrsen Will. Overfitting vs. Underfitting: A Complete Example | by Will Koehrsen | Towards Data Science, January 2018.