POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Aprendizagem Computacional aplicada à Deteção de Intrusões - Efeito das Técnicas de Balanceamento de Dados

Mestrado em Cibersegurança e Informática Forense

Hugo Pedro Bessa Almeida

Leiria, novembro de 2021

Esta página foi intencionalmente deixada em branco

POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Aprendizagem Computacional aplicada à Deteção de Intrusões - Efeito das Técnicas de Balanceamento de Dados

Mestrado em Cibersegurança e Informática Forense

Hugo Pedro Bessa Almeida

Dissertação realizada sob a orientação do Professor Doutor Carlos Grilo, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, novembro de 2021

Esta página foi intencionalmente deixada em branco

Originalidade e Direitos de Autor

A presente dissertação é original, elaborada unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para a elaborar.

Esta página foi intencionalmente deixada em branco

Agradecimentos

Este trabalho, nesta fase em que vivemos e na minha vida pessoal, foi um tremendo empreendimento e não seria possível sem algumas pessoas muito especiais para mim.

Pretendo agradecer em especial pela orientação do Professor Doutor Carlos Grilo, que desde o primeiro dia foi alguém que me sensibilizou pela sua disponibilidade, personalidade, paciência, conhecimento e grau de exigência tanto a nível científico como pessoal. O seu apoio sempre esteve presente, bem como sua perspetiva sábia e oportuna que muito contribuíram para a evolução deste trabalho, muito obrigado Professor.

Agradeço aos meus pais José e Eduarda, por serem um exemplo, uma força impulsionadora para mim, pela sua preocupação, constante motivação, amor e carinho. Espero um dia fazer o mesmo que fazem comigo.

Agradeço à minha esposa Alexandra, pela sua paciência, amor, e apoio em todos empreendimentos em que me coloco.

À minha filha Clarinha que é o meu maior feito, o meu orgulho, a razão de tudo, para sempre.

Por fim, meu agradecimento a todos os meus amigos e família que me apoiaram e motivaram durante o desenvolvimento deste trabalho.

Esta página foi intencionalmente deixada em branco

Resumo

A cibersegurança tem vindo a ganhar cada vez mais importância. Atualmente, as redes computacionais profissionais e domésticas estão expostas a um grande número de ataques maliciosos. Todo este crescimento e evolução torna cada vez mais complexa e pertinente a deteção e prevenção destas ameaças. Umas das áreas que tem sido preponderante nesta tarefa é a aprendizagem computacional. No entanto, grande parte dos trabalhos existentes nesta área concentra-se na realização de experiências com novos algoritmos de classificação, sendo o efeito das operações de limpeza, pré-processamento e balanceamento dos dados relegados para segundo plano. Isto é particularmente importante visto que, um dos problemas comuns em conjuntos de dados reais e, em particular, nos conjuntos de dados de tráfego de rede, é a falta de balanceamento dos dados.

O objetivo deste trabalho consiste em estudar os efeitos de duas técnicas de balanceamento dos dados opostas no processo da aprendizagem computacional, a saber, o *Random UnderSampling* e o *Random OverSampling*. Para tal, foi escolhido o conjunto de dados CSE-CIC-IDS-2018, cujo conteúdo tende a simular o tráfego passível de ser encontrado numa rede computacional empresarial.

Para atingir o objetivo proposto foram aplicadas várias técnicas de limpeza e de pré-processamento, foram também criadas várias versões de conjuntos de dados para treino, aplicados métodos de seleção de atributos e algoritmos de classificação.

Os resultados dos testes realizados permitem formular as seguintes conclusões: 1) existe um melhoramento dos resultados até um determinado nível de *undersampling* mas, para além desse limite, a redução da quantidade de dados leva a uma deterioração dos resultados; 2) o efeito do operador de *oversampling Random OverSampling* é muito pouco significativo; 3) estas conclusões mantêm-se quando se utilizam conjuntos de dados sobre os quais foram realizadas operações de seleção de atributos.

Palavras-chave: Cibersegurança, *Intrusion Detection Systems*, aprendizagem computacional, balanceamento de dados, pré-processamento, seleção de atributos.

Esta página foi intencionalmente deixada em branco

Abstract

Cybersecurity is becoming increasingly important. Today, professional and domestic computer networks are exposed to a large number of malicious attacks. All this growth and evolution makes the detection and prevention of these threats increasingly complex and pertinent. One of the areas that has been preponderant in this task is machine learning. However, most of the existing works in this area focus on experiments with new classification algorithms, and the effect of data cleaning, pre-processing and balancing operations are relegated to second place. This is particularly important since one of the common problems in real datasets, and specifically in network traffic datasets, is the lack of data balancing.

The objective of this work is to study the effects of two opposite data balancing techniques on the process of machine learning, namely, Random UnderSampling and Random OverSampling. For this purpose, the CSE-CIC-IDS-2018 dataset was chosen, whose content tends to simulate the traffic likely to be encountered in an enterprise computational network.

To achieve the proposed objective, several cleaning and pre-processing techniques were applied, several versions of data sets for training were also created, attribute selection methods and classification algorithms were applied.

The results of the tests carried out allow formulating the following conclusions: 1) there is an improvement in the results up to a certain level of undersampling but, beyond that limit, the reduction of the amount of data leads to a deterioration of the results; 2) the effect of the Random OverSampling operator is very little significant; 3) these conclusions are maintained when using datasets on which attribute selection operations were performed.

Keywords: Cybersecurity, Intrusion Detection Systems, machine learning, data balancing, pre-processing, attribute selection.

Esta página foi intencionalmente deixada em branco

Índice

Esta página foi intencionalmente deixada em branco

Lista de Figuras

Esta página foi intencionalmente deixada em branco

Lista de tabelas

Esta página foi intencionalmente deixada em branco

Lista de Gráficos

Esta página foi intencionalmente deixada em branco

Lista de siglas e acrónimos

ACCS	*Australian Center for Cybersecurity*
APIDS	*Application Protocol-based intrusion detection system*
CIA	*Confidentiality, Integrity, Availability*
CIC	*Canadian Institute for Cybersecurity*
CSE	*Communications Security Establishment*
DDoS	*Distributed Denial-of-Dervice*
DoS	*Denial-of-Service*
ESTG	Escola Superior de Tecnologia e Gestão
FN	*False Negative*
FP	*False Positive*
FTP	*File Transfer Protocol*
HIDS	*Host-based Intrusion Detection System*
HOIC	*High Orbital Cannon*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
IDPS	*Intrusion Detection and Prevention Systems*
IDS	*Intrusion Detection System*
IMAP	*Internet Message Access Protocol*
IP	*Internet Protocol*
IPS	*Intrusion Prevention System*
LOIC	*Low Orbital Cannon*
NaN	*Not a Number*
NB	*Naïve Bayes*
NIDS	*Network Intrusion Detection System*
PIDS	*Protocol based Intrusion Detection System*
ROS	*Random OverSampling*
RUS	*Random UnderSampling*
SMOTE	*Synthetic Minority Oversampling Technique*
SMTP	*Simple Mail Transfer Protocol*
SQL	*Structured Query Language*
SSH	*Secure Shell*
TN	*True Negative*
TP	*True Positive*

Esta página foi intencionalmente deixada em branco

1. Introdução

A cibersegurança é uma área que tem vindo a ganhar cada vez mais importância. Vários fatores têm contribuído para isso, sendo um deles a banalização do uso da internet e, por consequência, dos serviços alojados em *Cloud*. Este aspeto trouxe bastantes benefícios em termos de custos e automatização de processos, mas, por consequência, tornou as empresas e os utilizadores em geral mais expostos aos perigos vindos da internet. Mais recentemente, outro fator que levou a uma maior exposição, foi o aumento significativo do teletrabalho e do número de horas de utilização dos dispositivos devido ao aparecimento da pandemia do Covid 19.

As redes computacionais profissionais e domésticas, estão hoje expostas a um grande número de ataques maliciosos, e com vários perfis diferentes: *Worms*, *Spywares*, *Exploits*, *Denial of Service*, *Ransomware*, *Sql Injection*, *Spam* e *Phishing* para mencionar apenas alguns. Todo este crescimento e evolução torna cada vez mais complexa e pertinente a deteção e prevenção destas ameaças.

Uma das áreas que mais tem sido utilizada para lidar com o problema de deteção e prevenção de ataques, tem sido a aprendizagem computacional. Esta é uma área que tem como objetivo gerar modelos de classificação com base nos dados. No entanto, grande parte dos trabalhos existentes concentra-se na realização de experiências com novos algoritmos de classificação, sendo o efeito das operações de limpeza, pré-processamento e balanceamento dos dados relegados para segundo plano. Leevy e Khoshgoftaar, em [1], confirmam esta perspetiva, sendo um estudo acerca dos modelos de deteção de intrusão baseados no conjunto de dados CSE-CIC-IDS-2018. Os autores referem, em particular, que a maioria dos trabalhos analisados não tratam os problemas de falta de balanceamento dos dados.

De facto, um dos problemas comuns em conjuntos de dados reais e, em particular, nos conjuntos de dados de tráfego de rede é a falta de balanceamento dos dados. Um conjunto de dados considera-se balanceado relativamente a um determinado atributo se existe um equilíbrio entre o número de ocorrências dos diferentes valores desse atributo. Este equilíbrio é particularmente importante no caso do atributo que é utilizado para classificar os dados. Tomando como exemplo o contexto em estudo, se no atributo de tipo de tráfego, todos os tipos de tráfego maliciosos tiverem o mesmo número de ocorrências, que o tráfego benigno/normal, este conjunto de dados pode ser considerado balanceado se as categorias de interesse forem tráfego Benigno e Malicioso. Ora, na realidade, isto não acontece, visto que, felizmente, o número de ocorrências de tráfego benigno é bastante superior ao de ocorrências maliciosas, porque estas ocorrem em muito menor número. No entanto, para os algoritmos de aprendizagem computacional, o facto de um conjunto de dados não estar balanceado dificulta a obtenção de resultados aceitáveis, podendo levar à construção de modelos que resultem em demasiados falsos positivos, ou seja, tráfego benigno identificado como malicioso, ou falsos negativos, isto é, tráfego malicioso identificado como benigno. Nestas

situações há a necessidade de se aplicarem métodos que permitam tornar o conjunto de dados mais balanceado.

O objetivo deste trabalho consiste em estudar os efeitos de duas técnicas de balanceamento dos dados opostas no processo da aprendizagem computacional, a saber, o *Random UnderSampling* e o *Random OverSampling*. De forma breve, o primeiro consiste em eliminar aleatoriamente instâncias das classes mais numerosas e o segundo em duplicar instâncias de classes menos frequentes.

Para isso, são implementadas técnicas de limpeza e balanceamento de dados em diferentes cenários, sendo criados vários grupos de conjuntos de treino que foram utilizados em algoritmos de classificação distintos, que são testados num conjunto de teste de referência. No final do processo, são comparados os resultados obtidos, analisados e apresentadas explicações para os padrões encontrados.

Os restantes capítulos deste documento estão organizados da seguinte forma:

- Capítulo 2, Matéria relacionada – Através da matéria relacionada começamos por introduzir os conteúdos relacionados com os temas que são desenvolvidos nesta dissertação;
- Capítulo 3, Estado de arte – Neste capítulo é analisado o estado de arte dos conteúdos relacionados com o tema deste trabalho;
- Capítulo 4, Metodologias e ferramentas utilizadas – Capítulo onde é descrita a metodologia empregue e as ferramentas utilizadas neste trabalho;
- Capítulo 5, Seleção e limpeza do conjunto de dados – Neste capítulo são descritos aspetos relacionados com a seleção do conjunto de dados bem como as operações de limpeza realizadas sobre os dados;
- Capítulo 6, Construção do conjunto de dados – Neste capítulo são apresentados todos os passos efetuados para a construção dos conjuntos de dados, com as duas técnicas de balanceamento implementadas;
- Capítulo 7, Testes e análise de resultados – Neste capítulo são descritos os passos efetuados para a execução dos testes, e apresentados e discutidos os resultados obtidos.
- Capítulo 8, Conclusão e trabalho futuro – Por fim, é apresentada a conclusão e as perspetivas de conteúdos de melhoramento como trabalho futuro.

2. Matéria relacionada

Neste capítulo são introduzidos os conteúdos relacionados com os temas que são desenvolvidos nesta dissertação. Começamos por descrever os conceitos associados com Sistemas de Deteção de Intrusões, visto que é um destes sistemas que o conjunto de dados utilizado nesta dissertação replica. De seguida, são analisados os conceitos relacionados com a aprendizagem computacional e são ainda descritos os métodos utilizados no trabalho.

2.1. Sistemas de deteção de intrusões

Atualmente, cada vez mais elementos do nosso dia a dia estão interligados e conectados à Internet. Para além dos particulares, as empresas estão totalmente dependentes de serviços que funcionam remotamente via *cloud*, ou necessitam de conexão à internet, tais como a faturação, controlo de frotas, comunicação, desenvolvimento, controlo de versões, acesso remoto, entre outros. Esta constatação é obvia, no entanto, a consequência é a exposição que todos estes serviços geram. As ameaças são constantes e os riscos de intrusão são elevados e com consequências cada vez maiores [2], [3]. Uma intrusão pode ser definida como a tentativa deliberada de alguém ganhar acesso aos recursos de um sistema computacional sem autorização o que poderá afetar toda a segurança do sistema e a informação nele armazenada. Por intrusão podemos considerar ainda, *password cracking*, alterações às configurações do sistema, aproveitamento de vulnerabilidades ou falhas de software ou protocolos, e monitorização ilícita (*sniffing*) de tráfego [4].

Neste sentido, as principais características relacionadas com a segurança de conteúdos e a sua informação ficam comprometidas. Estas características, a saber, disponibilidade, integridade e confidencialidade dos elementos, compõem o modelo "CIA" *(do inglês Confidentiality, Integrity, Availability)*, que serve de base às políticas de segurança de informação dentro de uma organização [5]. Numa breve análise, essas orientações definem-se da seguinte forma:

- Disponibilidade – significa que a informação se mantém disponível a quem dela necessita; para isso, é necessário garantir que todo o software e hardware cumpre as funções de manutenção e disponibilização de informação; [6], [7], [8]
- Integridade – capacidade de garantir que o estado e o conteúdo da informação não é alterado durante a sua circulação nos sistemas informáticos da empresa, ou que não sejam alterados por elementos não autorizados; [7], [8]
- Confidencialidade – garante que informação importante não é acessível a elementos não autorizados. Para tal, é necessário todo um conjunto de proteções e políticas de acessos e privilégios que devem ser implementadas. [7], [8]

É, por isso, importante a implementação de sistemas que permitam a monitorização constante dos serviços do sistema e a consequente deteção e reação a qualquer intrusão que ocorra.

Um sistema de deteção de intrusões (IDS, do inglês *Intrusion Detection System*) é um software que monitoriza uma rede ou hospedeiro a fim de encontrar possíveis intrusões ou eventos anómalos.

2.1.1. Métodos de sistemas de deteção de intrusão

Os IDS têm vindo a evoluir bastante ao logo dos anos. Se inicialmente apenas procediam à deteção de eventos anómalos, posteriormente começaram a tornar-se mais eficientes e com mais ferramentas para deteção, notificação e reação automatizada. [9]

Em termos de métodos de deteção, os sistemas de deteção são caracterizados, essencialmente, das seguintes formas:

- *Signature-based* – o sistema deteta ataques e outros eventos com base em padrões conhecidos, por exemplo, o número de *bytes* ou números de 1 ou 0 (*signatures*) que constam nas amostras de tráfego de rede. Esses padrões também podem estar relacionados com instruções conhecidas que são partes de código *malware*. Devido a este modo de funcionamento, depende de uma base de dados, que deve estar devidamente atualizada, de padrões conhecidos que permitam identificar ataques. Este tipo de abordagem é simples e tem uma baixa percentagem de falsos positivos; [10], [4]
- *Anomaly-based* – este sistema recorre a aprendizagem computacional usando classificadores para catalogar o tráfego. Todo o tráfego que este sistema capta é comparado com um modelo de treino, definindo o resultado como normal ou anormal. [9] Um IDS baseado neste tipo deteção, tenta encontrar anomalias nos padrões de comportamento que se diferenciem do normal comportamento dos sistemas hospedeiros ou da rede que está a monitorizar. Dentro desses padrões de comportamento analisados podem constar a taxa de tráfego, portos usados, protocolos, etc.; [10]
- *Stateful Protocol Analysis* – Este processo, também conhecido como "*Specification based detection*" [11], consiste em analisar e comparar perfis de protocolos geralmente considerados benignos e as suas atividades para dessa forma detetar padrões desviantes. [12]

2.1.2. Área de abrangência e capacidade de reação dos sistemas de deteção

Como referido acima, os sistemas de deteção de intrusão são bastante úteis, mas são sistemas que apenas fazem deteção de anomalias. Para auxiliar neste combate existem ainda os sistemas reativos, que reagem em função das anomalias detetadas, e os sistemas de prevenção de intrusões (IPS), que poderão ter uma capacidade reativa e desenvolver certos procedimentos pré-configurados que permitam desativar ou bloquear certos serviços. [5]

O ideal é a utilização de sistemas que sejam uma combinação dos dois, os denominados Sistemas de Deteção e Prevenção de Intrusão (IDPS, do inglês *Intrusion Detection and Prevention Systems*). As funções principais de um IDPS é registar a informação relacionada com os eventos monitorizados, notificar os administradores de sistema de eventos

considerados relevantes e gerar relatórios de variados tipos. Muitos destes sistemas também tentam reagir às anomalias ou ameaças detetadas tentando várias técnicas, que podem ir desde executar tarefas predefinidas como suspensão de serviços, reconfiguração da *firewall* ou banir endereços IP. [12]

Geralmente, um sistema de deteção e prevenção de intrusão utiliza métodos híbridos de deteção, ou seja, combinações dos métodos já referidos neste documento, *anomaly detetion, stateful protocal analysis and signature-based.* [11]

Para além da sua capacidade reativa, os sistemas de deteção de intrusão podem ser categorizados com base no seu raio de ação ou área de abrangência. Assim, existem:

- **Sistemas de deteção de intrusão em rede (NIDS)** - Sistemas que monitorizam uma rede informática e que são colocados em um ou mais pontos estratégicos para aumentar a eficácia dessa vigilância. Um NIDS está localizado numa zona desmilitarizada (DMZ, do inglês *demilitarized zone*) logo após a *firewall*. Capta o tráfego da rede em tempo real, analisa-o e executa ações defensivas. Estas ações contra o tráfego malicioso podem ser alertas únicos ao administrador ou reações ativas como, por exemplo, o bloqueio de tráfego malicioso.[13], [14]

- **Sistemas de deteção de instrução no host (HIDS) –** Sistemas que inspecionam os dados provenientes dos sistemas hospedeiros e que analisam todo o conteúdo relacionado com sistemas operativos, *logs* de sistema, aplicações e bases de dados. Os HIDS conseguem detetar ameaças internas que não envolvem tráfego de rede. [7]

- **Sistemas de deteção de intrusão baseados em protocolos (PIDS) –** Analisam os protocolos que são usados na partilha de dados entre o sistema e o servidor. [15]

- **Sistemas de deteção de intrusão baseados nos protocolos das aplicações (APIDS) –** Sistemas que monitorizam os protocolos relacionados com aplicações específicas como, por exemplo, a linguagem SQL envolvida nas transações entre uma base de dados e o sistema. [15]

Os IDS que utilizam as funcionalidades e características dos NIDS e dos HIDS são denominados IDS híbridos. Um IDS híbrido utiliza tanto os NIDS como os HIDS para a recolha e análise de informações heterogéneas sobre ameaças cibernéticas. Um IDS que seja preventivo, por exemplo, que possa bloquear autonomamente ataques assim que estes ocorrem, é denominado Sistema de Prevenção de Intrusão (IPS, do inglês *Intrusion Prevention System*). [13], [14], [16]

2.2. Aprendizagem computacional

A aprendizagem computacional consiste num conjunto de técnicas de aprendizagem, imitação e previsão que podem ser utilizadas em tarefas de processamento de informação baseadas na identificação de padrões de comportamento com o mínimo de intervenção

humana [17]. Também pode ser descrita como um processo de resolução de um problema com base num conjunto de dados e consequente construção algorítmica de um modelo estatístico com base nesse conjunto de dados [18], [19]. Em [20], a aprendizagem computacional é definida de uma forma resumida, como um processo automatizado de extrair padrões de um conjunto de dados.

2.2.1. Utilização de aprendizagem computacional

Com base em [21], existem duas características em que se pode considerar a implementação de um sistema baseado em aprendizagem computacional em vez de um programa construído numa linguagem de programação procedimental para resolver um problema, a saber:

- Complexidade - Problemas do dia a dia que são difíceis de transpor para uma programação procedimental, por exemplo: reconhecimento facial, condução de um veículo, diversidade de contextos do problema. Por outro lado, problemas relacionados com a complexidade de trabalhar com grandes volumes de dados e necessidade de grandes capacidades de processamento, por exemplo: motores de busca, arquivos médicos, informação relacionada com astronomia ou meteorologia, redes sociais, etc.

- Adaptabilidade - Esta característica está relacionada com a flexibilidade associada aos dados em análise. Ao contrário de um programa procedimental, muitos dos dados recolhidos para processamento atualmente são bastante dinâmicos e heterogéneos, por exemplo, recorrendo a reconhecimento facial, tipos de corrida ou caminhada, escrita manual, tipos de ataques registados em tráfego de redes, assinaturas de vírus, *spam* de *emails*. Esta diversidade dos dados necessita de sistemas preparados para se adaptarem ao que quer que estejam a analisar e consigam dar o tratamento que é considerado o mais correto para obter resultados com uma alta taxa de sucesso.

2.2.2. Etapas do processo de aprendizagem computacional

O processo de implementação de um modelo de aprendizagem computacional desenvolve-se ao longo de várias etapas, como representado na figura seguinte.

Figura 1 - As diferentes fases do processo de aprendizagem computacional

- **Fase 1 – Recolha de dados (*Data collection*)**

A primeira fase consiste em recolher os dados - que podem ter origem em diferentes fontes - que estão diretamente relacionados com o problema que se pretende solucionar com o

modelo a ser construído. Em geral, o *conjunto de dados* obtido é composto por um conjunto de *instâncias*, cada uma delas consistindo num conjunto de pares *atributo*/*valor*. Cada *instância* representa um caso real do universo em estudo. As instâncias podem ser, por exemplo, fichas médicas, *logs* de *software*, registos de afluência, etc. Independentemente do tipo de dados, estes devem ser obtidos com a consciência do fim a que se destinam para facilitar as fases seguintes. [22]

- **Fase 2 – Pré-processamento dos dados**

Esta fase consiste na realização de operações de limpeza e preparação de todos os dados recolhidos. É frequente os dados recolhidos terem diferentes tipos de problemas. Por exemplo, pode haver valores omissos, errados ou inconsistentes, atributos em que todos os exemplos têm o mesmo valor e, logo, sem utilidade, entre muitos outros problemas.

Uma vez corrigidos os problemas encontrados, é frequente ainda proceder-se a um conjunto de operações sobre os dados que se destinam a colocá-los num formato que potencie ou facilite a construção de um melhor modelo. Por exemplo, pode ser necessário normalizar valores, criar atributos que sejam calculados a partir de um ou mais atributos existentes (o que pode levar à eliminação destes), seleção de atributos, entre outras operações.

- **Fase 3 – Construção e treino do modelo de dados**

É nesta fase que se procede ao treino do modelo. Para isso, é primeiro necessário dividir o conjunto de dados resultante do pré-processamento em dois conjuntos disjuntos: o *conjunto de treino* e o *conjunto de teste*. O conjunto de treino é uma amostra do conjunto de dados inicial que é utilizada durante o processo de treino do modelo. O conjunto de teste é também uma amostra do conjunto de dados inicial que é utilizado para avaliar o desempenho dos modelos gerados mediante determinadas métricas. [23], [24]

É frequente ser criado também outro subconjunto, denominado conjunto de validação, que é utilizado para avaliar o desempenho do modelo durante o processo de treino. Este conjunto é utilizado em situações em que se pretende encontrar o melhor modelo possível para o problema. Em situações em que o objetivo não é otimizar os modelos, como é o caso do presente trabalho, são utilizados apenas os conjuntos de treino e teste.

- **Fase 4 – Avaliação do modelo**

O processo de avaliação consiste em avaliar o modelo aplicando-o ao conjunto de teste de modo a verificar se o modelo é capaz de generalizar para além dos dados que foram utilizados no processo de treino.

- **Fase 5 – Implementação do modelo**

Fase final de passagem do modelo para o ambiente de produção. Nesta fase o modelo é alimentado com os dados gerados num cenário real, o que permitirá ao sistema fornecer resultados com base nos dados inseridos. A implementação do modelo é um processo bastante sensível já que necessita da coordenação de várias equipas, como os responsáveis

pela gestão dos dados, administradores de sistemas e programadores, para a implementação do modelo no ambiente informático da empresa com o mínimo de incompatibilidades.

2.2.3. Tipos de aprendizagem

No âmbito da aprendizagem computacional, existem três tipos principais de aprendizagem: supervisionada, não supervisionada e por reforço crítico.

- **Aprendizagem supervisionada -** O objetivo de um algoritmo de aprendizagem supervisionada é usar um conjunto de dados que inclui um atributo que corresponde ao conceito que se pretende aprender, ou seja, a saída desejada para cada instância. Por exemplo, se um modelo for criado de forma a receber como entrada os dados de um determinado fluxo de tráfego e devolver a probabilidade desse fluxo ser malicioso, o conjunto de treino deveria incluir um atributo cujos valores pertenceriam ao conjunto {benigno, malicioso}. [17], [18]

- **Aprendizagem não supervisionada -** Este tipo de algoritmos utiliza conjuntos de dados que não contêm a saída desejada para cada exemplo (*unlabeled*). O objetivo de um algoritmo de aprendizagem não supervisionada é criar um modelo que receba um conjunto de informações (vetor de atributos) e o transforme num outro conjunto de informações, ou num determinado valor que ajude a resolver um problema. Um dos exemplos mais referenciados na utilização deste tipo de aprendizagem, é o designado de *clustering*, que é uma técnica que permite identificar padrões de valores dentro do conjunto de dados, como dias de maior afluência, dias com maior temperatura, ou as mesas com mais afluência. [17], [18]

- **Aprendizagem por reforço crítico -** Neste caso, o algoritmo não é alimentado com um conjunto de dados de treino que inclua a saída desejada, mas, por cada saída calculada pelo modelo, é-lhe fornecido um sinal que indica se o resultado está correto ou errado. Este sinal é realimentado no modelo permitindo que os seus parâmetros possam ser ajustados progressivamente de modo a melhorar o seu desempenho à medida que o processo de treino progride. Este tipo de algoritmos é utilizado na resolução de problemas específicos e complexos de aprendizagem de longa duração, como jogos, robótica, etc. [20], [25]

2.2.4. Problemas de classificação e regressão

De acordo com a descrição acima, os atributos de um conjunto de dados utilizado num algoritmo de aprendizagem supervisionada podem ser divididos no conjunto de atributos de entrada x e o atributo correspondente à saída desejada y, conhecido por *label*. Este tipo de aprendizagem utiliza um algoritmo para construir um modelo $y = f(x)$ que, idealmente, deve dar a resposta correta para qualquer exemplo do domínio do problema. Ou seja, o objetivo é aperfeiçoar tanto quanto possível o modelo de modo a que, quando confrontado com novos dados de entrada x, este seja capaz de calcular a saída correta y com um grau de precisão satisfatório.

Estes algoritmos têm como objetivo solucionar problemas de dois tipos: Classificação e Regressão:[19], [26], [27]

- **Classificação -** Os problemas de classificação são problemas onde, mediante o conjunto de valores de entrada *x*, o modelo devolve um valor que corresponde à categoria *y* a que corresponde esse conjunto de valores. Podemos ter como exemplos, tipos de ataques de uma rede informática, tipos de doenças, géneros, e outros tipos de categorias. Existem vários algoritmos de classificação conhecidos como Regressão logística, Árvores de Decisão, *Random Forest*, *Multilayer Perceptron*, *Naïve Bayes*, *KNN*, entre outros. [19], [26], [27]
- **Regressão -** No caso dos problemas de regressão, o resultado não consiste na identificação de uma categoria, mas sim na previsão de um valor contínuo *y*. Este valor pode ser um número de calçado, uma altura, um salário, uma taxa de açúcar no sangue, um vencimento, entre outros. O seu modelo de previsão estabelece uma relação entre as variáveis independentes *x* e as variáveis dependentes *y*. Alguns dos algoritmos de regressão mais conhecidos são: Regressão Linear Simples, Regressão Logística e *Support Vector Machines*. [19], [26], [27]

2.3. Algoritmos de aprendizagem supervisionada

Neste trabalho foram utilizados vários algoritmos de aprendizagem supervisionada no problema de identificação de ataques de intrusão, que consiste num problema de classificação. Nesta secção são descritos os algoritmos utilizados.

2.3.1. KNN (IBK)

KNN *(K-Nearest Neighbour Algorithm)* é um algoritmo bastante popular de aprendizagem supervisionada. Este algoritmo baseia-se no pressuposto de que num determinado conjunto de dados, um determinado ponto (ou seja, uma instância) tem dados semelhantes nas suas proximidades, classificando os dados da categoria com mais ocorrências numa área definida pelo valor *K* que representa o número de vizinhos mais próximos [28]. Ou seja, em problemas de classificação, permite que uma instância seja classificada de acordo com a classificação mais comum de entre as K instâncias que lhe estejam mais próximas. Para além do valor de K, podem ser utilizados outros parâmetros, nomeadamente, pesos em caso de empate e diferentes cálculos de métricas para medir a distância entre as instâncias mais próximas.

Este algoritmo pode também ser utilizado em problemas de regressão. Neste caso, o algoritmo procederá calculando a média das instâncias mais próximas. [29], [30]

2.3.2. *Naïve Bayes* (NB)

O *Naïve Bayes* é um algoritmo de aprendizagem supervisionada que aplica o teorema de probabilidade de *Bayes* para prever a probabilidade de ocorrências de um acontecimento baseado em observações anteriores de eventos semelhantes. Este algoritmo é utilizado em aprendizagem computacional em várias situações como, por exemplo, identificar correio de

spam ou no cálculo da probabilidade de um conjunto de sintomas se refletir numa doença. É um algoritmo conhecido pela sua simplicidade. Um conjunto de dados é submetido e com base nos valores dos atributos, o sistema gera a probabilidade de as instâncias pertencerem a uma determinada categoria. Pode, por exemplo, ser utilizado para identificar tráfego malicioso ou normal com base nos diferentes atributos que compõem as instâncias do conjunto de dados. Como foi já referido, é um algoritmo simples e que necessita de poucas amostras para treino, no entanto, revela falhas por não identificar algumas interdependências entre os atributos para efeitos de classificação, o que pode afetar os resultados obtidos. [14], [21], [31], [32]

2.3.3. Redes neuronais (*Mutlilayer Perceptron Neural Network*)

As redes neuronais são um método de aprendizagem que consiste numa rede de unidades de processamento, designados neurónios, cujo funcionamento é vagamente similar ao dos neurónios biológicos. É bastante utilizado em contextos relacionados com classificação de imagens, reconhecimento de objetos, compreensão semântica, entre outros. Um *perceptron* é um modelo único de neurónio que tem servido de base para a maioria das redes neurais. Nas redes neuronais artificiais existem pelo menos três camadas de neurónios: a camada de entrada, a camada escondida (*hidden layers*) e a camada de saída (*output layer*), podendo existir várias camadas escondidas.

Cada ligação entre cada camada tem um certo peso (valor numérico associado). Na camada de entrada há normalmente um nó para cada atributo e na camada de saída há um para cada classe. Os pesos são atribuídos durante o processo de aprendizagem, realizada sobre o conjunto de treino.

Estas redes precisam de ser treinadas num conjunto de dados preparado para o efeito, e, uma vez treinadas, a rede neural está pronta para fazer as suas previsões continuamente. Geralmente, este algoritmo obtém resultados muito positivos, mas pode ser algo lento no cálculo dos seus resultados se o número de unidades for elevado, como acontece nas chamadas redes profundas. [25], [30], [33], [34]

2.3.4. Árvores de decisão - C4.5

As árvores de decisão são algoritmos de aprendizagem computacional usados de forma bastante comum para problemas de classificação e regressão. Estes algoritmos funcionam gerando uma árvore de decisão de acordo com os valores dos seus atributos.

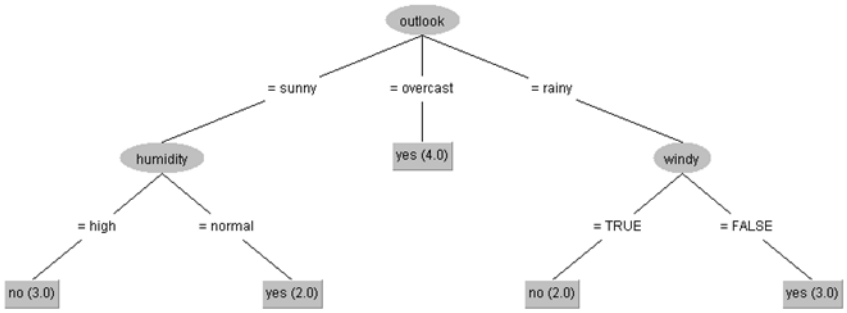

Figura 2 - Exemplo de uma árvore de decisão

O processo desenrola-se selecionando, para o nó raiz, o atributo mais discriminativo, ou seja, aquele com maior ganho de informação; de seguida, é gerado um ramo para cada valor do atributo selecionado, dividindo os exemplos de treino em subconjuntos. O processo repete-se de forma recursiva para cada um dos subconjuntos criando novos nós, sendo que, para cada nó, são descartados atributos que já tenham sido usados. Em cada ramo, o processo pára quando os exemplos de um subconjunto pertencerem todos à mesma classe. A Figura 2 mostra um exemplo de uma árvore de decisão criada pelo algoritmo usando o software Weka, abordado mais abaixo.

O objetivo principal deste procedimento é gerar a árvore de decisão mais pequena possível e obter uma leitura e mapeamento completos do conjunto de dados analisado. [27], [29], [35]

2.3.5. *Random Forest*

Random Forest é um algoritmo combinado (*ensemble learning*), ou seja, combina mais do que um algoritmo de uma variedade de objetos de busca, que criam múltiplas árvores de decisão. É assim criado um conjunto de árvores de decisão ao selecionar aleatoriamente diferentes subconjuntos do conjunto de dados de treino. Em tarefas de classificação, o valor devolvido consiste no resultado mais comum devolvido pelo conjunto de árvores. A floresta é considerada mais robusta quanto maior o número de árvores usadas no processo de decisão.

Este algoritmo constitui um modelo eficiente visto que depois de construir múltiplas árvores de decisão, integra-as numa única árvore de decisão de forma a obter a maior precisão possível no resultado final. É considerado uma extensão do algoritmo de *Bagging* e pode ser utilizado para classificação ou regressão. [34], [36], [37]

2.4. Métricas de avaliação de resultados

Para os testes que foram efetuados neste trabalho, recorrendo aos diferentes algoritmos de aprendizagem computacional, foram utilizadas várias métricas de avaliação de resultados. Quando se analisam os resultados de um algoritmo de aprendizagem computacional, em especial relacionados com sistemas de deteção de intrusão, todas as métricas estão relacionadas com o número de previsões positivas verdadeiras (*TP – True Positive*), previsões negativas verdadeiras (*TN – True Negative*), previsões positivas falsas (*FP – False Positive*) e previsões negativas falsas (*FN – False Negative*). Estas previsões são também

habitualmente representadas sob a forma de uma matriz denominada, matriz de confusão (Figura 3).

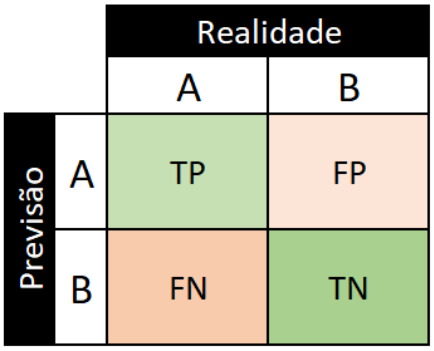

Figura 3 - Exemplo de matriz de confusão

Esta matriz tem como objetivo avaliar ou interpretar os resultados obtidos pelos algoritmos de classificação. Nesta matriz é obtida uma relação de verdadeiro ou falso entre o eixo dos dados previstos e o eixo dos dados reais observados.

De seguida, apresentam-se métricas importantes decorrentes da matriz de confusão.

2.4.1. Acurácia

A acurácia é uma métrica calculada a partir da divisão entre o número de Previsões Verdadeiras (*TP* e *TN*, do inglês *True Positive* e *True Negative*) e o número total de previsões. A acurácia é mais aconselhada para conjuntos de dados balanceados. [19], [38]

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}$$

2.4.2. Precisão

O valor da precisão é calculado a partir da divisão entre as Previsões Positivas Verdadeiras (*TP*) e o total de previsões positivas (*TP* + *FP*). Um resultado baixo, indica a presença de um valor elevado de Previsões Positivas falsas. [19], [38]

$$\text{Precisão} = \frac{TP}{TP + FP}$$

2.4.3. Revocação

A métrica revocação é calculada a partir da divisão das Previsões Positivas Verdadeiras (*TP*) e a soma das Previsões Positivas Verdadeiras (*TP*) e as Previsões Negativas Falsas (*FN*). Analisando as possíveis variações de resultados, é possível verificar que um baixo valor de revocação indica a presença de um elevado número de Previsões Negativas Falsas. [19], [38]

$$\text{Revocação} = \frac{TP}{TP + FN}$$

2.4.4. *F-Measure ou F-Score*

O *F-Measure* é um valor ponderado que resulta da divisão do dobro da multiplicação entre a revocação e a precisão com a soma destas duas métricas. É mais utilizada para conjuntos de dados não balanceados. [19], [38]

$$F - Measure = \frac{2 \times (\text{Revocação} \times \text{Precisão})}{\text{Revocação} + \text{Precisão}}$$

2.5. Técnicas de balanceamento de dados

De acordo com o referido em [39], a maioria dos sistemas de aprendizagem computacional assume que os seus conjuntos de dados de treino estão balanceados, isto é, que o volume de amostras esta distribuído de igual forma por cada categoria que está a ser analisada. No entanto, isto nem sempre acontece no mundo real, ou seja, pode acontecer que o número de instâncias correspondente a uma determinada classe é muito diferente do número de instâncias correspondente a outra classe. Neste caso, estamos perante um problema de desequilíbrio de classes, o que é algo bastante comum e constitui por vezes um obstáculo para a obtenção de bons índices de classificação por parte dos algoritmos de aprendizagem computacional.

Existem inúmeros exemplos deste problema, mas tendo em mente o trabalho em questão, durante a monitorização de um fluxo de dados de modo a identificar diferentes categorias de tráfego malicioso, certamente existirão diferentes tipos de ataques identificados em quantidades diferentes. No entanto, em geral, a situação mais frequente é a ocorrência de tráfego benigno, logo, a classe benigna irá conter bastantes mais exemplos do que qualquer uma das restantes. O mesmo pode acontecer, em geral, com fichas médicas, eventos de segurança, monitorização de qualidade de águas, ou seja, tarefas de monitorização onde a normalidade supera a anomalia ou exceção.

As consequências deste desequilíbrio, passam por uma viciação dos resultados, originando a obtenção de resultados tendencialmente influenciados pela maior presença de exemplos de determinadas classes em detrimento de outras. Por exemplo, o caso mencionado acima de monitorização de anomalias pode dar origem a falsos negativos em excesso por falta de exemplos anómalos em número suficiente para proporcionar uma aprendizagem balanceada. [39]–[41]

2.5.1. Métodos

Com base no referido anteriormente, existem então alguns métodos e técnicas para balancear os dados. É normalmente utilizado o termo *sampling*, um processo que visa equilibrar as discrepâncias de quantidades entre os exemplos de cada classe.

Estas técnicas podem ser divididas em duas abordagens:

Oversampling - Método que gera novas ocorrências das classes com poucos exemplos, a partir dos exemplos existentes. Este processo de criação pode ser efetuado de forma aleatória ou de forma sintética. [39]–[41]

Undersampling - Método que reduz o excesso de ocorrências das classes com maior número de exemplos, de forma a equilibrar o conjunto de dados no seu todo. [39]–[41]

Estas técnicas podem dar origem a alguns problemas específicos, como maior tempo de processamento/cálculo, *"overfitting"* (quando os resultados obtidos com o conjunto de treino são significativamente melhores do que com o com o conjunto de teste) e potencial perda de informação (neste caso, quando conteúdo é removido das categorias com mais ocorrências). Neste trabalho foram utilizados especificamente dois métodos:

- ***Random OverSampling*** – o objetivo deste método é compensar classes com menor frequência, ou seja, que possuem menos ocorrências do que as restantes. Isto é conseguido duplicando as amostras existentes até existir um número de ocorrências que permita considerar as classes balanceadas. Pensando num exemplo prático, consideremos um caso em que possuímos um conjunto de dados com 20 classes e, dessas, 5 são classes minoritárias (com um baixo número de amostras) e as restantes 15 possuem todas um mesmo número de amostras, substancialmente superior. Utilizando o método de *Random Oversampling*, as ocorrências de cada uma das classes minoritárias iriam ser copiadas aleatoriamente até perfazerem os números de ocorrências equivalentes aos das classes maioritárias, tornando assim o conjunto de dados equilibrado. [41], [42]

- ***Random UnderSampling*** - este método funciona na perspetiva contrária, ou seja, elimina amostras aleatórias da classe que possui mais ocorrências de forma que esta fique nivelada em número de ocorrências com as classes com menos amostras. Usando o mesmo exemplo mencionado anteriormente, com este método todas as classes com mais ocorrências do que as classes minoritárias iriam ter as suas amostras removidas de forma aleatória até o conjunto de dados ficar balanceado pelo número de amostras das classes com menos ocorrências. [41], [42]

Para além destes, ainda é possível destacar outros métodos mais complexos, nomeadamente: SMOTE (*Synthetic Minority Over-sampling Technique*); ADASYN (*Adaptive Synthetic Over-Sampling*); *Tomek links*.

- **SMOTE (*Synthetic Minority Over-sampling Technique*)**

  Algoritmo de *oversampling* heurístico. Gera exemplos ou amostras artificiais das classes minoritárias através das trocas entre as instâncias que se encontram mais próximas. É um dos métodos mais atualizados atualmente, tendo servido de ponto de partida para a criação de outros métodos de *oversampling*. De realçar que o *overfitting* é evitado e mistura o espetro da classe minoritária com o da classe maioritária. [39], [41]

- **ADASYN (*Adaptive Synthetic Over-Sampling*)**

  Este algoritmo é considerado uma extensão do SMOTE, onde existe a particularidade de serem criados mais exemplos sintéticos na área onde os exemplos da classe minoritária têm menos ocorrências. [18], [41]

- **Tomek links**

  Este algoritmo pode ser utilizado como método de *undersampling* ou para limpeza de dados. Se tivermos dois pares de instâncias *A* e *B* pertencentes a duas classes diferentes, o par (*A*, *B*) forma uma *Tomek link* se não existe uma instância *C* tal que $d(A, C) < d(A, B)$ ou se $d(B, C) < d(A, C)$**,** onde $d(x, y)$ é a distância entre o par (x, y). Quando um par de Tomek Links é detetado, ambas as instâncias são removidas em operações de limpeza de dados. Em operações de *undersampling* apenas a instância pertencente à classe maioritária é eliminada. [39], [41]

2.6. Seleção de atributos

Em casos reais, os conjuntos de dados podem possuir centenas de atributos e milhares ou milhões de instâncias. Mas nem sempre os atributos são todos importantes para descobrir os padrões referidos. Para encontrar a importância, ou grau de relevância entre os atributos, são utilizados algoritmos de seleção de atributos. [43]

O processo de seleção de atributos consiste essencialmente em escolher um subconjunto de atributos do conjunto de dados originalmente disponibilizado para criar um modelo que mantenha ou melhore os resultados. Este processo é bastante útil quando os conjuntos de dados possuem um grande número de atributos, sendo que apenas um pequeno subconjunto é provavelmente relevante para a resolução do problema. Apesar de a redução de dados poder parecer à primeira vista algo de problemático, a redução do conjunto de dados poderá mesmo melhorar os resultados obtidos pelos modelos de previsão.

Existem benefícios diretos, como a poupança de tempo de processamento e de recursos computacionais, mas estes não são a principal prioridade. Utilizando conjuntos de dados mais pequenos também reduz a probabilidade de *overfitting*. Menos atributos, também permitem a existência de uma menor área de pesquisa e menos possibilidades de decisões erradas e, por consequência, generalizações inconsistentes [15], [44]–[46]. Um benefício adicional consiste na redução do número de atributos que aparecem nos padrões detetados, o que ajuda a fazer com que esses padrões sejam de mais fácil perceção [47].

Os algoritmos de seleção de atributos podem ser classificados em duas categorias: *Filter Approach* e *Wrapper Approach*. [43], [47]

Filter Approach

Este tipo de abordagem precede o processo de classificação. É independente do algoritmo de aprendizagem, sendo computacionalmente simples, rápida e escalável. Usando o método

de filtragem, o conjunto de dados resultante pode posteriormente ser utilizado com diferentes algoritmos classificadores. [48]

Neste trabalho foram utilizados os seguintes métodos de seleção de atributos baseados nesta abordagem:

- *CorrelationAttributeEval* – segundo Jabbar e Mohammed, em [49], é um processo de filtro de correlação (também conhecido como filtro de correlação de *Pearson*) utilizado para calcular a correlação de cada atributo com a classe objetivo. O filtro atribui a cada atributo uma classificação ordenando-os de forma ascendente com base nessa classificação. Em [50] é também referido que este processo consiste num cálculo da correlação entre cada atributo e a classe objetivo para selecionar apenas os atributos que possuem uma correlação positiva ou moderadamente negativa, abandonando os atributos com uma correlação baixa (aproximadamente zero). De acrescentar que este método utiliza o método de procura *Ranker*, que em [27] é referido como um método de procura que classifica atributos individuais de acordo com a sua avaliação.
- *InfoGainAttributeEval* – calcula o ganho de informação de cada atributo relativamente à classe objetivo utilizando também o método de procura *Ranker*. [51]

Wrapper Approach

A abordagem Wrapper usa o próprio algoritmo de classificação para medir a importância do conjunto de atributos e, portanto, depende do modelo de classificação utilizado. Estes métodos obtêm geralmente melhor desempenho do que os de filtragem porque a seleção de atributos é otimizada para o algoritmo de classificação utilizado. No entanto, estes métodos podem revelar-se computacionalmente demasiado dispendiosos para conjuntos de dados de grandes dimensões, uma vez que cada atributo tem de ser avaliado com o algoritmo de classificação escolhido. Esta abordagem identifica os atributos que considera relevantes e prescreve a remoção daqueles que não contribuem para obter melhor acurácia com o modelo. [48]

O método *WrapperSubsetEval*, é um método deste tipo que avalia o desempenho de um algoritmo de aprendizagem selecionado pelo utilizador para diferentes combinações de atributos, devolvendo a melhor combinação encontrada [52]. Neste trabalho realizaram-se testes com este algoritmo, no entanto, este revelou-se computacionalmente muito exigente, pelo que não são apresentados os resultados com ele obtidos.

3. Estado da arte

Neste capítulo é analisado o estado de arte das matérias relacionadas com o tema deste trabalho. Esta análise acompanha as diferentes vertentes consideradas importantes ao longo do trabalho, nomeadamente, o conjunto de dados, pré-processamento e balanceamento do conjunto de dados, seleção de atributos, classificação e tratamento dos tipos de ataques.

3.1. Conjuntos de dados

A escolha do conjunto de dados foi o ponto de partida para o início deste trabalho pelo que, de seguida, é abordado o que é um conjunto de dados de tráfego de redes e analisados alguns dos conjuntos de dados relacionados com sistemas de deteção de intrusão.

Segundo *Martinez Garre et al*., em [53], um conjunto de dados é uma tabela onde cada coluna corresponde a um atributo e cada linha corresponde a uma amostra real, completa e bem catalogada desse conjunto de dados. Em [54] é referido que os conjuntos de dados de tráfego de redes podem ser categorizados em conjuntos *Packet-Based* e *Flow-Based*. Segundo o mesmo artigo, a diferença é que, enquanto os conjuntos de dados da categoria *Flow-based* contêm dados essencialmente com meta-informação acerca das conexões de rede, os dados da categoria *Packet-based* contêm também os *payloads*, ou seja, os pacotes de dados efetivamente transmitidos.

3.1.1. Conjuntos de dados para sistemas de deteção de intrusão

Alguns dos conjuntos de dados mais conhecidos para proceder a análise de tráfego, são o KDD99, DARPA 1998 e DARPA 1999. No entanto, tendo em conta as datas em que foram criados, o seu conteúdo já não se revela capaz de simular situações atuais. [7], [55]

Atualmente, existem alguns conjuntos de dados com conteúdo adaptado ou gerado artificialmente [54]. Com base na investigação efetuada, importa referir alguns destes conjuntos considerados relevantes por diversos autores e de alguma forma relacionados com o conjunto de dados selecionado para este trabalho:

- **NSL-KDD** - Este conjunto de dados é uma evolução do KDD99 e veio dar resposta a alguns problemas encontrados no conjunto de dados original, tais como uma grande quantidade de entradas duplicadas. Em [56] regista-se que esses valores chegam a atingir 78% do conjunto de treino. Para além disso, e como consequência das entradas duplicadas, este conjunto de dados ainda sofria um balanceamento impróprio das classes, algo que o NSL-KDD veio resolver eliminando os registos duplicados. O resultado final é um conjunto de dados com 125.973 registos no seu conjunto de treino e 22.544 no seu conjunto de teste, composto por 41 atributos que representam um evento normal/benigno ou um ataque. [7], [55]

- **UNSW-NB15 -** Este conjunto de dados foi criado pelo *Cyber Range Lab* no *Australian Center for Cybersecurity* (ACCS) em 2015. O seu conteúdo híbrido foi gerado pela

ferramenta IXIA *PerfectStorm tool*, permitindo conciliar tráfego benigno gerado numa rede com ataques maliciosos gerados de forma automatizada. Foi também utilizada a ferramenta *tcpdump* para capturar ficheiros *pcap* com diferentes formas de ataque a vulnerabilidades como, *exploits*, *backdoors*, *DoS*, entre outros. De realçar que está disponível em formato *Packet* e formato *Flow*. [55], [57]

- **CIDDS-001** – Este é um dos conjuntos de dados mais utilizados em investigações relacionadas com deteção de intrusões baseadas em rede. Este conjunto de dados contém dados de fluxos de rede unidirecionais e foi gerado recorrendo a *scripts* da linguagem *Python* para simular comportamento humano em máquinas de uma rede virtual. Tem particularidades bastante realistas como, por exemplo, variações mediante o horário de expediente e respetivos ciclos de operação. Contém dados benignos, considerados normais, mas também eventos maliciosos como ataques de força bruta, negação de serviço, *ping scans*, e *port scans*. [58]

- **CICIDS 2017** – Este conjunto de dados está na origem do conjunto de dados utilizado neste trabalho. Publicado pela Universidade de *New Brunswick*, contém fluxos de dados catalogados como benignos, bem como malignos, com vários tipos de ataques, nomeadamente, força bruta, negação de serviço, Botnet, SSH e Heartbleed, entre outros. Na sua criação foram considerados diversos dispositivos de rede, desde *modems*, *firewalls*, *switches*, *routers* e também diferentes sistemas operativos (*Microsoft*, *Apple* e *Linux*). Contém 80 atributos relacionados com fluxos de rede capturados a partir do tráfego gerado. Os dados contemplam tráfego de rede bidirecional em formato *Packet* e *Flow* gerado durante 5 dias. [7], [38], [55], [58]

3.1.2. CSE-CIC-IDS2018

Um conjunto de dados pretende simular e demonstrar um comportamento ou uma situação real de um determinado cenário. Independentemente do cenário em questão, o conjunto de dados deve ser construído de forma a facilitar as previsões para os sistemas de aprendizagem computacional [22]. Nesta secção é analisado o conjunto de dados selecionado para o trabalho, o CSE-CIC-IDS2018, e explicados os aspetos considerados relevantes que levaram à escolha deste conjunto.

Descrição

O conjunto de dados CSE-CIC-IDS2018 é o resultado de uma colaboração entre o *Communications Security Establishment* (CSE) e o *Canadian Institute for Cybersecurity* (CIC). A simulação de ambientes que permitem o estudo de eventos anómalos em redes computacionais é algo bastante complexo, pois é necessário um conjunto de procedimentos bastante diversificados, configurações e validações que permitam replicar situações que possibilitem a deteção de ataques, também eles diversificados, com base nas suas caraterísticas. O principal objetivo deste trabalho foi criar um conjunto de dados que em tudo espelhasse os dados de tráfego obtidos no mundo real, ao nível dos dados considerados normais e da deteção de ocorrências de diferentes tipos de ataques. [13], [59], [60]

Origem

O conjunto de dados CSE-CIC_IDS2018 é considerado uma evolução da versão de 2017. Embora tenha uma estrutura semelhante a essa versão, com os mesmos 80 atributos, foi preparado para um volume muito maior de tráfego de rede contendo aproximadamente 16.000.000 instâncias divididas em 10 dias. [1]

Organização do conjunto de dados

Este conjunto de dados é composto por registos de tráfego de rede com mais de 80 atributos e padrões diferentes. Para o processo de extração, foi utilizado o software CICIFlowMeter-V3. Os dados estão organizados por dia e foram capturados em dez dias de trabalho entre quarta-feira, 14 de fevereiro de 2018 e sexta-feira, 2 de março de 2018. Em cada dia foram registados dados de tráfego de rede e registos de eventos de diferentes máquinas. [59], [61]

Este conjunto de dados inclui registos de diferentes tipos de intrusões visando diferentes tipos de aplicações, portos, e outros recursos da rede. Para simular os diferentes eventos que assolam um sistema em rede foram criados, dois tipos de perfil:

- **Benigno (*B-Profile*)**

  Representa todos os eventos diários expectáveis num ambiente deste tipo. A maioria do tráfego é HTTP e HTTPS. No entanto, neste tipo de eventos também estão simulados eventos relacionados com SMTP, POP3, IMAP, SSH e FTP. [59]

- **Malicioso (*M-Profile*)**

  Perfil que envolve cenários de ataque de diversos tipos. Desta forma, é possível recriar eventos que são comuns no dia a dia de uma rede. Para acrescer a esta aproximação da realidade, existem também variações visíveis ao nível do número de ocorrências de cada evento de uma dada ameaça [59]. Dentro deste perfil, de acordo com os autores, existem vários cenários de ataques dos quais se destacam:

  - Ataque infiltrado de origem interna: Neste cenário, são enviados ficheiros via email para as vítimas aproveitando vulnerabilidades da rede e dos sistemas. Após a vulnerabilidade ser explorada de forma bem-sucedida, são desenvolvidas atividades de monitorização e análise da rede internet à procura novas vulnerabilidades; [59], [62]
  - Negação de Serviço – HTTP: Para simular ataques de negação de serviço foram utilizados vários instrumentos diferentes, nomeadamente *Slowloris* e *SlowHTTPTest*. Estas ferramentas têm a capacidade de aproveitar vulnerabilidades para atacar páginas de internet com um número de pedidos excessivo, provocando assim a incapacidade destas para responder aos normais pedidos de quem com elas pretende comunicar; [59], [63]
  - Negação de Serviço com Origem Distribuída: Este tipo de ataque é semelhante ao anterior, mas com algumas camadas de complexidade. Neste caso, o atacante tenta invadir várias máquinas para desta forma poder proceder ao ataque da vítima de

várias origens diferentes. Para este trabalho, no conjunto de dados foram utilizadas ferramentas de DDOS como: *GoldenEye, Hulk*, *Low Orbital Cannon* (LOIC) e *High Orbital Cannon* (HOIC); [59], [63]

- Ataques a aplicações *Web*: Neste cenário são implementados diferentes ataques a aplicações *Web* que estejam a funcionar no sistema de rede. Numa fase inicial, são realizadas análises às páginas de internet na rede com ferramentas de pesquisas de vulnerabilidades próprias, produzindo de seguida os diferentes ataques em função das vulnerabilidades encontradas. Neste caso, entre outros, temos em análise o *SQL Injection*, forma de aquisição de informação confidencial de bases de dados através da injeção de comandos *SQL* diretamente através das interfaces das páginas de internet; [28], [59]
- Ataques de Força Bruta (*Brute-Force*): São técnicas bastante comuns nos ambientes de redes tentando aproveitar configurações de nomes de utilizador e palavra-chave que não sigam as regras de segurança mais conhecidas na conceção e manutenção de password. [59], [62]

Os ficheiros CSV resultantes referem-se a 10 dias, com ataques de diferentes tipos registados em cada dia. [59]. A tabela 1 apresenta a lista dos 10 ficheiros:

Tabela 1 – Ficheiros e ataques do conjunto de dados CSE-CIC_IDS2018

Ficheiro CSV	Ataques	Data	Hora de início	Hora de fim
Wednesday-14-02-2018_TrafficForML_CICFlowMeter	FTP – BruteForce	Wed-14-02-2018	10:32	12:09
	SSH – BruteForce		14:01	15:31
Thursday-15-02-2018_TrafficForML_CICFlowMeter	DoS-GoldenEye	Thurs-15-02-2018	09:26	10:09
	DoS-Slowloris		10:59	11:40
Friday-16-02-2018_TrafficForML_CICFlowMeter	DoS-SlowHTTPTest	Fri-16-02-2018	10:12	11:08
	DoS-Hulk		13:45	14:19
Thuesday-20-02-2018_TrafficForML_CICFlowMeter	DDoS attacks-LOIC-HTTP	Tues-20-02-2018	10:12	11:17
	DDoS-LOIC-UDP		13:13	13:32
Wednesday-21-02-2018_TrafficForML_CICFlowMeter	DDOS-LOIC-UDP	Wed-21-02-2018	10:09	10:43
	DDOS-HOIC		14:05	15:05
Thursday-22-02-2018_TrafficForML_CICFlowMeter	Brute Force -Web	Thurs-22-02-2018	10:17	11:24
	Brute Force -XSS		13:50	14:29
	SQL Injection		16:15	16:29
Friday-23-02-2018_TrafficForML_CICFlowMeter	Brute Force -Web	Fri-23-02-2018	10:03	11:03
	Brute Force -XSS		13:00	14:10
	SQL Injection		15:05	15:18
Wednesday-28-02-2018_TrafficForML_CICFlowMeter	Infiltration	Wed-28-02-2018	10:50	12:05
			13:42	14:40
Thursday-01-03-2018_TrafficForML_CICFlowMeter	Infiltration	Thurs-01-03-2018	9:57	10:55
			14:00	15:37
			14:00	15:37
Friday-02-03-2018_TrafficForML_CICFlowMeter	Bot	Fri-02-03-2018	10:11	11:34
			14:24	15:55

Atributos

Neste conjunto de dados estão presentes mais de 80 atributos que caracterizam os eventos que ocorrem numa rede. Para isso, foi utilizado o software *CICFlowMeter*, já referido anteriormente, que permite gerar fluxos de tráfego de rede. Este software, escrito em linguagem Java, permite gerar fluxos bidirecionais [59], [62], [64]. O output da aplicação, são ficheiros em formato CSV, divididos por dias, onde constam os atributos descritos na Tabela 1 do Anexo A. O ficheiro correspondente ao dia "*Tuesday-20-02-2018*" tem uma estrutura diferente dos restantes, possuindo 4 atributos extra, a saber: Flow ID, Src IP, Src Port, e Dst IP.

Ainda relativamente aos atributos inseridos neste conjunto de dados, segundo [65], estes podem ser divididos em 8 categorias, descritas na Tabela 2:

Tabela 2 - Categorias de atributos do conjunto de dados

Atributos	Descrição
1-4	Atributos simples de ligações de rede
5-16	Atributos relacionados com pacotes de rede
17-22	Atributos relacionados com fluxos de rede
23-45	Atributos relacionados com dados estatísticos de fluxos de rede
46-63	Atributos de tráfego de rede relacionados com conteúdo
64-67	Atributos relacionados com subfluxos de rede
68-79	Atributos de tráfego de uso geral
80-83	Atributos simples de ligações de rede

Obtenção do conjunto de dados

O conjunto de dados CSE-CIC-IDS2018 está disponível publicamente através do AWS - *Amazon Web Services* e, para obter acesso, é necessário instalar o AWS ILC (*Command Line Interface*), que é uma ferramenta unificada para gerir múltiplos serviços AWS. Dentro destes serviços, subsequentemente, é necessário interagir com o *Amazon S3* (*Amazon Simple Storage Service*), que é um serviço de armazenamento de conteúdos que pode ser utilizado em vários contextos, desde o armazenamento de aplicações, websites, até às cópias de segurança de ficheiros. [59]

3.2. Aprendizagem computacional aplicada à deteção de ataques

Segundo Delplace *et al*., em [66], os ataques informáticos tem vindo a crescer, tanto em frequência, como em sofisticação ao longo dos anos. Tal sofisticação e complexidade tem como consequência uma necessidade de mais avanços e inovação contínua nas estratégias de defesa. Isto significa que os métodos de intrusão e deteção, apesar de bastante utilizados e recomendados, já não são suficientes para responder ao tipo de ameaças que estes ataques representam.

Em [67], uma outra perspetiva, é referido que, tendo em conta toda a quantidade de diferentes ameaças, tecnologias e o aumento de procedimentos automatizados implementados pelos

atacantes, é cada vez mais urgente a utilização de métodos mais avançados. A construção manual de regras já não é de forma alguma exequível a esta escala, e assinaturas ou artefactos de eventos gerados automaticamente necessitam muitas vezes de contexto, o que faz com que sejam catalogados como falsos negativos quando são detetadas pequenas variações de assinaturas conhecidas.

Enquanto isso, a capacidade computacional aumenta e os seus custos diminuem o que faz com que a aprendizagem computacional seja vista cada vez mais como uma alternativa válida ou um mecanismo adicional para procedimentos de defesa contra os diferentes tipos de ataques maliciosos. É de registar que a aprendizagem computacional tem vindo a gerar bastante interesse e possui uma grande abrangência ao nível das aplicações e áreas de estudo e, neste caso, na cibersegurança. Com hardware mais acessível e capacidades de processamento mais acessíveis, os algoritmos de aprendizagem computacional podem ser utilizados para analisar, identificar e classificar diferentes tipos de ataque a partir de conjuntos de dados bastante volumosos. [66]

3.3. Pré-processamento e balanceamento dos conjuntos de dados

Como referido no capítulo anterior, uma das características em comum de muitos conjuntos de dados reais é o facto de serem desequilibrados. Isso acontece, por exemplo, em conjuntos de dados em que a classe objetivo diz respeito à ocorrência ou não de uma doença, transações com cartões de crédito, para deteção de fraude, e sistemas de deteção de intrusões, entre outros [68]. Para contornar este desequilíbrio natural, torna-se, portanto, bastante relevante analisar como foram tratados os conjuntos de dados por diferentes autores para encontrar um balanceamento nos seus conjuntos. Nesta análise, estão incluídas as técnicas de pré-processamento, as divisões de subconjuntos de dados e os procedimentos utilizados para balanceamento dos mesmos, quando utilizados. Reforçamos, no entanto, que uma parte significativa dos artigos não trata este problema [1].

3.3.1. CSE-CIC-IDS2018

Em [13], aos autores referem que, no âmbito das operações de pré-processamento, e para tentar tornar as previsões dos modelos criados mais objetivas, removeram atributos relativos a endereços IP e portas de *hosts*, embora tenham mantido portas de destino já que estas podem ajudar na identificação de determinados ataques. Foram ainda removidos exemplos e atributos com valores em falta, embora não existam referências quanto ao número. Os autores referem ainda que para a divisão dos subconjuntos de treino e de teste estabeleceram uma proporção estratificada de 20:80. Esta proporção de divisão levantou algumas questões quanto aos fatores que lhe deram origem, especialmente por não ser usual nem existir nenhuma justificação. Depois de alguma investigação, as parcelas reais são inconclusivas, ainda para mais quando no artigo [1], é descrito o trabalho efetuado em [13] referindo esta divisão como 80:20. Não existem referências quanto a técnicas de balanceamento empregues. No entanto, são detetadas discrepâncias nos resultados das taxas de deteção, que

no caso dos ataques *Web*, são abaixo da média. Uma possibilidade avançada pelos autores é que poderão faltar no conjunto de dados atributos que contribuam para uma melhor classificação deste tipo de ataques.

Já em [60], os autores procedem à remoção de atributos que têm pouco ou nenhum impacto na determinação do tipo de tráfego como, por exemplo, os registos temporais e os endereços IP. Neste caso, a divisão dos subconjuntos de dados foi feita em 3 partes, para o conjunto de treino, o conjunto de teste e o conjunto de validação. Esta divisão foi estabelecida na proporção de 90:9:1. Especificamente, os autores notaram um elevado número de ocorrências de tráfego benigno, o que poderia enviesar os resultados, e os fez aplicar uma técnica de *undersampling* simples, escolhendo 2 milhões de instâncias benignas de forma aleatória. Para além disso, foi aplicado o algoritmo SMOTE nas amostras de *WEB attack* e *Infiltration attack*. De acrescentar que o *oversampling* apenas foi aplicado no conjunto de dados de treino e que, depois de dividido, este subconjunto teve as suas instâncias misturadas.

Em [69], os autores, como tem sido comum no tratamento deste conjunto de dados, removeram os exemplos com atributos com valores inválidos (*Infinity*, *NaN*, ou valores em falta). Neste caso, converteram também os registos temporais (*timestamp*) para valores numéricos. É referido que neste processo foram removidas aproximadamente 20.000 instâncias. Neste artigo foram ainda utilizados dois tipos de divisões dos subconjuntos de dados de treino e de teste, 70:30 e 80:20. Não existem referências quanto a operações de balanceamento de dados.

No caso de [70], os autores removeram as instâncias consideradas inválidas, com valores em falta, ou sem significado relevante para a sua investigação. Pela informação registada, foram removidas 59 instâncias inválidas e 4 atributos irrelevantes, tendo ficado com 16,232,943 instâncias e 77 colunas de atributos. Foram também normalizados vários atributos considerados adequados para o efeito utilizando uma função Min-Max. Foi ainda aplicada uma técnica de *undersampling* nas instâncias de tráfego benigno reduzindo de 9.43 milhões para 1 milhão de instâncias de forma aleatória. O conjunto de dados foi dividido para o conjunto de treino e conjuntos de dados, respetivamente em 70:30.

Já em [71], em termos de pré-processamento, foram removidas as instâncias com dados em falta e com valores *Infinity*. A divisão dos subconjuntos de treino e validação foi de 80:20. Não existem menções quanto ao balanceamento de dados.

Por fim, em [72], os autores procedem à remoção do atributo *Protocol* que, segundo eles, é redundante devido ao atributo *Dst Port* (porta de destino) conter valores equivalentes de protocolo para cada porta de destino. O registo temporal também foi removido para não enviesar qualquer conclusão relacionada com a ocorrência dos ataques, ou seja, os algoritmos devem ser capazes de detetar ataques independentemente do momento em que ocorrem. Foram excluídos atributos com valor 0, valor negativo, *NaN* e *Infinity*. Os autores não utilizaram nenhuma técnica de balanceamento de dados mas referem a sua importância e reforçam que o assunto deverá merecer a sua atenção no futuro. Quanto à divisão dos conjuntos de dados, devido às especificidades das experiências, tais não aparecem referidas.

Nos próximos parágrafos descrevemos dois trabalhos onde foi estudado o efeito de métodos de *undersampling* e os resultados obtidos pelos algoritmos de aprendizagem.

O artigo [73] foca-se em estudar o efeito de diferentes rácios de *undersampling* no conjunto de dados CSE-CIC-IDS2018. No que diz respeito ao pré-processamento, foram removidos os atributos *Protocol* e *Timestamp*, bem como o atributo *Destination_port* por ter sido considerado fora do espetro do trabalho. Os autores removeram os quatro atributos extra do ficheiro "Thuesday-20-02-2018_TrafficForML_CICFlowMeter.csv". Foram também eliminadas instâncias com dados negativos nos atributos *Fwd_Header_Length*, *Flow_Duration*, e *Flow_IAT_Min*. Os autores removeram 8 atributos com o valor 0 em todas as instâncias, *Bwd_PSH_Flags*, *Bwd_URG_Flags*, *Fwd_Avg_Bytes_Bulk*, *Fwd_Avg_Packets_Bulk*, *Fwd_Avg_Bulk_Rate*, *Bwd_Avg_Bytes_Bulk*, *Bwd_Avg_Packets_Bulk*, *Bwd_Avg_Bulk_Rate*. Foram ainda removidos os atributos *Init_Win_bytes_forward* e *Init_Win_bytes_backward* por conterem demasiados valores negativos. Os autores referem que se optassem por remover as respetivas instâncias, isso iria provocar a remoção de uma grande parte do conjunto de dados. Da mesma forma, o atributo *Flow_Duration* também não foi utilizado por possuir valores demasiado baixos e muitas ocorrências com valor zero. Foram ainda removidas as instâncias onde o atributo *Flow Bytes/s* ou o *Flow Packets/s* continham valores não numéricos (*NaN*) ou infinitos (*Infinity*). Para esta análise foram apenas selecionadas classes de ataques *Web* compostas por *SQL Injection*, *Brute Force-Web* e *Brute Force-XSS*. Neste estudo foi utilizado o *Random UnderSampling* (RUS) que consiste em balancear as classes reduzindo a classe maioritária (benignos) aleatoriamente até à proporção necessária correspondente ao valor da classe minoritária (ataques *Web*). Foram utilizados os seguintes rácios de instâncias benignas e instâncias correspondentes a ataques Web: 999:1, 99:1, 95:5, 9:1, 3:1, 65:35, 1:1 e nenhum. Os autores mostram que os resultados são melhores para conjuntos de dados mais balanceados.

Em [65], são utilizados vários algoritmos clássicos de aprendizagem computacional e realizado um estudo sobre o efeito de técnicas de balanceamento como o *Random OverSampling*, *Random UnderSampling*, SMOTE e o *Difficult Set Sampling Technique* (DSSTE), proposto no artigo. Em termos de conjuntos de dados, foram utilizados o NSL-KDD e CSE-CIC-IDS2018. Os procedimentos de pré-processamento foram: remoção de amostras com valores duplicados mantendo apenas uma ocorrência; remoção de valores não numéricos (*NaN*) ou infinitos (*Infinity*); no caso do CSE-CIC-IDS2018, foram ainda removidos atributos como *Timestamp*, *Destination Address*, *Source Address*, *Source Port*; Para além disso, os atributos *Init Bwd Win Byts* e *Init Fwd Win Byts* viram as ocorrências do valor -1 substituídas por 1. A proporção de divisão entre treino e teste é de 80:20. Os autores concluem que os melhores resultados são obtidos com o algoritmo proposto, o DSSTE. Este método, segundo os autores, reduz o não balanceamento do conjunto de treino original e fornece indicações acerca de quais os atributos de classes minoritárias que necessitam de ser aprendidos. Isto permite aos algoritmos classificadores aprenderem as diferenças durante a fase de treino e melhorar o desempenho do modelo.

3.4. Seleção de atributos

Nesta secção, descrevem-se alguns trabalhos que utilizam técnicas de seleção de atributos com o conjunto de dados CSE-CIC-IDS2018.

Em [70] é utilizado o método de seleção de atributos *XGboost*, combinado com o algoritmo *Random Forest*, devido aos seus altos índices de acurácia e robustez. Esta técnica é implementada para mitigar o problema de não balanceamento dos atributos e extrair atributos chave para posterior análise.

Já em [71], são utilizados os métodos *Spearman's rank correlation coefficient* e o *Chisquare test*. Os melhores resultados foram obtidos com o *Spearman's rank correlation coefficient* que conseguiu extrair 23 dos 80 atributos originais do conjunto de dados.

Em [72] é estudado o impacto da seleção de atributos no desempenho de alguns algoritmos de aprendizagem e também o impacto de um determinado atributo. Foi utilizado um método de seleção de atributos baseado em ensembles. Mais concretamente, foram utilizados os métodos de seleção de atributos *filter based Information Gain, Gain Ratio, Chi-Squared* e os métodos *wrapper based Random Forest, CatBoost, XGBoost e LightGBM*). Num primeiro momento, estes métodos foram utilizados em separado, produzindo cada um uma lista de atributos ordenada por ordem descendente de preferência. Para cada uma dessas listas foram considerados apenas os 20 primeiros atributos. Posteriormente, foram criados 5 grupos constituídos por atributos que constassem, respetivamente, de 4, 5, 6 ou 7 listas, para além do grupo com o conjunto original de atributos. Os autores concluem que os melhores resultados são obtidos quando se utiliza o grupo que contem 4 listas de atributos. Este grupo é o que obtém uma performance similar ou melhor ao grupo com todos os atributos.

3.5. Classificação e tratamento dos tipos de ataques

Para além do pré-processamento, das técnicas de balanceamento e da seleção de atributos, é relevante perceber qual o tipo de classificação e tratamento dado aos diferentes ataques presentes no conjunto de dados CSE-CIC-IDS2018, uma vez que este varia entre diferentes trabalhos.

Em [60], apesar de várias aplicações de técnicas de balanceamento, foi mantida a estrutura original dos ataques. Ou seja, o modelo foi concebido para ser capaz de distinguir entre os diferentes tipos de ataque presentes no conjunto de dados, bem como tráfego normal, o que significa que é realizada classificação multiclasse. O mesmo acontece em [13]. No entanto, por trabalharem com redes neuronais, os autores decidiram transformar o atributo que identifica os ataques que, no caso do CSE-CIC-IDS2018, é o atributo *Label*. Mais concretamente, os valores deste atributo, as *labels*, foram transformados em valores numéricos correspondentes a cada um dos ataques possíveis e ao tráfego normal.

No artigos [67] e [78], os autores transformam os 14 ataques originais, que dão origem ao tráfego malicioso, mais o tráfego normal, designado por benigno, em 7 tipos de ocorrências, nomeadamente: Benigno; *BruteForce*; DoS; Bot; DDoS; *Web Attacks* e *Infiltration*. O

mesmo acontece em [70] mas, neste caso, as *labels* foram primeiro convertidas para inteiros de 0 a 6, correspondentes aos diferentes tipos de ataque (mais o tráfego benigno) e, posteriormente, convertidas para o formato binário *one-hot encoding*.

Em [75], o atributo *Label* foi alterado de modo a só se poder distinguir entre tráfego benigno e malicioso, transformando o problema num problema de classificação binária. Para o conjunto de dados CSE-CIC-IDS2018, mas também para outros, foi aplicada a classificação binária entre tráfego benigno e malicioso e, num segundo teste, foram utilizadas as *labels* originais, ou seja, foi aplicada classificação multiclasse.

A classificação binária é também aplicada em [76], embora sejam utilizados apenas alguns ataques do conjunto de dados CSE-CIC-IDS2018. Já em [77] é também referido que as duas classes (benigno e malicioso) foram balanceadas, não sendo explicitada a técnica utilizada.

Por fim, em [78] os autores apresentaram uma variação da abordagem de classificação binária. Mais concretamente, os autores criaram um conjunto de cenários de classificação binária, em que o modelo é treinado com tráfego benigno e tráfego correspondente a um determinado ataque num dia específico, sendo o teste realizado com outro ataque. Por exemplo, num dos cenários foram utilizados os dados do dia 16/02/2018, onde constam dois tipos de ataque, para além de tráfego benigno; Um dos ataques foi utilizado para treino e o outro para teste. Esta é uma abordagem interessante porque estuda o desempenho dos modelos quando confrontados com um ataque diferente daquele com que foram treinados. Para além destes cenários, foram também testados outros em que o modelo é testado com o mesmo ataque com que foi treinado, mas com instâncias que ocorreram noutro dia.

3.6. Algoritmos de Aprendizagem

É de salientar que grande parte dos trabalhos pesquisados, concentram-se essencialmente em avaliar o desempenho de diferentes algoritmos de aprendizagem, sendo que os procedimentos de preparação dos dados são raramente o objeto de estudo.

Nesta secção descrevemos os algoritmos de aprendizagem utilizados em alguns trabalhos que utilizaram o conjunto de dados CSE-CIC-IDS-2018.

Em [68], onde é realizado um estudo comparativo entre os conjuntos de dados atuais. Nele encontram-se referidos alguns dos principais conjuntos de dados, em particular, o CSE-CIC IDS2018. O objetivo do referido estudo foi estabelecer uma comparação entre alguns dos conjuntos de dados de IDS mais utilizados, tratando-os através de processos de normalização Max-Min na fase de pré-processamento e usando, para classificação, os algoritmos Support Vector Machines (SVN), KNN e Árvores de Decisão. Neste estudo, os autores concluem que aplicando os algoritmos a alguns dos principais conjuntos de dados relacionados com IDS (CSE-CIC IDS-2018, UNSW-NB15, ISCX-2012, NSL-KDD e CIDDS-001), são obtidos resultados muito similares entre eles, embora seja dado especial destaque aos resultados obtidos pelas árvores de decisão, cujas taxas de sucesso variam entre 99% e 100% nos conjuntos de dados CSE-CIC IDS-2018, ISCX-2012, NSL-KDD e CIDDS-001.

Em [78] é utilizado um conjunto de algoritmos, nomeadamente CLONALG *Artificial Immune System*, *Learning Vector Quantization* e *Back-Propagation Multi-Layer Perceptron*. Segundo os autores, os resultados demonstram a adequação da utilização do conjunto de dados para testar algoritmos de deteção de intrusão na rede baseados no comportamento e a eficiência do algoritmo *Back-Propagation Multi-Layer Perceptron* para detetar os ataques *Zero-day* em comparação com o CLONALG *Artificial Immune System* e *Learning Vector Quantization*.

[73] também aborda a questão dos dados não balanceados, mas neste caso foca-se em estudar diferentes rácios de *undersampling* no conjunto de dados CSE-CIC-IDS2018. Para além disso, ainda aplicam no seu estudo 7 algoritmos de classificação diferentes: Árvores de Decisão, *Random Forest, CatBoost, LightGBM, XGBoost, Naïve Bayes* e Regressão Logística. Os autores concluem que os resultados obtidos com diferentes algoritmos de classificação são estatisticamente diferentes e que a interação entre os diferentes rácios de *undersampling* e os algoritmos de classificação também é significativa na obtenção de resultados.

Em [13] são utilizadas redes neuronais profundas (*deep neural networks*) num modelo de composição modular para tentar diminuir a taxa de ocorrência de falsos positivos nos sistemas de deteção de intrusões. As experiências são utilizadas no conjunto de dados CSE-CIC-IDS-2018 e os modelos finais podem ser utilizados em sistemas de deteção de intrusões para gerar alertas e prevenir novos ataques. Os autores reportam que os resultados experimentais mostram melhorias na deteção de alguns tipos de ataques em valores próximos dos 100% quando comparados com outros trabalhos relacionados com o tema. Já em [60], também são utilizadas redes neuronais *Long Short Term Memory* adicionando um mecanismo designado por *Attention Mechanism*, que tenta focar em diferentes aspetos relevantes para poder sumarizar todo o conteúdo importante. Os resultados experimentais atingem valores na ordem dos 96% o que, segundo os autores, é superior a outros algoritmos de aprendizagem computacional. Ainda relacionado com as redes neuronais profundas, [79] propõe a utilização da técnica *Binary Particle Swarm Optimization* para solucionar os problemas de seleção de atributos. Os resultados são avaliados com classificadores *Support Vector Machines, K - Nearest Neighbour*, Árvores de Decisão e Regressão Logística. Os autores referem que os resultados demonstram uma acurácia alta de 95%.

Por sua vez, em [71] é comparado o desempenho de 7 algoritmos de classificação para identificar quais os mais apropriados para uma solução de *Ensemble Learning*. Para esta experiência foram utilizados diferentes algoritmos, nomeadamente, *Random Forest, Gaussian Naive Bayes*, Árvores de Decisão, *Quadratic Discriminant Analysis Gradient Boosting* e Regressão Logística, aplicados ao conjunto de dados CSE-CIC-IDS-201. Depois dos testes efetuados emergiram como opções viáveis o *Gradient Boosting*, a Regressão Logística e as Árvores de Decisão, com uma acurácia e precisão de 98,8%.

Esta página foi intencionalmente deixada em branco

4. Metodologia e ferramentas utilizadas

Neste capítulo descreve-se a metodologia utilizada na realização do trabalho bem como as ferramentas utilizadas.

4.1. Metodologia

Enquadrando-se na área da aprendizagem computacional, este trabalho debruça-se especificamente em documentar e analisar os diferentes aspetos das fases de pré-processamento, balanceamento e seleção de atributos de um conjunto de dados, bem como a análise dos resultados das experiências realizadas.

A metodologia empregue segue de forma próxima as diferentes fases de um projeto na área da aprendizagem computacional (ver Figura 1). Assim, para atingir os objetivos propostos, o trabalho foi organizado nas seguintes fases:

- Seleção e limpeza do conjunto de dados – Nesta fase foram efetuados procedimentos iniciais para o trabalho, como a seleção do conjunto de dados e a sua limpeza;
- Construção de conjuntos de dados – Nesta fase foram construídos diversos conjuntos de treino e um conjunto de teste. Cada conjunto de treino representa um cenário que se pretende testar, nomeadamente, *undersampling* e *oversampling*. Quanto ao conjunto de teste, este foi o mesmo em todos os testes de modo a que os resultados pudessem ser comparados;
- Execução de testes e análise de resultados – Fase onde foram realizados os testes e analisados os resultados.

Os próximos capítulos descrevem o trabalho realizado em cada uma destas fases.

4.2. Ferramentas utilizadas

Para este trabalho foram utilizadas diferentes ferramentas, principalmente para tratar os ficheiros do conjunto de dados original e elaborar operações de tratamentos de dados, que são descritas nas secções seguintes:

4.2.1. *Bash Scripting*

Na fase inicial deste trabalho foram criados *scripts* para proceder à automatização de processos, como a edição dos conjuntos de dados de treino nos diferentes cenários de cada experiência.

```
#Preparations
rm testset.csv
cp header.csv testset.csv

#START

echo "Starting 30% Test File Benign Building..."
sed '1,115406!d' '1 Benign.csv' >> testset.csv
sed '1,165126!d' '2 Benign.csv' >> testset.csv
sed '1,89329!d' '3 Benign.csv' >> testset.csv
sed '1,72163!d' '5 Benign.csv' >> testset.csv
sed '1,180242!d' '6 Benign.csv' >> testset.csv
sed '1,180928!d' '7 Benign.csv' >> testset.csv
sed '1,90459!d' '8 Benign.csv' >> testset.csv
sed '1,41099!d' '9 Benign.csv' >> testset.csv
```

Figura 4 - Exemplo de script para criação de 30% das ocorrências benignas, retiradas em cada dia

Estes *scripts* foram criados usando a *Bash Shell*, e tinham como principal função definir as áreas de edição entre as instâncias do conjunto de dados original. No caso da construção de conjuntos de treino com *oversampling*, os scripts foram utilizados para obter uma distribuição mais homogénea das instâncias selecionadas. No caso da construção de conjuntos de treino com *undersampling*, foram utilizados para aumentar a rapidez e eficiência na eliminação de instâncias.

4.2.2. WEKA *Workbench*

O WEKA *Workbench* é descrito como uma coleção de algoritmos de aprendizagem computacional e ferramentas de análise e pré-processamento de dados. O seu nome significa "*Waikato Environment for Knowledge Analysis*" e foi desenvolvido pela universidade de *Waikato* na Nova Zelândia. O sistema foi desenvolvido em Java e distribuído de forma gratuita sobre os termos das licenças GNU. Funciona nos principais sistemas conhecidos, nomeadamente Linux, Windows e Macintosh.

Neste trabalho foi utilizada a versão 3.8.5, conforme se pode verificar na imagem anterior. Um dos maiores fatores da popularidade é a interface gráfica da ferramenta, que simplifica e automatiza grande parte das tarefas a realizar, permitindo o acesso rápido a uma grande diversidade de ferramentas relacionadas com mineração de dados e aprendizagem computacional. Permite a utilização de ferramentas de pré-processamento dos dados, algoritmos de classificação, regressão, *clustering* e seleção de atributos, entre outras ferramentas.

O Weka tem um tipo de ficheiro nativo, o ARFF, mas suporta outros tipos, nomeadamente CSV e *LibSVM*.

- **Funcionalidades utilizadas**

No menu inicial (Figura 5) constam várias aplicações, das quais se destaca o *Explorer* (Figura 6) ferramenta utilizada em grande parte deste trabalho.

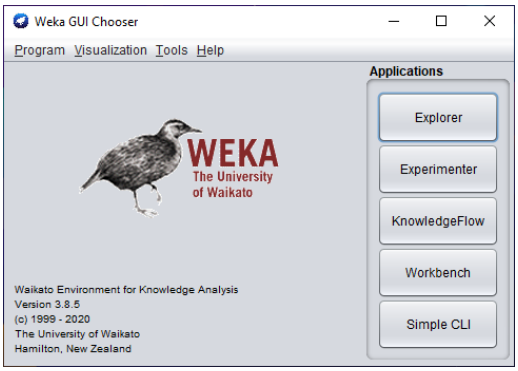

Figura 5 - Interface gráfico da ferramenta WEKA – Menu Inicial

Esta interface possui vários separadores, alguns dos quais podem ser adicionados durante a instalação ou posteriormente sob a forma de *plugin*. Ao abrir o Weka *Explorer*, é apresentado por omissão o painel "*Preprocess*" (Figura 6), que permite realizar várias operações de limpeza e pré-processamento do conjunto de dados. Logo à partida, é possível selecionar um atributo e visualizar algumas estatísticas a ele relacionadas.

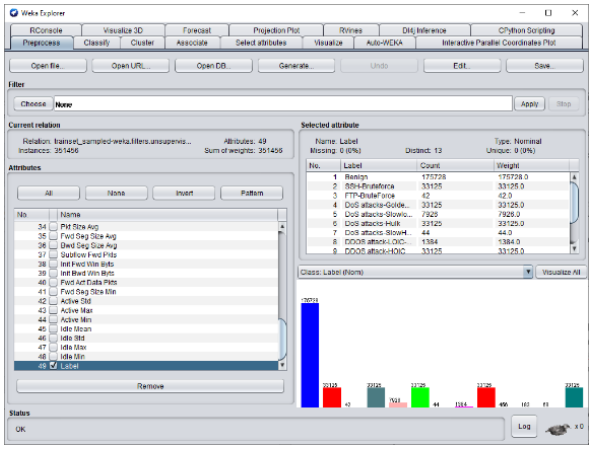

Figura 6 - Interface gráfico Weka Explorer no painel Preprocess

É possível selecionar um subconjunto de atributos a remover e, ainda, usando a opção *Edit* (Figura 7), é possível corrigir erros detetados como, por exemplo, atributos com valores inválidos.

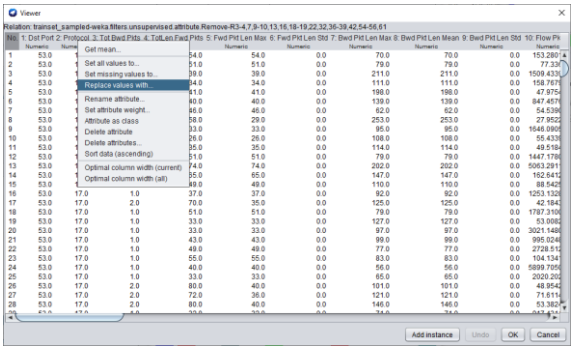

Figura 7 - Weka Explorer - Editor de atributos

Para utilizar algoritmos de classificação e regressão, é necessário escolher o separador *Classify* (Figura 8).

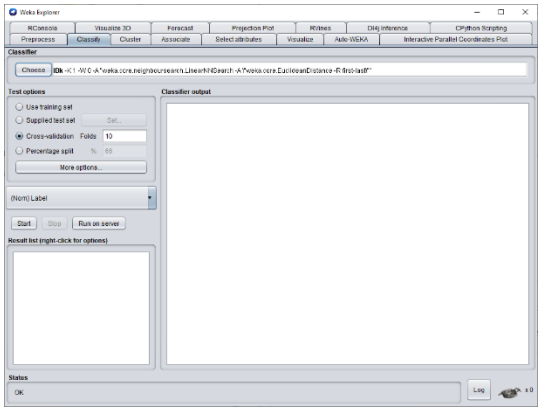

Figura 8 - Interface Weka - Separador *Classify*.

Neste separador, através do botão "*Choose*", é possível escolher a lista de algoritmos classificadores, os quais estão agrupados em várias categorias (Figura 9).

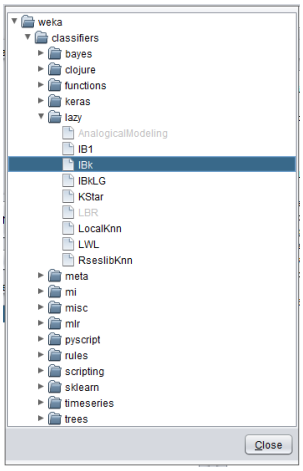

Figura 9 - Lista de algoritmos de classificação e regressão

Por omissão, nas opções de teste está selecionada a opção *Cross-Validation* com 10 *folds*, mas para este trabalho foi sempre utilizado um conjunto de dados de teste preparado para o efeito. Quanto ao conjunto de dados de treino, esse já deverá ter sido adicionado no separador de pré-processamento, como foi já referido anteriormente.

Assim, depois dos conjuntos de dados adicionados, e algoritmos escolhidos, basta premir o botão *Start (Figura 10)* e aguardar pelo resultado.

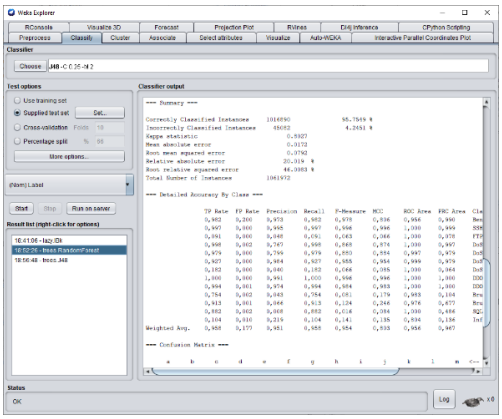

Figura 10 - Interface Weka - Exemplo de teste com vários algoritmos

Dependendo do modo ou do algoritmo escolhido, o separador também fornece mecanismos de interpretação visual dos modelos, árvores de decisão (Figura 11), entre outros.

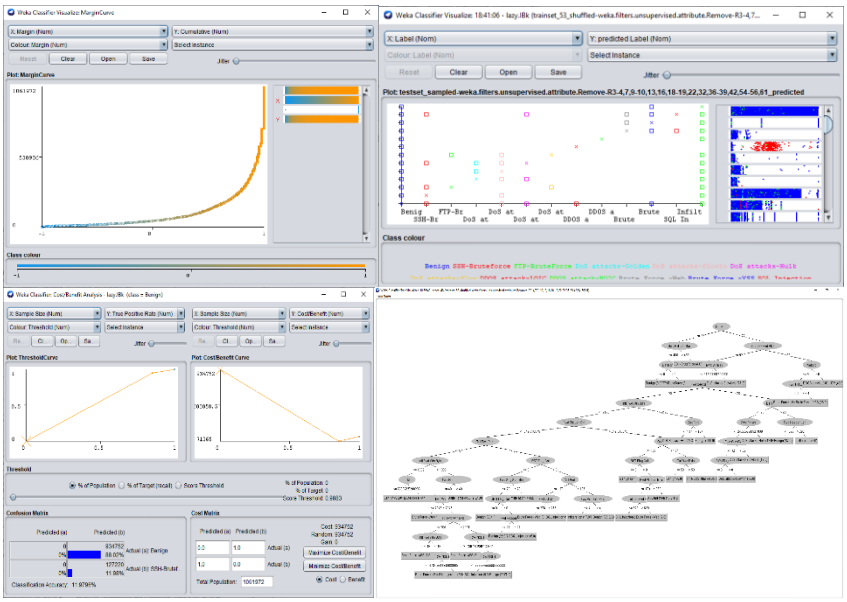

Figura 11 - Interface Weka - Diferentes tipos de interpretação visual dos modelos

Por fim, outra das funcionalidades utilizadas, e com bastante utilidade, foi a seleção de atributos. Neste separador é possível efetuar configurações, como escolher o avaliador de atributos (no caso da Figura 12, *CfsSubsetEval*) e o método de pesquisa (no caso mesma figura, *GreedyStepwise*). É ainda possível escolher entre usar o conjunto de dados de treino completo ou através de *cross-validation*.

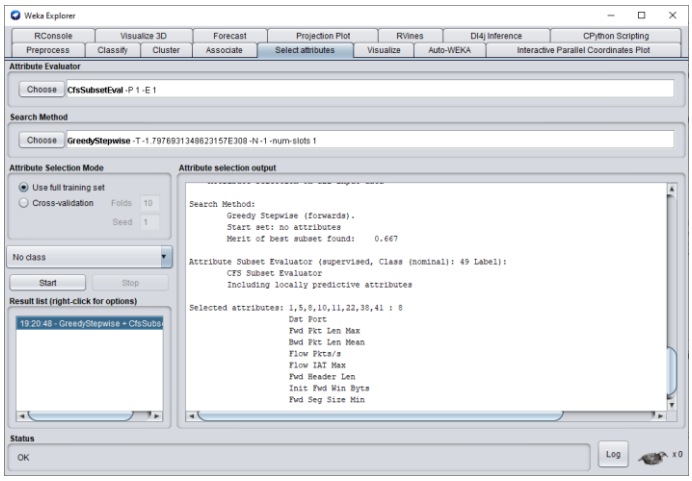

Figura 12 - Interface Weka - Seleção de atributos

Independentemente do modo escolhido, é necessário assegurar os devidos requisitos técnicos para o sistema funcionar, nomeadamente a *Java Virtual Machine* e possuir capacidade de processamento e memória suficiente para obter resultados satisfatórios e em tempo útil. [27], [44], [46]

4.2.3. WEKA API

O software WEKA, disponibiliza uma API que permite obter acesso às suas funcionalidades através de programação (Figura 13). Esta funcionalidade é uma vantagem que permitiu suprimir uma limitação do software original, que foi a impossibilidade de repetir os mesmos testes várias vezes. Desta forma, usando o WEKA API com a linguagem *Java*, foi criado um programa que permitia repetir cada experiência várias vezes com sementes (*seeds*) diferentes de modo a obter resultados com significado estatístico.

```
package ids;
// 2/9/21
import weka.classifiers.Evaluation;
import weka.classifiers.functions.MultilayerPerceptron;
import weka.classifiers.trees.RandomForest;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;
```

Figura 13 - Exemplo de utilização das bibliotecas da Weka Api

5. Seleção e limpeza do conjunto de dados

Neste capítulo são descritos aspetos relacionados com a seleção do conjunto de dados bem como as operações de limpeza dos dados, indispensáveis para que estes possam ser utilizados posteriormente na criação de modelos utilizando algoritmos de aprendizagem computacional.

5.1. Seleção do conjunto de dados

Como já foi referido, para este trabalho foi selecionado o conjunto de dados CSE-CIC-IDS-2018, o qual possui um volume e diversidade de instâncias suficiente para proceder aos diferentes testes e análises. Para além destes aspetos, este conjunto de dados está relacionado com a área da cibersegurança, característica que era um requisito para este trabalho.

A preparação do conjunto obedeceu a vários aspetos e a diferentes etapas. Alguns destes aspetos estavam relacionados com o aumento de eficiência dos algoritmos de classificação.

Os 10 ficheiros CSV do CSE-CIC-IDS-2018 perfazem aproximadamente 6,9 gigabytes de tráfego. O tamanho dos ficheiros não está distribuído de forma uniforme. Por exemplo, o ficheiro correspondente ao dia "Tuesday-20-02-2018" tem quase 4 gigabytes, ou seja, mais de metade do tamanho total do conjunto de dados. Os restantes ficheiros possuem tamanhos que oscilam entre os 100 e os 400 megabytes. Devido ao seu tamanho e ao facto de ter mais atributos que os outros ficheiros, o ficheiro correspondente ao dia "Tuesday-20-02-2018" não foi utilizado. De modo a ser possível a realização dos testes em tempo útil, o ficheiro correspondente ao dia "Friday-02-03-2018" também não foi utilizado. Este ficheiro foi escolhido pelo facto de as instâncias maliciosas corresponderem a apenas um tipo de ataque, semelhante em número de instâncias a outro dos ataques existentes no conjunto de dados.

É importante mencionar alguns fatores que foram tidos em conta nas decisões tomadas nesta fase, nomeadamente:

- Tamanho do conjunto de dados – face ao hardware disponível, os ficheiros que compunham o conjunto de dados original, provaram ser volumosos, pelo que se tornou necessário ajustar o tamanho do conjunto de dados de teste para obter resultados em tempo útil;
- Tempo de processamento dos modelos – Com um conjunto de dados de treino maior, o tempo de processamento mostrou-se algo demorado para ser possível fazer as experiências pretendidas em tempo considerado aceitável.
- Recursos de hardware disponíveis – Várias soluções de hardware foram testadas. Tendo em consideração os recursos limitados, foram encontradas soluções que tornaram toda a estrutura viável.

5.2. Limpeza do conjunto de dados original

O processo de limpeza do conjunto de dados é bastante importante visto que é muito comum os conjuntos de dados conterem anomalias. Entre outras anomalias, pode haver valores em falta, valores negativos quando não é expetável, ou existirem atributos com valores inadequados para serem utilizados com alguns algoritmos de aprendizagem, em particular, valores nominais quando se pretende utilizar uma rede neuronal, por exemplo.

Assim, foram efetuados os seguintes procedimentos de limpeza ao conjunto de dados original:

1. **Eliminação do atributo *Timestamp* (Referência Temporal)** – Estruturalmente, todos os ficheiros representam dias, e todas as instâncias desses ficheiros têm o atributo *Timestamp* com o valor da referência temporal da ocorrência desse evento. Cada evento é caraterizado como benigno ou um determinado tipo de ataque malicioso. Uma vez que estamos perante um problema de classificação, estes dados de cariz temporal não eram necessários e poderiam mesmo dificultar a obtenção de bons resultados, pois os valores deste atributo são diferentes para todas as instâncias;

2. **Substituição de valores negativos** – O conjunto de dados incluía instâncias com valores negativos que não são característicos dos atributos que os continham. Estes atributos estão relacionados com fluxos de dados (*Flow Duration*, *Init Fwd Win Byts* e *Init Bwd Win Byts*) e apresentam no seu estado original o valor -1 em algumas instâncias, o que não é um valor aceitável. Estes valores são explicados no documento "*Malmenator - Network Anomaly Detection*" [80] como sendo resultado de um problema não especificado do software CICFlowmeter no momento em que gerou estes dados. Assim, seguindo o exemplo do mesmo documento, foram substituídas todas as ocorrências de -1 por 0;

3. **Eliminação de instâncias com valores *Infinite* e *NaN***: Nos atributos *Flow Byts/s* e *Flow Pkts/s* foram detetadas ocorrências de valores *Infinite* e *NaN*, que foram removidas;

4. **Eliminação de atributos com o mesmo valor em todas as instâncias**: O conjunto de dados continha também vários atributos com o mesmo valor para todas as instâncias que foram eliminados. Mais concretamente, todas as instâncias tinham o valor 0 para os atributos *Bwd PSH Flags*, *Fwd URG Flags*, *Bwd URG Flags*, *CWE Flag Count*, *Fwd Byts/b Avg*, *Fwd Pkts/b Avg*, *Fwd Blk Rate Avg*, *Bwd Byts/b Avg*, *Bwd Pkts/b Avg* e *Bwd Blk Rate Avg*. No caso do atributo *Protocol*, após a operação 3, todas as instâncias remanescentes tinham o valor 6;

5. **Eliminação de instâncias duplicadas:** Por fim, as instâncias que sobraram de todos estes procedimentos foram analisadas e foram encontradas ocorrências duplicadas em mais do que um caso, tendo sido então removidas.

Depois de todos estes procedimentos, o conjunto de dados ficou então preparado para ser trabalhado e testado.

Em forma de balanço, é de registar que como resultado das operações de pré-processamento, restaram 69 dos 83 atributos originais. O tamanho original do conjunto de dados é de 16.232.943 instâncias, dos quais 7.948.738 instâncias pertenciam ao ficheiro "Tuesday-20-02-2018.csv". Ou seja, não contando com as instâncias deste ficheiro, que não foram utilizadas, ficamos com 8.284.205 instâncias. Aplicando as operações de pré-processamento restam 5.392.393 instâncias. Destas, 4.673.766 (86,7%) correspondem a tráfego benigno e 636 096 (13,6%) correspondem a tráfego malicioso.

Esta página foi intencionalmente deixada em branco

6. Construção de conjuntos de dados

Neste capítulo são apresentados todos os passos efetuados para a construção dos conjuntos de dados, com as duas técnicas de balanceamento implementadas.

6.1. Seleção de dados para treino e teste

O conjunto de dados resultante das operações de limpeza é composto por 69 atributos, em que o atributo correspondente à classe objetivo é o atributo *Label*, que contém a descrição dos diferentes ataques. Assim, o conjunto de dados utilizado é composto pelos 12 ataques já identificados anteriormente, mais a identificação de fluxo benigno, ou normal. Na tabela 3 estão as ocorrências referentes a cada tipo de *label*, por dia, representando 100% do conjunto de dados.

Tabela 3 - Número de ocorrências dos diferentes ataques no conjunto de dados estudado

Dia	Instâncias	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOIC-UDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	671133	577032	94101	94048	53										
2	876946	825632	51314			41406	9908								
3	591901	446647	145254					145199	55						
5	561405	360814	200591							1 730	198861				
6	901551	901210	341									228	79	34	
7	905185	904642	543									342	150	51	
8	513714	452293	61421												61421
9	370558	205496	82531												82531
Totais	5392393	4673766	636 096	94048	53	41406	9908	145199	55	1730	198861	570	229	85	143952

Analisando a tabela, nota-se a discrepância do número de ocorrências dos diferentes tipos de ataques. O número de ocorrências mais alto detetado é de 198861 ocorrências, do ataque de negação de serviço distribuído HOIC, enquanto o menor número é de apenas 53 ocorrências detetadas, do ataque de força bruta via FTP. Existe também uma discrepância evidente entre o número de instâncias correspondentes a tráfego benigno e tráfego malicioso.

Com base nas condicionantes referidas anteriormente, e na distribuição de ocorrências por ataque patente na tabela anterior, foram inicialmente definidos dois conjuntos de dados: o conjunto de dados para treino com 80% do conjunto de dados original e o conjunto de dados de teste com os 20% restantes.

Na Tabela 4 é apresentado o conjunto de dados para treino a ser utilizado que serviu de base para a construção dos diferentes conjuntos de treino a serem utilizados nas experiências com o objetivo de estudar o efeito dos métodos de *undersampling* e *oversampling*.

Tabela 4 - Conjunto de dados para treino

Dia	Instâncias	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	536906	461626	75280	75238	42										
2	701557	660506	41051			33125	7926								
3	473521	357318	116203					116159	44						
5	449124	288651	160473							1384	159089				
6	721241	720968	272									182	63	27	
7	724148	723714	435									274	120	41	
8	410971	361834	49137												49137
9	296446	164397	66025												66025
	4313914	3739013	508876	75238	42	33125	7926	116159	44	1384	159089	456	183	68	115162

Quanto ao conjunto de teste, este encontra-se representado na Tabela 5. Este conjunto de dados é bastante importante porque foi utilizado de forma inalterada em todas as experiências efetuadas, ou seja, permanece inalterado durante todo o trabalho. Este conjunto de teste tem a particularidade de manter a proporção original do conjunto de dados, permitindo testar os algoritmos num cenário próximo da realidade.

Tabela 5 - Conjunto de dados de teste

Dia	Instâncias	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	134 227	115 406	18 821	18 810	11										
2	175 389	165 126	10 263			8 281	1 982								
3	118 380	89 329	29 051					29 040	11						
5	112 281	72 163	40 118							346	39 772				
6	180 310	180 242	69									46	16	7	
7	181 037	180 928	108									68	30	10	
8	102 743	90 459	12 284												12 284
9	74 112	41 099	16 506												16 506
	1078479	934 753	127 220	18 810	11	8 281	1 982	29 040	11	346	39 772	114	46	17	28 790

6.2. Estratégia geral para a construção de conjuntos de dados de treino para realização de experiências

Neste trabalho, todas as experiências incidiram sobre o conjunto de treino, visto que o conjunto de teste foi comum a todas as experiências e serviu para comparação de resultados.

De modo a analisar os efeitos das operações de balanceamento e o impacto que a seleção de atributos tem nessas operações, foi definida a estratégia para construção de conjuntos de dados de treino para as diferentes experiências a partir dos dados selecionados para treino (ver secção anterior). Estas experiências foram divididas em dois cenários principais: *Sem seleção de atributos* e *Com seleção de atributos* (Figura 14). Foram definidos estes dois cenários de forma a poder perceber-se se as conclusões sobre o efeito das operações de balanceamento se alteram quando se procede à seleção de atributos, ou seja, o quão robustas são as conclusões. Para cada um destes cenários foram realizadas experiências utilizando os métodos de balanceamento *Random UnderSampling* e *Random OverSampling*. Para cada um destes métodos, foram construídos dois tipos de conjuntos de treino. No primeiro tipo, que denominámos por B=TotalA, os conjuntos de treino foram construídos de modo a que o número total de instâncias de tráfego benigno correspondesse ao número total de instâncias correspondente a tráfego malicioso, sendo os diferentes níveis de balanceamento realizados apenas entre os ataques. No segundo tipo, que denominámos por B=ARef, o número de

instâncias correspondentes a tráfego benigno foi equiparado ao número de instâncias do ataque de referência, ou seja, a classe que é usada para determinar o número de instâncias das outras classes. Note-se que para cada um destes tipos foram criados diferentes conjuntos de treino com diferentes níveis de balanceamento para se poder estudar o efeito desta operação. É também de referir que, em qualquer dos tipos de conjuntos de treino, foram utilizadas as *labels* do conjunto de dados original (com todos os ataques), o que significa que se aplicou classificação multiclasse.

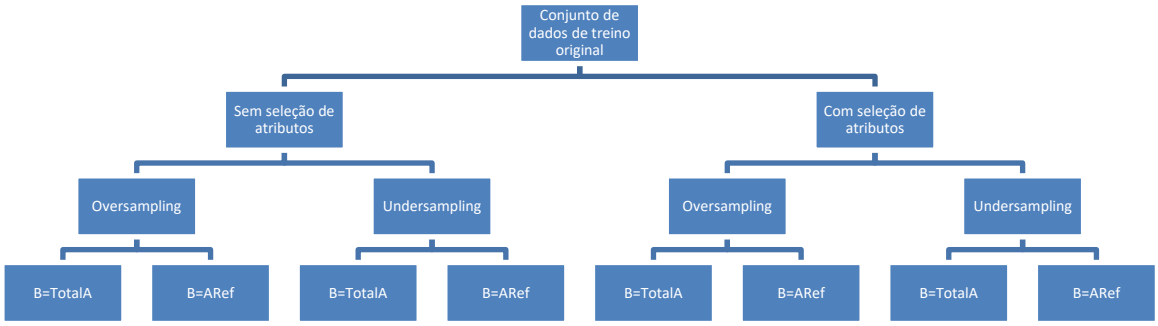

Figura 14 - Esquema da planificação de experiências efetuadas

O procedimento geral para construção dos conjuntos de treino para cada experiência proposta consistiu em fazer variar os valores das ocorrências de todos os ataques num espectro entre o ataque com menos ocorrências (FTP – *Bruteforce*, com 42 ocorrências) e o ataque com mais ocorrências. No entanto, por limitações computacionais, o ataque com mais instâncias, utilizado como ataque de referência foi o DOS *GoldenEye*, com 33125 ocorrências.

Assim, para cada tipo de conjunto de dados foram realizadas 8 experiências com níveis de balanceamento diferentes, mais concretamente, considerando o número de instâncias disponíveis para cada ataque (Tabela 6).

Tabela 6 – Número de instâncias por ataque

Nº de Instâncias	Ataque / Classe
42	FTP – *Bruteforce*
44	DoS *SlowHTTPTest*
68	Sql *Injection*
183	Brute *Force-XSS*
456	Brute *Force-Web*
1384	DDoS *attack*-LOIC-UDP
7926	DoS *Slowloris*
33125	DOS *GoldenEye*

Nas próximas secções é descrita a metodologia de criação dos conjuntos de treino usando *Random UnderSampling* e *Random OverSampling*. Note-se que em ambos os casos se utilizaram os princípios descritos nesta secção.

6.3. Random UnderSampling

No caso das experiências com *Random OverSampling*, o procedimento consiste em reduzir o número de instâncias dos ataques com um maior número de instâncias do que as do ataque de referência para um valor igual ao número de instâncias desse ataque. No caso dos ataques com um número inferior de instâncias relativamente ao ataque de referência, o número de instâncias mantém-se inalterado.

Tomemos como exemplo o caso em que o ataque de referência é o DDoS LOICUDP, ilustrado na Tabela 7. Usando a Tabela 6 como referência, neste exemplo, é possível verificar que esta experiência corresponde ao *undersampling* de todos os ataques com mais instâncias que o ataque DDOS LOICUDP, para um valor igual ao dessa classe. Repare-se que a distribuição de instâncias ilustrada na Tabela 7 corresponde a um conjunto de dados do tipo B=TotalA, uma vez que o número de instâncias de tráfego benigno é igual ao total de instâncias de tráfego malicioso. A Tabela 8 ilustra o mesmo exemplo para um conjunto de dados do tipo B=ARef, em que o número de instâncias de tráfego benigno é igual ao número de instâncias do ataque de referência.

Tabela 7 - Distribuição do número de instâncias para um conjunto de treino B=TotalA com *undersampling*.

Dia	Instâncias	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	2 852	1 426	1 426	1 384	42										
2	5 536	2 768	2 768			1 384	1 384								
3	2 856	1 428	1 428					1 384	44						
5	5 536	2 768	2 768							1 384	1 384				
6	544	272	272									182	63	27	
7	870	435	435									274	120	41	
8	1 108	554	554												554
9	1 660	830	830												830
	20 962	10 481	10 481	**1 384**	42	**1 384**	**1 384**	**1 384**	44	**1 384**	**1 384**	456	183	68	**1 384**

Tabela 8 - Distribuição do número de instâncias para o conjunto de treinto B=ARef com *undersampling*.

Dia	Instâncias	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	1599	173	1 426	1 384	42										
2	2941	173	2 768			1 384	1 384								
3	1601	173	1 428					1 384	44						
5	2941	173	2 768							1 384	1 384				
6	445	173	272									182	63	27	
7	608	173	435									274	120	41	
8	727	173	554												554
9	1003	173	830												830
	11865	1384	10 481	**1 384**	42	**1 384**	**1 384**	**1 384**	44	**1 384**	**1 384**	456	183	68	**1 384**

A seleção das instâncias para o *undersampling* foi feita de forma aleatória, mas com a preocupação de manter a proporção das ocorrências nos ataques. Para além disso, foi desenvolvido um mecanismo, recorrendo a *scripting*, que permitiu recriar cada uma das experiências dos cenários desenvolvidos evitando ao máximo a reutilização de amostras.

6.4. Random OverSampling

Nas experiências com *Random OverSampling*, a estratégia de implementação passou por aumentar o número de instâncias dos ataques com menos instâncias que as instâncias do ataque de referência e, simultaneamente, diminuir o número de instâncias das classes com mais instâncias. Nas tabelas 9 e 10 podemos verificar a extrapolação das classes com menor número de instâncias para o mesmo valor da classe de referência e a redução das classes superiores para o mesmo valor respetivamente para as vertentes, B=TotalA e B=ARef.

Tabela 9 - Distribuição do número de instâncias para um conjunto de treino B=TotalA com *oversampling*.

Dia	Instâncias	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	5536	2 768	2 768	1 384	1 384										
2	5536	2 768	2 768			1 384	1 384								
3	5536	2 768	2 768					1 384	1 384						
5	5536	2 768	2 768							1 384	1 384				
6	3166	1 583	1 583									552	481	550	
7	5138	2 569	2 569									832	903	834	
8	1108	554	554												554
9	1160	830	830												830
	33216	16 608	16 608	**1 384**	1384	**1 384**	**1 384**	**1 384**	1384	**1 384**	**1 384**	1384	1384	1384	**1 384**

Tabela 10 - Distribuição do número de instâncias para um conjunto de treino B=ARef com *oversampling*.

Dia	Instâncias	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	2 941	173	2 768	1 384	1 384										
2	2 941	173	2 768			1 384	1 384								
3	2 941	173	2 768					1 384	1 384						
5	2 941	173	2 768							1 384	1 384				
6	1 756	173	1 583									552	481	550	
7	2 742	173	2 569									832	903	834	
8	727	173	554												554
9	1 003	173	830												830
	17 992	1384	16 608	**1 384**	1384	**1 384**	**1 384**	**1 384**	1384	**1 384**	**1 384**	1384	1384	1384	**1 384**

6.5. Conjuntos de dados com seleção de atributos

Nos testes envolvendo a seleção de atributos foi criado um plano que assentou na utilização do software *Weka WorkBench* e a sua ferramenta *Select Atributes.*

Partindo dos mesmos conjuntos de dados utilizados nos testes com *undersampling* e *oversampling*, mencionados no diagrama da Figura 14 e no número de instâncias por ataque de referência da Tabela 6, foram aplicados os algoritmos de seleção de dados com os avaliadores de atributos *CorrelationAtributeEval* e *InfoGainAtributeEval*. Com estes avaliadores foi utilizado o método de pesquisa *Ranker*.

O processo de criação dos conjuntos de dados com cada um destes algoritmos de seleção de atributos foi semelhante. Cada algoritmo foi executado com os conjuntos de dados de treino (*undersampling* B=ARef; *undersampling* B=TotalA; *oversampling* B=ARef e *oversampling* B=TotalA). Para cada um destes conjuntos de dados, foram executados os 8 testes de números de instâncias por ataque (42, 44, 68, 183, 456, 1384, 7926, 33125) à semelhança dos pontos anteriores para o *Random OverSampling* e *Random UnderSampling*. A Tabela 11 mostra os resultados obtidos com o algoritmo *CorrelationAtributeEval*. Como se pode ver

nesta tabela, numa primeira fase foram selecionados para cada teste até um máximo de 10 atributos de entre os que o algoritmo considerou mais relevantes. Após este passo, o conjunto final de atributos selecionados foi definido como a união dos conjuntos de atributos obtidos em todos os testes

Este procedimento foi executado para os dois algoritmos, tendo sido obtido um conjunto de atributos distinto para cada um. Cada um destes conjuntos de atributos foi depois utilizado em testes com conjuntos de dados semelhantes ao que são descritos nas secções 6.3 e 6.4. Ou seja, para cada um dos conjuntos descritos naquelas secções, é realizado um grupo de testes com a versão em que todos os atributos estão presentes e dois outros grupos de testes com os dois subconjuntos de atributos selecionados com os dois algoritmos de seleção de atributos.

Após a sua aplicação, foram selecionados 28 atributos utilizando o algoritmo *CorrelationAtributeEval* e 20 utilizando o algoritmo *InfoGainAtributeEval*. A lista de atributos selecionados com cada um dos algoritmos encontra-se no Anexo D.

Tabela 11 – Exemplo de resultados de seleção de atributos por cada conjunto de dados para *CorrelationAtributeEval* com *Ranker*

	42	44	68	183	456	1384	7926	33125
Undersampling B=TotalA	Ranked attributes: 0.3191 1 Dst Port 0.2738 60 Fwd Seg Size Min 0.2036 58 Init Bwd Win Byts 0.194 8 Fwd Pkt Len Max 0.1852 11 Fwd Pkt Len Std 0.1688 13 Bwd Pkt Len Min 0.1636 34 Bwd Header Len 0.1623 54 Subflow Fwd Byts 0.1623 6 TotLen Fwd Pkts 0.1616 33 Fwd Header Len Selected attributes: 1,60,58,8,11,13,34,54,6,33,	Ranked attributes: 0.319 1 Dst Port 0.2715 60 Fwd Seg Size Min 0.2061 58 Init Bwd Win Byts 0.1951 8 Fwd Pkt Len Max 0.1858 11 Fwd Pkt Len Std 0.167 13 Bwd Pkt Len Min 0.1643 34 Bwd Header Len 0.1627 6 TotLen Fwd Pkts 0.1627 54 Subflow Fwd Byts 0.162 33 Fwd Header Len Selected attributes: 1,60,58,8,11,13,34,6,54,33	Ranked attributes: 0.3162 1 Dst Port 0.2566 60 Fwd Seg Size Min 0.2179 58 Init Bwd Win Byts 0.1818 8 Fwd Pkt Len Max 0.1739 11 Fwd Pkt Len Std 0.1687 6 TotLen Fwd Pkts 0.1687 54 Subflow Fwd Byts 0.1681 33 Fwd Header Len 0.1679 53 Subflow Fwd Pkts 0.1679 4 Tot Fwd Pkts Selected attributes: 1,60,58,8,11,6,54,33,53,4,	Ranked attributes: 0.3167 1 Dst Port 0.2565 60 Fwd Seg Size Min 0.2295 58 Init Bwd Win Byts 0.1856 54 Subflow Fwd Byts 0.1856 6 TotLen Fwd Pkts 0.1849 33 Fwd Header Len 0.1847 53 Subflow Fwd Pkts 0.1847 4 Tot Fwd Pkts 0.1847 59 Fwd Act Data Pkts 0.1829 8 Fwd Pkt Len Max Selected attributes: 1,60,58,54,6,33,53,4,59,8,	Ranked attributes: 0.3292 1 Dst Port 0.2564 60 Fwd Seg Size Min 0.2386 58 Init Bwd Win Byts 0.2053 3 Flow Duration 0.2045 22 Fwd IAT Tot 0.2021 8 Fwd Pkt Len Max 0.2006 54 Subflow Fwd Byts 0.2006 6 TotLen Fwd Pkts 0.2002 33 Fwd Header Len 0.2 53 Subflow Fwd Pkts Selected attributes: 1,60,58,3,22,8,54,6,33,53	Ranked attributes: 0.34907 1 Dst Port 0.27752 60 Fwd Seg Size Min 0.24881 58 Init Bwd Win Byts 0.24183 8 Fwd Pkt Len Max 0.23371 11 Fwd Pkt Len Std 0.22338 3 Flow Duration 0.2231 22 Fwd IAT Tot 0.22143 54 Subflow Fwd Byts 0.22143 6 TotLen Fwd Pkts 0.22127 33 Fwd Header Len Selected attributes: 1,60,58,8,11,3,22,54,6,33,	Ranked attributes: 0.35282 60 Fwd Seg Size Min 0.32663 1 Dst Port 0.20112 58 Init Bwd Win Byts 0.19999 8 Fwd Pkt Len Max 0.19824 28 Bwd IAT Mean 0.18809 11 Fwd Pkt Len Std 0.18653 30 Bwd IAT Max 0.17653 31 Bwd IAT Min 0.16241 45 PSH Flag Cnt 0.16205 13 Bwd Pkt Len Min Selected attributes: 60,1,58,8,28,11,30,31,45,13	Ranked attributes: 0.34771 1 Dst Port 0.33828 60 Fwd Seg Size Min 0.21906 8 Fwd Pkt Len Max 0.20496 11 Fwd Pkt Len Std 0.20445 58 Init Bwd Win Byts 0.18288 12 Bwd Pkt Len Max 0.18207 10 Fwd Pkt Len Mean 0.18207 51 Fwd Seg Size Avg 0.1676 45 PSH Flag Cnt 0.16108 57 Init Fwd Win Byts Selected attributes: 1,60,8,11,58,12,10,51,45,57
Undersampling B=ARef	Ranked attributes: 0.2484 60 Fwd Seg Size Min 0.1933 57 Init Fwd Win Byts 0.1818 22 Fwd IAT Tot 0.1817 3 Flow Duration 0.1781 48 ECE Flag Cnt 0.1781 44 RST Flag Cnt 0.1654 49 Down/Up Ratio 0.1566 8 Fwd Pkt Len Max 0.1537 38 Pkt Len Max 0.1495 59 Fwd Act Data Pkts Selected attributes: 60,57,22,3,48,44,49,8,38,59	Ranked attributes: 0.2487 60 Fwd Seg Size Min 0.1954 57 Init Fwd Win Byts 0.1841 22 Fwd IAT Tot 0.184 3 Flow Duration 0.1778 48 ECE Flag Cnt 0.1778 44 RST Flag Cnt 0.1625 49 Down/Up Ratio 0.1559 8 Fwd Pkt Len Max 0.1519 38 Pkt Len Max 0.1494 59 Fwd Act Data Pkts Selected attributes: 60,57,22,3,48,44,49,8,38,59	Ranked attributes: 0.2417 60 Fwd Seg Size Min 0.1968 57 Init Fwd Win Byts 0.179 22 Fwd IAT Tot 0.1787 3 Flow Duration 0.172 44 RST Flag Cnt 0.172 48 ECE Flag Cnt 0.1626 8 Fwd Pkt Len Max 0.1593 49 Down/Up Ratio 0.1523 11 Fwd Pkt Len Std 0.1495 38 Pkt Len Max Selected attributes: 60,57,22,3,44,48,8,49,11,38	Ranked attributes: 0.2377 60 Fwd Seg Size Min 0.1926 57 Init Fwd Win Byts 0.179 22 Fwd IAT Tot 0.1785 3 Flow Duration 0.1702 48 ECE Flag Cnt 0.1702 44 RST Flag Cnt 0.1606 8 Fwd Pkt Len Max 0.1534 49 Down/Up Ratio 0.1506 11 Fwd Pkt Len Std 0.1498 59 Fwd Act Data Pkts Selected attributes: 60,57,22,3,48,44,8,49,11,59	Ranked attributes: 0.2423 60 Fwd Seg Size Min 0.1894 57 Init Fwd Win Byts 0.1822 22 Fwd IAT Tot 0.1819 3 Flow Duration 0.1721 44 RST Flag Cnt 0.1721 48 ECE Flag Cnt 0.1595 8 Fwd Pkt Len Max 0.1496 11 Fwd Pkt Len Std 0.1493 59 Fwd Act Data Pkts 0.1493 53 Subflow Fwd Pkts Selected attributes: 60,57,22,3,44,48,8,11,59,53	Ranked attributes: 0.246 60 Fwd Seg Size Min 0.1908 57 Init Fwd Win Byts 0.1814 22 Fwd IAT Tot 0.181 3 Flow Duration 0.1741 48 ECE Flag Cnt 0.1741 44 RST Flag Cnt 0.1572 8 Fwd Pkt Len Max 0.1507 49 Down/Up Ratio 0.1494 12 Bwd Pkt Len Max 0.1491 59 Fwd Act Data Pkts Selected attributes: 60,57,22,3,48,44,8,49,12,59	Ranked attributes: 0.2466 60 Fwd Seg Size Min 0.1898 57 Init Fwd Win Byts 0.1817 22 Fwd IAT Tot 0.1813 3 Flow Duration 0.1718 44 RST Flag Cnt 0.1718 48 ECE Flag Cnt 0.1566 49 Down/Up Ratio 0.1511 8 Fwd Pkt Len Max 0.1491 59 Fwd Act Data Pkts 0.1491 53 Subflow Fwd Pkts Selected attributes: 60,57,22,3,44,48,49,8,59,53	Ranked attributes: 0.2478 60 Fwd Seg Size Min 0.19 57 Init Fwd Win Byts 0.1819 22 Fwd IAT Tot 0.1816 3 Flow Duration 0.1682 48 ECE Flag Cnt 0.1682 44 RST Flag Cnt 0.1542 8 Fwd Pkt Len Max 0.1532 49 Down/Up Ratio 0.1491 59 Fwd Act Data Pkts 0.1491 4 Tot Fwd Pkts Selected attributes: 60,57,22,3,48,44,8,49,59,4
Oversampling B=TotalA	Ranked attributes: 0.3191 1 Dst Port 0.2738 60 Fwd Seg Size Min 0.2036 58 Init Bwd Win Byts 0.194 8 Fwd Pkt Len Max 0.1852 11 Fwd Pkt Len Std 0.1688 13 Bwd Pkt Len Min 0.1636 34 Bwd Header Len 0.1623 54 Subflow Fwd Byts 0.1623 6 TotLen Fwd Pkts 0.1616 33 Fwd Header Len Selected attributes: 1,60,58,8,11,13,34,54,6,33	Ranked attributes: 0.3188 1 Dst Port 0.2728 60 Fwd Seg Size Min 0.206 58 Init Bwd Win Byts 0.1957 8 Fwd Pkt Len Max 0.1864 11 Fwd Pkt Len Std 0.1669 13 Bwd Pkt Len Min 0.1638 34 Bwd Header Len 0.1622 54 Subflow Fwd Byts 0.1622 6 TotLen Fwd Pkts 0.1615 33 Fwd Header Len Selected attributes: 1,60,58,8,11,13,34,54,6,33	Ranked attributes: 0.3122 1 Dst Port 0.2768 60 Fwd Seg Size Min 0.2162 58 Init Bwd Win Byts 0.1923 8 Fwd Pkt Len Max 0.1846 11 Fwd Pkt Len Std 0.1635 13 Bwd Pkt Len Min 0.1598 54 Subflow Fwd Byts 0.1598 6 TotLen Fwd Pkts 0.1592 33 Fwd Header Len 0.159 4 Tot Fwd Pkts Selected attributes: 1,60,58,8,11,13,54,6,33,4	Ranked attributes: 0.3013 1 Dst Port 0.28 60 Fwd Seg Size Min 0.219 58 Init Bwd Win Byts 0.1938 8 Fwd Pkt Len Max 0.1849 11 Fwd Pkt Len Std 0.1616 13 Bwd Pkt Len Min 0.156 6 TotLen Fwd Pkts 0.156 54 Subflow Fwd Byts 0.1555 33 Fwd Header Len 0.1554 53 Subflow Fwd Pkts Selected attributes: 1,60,58,8,11,13,6,54,33,53	Ranked attributes: 0.323 1 Dst Port 0.2589 60 Fwd Seg Size Min 0.2127 8 Fwd Pkt Len Max 0.2115 58 Init Bwd Win Byts 0.204 11 Fwd Pkt Len Std 0.1583 54 Subflow Fwd Byts 0.1583 6 TotLen Fwd Pkts 0.1578 33 Fwd Header Len 0.1576 53 Subflow Fwd Pkts 0.1576 4 Tot Fwd Pkts Selected attributes: 1,60,8,58,11,54,6,33,53,4,	Ranked attributes: 0.3184 1 Dst Port 0.2691 60 Fwd Seg Size Min 0.2095 58 Init Bwd Win Byts 0.1977 8 Fwd Pkt Len Max 0.1904 11 Fwd Pkt Len Std 0.1618 6 TotLen Fwd Pkts 0.1618 54 Subflow Fwd Byts 0.1611 33 Fwd Header Len 0.161 53 Subflow Fwd Pkts 0.161 4 Tot Fwd Pkts Selected attributes: 1,60,58,8,11,6,54,33,53,4	Ranked attributes: 0.3159 1 Dst Port 0.2681 60 Fwd Seg Size Min 0.2021 58 Init Bwd Win Byts 0.1937 8 Fwd Pkt Len Max 0.1878 11 Fwd Pkt Len Std 0.1617 54 Subflow Fwd Byts 0.1617 6 TotLen Fwd Pkts 0.1611 33 Fwd Header Len 0.161 4 Tot Fwd Pkts 0.161 53 Subflow Fwd Pkts Selected attributes: 1,60,58,8,11,54,6,33,4,53	Ranked attributes: 0.317 1 Dst Port 0.2656 60 Fwd Seg Size Min 0.2043 58 Init Bwd Win Byts 0.1978 8 Fwd Pkt Len Max 0.1905 11 Fwd Pkt Len Std 0.1617 54 Subflow Fwd Byts 0.1617 6 TotLen Fwd Pkts 0.1611 33 Fwd Header Len 0.161 53 Subflow Fwd Pkts 0.161 4 Tot Fwd Pkts 0.1609 59 Fwd Act Data Pkts Selected attributes: 1,60,58,8,11,54,6,33,53,4,59
Oversampling B=ARef	Ranked attributes: 0.2484 60 Fwd Seg Size Min 0.1933 57 Init Fwd Win Byts 0.1818 22 Fwd IAT Tot 0.1817 3 Flow Duration 0.1781 48 ECE Flag Cnt 0.1781 44 RST Flag Cnt 0.1654 49 Down/Up Ratio 0.1566 8 Fwd Pkt Len Max 0.1537 38 Pkt Len Max 0.1495 59 Fwd Act Data Pkts Selected attributes: 60,57,22,3,48,44,49,8,38,59	Ranked attributes: 0.2487 60 Fwd Seg Size Min 0.1954 57 Init Fwd Win Byts 0.1841 22 Fwd IAT Tot 0.184 3 Flow Duration 0.1778 48 ECE Flag Cnt 0.1778 44 RST Flag Cnt 0.1625 49 Down/Up Ratio 0.1559 8 Fwd Pkt Len Max 0.1519 38 Pkt Len Max 0.1494 59 Fwd Act Data Pkts Selected attributes: 60,57,22,3,48,44,49,8,38,59	Ranked attributes: 0.2417 60 Fwd Seg Size Min 0.1968 57 Init Fwd Win Byts 0.179 22 Fwd IAT Tot 0.1787 3 Flow Duration 0.172 44 RST Flag Cnt 0.172 48 ECE Flag Cnt 0.1626 8 Fwd Pkt Len Max 0.1593 49 Down/Up Ratio 0.1523 11 Fwd Pkt Len Std 0.1495 38 Pkt Len Max Selected attributes: 60,57,22,3,44,48,8,49,11,38	Ranked attributes: 0.2377 60 Fwd Seg Size Min 0.1926 57 Init Fwd Win Byts 0.179 22 Fwd IAT Tot 0.1785 3 Flow Duration 0.1702 48 ECE Flag Cnt 0.1702 44 RST Flag Cnt 0.1606 8 Fwd Pkt Len Max 0.1534 49 Down/Up Ratio 0.1506 11 Fwd Pkt Len Std 0.1498 59 Fwd Act Data Pkts Selected attributes: 60,57,22,3,48,44,8,49,11,59	Ranked attributes: 0.2423 60 Fwd Seg Size Min 0.1894 57 Init Fwd Win Byts 0.1822 22 Fwd IAT Tot 0.1819 3 Flow Duration 0.1721 44 RST Flag Cnt 0.1721 48 ECE Flag Cnt 0.1595 8 Fwd Pkt Len Max 0.1496 11 Fwd Pkt Len Std 0.1493 59 Fwd Act Data Pkts 0.1493 53 Subflow Fwd Pkts Selected attributes: 60,57,22,3,44,48,8,11,59,53	Ranked attributes: 0.246 60 Fwd Seg Size Min 0.1908 57 Init Fwd Win Byts 0.1814 22 Fwd IAT Tot 0.181 3 Flow Duration 0.1741 48 ECE Flag Cnt 0.1741 44 RST Flag Cnt 0.1572 8 Fwd Pkt Len Max 0.1507 49 Down/Up Ratio 0.1494 12 Bwd Pkt Len Max 0.1491 59 Fwd Act Data Pkts Selected attributes: 60,57,22,3,48,44,8,49,12,59	Ranked attributes: 0.2466 60 Fwd Seg Size Min 0.1898 57 Init Fwd Win Byts 0.1817 22 Fwd IAT Tot 0.1813 3 Flow Duration 0.1718 44 RST Flag Cnt 0.1718 48 ECE Flag Cnt 0.1566 49 Down/Up Ratio 0.1511 8 Fwd Pkt Len Max 0.1491 59 Fwd Act Data Pkts 0.1491 53 Subflow Fwd Pkts Selected attributes: 60,57,22,3,44,48,49,8,59,53	Ranked attributes: 0.2478 60 Fwd Seg Size Min 0.19 57 Init Fwd Win Byts 0.1819 22 Fwd IAT Tot 0.1816 3 Flow Duration 0.1682 48 ECE Flag Cnt 0.1682 44 RST Flag Cnt 0.1542 8 Fwd Pkt Len Max 0.1532 49 Down/Up Ratio 0.1491 59 Fwd Act Data Pkts 0.1491 4 Tot Fwd Pkts Selected attributes: 60,57,22,3,48,44,8,49,59,4

Esta página foi intencionalmente deixada em branco

7. Testes e análise dos resultados

Neste capítulo, são descritos os resultados dos testes realizados, sendo também avançadas algumas explicações desses mesmos resultados.

7.1. Configuração dos testes

Para realizar os testes pretendidos foram utilizadas as ferramentas Weka *Workbench* e Weka API. A versão *Workbench* foi utilizada para edição e remoção de instâncias e seleção de atributos. Com a versão API, foram criados algoritmos em Java, cujo objetivo foi executar cada um dos algoritmos de classificação de 30 vezes sobre o mesmo conjunto de dados de treino. Os algoritmos de classificação selecionados, já mencionados anteriormente, foram o IBK, *Multilayer Perceptron* e *Random Forest*. De realçar ainda que em todos os testes foram utilizadas as parametrizações por omissão no Weka dos algoritmos utilizados, já que o objetivo não passou por tentar encontrar a melhor parametrização para cada um dos algoritmos, mas sim testar os efeitos de balanceamento dos dados. No total, foram realizados 2*2*3*8=96 testes. Estes valores correspondem, respetivamente, aos cenários *undersampling* e *oversampling* (2), B=TotalA e B=ARef (2), um cenário sem seleção de atributos e dois com seleção de atributos (3), e 8 ataques de referência.

Quanto às métricas utilizadas, para cada teste, foram utilizadas a Acurácia, a Precisão, a Revocação e F-*Measure*. Nas próximas secções os resultados são apresentados com base no *F-Measure*, visto ser uma métrica mais robusta quando os dados não estão balanceados. Os resultados das restantes métricas estão disponíveis nos Anexos B e C.

Em termos de *hardware*, foram utilizados: um computador com processador *Intel Core* i7-1065G7 *Quad-Core* com 24Gb de memória RAM com sistema operativo *Windows*; utilização de máquinas *Linode* (8 CPU, 16 Gb RAM) com sistema operativo Debian para processamento remoto e, por fim, a utilização de uma máquina facultada pelo Politécnico de Leiria (*Intel Xeon Gold* 6226 2.70 GHz, 16 Gb Ram) com sistema operativo *Windows*.

7.2. Testes sem seleção de atributos

Nesta secção são apresentados os resultados que foram obtidos com os conjuntos de dados com todos os atributos que não foram eliminados durante o processo de limpeza de dados. São apresentados resultados com *undersampling* e *oversampling* e, para cada um destes cenários, resultados obtidos com conjuntos de dados B=TotalA e B=ARef.

7.2.1. *Random UnderSampling*

As tabelas 12 e 13 contêm o número de instâncias por classe dos conjuntos de dados para cada um dos 8 ataques de referência para os cenários B=TotalA e B=ARef, respetivamente, quando foi utilizado *undersampling*. Cada um destes conjuntos foi testado com todos os algoritmos referidos na secção anterior.

Tabela 12 - Conjuntos de dados dos 8 cenários de *undersampling* com B=TotalA

Teste	Ataque de Referência	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	42	504	504	42	42	42	42	42	42	42	42	42	42	42	42
2	44	526	526	44	42	44	44	44	44	44	44	44	44	44	44
3	68	766	766	68	42	68	68	68	44	68	68	68	68	68	68
4	183	1801	1801	183	42	183	183	183	44	183	183	183	183	68	183
5	456	3985	3985	456	42	456	456	456	44	456	456	456	183	68	456
6	1384	10481	10481	1384	42	1384	1384	1384	44	1384	1384	456	183	68	1384
7	7926	49733	49733	7926	42	7926	7926	7926	44	1384	7926	456	183	68	7926
8	33125	175728	175728	33125	42	33125	7926	33125	44	1384	33125	456	183	68	33125

Tabela 13 - Conjuntos de dados dos 8 cenários de *undersampling* com B=ARef

Teste	Ataque de Referência	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	42	42	504	42	42	42	42	42	42	42	42	42	42	42	42
2	44	44	526	44	42	44	44	44	44	44	44	44	44	44	44
3	68	68	766	68	42	68	68	68	44	68	68	68	68	68	68
4	183	183	1801	183	42	183	183	183	44	183	183	183	183	68	183
5	456	456	3985	456	42	456	456	456	44	456	456	456	183	68	456
6	1384	1384	10481	1384	42	1384	1384	1384	44	1384	1384	456	183	68	1384
7	7926	7926	49733	7926	42	7926	7926	7926	44	1384	7926	456	183	68	7926
8	33125	33125	175728	33125	42	33125	7926	33125	44	1384	33125	456	183	68	33125

No Gráfico 1 são apresentados os resultados obtidos com estes conjuntos de dados.

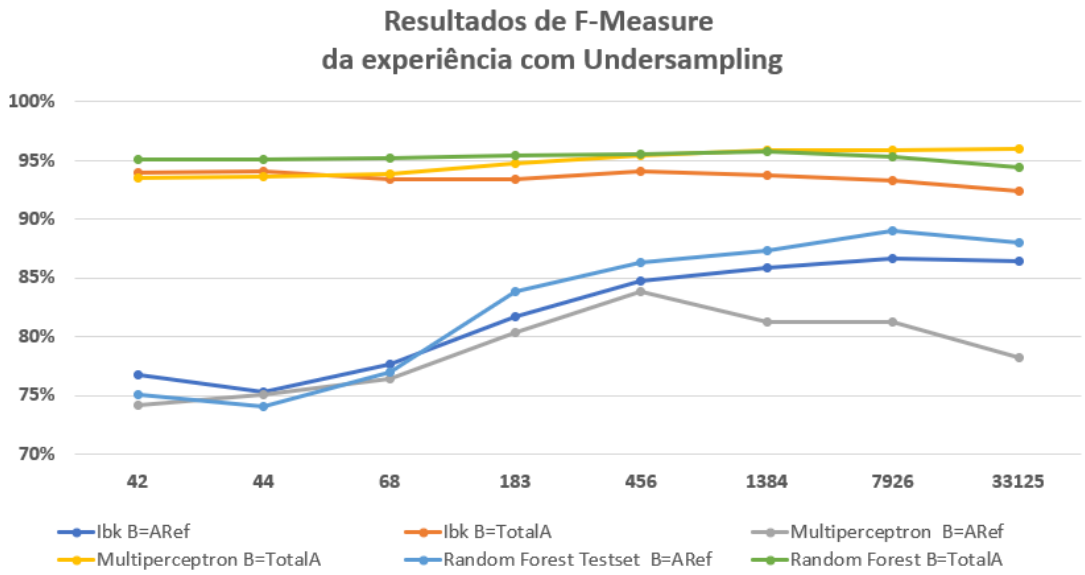

Gráfico 1 - *F-Measure* das experiências com *undersampling*

Analisando o Gráfico 1, existem alguns padrões que se podem detetar, nomeadamente:

- Os resultados obtidos com os conjuntos B=TotalA, são superiores aos que são obtidos com os conjuntos B=ARef e menos sensíveis a alterações; uma explicação para este resultado é o facto de os conjuntos B=TotalA terem bastantes mais instâncias benignas, o que leva os modelos a classificarem o tráfego como benigno de forma correta bastantes mais vezes. Este facto pode ser comprovado analisando as matrizes de confusão. As figuras 15 e 16 mostram as matrizes de confusão dos casos B=TotalA em *undersampling* com ataque de referência 68 e B=ARef em

undesampling com o mesmo ataque de referência. Como se pode ver, a quantidade de tráfego benigno corretamente identificado aumenta quase para o dobro, sendo que a grandeza dos acertos para as outras classes não sofre grandes alterações;

```
=== Confusion Matrix ===

      a     b     c     d     e     f     g     h     i     j     k     l     m   <-- classified as
 913353   169     6   127   346   571    19     3  1298  2208  1348   825 14479 |     a = Benign
     14 18758     1     0     0    31     0     0     0     0     0     6     0 |     b = SSH-Bruteforce
      0     0     1     0     0     0    10     0     0     0     0     0     0 |     c = FTP-BruteForce
      0     0     0  8270     9     2     0     0     0     0     0     0     0 |     d = DoS attacks-GoldenEye
     38     0     0     0  1940     0     0     0     0     0     0     4     0 |     e = DoS attacks-Slowloris
     23     0     0    66     0 28451     0     0   500     0     0     0     0 |     f = DoS attacks-Hulk
      0     0     9     0     0     0     2     0     0     0     0     0     0 |     g = DoS attacks-SlowHTTPTest
      0     0     0     0     0     0     0   346     0     0     0     0     0 |     h = DDOS attack-LOIC-UDP
      2     0     0     0     0     0     0     0 39647     6   109     8     0 |     i = DDOS attack-HOIC
      6     0     0     0     0     0     0     0     0    82    13    13     0 |     j = Brute Force -Web
      2     0     0     0     0     0     0     0     0     2    41     1     0 |     k = Brute Force -XSS
      1     0     0     0     0     0     0     0     0     2     0    14     0 |     l = SQL Injection
  25774     0     1    70    20    20    12     0    13    33    38   101  2708 |     m = Infilteration
```

Figura 15 – Teste com Random Forest de B=TotalA em *undersampling* com ataque de referência 68

```
=== Confusion Matrix ===

      a     b     c     d     e     f     g     h     i     j     k     l     m   <-- classified as
 599120   178   114 26030  2433  4447   203     4  6330 15780  4594 38797 236722 |     a = Benign
      1 18758     9     1     0     3     4     0     0     0     0    34     0 |     b = SSH-Bruteforce
      0     0     0     0     0     0    11     0     0     0     0     0     0 |     c = FTP-BruteForce
      0     0     0  8269    10     2     0     0     0     0     0     0     0 |     d = DoS attacks-GoldenEye
     33     0     0     2  1939     0     0     0     0     2     0     5     1 |     e = DoS attacks-Slowloris
      2     0     0    59     0 28002     0     0   968     0     0     9     0 |     f = DoS attacks-Hulk
      0     0     8     0     0     0     3     0     0     0     0     0     0 |     g = DoS attacks-SlowHTTPTest
      0     0     0     0     0     0     0   346     0     0     0     0     0 |     h = DDOS attack-LOIC-UDP
      2     0     0     0     0     0     0     0 39648     1   113     8     0 |     i = DDOS attack-HOIC
      0     0     0     0     0     0     0     0     0    99     0    14     1 |     j = Brute Force -Web
      1     0     0     0     0     0     0     0     0     2    42     1     0 |     k = Brute Force -XSS
      0     0     0     0     0     0     0     0     0     3     0    14     0 |     l = SQL Injection
  11891     2     4   496   578    54    57     0   143   806   131  1277 13351 |     m = Infilteration
```

Figura 16 – Teste com Random forest de B=ARef em *undersampling* com ataque de referência 68

- Excetuando uma ligeira descida em alguns casos, em particular, quando passamos do ataque de referência com 42 instâncias para o de 44, o valor da F-*Measure* aumenta à medida que o número de instâncias do ataque de referência aumenta, voltando a diminuir quando o número de instâncias do ataque de referência ultrapassa um determinado limiar. Para se perceber a explicação abaixo convém primeiro realçar que 1) os conjuntos de dados correspondentes aos resultados mais à direita no gráfico estão mais próximos do conjunto original, ou seja, que são menos balanceados, e 2) que o tamanho dos conjuntos de dados é menor para ataques de referência com menos instâncias. Assim, a análise do gráfico da direita para a esquerda, parece indicar que o efeito do balanceamento dos dados é benéfico até um determinado ponto (7926, 1384, 456, dependendo dos casos), apesar de isso implicar uma diminuição do número de instâncias do conjunto de dados. No entanto, a partir de um certo ponto, os resultados pioram, o que parece indicar que, a partir desse ponto, a perda de dados resultante do balanceamento tem um impacto negativo maior do que os eventuais ganhos.

Pode-se verificar nos anexos B e C que as mesmas conclusões podem ser retiradas para as restantes métricas, exceto para a precisão. Em geral, o valor da precisão aumenta com o número de instâncias da classe de referência. A análise das matrizes de confusão permite

verificar que este resultado acontece devido ao aumento de previsões corretas, sobretudo nas classes mais frequentes, com o aumento do número de instâncias do ataque de referência. Tal, é possível verificar comparando as variações de resultados entre a Figura 16 e a Figura 17.

```
=== Confusion Matrix ===

      a     b     c     d     e     f     g     h     i     j     k     l      m   <-- classified as
 774212    94    12    67   151   238    16     0   546  3880   313   281 154942 |      a = Benign
      4 18804     1     0     0     1     0     0     0     0     0     0      0 |      b = SSH-Bruteforce
      0     0     1     0     0     0    10     0     0     0     0     0      0 |      c = FTP-BruteForce
      0     0     0  8280     0     1     0     0     0     0     0     0      0 |      d = DoS attacks-GoldenEye
      0     0     0     0  1982     0     0     0     0     0     0     0      0 |      e = DoS attacks-Slowloris
      0     0     0     0     0 29040     0     0     0     0     0     0      0 |      f = DoS attacks-Hulk
      0     0     9     0     0     0     2     0     0     0     0     0      0 |      g = DoS attacks-SlowHTTPTest
      0     0     0     0     0     0     0   346     0     0     0     0      0 |      h = DDOS attack-LOIC-UDP
      0     0     0     0     0     0     0     0 39754     1    11     4      2 |      i = DDOS attack-HOIC
      3     0     0     0     0     0     0     0     0   104     0     6      1 |      j = Brute Force -Web
      0     0     0     0     0     0     0     0     0     3    43     0      0 |      k = Brute Force -XSS
      0     0     0     0     0     0     0     0     0     4     0    13      0 |      l = SQL Injection
   9855     0     1     4     2     0     1     0     5    56     2    12  18852 |      m = Infilteration
```

Figura 17 - Teste com Random forest de B=ARef em undersampling com ataque de referência 7926

7.2.2. *Random Oversampling*

As tabelas 14 e 15 contêm o número de instâncias por classe dos conjuntos de dados para cada um dos 8 ataques de referência para os cenários B=TotalA e B=ARef, respetivamente, quando foi utilizado *oversampling*.

Tabela 14 - Conjuntos de dados dos 8 cenários de *oversampling* com B=TotalA

Teste	Ataque de Referência	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	42	504	504	42	42	42	42	42	42	42	42	42	42	42	42
2	44	528	528	44	44	44	44	44	44	44	44	44	44	44	44
3	68	816	816	68	68	68	68	68	68	68	68	68	68	68	68
4	183	2196	2196	183	183	183	183	183	183	183	183	183	183	183	183
5	456	5472	5472	456	456	456	456	456	456	456	456	456	456	456	456
6	1384	16608	16608	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384
7	7926	95112	95112	7926	7926	7926	7926	7926	7926	7926	7926	7926	7926	7926	7926
8	33125	397500	397500	33125	33125	33125	33125	33125	33125	33125	33125	33125	33125	33125	33125

Tabela 15 - Conjuntos de dados dos 8 cenários de *oversampling* com B=Aref

Teste	Ataque de Referência	Benignas	Maliciosas	SSH Bruteforce	FTP Bruteforce	DoS GoldenEye	DosS Slowloris	DoS Hulk	DoS SlowHTTPTest	DDoS LOICUDP	DDOS HOIC	Brute Force Web	Brute Force XSS	Sql Injection	Infiltration
1	42	42	504	42	42	42	42	42	42	42	42	42	42	42	42
2	44	44	528	44	44	44	44	44	44	44	44	44	44	44	44
3	68	68	816	68	68	68	68	68	68	68	68	68	68	68	68
4	183	183	2196	183	183	183	183	183	183	183	183	183	183	183	183
5	456	456	5472	456	456	456	456	456	456	456	456	456	456	456	456
6	1384	1384	16608	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384	1384
7	7926	7926	95112	7926	7926	7926	7926	7926	7926	7926	7926	7926	7926	7926	7926
8	33125	33125	397500	33125	33125	33125	33125	33125	33125	33125	33125	33125	33125	33125	33125

No Gráfico 2 são apresentados os resultados obtidos com estes conjuntos de dados.

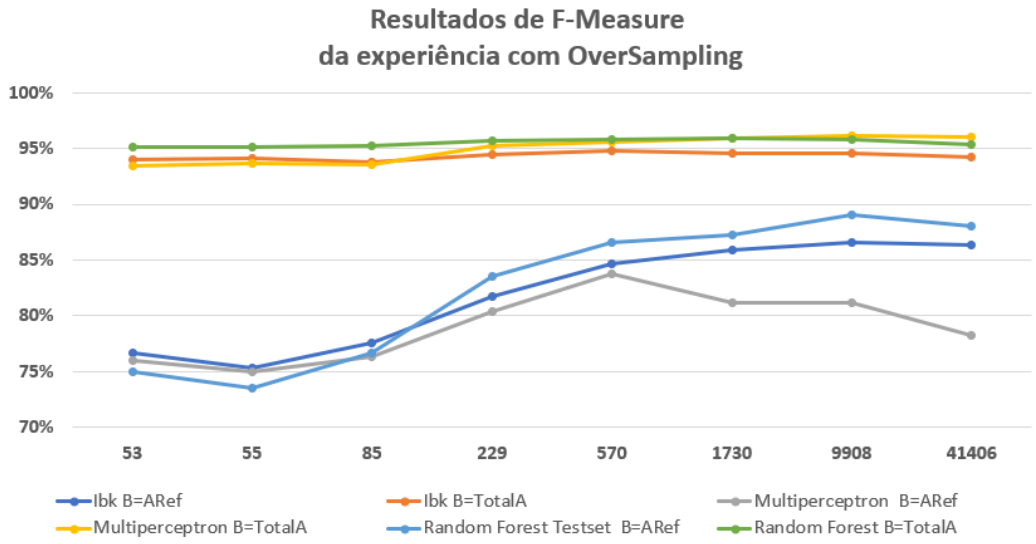

Gráfico 2 - *F-Measure* da experiência com *oversampling*

Os resultados, quando comparados com os do Gráfico 1, apresentam valores muito semelhantes embora, em alguns casos, ligeiramente superiores (comparar, por exemplo, os valores obtidos para o ataque de referência com 42 instâncias). Esta constatação parece indicar que a influência do *oversampling* é muito baixa. Lembramos que nestes conjuntos de dados também se procedeu ao *undersampling* das classes com mais instâncias do que a classe (ataque) de referência. Isso significa que a única diferença entre os dois grupos de conjuntos de dados (*undersampling* e *oversampling*) é o *oversampling* realizado às classes com menos instâncias que a classe de referência, permitindo desta forma isolar o efeito deste aspeto relativamente aos resultados obtidos apenas com *undersampling*. Assim, uma explicação possível para os resultados é que o efeito do *undersampling* operado sobre as classes com mais instâncias que a classe de referência domina completamente o efeito do *oversampling* operado sobre as classes com menos instâncias. No entanto, considerando que o número de instâncias resultantes de *oversampling* aumenta com o número de instâncias da classe de referência, esperar-se-ia uma maior influência desta operação nos pontos mais à direita no gráfico, o que não se verifica, dada a semelhança com os resultados obtidos apenas com *undersampling*. Esta análise parece indicar que o efeito da operação de *oversampling* utilizando o método *Random OverSampling* é de facto muito pouco significativo.

7.3. Testes com seleção de atributos

Nesta secção são apresentados os resultados com os conjuntos de dados sobre os quais foi realizada a seleção de atributos. Os algoritmos utilizados, já mencionados anteriormente, são o *CorrelationAtributeEval* e o *InfoGainAtributeEval* com algoritmo de pesquisa *Ranker*. Não são apresentadas tabelas com o número de instâncias por classe de cada conjunto de dados uma vez que se mantêm os valores apresentados nas tabelas 12 e 13, para *undersampling*, e nas tabelas 14 e 15, para *oversampling*. Os algoritmos de aprendizagem utilizados com cada conjunto de dados também foram os mesmos. De realçar que a única

diferença entre os conjuntos de teste utilizados nestes testes e nos testes sem seleção de atributos é a remoção dos atributos não selecionados com cada algoritmo de seleção.

Nos gráficos 3 e 4 são apresentados os resultados obtidos com o algoritmo com *CorrelationAtributeEval*, com *undersampling* e *oversampling*, respetivamente. Por sua vez, nos gráficos 5 e 6 são apresentados os resultados obtidos com o algoritmo com *InfoGainAtributeEval*, com *undersampling* e *oversampling*, respetivamente.

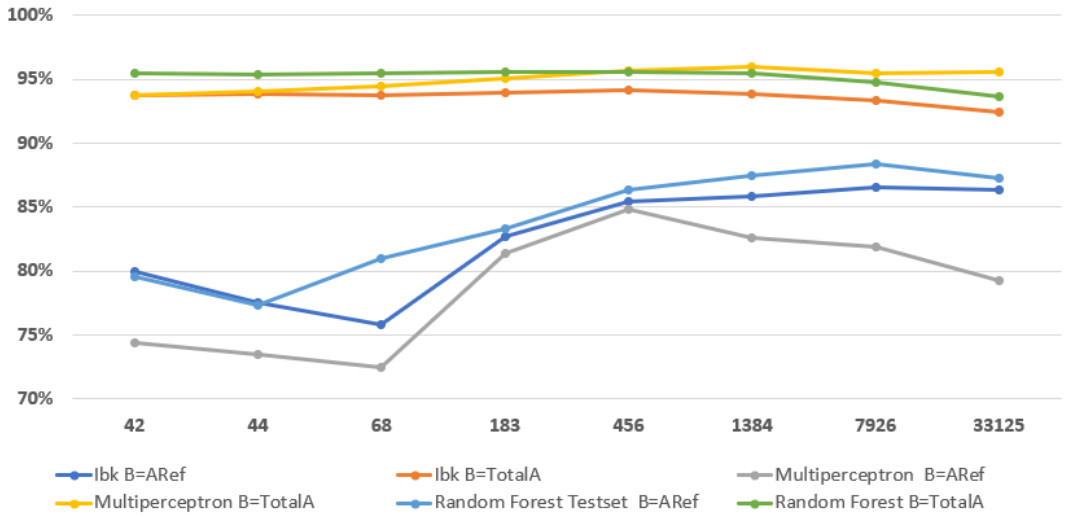

Gráfico 3 - *F-Measure* de *CorrelationAtributeEval* com *ranker* e com *undersampling*

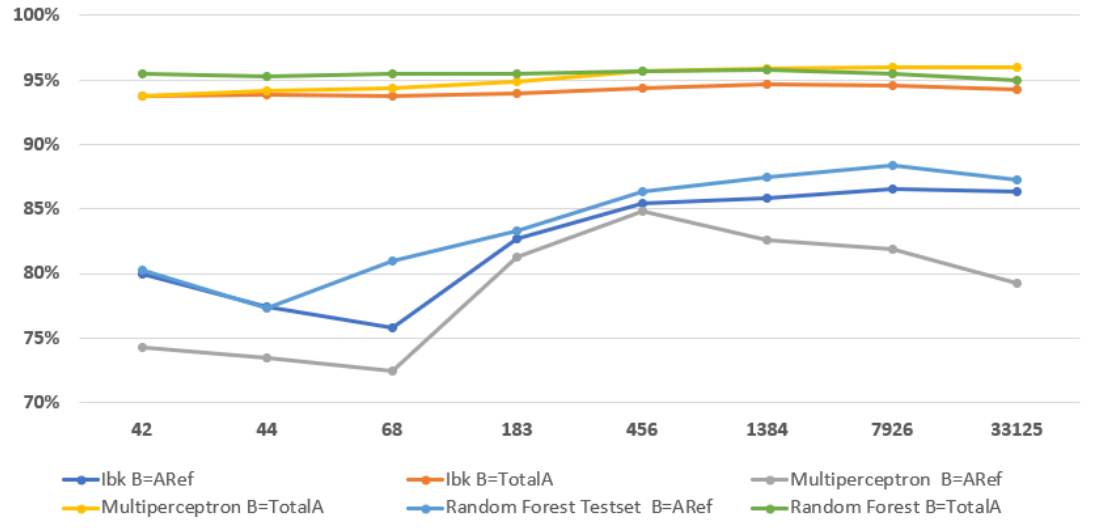

Gráfico 4 - *F-Measure* de *CorrelationAtributeEval* com *ranker* e com *oversampling*

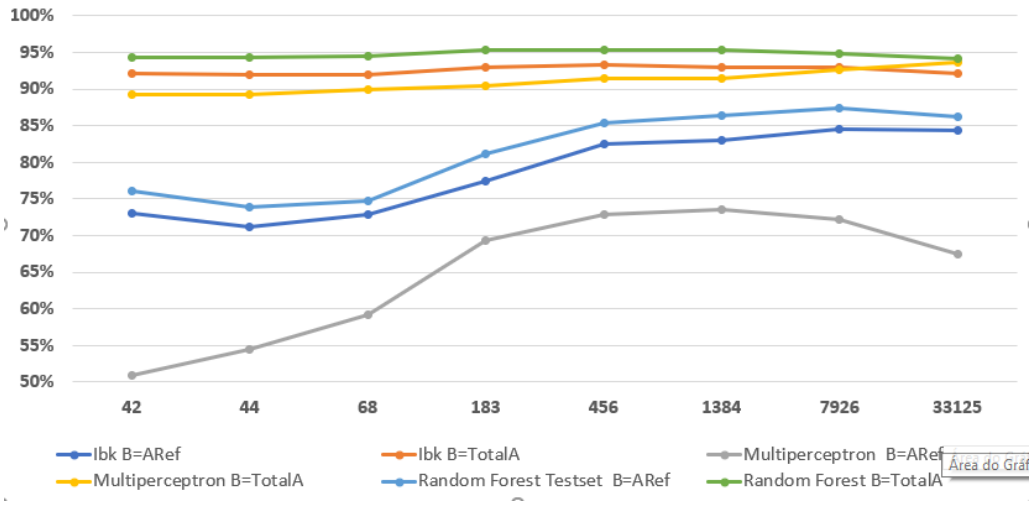

Gráfico 5 - *F-Measure* de *InfoGainAtributeEval* com *ranker* com *undersampling*

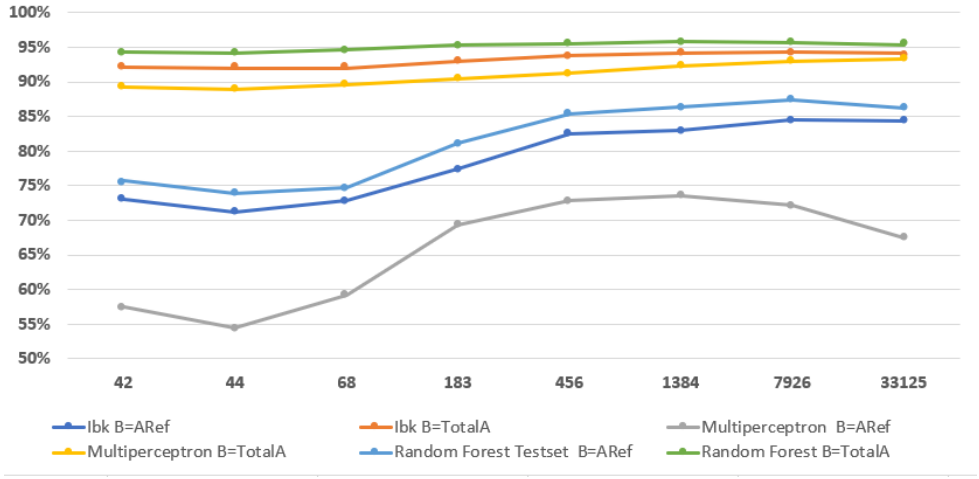

Gráfico 6 - *F-Measure* de *InfoGainAtributeEval* com *ranker* em *oversampling*

Analisando estes gráficos, podemos concluir que, excetuando os valores absolutos e algumas variações nas curvas, os padrões das curvas obtidas com seleção de atributos são essencialmente os mesmos que são obtidos sem seleção de atributos. Isso significa que os resultados obtidos relativamente ao efeito das operações de balanceamento utilizadas neste trabalho se mostram robustos a operações de seleção de atributos e que podem ser retiradas as mesmas conclusões.

Esta página foi intencionalmente deixada em branco

8. Conclusão e trabalho futuro

O objetivo deste trabalho consistiu em estudar os efeitos de duas técnicas de balanceamento dos dados no processo da aprendizagem computacional, o *Random UnderSampling* e o *Random OverSampling*. A motivação para a realização deste trabalho resultou do facto de uma grande maioria dos conjuntos de dados reais, serem bastante desequilibrados em termos de balanceamento. A acrescer a isso, grande parte dos trabalhos que abordam a temática de aprendizagem computacional na área de cibersegurança, concentra-se mais na testagem de novos algoritmos de classificação, e muito menos no pré-processamento dos dados, em particular, nas operações de balanceamento dos dados.

Para este trabalho foi selecionado o conjunto de dados CSE-CIC-IDS-2018, um conjunto de dados criado para simular cenários reais de redes computacionais bastante realistas pelos padrões atuais.

A este conjunto de dados foram, numa primeira fase, aplicadas técnicas de limpeza. Numa segunda fase foram aplicadas técnicas de balanceamento de dados, mais concretamente, os métodos de *Random UnderSampling* e *Random OverSampling*, para construir conjuntos de treino que correspondessem a diferentes cenários. Para além dos cenários com *undersampling* e *oversampling*, consideraram-se também outras variantes como a denominada B=TotalA, em que o número de instâncias benignas era igual ao total do número de instâncias maliciosas, e a B=ARef, em que o número de instâncias benignas era igual ao número de instâncias do ataque de referência. Consideraram-se ainda cenários com e sem seleção de atributos. Por fim, foram realizados testes com três algoritmos de classificação distintos.

Os resultados indicam que existe um efeito benéfico até determinado grau de *undersampling* mas que esse efeito desaparece quando a redução de dados ultrapassa um determinado limiar. Por outro lado, os resultados também mostram que o efeito da operação de *oversampling*, utilizando o operador *Random OverSampling* é muito pouco significativo. Merece também ser referido que são obtidos resultados significativamente melhores com os cenários B=TotalA do que com os cenários B=ARef. Finalmente, verificou-se que os resultados são robustos à seleção de atributos, podendo ser retiradas as mesmas conclusões.

Relativamente ao trabalho futuro, este poderá passar por três abordagens, nomeadamente:

- Estudar o efeito do *oversampling* em conjuntos de dados em que as instâncias dos ataques mais frequentes que o ataque de referência não são alvo de *undersampling*, de forma a confirmar as explicações apresentadas para os resultados obtidos neste trabalho;
- Estudar o efeito da aplicação de outros métodos de *oversampling*, como o SMOTE e outros referidos neste documento;

- Por fim, seria pertinente realizar testes com ataques de referência para além das 33125 instâncias, de modo a confirmar a tendência descendente dos resultados para ataques de referência mais frequentes.

Referências Bibliográficas

[1] J. L. Leevy e T. M. Khoshgoftaar, «A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data», *J. Big Data*, vol. 7, n. 1, p. 104, Dez. 2020, doi: 10.1186/s40537-020-00382-x.

[2] M. Verkerken, L. D'hooge, T. Wauters, B. Volckaert, e F. De Turck, «Unsupervised Machine Learning Techniques for Network Intrusion Detection on Modern Data», em *2020 4th Cyber Security in Networking Conference (CSNet)*, Lausanne, Switzerland, Out. 2020, pp. 1–8. doi: 10.1109/CSNet50428.2020.9265461.

[3] G. Karatas, O. Demir, e O. K. Sahingoz, «Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset», *IEEE Access*, vol. 8, pp. 32150–32162, 2020, doi: 10.1109/ACCESS.2020.2973219.

[4] I. A. Solomon, A. Jatain, e S. B. Bajaj, «Neural Network Based Intrusion Detection: State of the Art», *SSRN Electron. J.*, 2019, doi: 10.2139/ssrn.3356505.

[5] K. Coulibaly, «An overview of Intrusion Detection and Prevention Systems», *ArXiv200408967 Cs*, Abr. 2020, Acedido: Jul. 24, 2021. [Em linha]. Disponível em: http://arxiv.org/abs/2004.08967

[6] S. Qadir e S. M. K. Quadri, «Information Availability: An Insight into the Most Important Attribute of Information Security», *J. Inf. Secur.*, vol. 07, n. 03, pp. 185–194, 2016, doi: 10.4236/jis.2016.73014.

[7] A. Khraisat, I. Gondal, P. Vamplew, e J. Kamruzzaman, «Survey of intrusion detection systems: techniques, datasets and challenges», *Cybersecurity*, vol. 2, n. 1, p. 20, Dez. 2019, doi: 10.1186/s42400-019-0038-7.

[8] J. V. D. Ham, «Toward a Better Understanding of "Cybersecurity"», *Digit. Threats Res. Pract.*, vol. 2, n. 3, pp. 1–3, Jul. 2021, doi: 10.1145/3442445.

[9] T. Hamed, J. B. Ernst, e S. C. Kremer, «A Survey and Taxonomy of Classifiers of Intrusion Detection Systems», em *Computer and Network Security Essentials*, K. Daimi, Ed. Cham: Springer International Publishing, 2018, pp. 21–39. doi: 10.1007/978-3-319-58424-9_2.

[10] A. Gupta e L. S. Sharma, «A categorical survey of state-of-the-art intrusion detection system-*Snort*», *Int. J. Inf. Comput. Secur.*, vol. 13, n. 3/4, p. 337, 2020, doi: 10.1504/IJICS.2020.109481.

[11] A.-S. K. Pathan, *The state of the art in intrusion prevention and detection*. 2016.

[12] K. A. Scarfone e P. M. Mell, «Guide to Intrusion Detection and Prevention Systems (IDPS)», National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-94, 2007. doi: 10.6028/NIST.SP.800-94.

[13] R. Atefinia e M. Ahmadi, «Network intrusion detection using multi-architectural modular deep neural network», *J. Supercomput.*, vol. 77, n. 4, pp. 3571–3593, Abr. 2021, doi: 10.1007/s11227-020-03410-y.

[14] L. N. Tidjon, M. Frappier, e A. Mammar, «Intrusion Detection Systems: A Cross-Domain Overview», *IEEE Commun. Surv. Tutor.*, vol. 21, n. 4, pp. 3639–3681, 2019, doi: 10.1109/COMST.2019.2922584.

[15] H. Nachan, D. Poddar, S. Sarode, P. Kumhar, e S. Birla, «Intrusion Detection System: A Survey».

[16] Z. Q. Wang e D. K. Zhang, «HIDS and NIDS Hybrid Intrusion Detection System Model Design», *Adv. Eng. Forum*, vol. 6–7, pp. 991–994, Set. 2012, doi: 10.4028/www.scientific.net/AEF.6-7.991.

[17] D. Barber, *Bayesian reasoning and machine learning*. Cambridge ; New York: Cambridge University Press, 2012.
[18] A. Burkov, *The hundred-page machine learning book*. Polen: Andriy Burkov, 2019.
[19] A. V. Joshi, *Machine Learning and Artificial Intelligence*. Cham: Springer, 2020.
[20] J. D. Kelleher, B. Mac Namee, e A. D'Arcy, *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*, Second edition. Cambridge, Massachusetts: The MIT Press, 2020.
[21] S. Shalev-Shwartz e S. Ben-David, *Understanding machine learning: from theory to algorithms*. New York, NY, USA: Cambridge University Press, 2014.
[22] M. K. M. Nasution, «A method for constructing a dataset to reveal the industrial behaviour of big data», *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1003, p. 012156, Dez. 2020, doi: 10.1088/1757-899X/1003/1/012156.
[23] U. R. Hodeghatta e U. Nayak, «Supervised Machine Learning—Classification», em *Business Analytics Using R - A Practical Approach*, Berkeley, CA: Apress, 2017, pp. 131–160. doi: 10.1007/978-1-4842-2514-1_6.
[24] P. Harrington, *Machine learning in action*. Shelter Island, N.Y: Manning Publications Co, 2012.
[25] C. M. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.
[26] M. Jindal, J. Gupta, e B. Bhushan, «Machine learning methods for IoT and their Future Applications», em *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, Out. 2019, pp. 430–434. doi: 10.1109/ICCCIS48478.2019.8974551.
[27] I. H. Witten e I. H. Witten, Eds., *Data mining: practical machine learning tools and techniques*, Fourth Edition. Amsterdam: Elsevier, 2017.
[28] R. Panigrahi, S. Borah, A. K. Bhoi, e P. K. Mallick, «Intrusion Detection Systems (IDS)—An Overview with a Generalized Framework», em *Cognitive Informatics and Soft Computing*, vol. 1040, P. K. Mallick, V. E. Balas, A. K. Bhoi, e G.-S. Chae, Eds. Singapore: Springer Singapore, 2020, pp. 107–117. doi: 10.1007/978-981-15-1451-7_11.
[29] R. D. Ravipati e M. Abualkibash, «A SURVEY ON DIFFERENT MACHINE LEARNING ALGORITHMS AND WEAK CLASSIFIERS BASED ON KDD AND NSL-KDD DATASETS», *Int. J. Artif. Intell. Appl.*, vol. 10, n. 03, pp. 01–11, Mai. 2019, doi: 10.5121/ijaia.2019.10301.
[30] A. Jamali, «Evaluation and comparison of eight machine learning models in land use/land cover mapping using Landsat 8 OLI: a case study of the northern region of Iran», *SN Appl. Sci.*, vol. 1, n. 11, p. 1448, Nov. 2019, doi: 10.1007/s42452-019-1527-8.
[31] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, e A. Wahab, «A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions», *Electronics*, vol. 9, n. 7, p. 1177, Jul. 2020, doi: 10.3390/electronics9071177.
[32] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[33] M. W. Gardner e S. R. Dorling, «Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences», *Atmos. Environ.*, vol. 32, n. 14–15, pp. 2627–2636, Ago. 1998, doi: 10.1016/S1352-2310(97)00447-0.
[34] A. S. Shah, M. Shah, M. Fayaz, F. Wahid, H. K. Khan, e A. Shah, «Forensic Analysis of Offline Signatures Using Multilayer Perceptron and Random Forest», *Int. J. Database Theory Appl.*, vol. 10, n. 1, pp. 139–148, Jan. 2017, doi: 10.14257/ijdta.2017.10.1.13.

[35] T. Acharya, D. Lee, I. Yang, e J. Lee, «Identification of Water Bodies in a Landsat 8 OLI Image Using a J48 Decision Tree», *Sensors*, vol. 16, n. 7, p. 1075, Jul. 2016, doi: 10.3390/s16071075.

[36] A. Subudhi, M. Dash, e S. Sabut, «Automated segmentation and classification of brain stroke using expectation-maximization and random forest classifier», *Biocybern. Biomed. Eng.*, vol. 40, n. 1, pp. 277–289, Jan. 2020, doi: 10.1016/j.bbe.2019.04.004.

[37] V. F. Rodriguez-Galiano, B. Ghimire, J. Rogan, M. Chica-Olmo, e J. P. Rigol-Sanchez, «An assessment of the effectiveness of a random forest classifier for land-cover classification», *ISPRS J. Photogramm. Remote Sens.*, vol. 67, pp. 93–104, Jan. 2012, doi: 10.1016/j.isprsjprs.2011.11.002.

[38] I. Sharafaldin, A. Habibi Lashkari, e A. A. Ghorbani, «Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization»:, em *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, Funchal, Madeira, Portugal, 2018, pp. 108–116. doi: 10.5220/0006639801080116.

[39] G. E. A. P. A. Batista, R. C. Prati, e M. C. Monard, «A study of the behavior of several methods for balancing machine learning training data», *ACM SIGKDD Explor. Newsl.*, vol. 6, n. 1, pp. 20–29, Jun. 2004, doi: 10.1145/1007730.1007735.

[40] G. Hoang, A. Bouzerdoum, e S. Lam, «Learning Pattern Classification Tasks with Imbalanced Data Sets», em *Pattern Recognition*, P.-Y. Yin, Ed. InTech, 2009. doi: 10.5772/7544.

[41] E. Rendón, R. Alejo, C. Castorena, F. J. Isidro-Ortega, e E. E. Granda-Gutiérrez, «Data Sampling Methods to Deal With the Big Data Multi-Class Imbalance Problem», *Appl. Sci.*, vol. 10, n. 4, p. 1276, Fev. 2020, doi: 10.3390/app10041276.

[42] T. R. Hoens e N. V. Chawla, «Imbalanced Datasets: From Sampling to Classifiers», em *Imbalanced Learning*, H. He e Y. Ma, Eds. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2013, pp. 43–59. doi: 10.1002/9781118646106.ch3.

[43] S. Gnanambal, M. Thangaraj, V. T. Meenatchi, e V. Gayathri, «Classification algorithms with attribute selection: an evaluation study using WEKA», *Int. J. Adv. Netw. Appl.*, vol. 9, n. 6, pp. 3640–3644, 2018.

[44] T. C. Smith e E. Frank, «Introducing Machine Learning Concepts with WEKA», em *Statistical Genomics*, vol. 1418, E. Mathé e S. Davis, Eds. New York, NY: Springer New York, 2016, pp. 353–378. doi: 10.1007/978-1-4939-3578-9_17.

[45] P. Cichosz, *Data mining algorithms: explained using R*. Chichester, West Sussex, UK ; Malden, MA, USA: John Wiley & Sons Inc, 2015.

[46] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, e I. H. Witten, «The WEKA data mining software: an update», *ACM SIGKDD Explor. Newsl.*, vol. 11, n. 1, pp. 10–18, Nov. 2009, doi: 10.1145/1656274.1656278.

[47] A. G. Karegowda, A. S. Manjunath, e M. A. Jayaram, «Comparative study of attribute selection using gain ratio and correlation based feature selection», *Int. J. Inf. Technol. Knowl. Manag.*, vol. 2, n. 2, pp. 271–277, 2010.

[48] M. Usman, O. Owolabi, e A. Ajibola, «Feature Selection: It Importance in Performance Prediction», Jun. 2020.

[49] A. F. Jabbar e I. J. Mohammed, «Development of an Optimized Botnet Detection Framework based on Filters of Features and Machine Learning Classifiers using CICIDS2017 Dataset», *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 928, p. 032027, Nov. 2020, doi: 10.1088/1757-899X/928/3/032027.

[50] S. Kurnaz e M. E. Khudhur, «Comparative and Analysis Study for Malicious Executable by Using Various Classification Algorithms», 2018.

[51] A. Teixeira, Ed., *Computational processing of the Portuguese language: 8th international conference, PROPOR 2008, Aveiro, Portugal, September 8-10, 2008 ; proceedings*. Berlin Heidelberg: Springer, 2008.

[52] R. Sadeghi, R. Zarkami, K. Sabetraftar, e P. Van Damme, «Application of genetic algorithm and greedy stepwise to select input variables in classification tree models for the prediction of habitat requirements of Azolla filiculoides (Lam.) in Anzali wetland, Iran», *Ecol. Model.*, vol. 251, pp. 44–53, Fev. 2013, doi: 10.1016/j.ecolmodel.2012.12.010.

[53] J. T. Martínez Garre, M. Gil Pérez, e A. Ruiz-Martínez, «A novel Machine Learning-based approach for the detection of SSH botnet infection», *Future Gener. Comput. Syst.*, vol. 115, pp. 387–396, Fev. 2021, doi: 10.1016/j.future.2020.09.004.

[54] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, e A. Hotho, «A survey of network-based intrusion detection data sets», *Comput. Secur.*, vol. 86, pp. 147–167, Set. 2019, doi: 10.1016/j.cose.2019.06.005.

[55] D. B. Han e P. Jha, «Malmenator - Network Anomaly Detection». University of Hong Kong, Mai. 04, 2020.

[56] M. Tavallaee, E. Bagheri, W. Lu, e A. A. Ghorbani, «A detailed analysis of the KDD CUP 99 data set», em *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Jul. 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.

[57] N. Moustafa e J. Slay, «UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)», em *2015 Military Communications and Information Systems Conference (MilCIS)*, Nov. 2015, pp. 1–6. doi: 10.1109/MilCIS.2015.7348942.

[58] J. Carneiro, N. Oliveira, N. Sousa, E. Maia, e I. Praça, «Machine Learning for Network-based Intrusion Detection Systems: an Analysis of the CIDDS-001 Dataset», *ArXiv210702753 Cs*, Jul. 2021, Acedido: Ago. 30, 2021. [Em linha]. Disponível em: http://arxiv.org/abs/2107.02753

[59] «IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB». https://www.unb.ca/cic/datasets/ids-2018.html (acedido Jul. 25, 2021).

[60] P. Lin, K. Ye, e C.-Z. Xu, «Dynamic Network Anomaly Detection System by Using Deep Learning Techniques», em *Cloud Computing – CLOUD 2019*, vol. 11513, D. Da Silva, Q. Wang, e L.-J. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 161–176. doi: 10.1007/978-3-030-23502-4_12.

[61] A. L. G. Rios, Z. Li, K. Bekshentayeva, e L. Trajkovic, «Detection of Denial of Service Attacks in Communication Networks», em *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, Seville, Spain, Out. 2020, pp. 1–5. doi: 10.1109/ISCAS45731.2020.9180445.

[62] D. Ravikumar, *Towards Enhancement of Machine Learning Techniques Using CSE-CIC-IDS2018 Cybersecurity Dataset*. Rochester Institute of Technology, 2021.

[63] M. Gniewkowski, «An Overview of DoS and DDoS Attack Detection Techniques», em *Theory and Applications of Dependable Computer Systems*, vol. 1173, W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, e J. Kacprzyk, Eds. Cham: Springer International Publishing, 2020, pp. 233–241. doi: 10.1007/978-3-030-48256-5_23.

[64] D. B. Han, P. Jha, e D. D. Schnieders, «Network Based Intelligent Malware Detection». University of Hong Kong.

[65] L. Liu, P. Wang, J. Lin, e L. Liu, «Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning and Deep Learning», *IEEE Access*, vol. 9, pp. 7550–7563, 2021, doi: 10.1109/ACCESS.2020.3048198.

[66] A. Delplace, S. Hermoso, e K. Anandita, «Cyber Attack Detection thanks to Machine Learning Algorithms», *ArXiv200106309 Cs Stat*, Jan. 2020, Acedido: Set. 17, 2021. [Em linha]. Disponível em: http://arxiv.org/abs/2001.06309

[67] C. Wressnegger, «Efficient machine learning for attack detection», *It - Inf. Technol.*, vol. 62, n. 5–6, pp. 279–286, Dez. 2020, doi: 10.1515/itit-2020-0015.

[68] S. Tyagi e S. Mittal, «Sampling Approaches for Imbalanced Data Classification Problem in Machine Learning», em *Proceedings of ICRIC 2019*, vol. 597, P. K. Singh, A. K. Kar, Y. Singh, M. H. Kolekar, e S. Tanwar, Eds. Cham: Springer International Publishing, 2020, pp. 209–221. doi: 10.1007/978-3-030-29407-6_17.

[69] R. B. Basnet, R. Shash, C. Johnson, L. Walgren, e T. Doleck, «Towards Detecting and Classifying Network Intrusion Traffic Using Deep Learning Frameworks.», *J Internet Serv Inf Secur*, vol. 9, n. 4, pp. 1–17, 2019.

[70] Y. Hua, «An Efficient Traffic Classification Scheme Using Embedded Feature Selection and LightGBM», em *2020 Information Communication Technologies Conference (ICTC)*, Nanjing, China, Mai. 2020, pp. 125–130. doi: 10.1109/ICTC49638.2020.9123302.

[71] Q. R. S. Fitni e K. Ramli, «Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems», em *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, Bali, Indonesia, Jul. 2020, pp. 118–124. doi: 10.1109/IAICT50021.2020.9172014.

[72] J. L. Leevy, J. Hancock, R. Zuech, e T. M. Khoshgoftaar, «Detecting cybersecurity attacks across different network features and learners», *J. Big Data*, vol. 8, n. 1, p. 38, Dez. 2021, doi: 10.1186/s40537-021-00426-w.

[73] R. Zuech, J. Hancock, e T. M. Khoshgoftaar, «Detecting web attacks using random undersampling and ensemble learners», *J. Big Data*, vol. 8, n. 1, p. 75, Dez. 2021, doi: 10.1186/s40537-021-00460-8.

[74] M. A. Khan e J. Kim, «Toward Developing Efficient Conv-AE-Based Intrusion Detection System Using Heterogeneous Dataset», *Electronics*, vol. 9, n. 11, p. 1771, Out. 2020, doi: 10.3390/electronics9111771.

[75] M. Sarhan, S. Layeghy, N. Moustafa, e M. Portmann, «NetFlow Datasets for Machine Learning-based Network Intrusion Detection Systems», *ArXiv201109144 Cs*, vol. 371, pp. 117–135, 2021, doi: 10.1007/978-3-030-72802-1_9.

[76] L. D'hooge, T. Wauters, B. Volckaert, e F. De Turck, «Inter-dataset generalization strength of supervised machine learning methods for intrusion detection», *J. Inf. Secur. Appl.*, vol. 54, p. 102564, Out. 2020, doi: 10.1016/j.jisa.2020.102564.

[77] M. Soltani, M. J. Siavoshani, e A. H. Jahangir, «A content-based deep intrusion detection system», *Int. J. Inf. Secur.*, Set. 2021, doi: 10.1007/s10207-021-00567-2.

[78] P. Ferreira e M. Antunes, «Benchmarking bioinspired machine learning algorithms with CSE-CIC-IDS2018 network intrusions dataset», 2020.

[79] R. I. Farhan, A. T. Maolood, e N. F. Hassan, «Optimized Deep Learning with Binary PSO for Intrusion Detection on CSE-CIC-IDS2018 Dataset», *J. Al-Qadisiyah Comput. Sci. Math.*, vol. 12, n. 3, p. Page 16-27, 2020.

[80] D. B. Han, P. Jha, e D. D. Schnieders, «Malmenator Network Anomaly Detection». University of Hong Kong, Fev. 02, 2020.

Esta página foi intencionalmente deixada em branco

Anexo A

Tabela 1 - Atributos originais do conjunto de dados estudado (tabela adaptada de [59])

Nome atributo	Descrição
fl_dur	Duração do fluxo
tot_fw_pk	Pacotes totais na direção para frente
tot_bw_pk	Pacotes totais no sentido inverso
tot_l_fw_pkt	Tamanho total do pacote na direção para frente
fw_pkt_l_max	Tamanho máximo do pacote na direção para frente
fw_pkt_l_min	Tamanho mínimo do pacote na direção para frente
fw_pkt_l_avg	Tamanho médio do pacote na direção para frente
fw_pkt_l_std	Tamanho do desvio padrão do pacote na direção para frente
Bw_pkt_l_max	Tamanho máximo do pacote na direção para trás
Bw_pkt_l_min	Tamanho mínimo do pacote na direção para trás
Bw_pkt_l_avg	Tamanho médio do pacote na direção inversa
Bw_pkt_l_std	Tamanho do desvio padrão do pacote na direção para trás
fl_byt_s	Taxa de bytes de fluxo que é o número de pacotes transferidos por segundo
fl_pkt_s	Taxa de pacotes de fluxo que é o número de pacotes transferidos por segundo
fl_iat_avg	Tempo médio entre dois fluxos
fl_iat_std	Desvio padrão de tempo de dois fluxos
fl_iat_max	Tempo máximo entre dois fluxos
fl_iat_min	Tempo mínimo entre dois fluxos
fw_iat_tot	Tempo total entre dois pacotes enviados na direção para frente
fw_iat_avg	Tempo médio entre dois pacotes enviados na direção para frente
fw_iat_std	Tempo de desvio padrão entre dois pacotes enviados na direção para frente
fw_iat_max	Tempo máximo entre dois pacotes enviados na direção para frente
fw_iat_min	Tempo mínimo entre dois pacotes enviados na direção para frente

bw_iat_tot	Tempo total entre dois pacotes enviados na direção inversa
bw_iat_avg	Tempo médio entre dois pacotes enviados na direção inversa
bw_iat_std	Tempo de desvio padrão entre dois pacotes enviados na direção para trás
bw_iat_max	Tempo máximo entre dois pacotes enviados na direção inversa
bw_iat_min	Tempo mínimo entre dois pacotes enviados no sentido inverso
fw_psh_flag	Número de vezes que a bandeira PSH foi definida em pacotes que viajam na direção para frente (0 para UDP)
bw_psh_flag	Número de vezes que o sinalizador PSH foi definido em pacotes que viajam na direção para trás (0 para UDP)
fw_urg_flag	Número de vezes que a bandeira URG foi definida em pacotes que viajam na direção para frente (0 para UDP)
bw_urg_flag	Número de vezes que a bandeira URG foi definida em pacotes que viajam na direção para trás (0 para UDP)
fw_hdr_len	Total de bytes usados para cabeçalhos na direção para frente
bw_hdr_len	Total de bytes usados para cabeçalhos na direção para frente
fw_pkt_s	Número de pacotes encaminhados por segundo
bw_pkt_s	Número de pacotes para trás por segundo
pkt_len_min	Comprimento mínimo de um fluxo
pkt_len_max	Comprimento máximo de um fluxo
pkt_len_avg	Comprimento médio de um fluxo
pkt_len_std	Comprimento do desvio padrão de um fluxo
pkt_len_va	Tempo mínimo entre chegada do pacote
fin_cnt	Número de pacotes com FIN
syn_cnt	Número de pacotes com SYN
rst_cnt	Número de pacotes com RST
pst_cnt	Número de pacotes com PUSH
ack_cnt	Número de pacotes com ACK
urg_cnt	Número de pacotes com URG
cwe_cnt	Número de pacotes com CWE
ece_cnt	Número de pacotes com ECE
down_up_ratio	Taxa de download e upload

pkt_size_avg	Tamanho médio do pacote
fw_seg_avg	Tamanho médio observado na direção para frente
bw_seg_avg	Tamanho médio observado na direção inversa
fw_byt_blk_avg	Taxa média do número de bytes em massa na direção para frente
fw_pkt_blk_avg	Taxa média do número de pacotes em massa na direção direta
fw_blk_rate_avg	Número médio de taxa em massa na direção para frente
bw_byt_blk_avg	Taxa média do número de bytes em massa no sentido inverso
bw_pkt_blk_avg	Taxa média do número de pacotes em massa na direção para trás
bw_blk_rate_avg	Número médio de taxa em massa na direção inversa
subfl_fw_pk	O número médio de pacotes em um subfluxo na direção para frente
subfl_fw_byt	O número médio de bytes em um subfluxo na direção para frente
subfl_bw_pkt	O número médio de pacotes em um subfluxo na direção para trás
subfl_bw_byt	O número médio de bytes em um subfluxo na direção para trás
fw_win_byt	Número de bytes enviados na janela inicial na direção para frente
bw_win_byt	# de bytes enviados na janela inicial na direção para trás
Fw_act_pkt	Nº de pacotes com pelo menos 1 byte de carga útil de dados TCP na direção para frente
fw_seg_min	Tamanho mínimo do segmento observado na direção para frente
atv_avg	Tempo médio em que um fluxo estava ativo antes de ficar ocioso
atv_std	Tempo de desvio padrão em que um fluxo estava ativo antes de ficar ocioso
atv_max	Tempo máximo em que um fluxo esteve ativo antes de se tornar ocioso
atv_min	Tempo mínimo em que um fluxo esteve ativo antes de se tornar ocioso
idl_avg	Tempo médio em que um fluxo ficou ocioso antes de se tornar ativo
idl_std	Tempo de desvio padrão em que um fluxo estava ocioso antes de se tornar ativo
idl_max	Tempo máximo em que um fluxo ficou ocioso antes de se tornar ativo
idl_min	Tempo mínimo em que um fluxo ficou ocioso antes de se tornar ativo

Esta página foi intencionalmente deixada em branco

Anexo B

Tabela 1 - Resultados de Acurácia em *undersampling*

Accuracy	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	66.8042 %	65.0446 %	68.2275 %	73.0577 %	77.4423 %	79.0753 %	80.1425 %	79.7963%
Ibk B=TotalA	92.8529 %	92.7272 %	91.5343 %	91.7708 %	92.7952 %	92.1332%	91.3537%	89.6346 %
Multiperceptron B=ARef	64.420%	65.4869%	70.6290%	72.3076%	77.0349%	73.2958%	73.1885%	69.0182%
Multiperceptron B=TotalA	91.9884%	92.2077%	92.6395%	94.2029%	95.1642%	95.8368%	95.9429%	96.0678%
Random Forest B=ARef	64.831%	63.483%	67.232%	76.226%	79.875%	81.299%	84.029%	82.370%
Random Forest B=TotalA	95.3964%	95.3335%	95.4661%	95.5362%	95.5677%	95.6607%	94.9735%	93.3271%

Tabela 2 - Resultados de Precisão em *undersampling*

Precisão	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	94.6000%	94.5000%	94.4000%	95.8000%	95.7000%	96.1000%	96.2000%	96.2949%
Ibk B=TotalA	95.3254%	95.6041%	95.6568%	95.3468%	95.6612%	95.7320%	95.7652%	95.7769%
Multiperceptron B=ARef	94.018%	94.1337%	94.4664%	95.3838%	95.4836%	95.7232%	95.9922%	96.2669%
Multiperceptron B=TotalA	95.1744%	95.3182%	95.3361%	95.4271%	95.8147%	95.9806%	95.8447%	95.8273%
Random Forest B=ARef	93.976%	94.188%	93.882%	95.274%	95.691%	96.005%	96.355%	96.557%
Random Forest B=TotalA	94.8597%	94.9456%	95.0656%	95.3275%	95.5079%	95.7591%	95.7776%	95.7802%

Tabela 3 - Resultados de Revocação em *undersampling*

Revocação	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	66.8000%	65.0000%	68.2000%	73.1000%	77.4000%	79.1000%	80.1000%	79.7963%
Ibk B=TotalA	92.8529%	92.7272%	91.5343%	91.7708%	92.7952%	92.1332%	91.4000%	89.6346%
Multiperceptron B=ARef	64.420%	65.4869%	67.4751%	72.3076%	77.0349%	73.2958%	73.1885%	69.0182%
Multiperceptron B=TotalA	91.9884%	92.2077%	92.6395%	94.2029%	95.1642%	95.8368%	95.9429%	96.0746%
Random Forest B=ARef	64.831%	63.483%	67.232%	76.226%	79.875%	81.299%	84.029%	82.370%
Random Forest B=TotalA	95.3964%	95.3335%	95.4661%	95.5362%	95.5677%	95.6607%	94.9735%	93.3271%

Tabela 4 - Resultados de F-Measure em *undersampling*

F - Measure	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	76.7000%	75.3000%	77.6000%	81.7000%	84.7000%	85.9000%	86.6000%	86.3919%
Ibk B=TotalA	94.0000%	94.1000%	93.4182%	93.4135%	94.1012%	93.7529%	93.3188%	92.3389%
Multiperceptron B=ARef	74.182%	75.0276%	76.3788%	80.3814%	83.8148%	81.2079%	81.1973%	78.2449%
Multiperceptron B=TotalA	93.4686%	93.6636%	93.8985%	94.7633%	95.4259%	95.8790%	95.8368%	95.9389%
Random Forest B=ARef	75.053%	74.017%	76.956%	83.855%	86.335%	87.291%	89.052%	88.046%
Random Forest B=TotalA	95.0778%	95.1116%	95.2369%	95.4076%	95.5239%	95.7032%	95.3548%	94.4451%

Tabela 5 - Resultados de Acurácia em *oversampling*

Acurácia	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	66.8042%	65.0446%	68.2275%	73.0577%	77.4422%	79.0752%	80.1424%	79.7963%
Ibk B=TotalA	92.8529%	92.7877%	92.4603%	93.9158%	94.1603%	93.5383%	93.4837%	92.8047%
Multiperceptron B=ARef	66.8708%	65.4869%	67.4751%	72.3076%	77.0349%	73.2958%	73.1885%	69.0182%
Multiperceptron B=TotalA	91.9884%	92.2272%	92.9458%	95.2200%	95.6895%	95.8888%	96.3082%	96.1136%
Random Forest B=ARef	64.7008%	62.9421%	66.9033%	75.6438%	80.2881%	81.2993%	84.0287%	82.3700%
Random Forest B=TotalA	95.3964%	95.3260%	95.6473%	96.1012%	96.1681%	96.0585%	95.7413%	94.8580%

Tabela 6 - Resultados de Precisão em *oversampling*

Precisão	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	94.6000%	94.5000%	94.4000%	95.8000%	95.7000%	96.1000%	96.2000%	96.3000%
Ibk B=TotalA	95.3000%	95.6000%	95.2000%	95.2000%	95.5000%	95.8000%	95.9000%	95.9000%
Multiperceptron B=ARef	93.9270%	94.1337%	94.4664%	95.3838%	95.4836%	95.7232%	95.9922%	96.2669%
Multiperceptron B=TotalA	95.1744%	95.2368%	94.6528%	95.4980%	95.7169%	95.9370%	95.9962%	95.8991%
Random Forest B=ARef	93.9822%	94.1566%	93.8968%	95.4029%	95.7200%	96.0049%	96.3545%	96.5572%
Random Forest B=TotalA	94.8597%	94.9462%	94.9806%	95.3691%	95.5912%	95.8500%	95.8834%	95.8767%

Tabela 7 - Resultados de Revocação em *oversampling*

Revocação	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	66.8000%	65.0000%	68.2000%	73.1000%	77.4000%	79.1000%	80.1000%	79.8000%
Ibk B=TotalA	92.9000%	92.8000%	92.5000%	93.9000%	94.2000%	93.5000%	93.5000%	92.8000%
Multiperceptron B=ARef	66.8708%	65.4869%	67.4751%	72.3076%	77.0349%	73.2958%	73.1885%	69.0182%
Multiperceptron B=TotalA	91.9884%	92.2272%	92.9458%	95.2200%	95.6895%	95.8888%	96.3082%	96.1117%
Random Forest B=ARef	64.7008%	62.9421%	66.9033%	75.6438%	80.2881%	81.2993%	84.0287%	82.3700%
Random Forest B=TotalA	95.3964%	95.3260%	95.6473%	96.1012%	96.1681%	96.0585%	95.7413%	94.8580%

Tabela 8 - Resultados de F-Measure em *oversampling*

F - Measure	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	76.7000%	75.3000%	77.6000%	81.7000%	84.7000%	85.9000%	86.6000%	86.4000%
Ibk B=TotalA	94.0000%	94.1000%	93.8000%	94.5000%	94.8000%	94.6000%	94.6000%	94.2000%
Multiperceptron B=ARef	75.9500%	75.0276%	76.3788%	80.3814%	83.8148%	81.2079%	81.1973%	78.2449%
Multiperceptron B=TotalA	93.4686%	93.6389%	93.5921%	95.2418%	95.5746%	95.8775%	96.1141%	95.9713%
Random Forest B=ARef	74.9508%	73.5471%	76.7061%	83.5047%	86.6099%	87.2909%	89.0522%	88.0458%
Random Forest B=TotalA	95.0778%	95.1043%	95.2712%	95.6594%	95.8268%	95.9377%	95.8083%	95.3358%

Tabela 9 - Resultados da Acurácia do *CorrelationAtributeEval* em *undersampling*

Acurácia	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	71.0315%	67.6368%	65.6496%	74.5144%	78.4023%	78.9720%	79.9932%	79.6964%
Ibk B=TotalA	92.2927%	92.3298%	92.0941%	92.6991%	92.7899%	92.2132%	91.3628%	89.6839%
Multiperceptron B=ARef	64.9692%	63.9989%	62.8054%	73.6649%	78.1050%	75.1784%	74.1971%	70.2803%
Multiperceptron B=TotalA	92.6018%	92.8384%	93.7098%	94.8595%	95.6701%	96.0318%	95.3403%	95.4715%
Random Forest B=ARef	71.7918%	68.4299%	72.0456%	75.2539%	79.8283%	81.5884%	82.9406%	81.1397%
Random Forest B=TotalA	95.4536%	95.1736%	95.4446%	95.5579%	95.6022%	95.0785%	93.8426%	91.8920%

Tabela 10 - Resultados de Precisão do *CorrelationAtributeEval* em *undersampling*

Precisão	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	94.8848%	94.9454%	94.5461%	95.7729%	95.8619%	96.1406%	96.2305%	96.3387%
Ibk B=TotalA	95.4339%	95.4872%	95.6654%	95.4637%	95.6658%	95.7358%	95.8048%	95.8028%
Multiperceptron B=ARef	93.6469%	93.9348%	94.0897%	95.2038%	95.6497%	95.7873%	96.0862%	96.0629%
Multiperceptron B=TotalA	95.2482%	95.3945%	95.3739%	95.4407%	95.7394%	95.9481%	95.7832%	95.7676%
Random Forest B=ARef	93.6840%	93.9450%	95.3806%	96.0294%	95.9925%	96.1815%	96.4510%	96.5543%
Random Forest B=TotalA	95.4734%	95.4832%	95.5231%	95.4974%	95.6412%	95.7974%	95.8063%	95.8042%

Tabela 11 - Resultados de Revocação do *CorrelationAtributeEval* em *undersampling*

Revocação	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	71.0315%	67.6368%	65.6496%	74.5144%	78.4023%	78.9720%	79.9932%	79.6964%
Ibk B=TotalA	92.2927%	92.3298%	92.0941%	92.6991%	92.7899%	92.2132%	91.3628%	89.6839%
Multiperceptron B=ARef	64.9692%	63.9989%	62.8054%	73.6649%	78.1050%	75.1784%	74.1971%	70.2803%
Multiperceptron B=TotalA	92.6018%	92.8384%	93.7098%	94.8595%	95.6701%	96.0318%	95.3403%	95.4715%
Random Forest B=ARef	71.7918%	68.4299%	72.0456%	75.2539%	79.8283%	81.5884%	82.9406%	81.1397%
Random Forest B=TotalA	95.4536%	95.1736%	95.4446%	95.5579%	95.6022%	95.0785%	93.8426%	91.8920%

Tabela 12 - Resultados de F-Measure do *CorrelationAtributeEval* em *undersampling*

F - Measure	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	79.9597%	77.4739%	75.7912%	82.7150%	85.3952%	85.8277%	86.5105%	86.3353%
Ibk B=TotalA	93.7494%	93.8143%	93.7394%	93.9836%	94.0997%	93.8010%	93.3384%	92.3734%
Multiperceptron B=ARef	74.3773%	73.4526%	72.4798%	81.3240%	84.7833%	82.5564%	81.8542%	79.2108%
Multiperceptron B=TotalA	93.7496%	94.0272%	94.4852%	95.0221%	95.6345%	95.9644%	95.4959%	95.5707%
Random Forest B=ARef	79.5996%	77.2792%	80.9351%	83.3068%	86.3464%	87.4881%	88.4002%	87.2790%
Random Forest B=TotalA	95.4232%	95.3150%	95.4700%	95.5139%	95.6123%	95.4197%	94.7449%	93.6428%

Tabela 13 - Resultados da Acurácia do *CorrelationAtributeEval* em *oversampling*

Acurácia	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	71.0315%	67.6368%	65.6496%	74.5144%	78.4023%	78.9720%	79.9932%	79.6964%
Ibk B=TotalA	92.2927%	92.3297%	92.0941%	92.6991%	93.2814%	93.6599%	93.4878%	92.9137%
Multiperceptron B=ARef	65.0718%	63.9989%	62.8054%	73.6649%	78.1050%	75.1784%	74.1971%	70.2803%
Multiperceptron B=TotalA	92.6018%	93.0067%	93.6727%	94.5840%	95.6083%	95.9533%	96.1335%	96.1687%
Random Forest B=ARef	72.6029%	68.4299%	72.0456%	75.2539%	79.8283%	81.5884%	82.9406%	81.1397%
Random Forest B=TotalA	95.4536%	95.1515%	95.5403%	95.5664%	95.7529%	95.6333%	95.1052%	94.1989%

Tabela 14 - Resultados de Precisão do *CorrelationAtributeEval* em *oversampling*

Precisão	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	94.8848%	94.9454%	94.5461%	95.7729%	95.8619%	96.1406%	96.2305%	96.3387%
Ibk B=TotalA	95.4339%	95.4872%	95.6654%	95.4637%	95.6895%	95.8127%	95.9133%	95.9122%
Multiperceptron B=ARef	93.7167%	93.9348%	94.0897%	95.2038%	95.6497%	95.7873%	96.0862%	96.0629%
Multiperceptron B=TotalA	95.2482%	95.4341%	95.2747%	95.4028%	95.7715%	95.8707%	95.9023%	95.8357%
Random Forest B=ARef	93.8133%	93.9450%	95.3806%	96.0294%	95.9925%	96.1815%	96.4510%	96.5543%
Random Forest B=TotalA	95.4734%	95.4913%	95.5095%	95.4984%	95.6864%	95.8836%	95.9225%	95.9132%

Tabela 15 - Resultados de Revocação do *CorrelationAtributeEval* em *oversampling*

Revocação	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	71.0315%	67.6368%	65.6496%	74.5144%	78.4023%	78.9720%	79.9932%	79.6964%
Ibk B=TotalA	92.2927%	92.3297%	92.0941%	92.6991%	93.2814%	93.6599%	93.4878%	92.9137%
Multiperceptron B=ARef	65.0718%	63.9989%	62.8054%	73.6649%	78.1050%	75.1784%	74.1971%	70.2803%
Multiperceptron B=TotalA	92.6018%	93.0067%	93.6727%	94.5840%	95.6083%	95.9533%	96.1335%	96.1687%
Random Forest B=ARef	72.6029%	68.4299%	72.0456%	75.2539%	79.8283%	81.5884%	82.9406%	81.1397%
Random Forest B=TotalA	95.4536%	95.1515%	95.5403%	95.5664%	95.7529%	95.6333%	95.1052%	94.1989%

Tabela 16 - Resultados de F-Measure do *CorrelationAtributeEval* em *oversampling*

F - Measure	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	79.9597%	77.4739%	75.7912%	82.7150%	85.3952%	85.8277%	86.5105%	86.3353%
Ibk B=TotalA	93.7494%	93.8143%	93.7394%	93.9836%	94.3843%	94.6435%	94.5833%	94.2581%
Multiperceptron B=ARef	74.3022%	73.4526%	72.4798%	81.3240%	84.7833%	82.5564%	81.8542%	79.2108%
Multiperceptron B=TotalA	93.7496%	94.1202%	94.3706%	94.8478%	95.6499%	95.8803%	95.9810%	95.9604%
Random Forest B=ARef	80.2631%	77.2792%	80.9351%	83.3068%	86.3464%	87.4881%	88.4002%	87.2790%
Random Forest B=TotalA	95.4232%	95.3072%	95.5074%	95.5194%	95.7092%	95.7524%	95.4905%	94.9854%

Tabela 17 - Resultados da Acurácia do *InfoGainAtributeEval* em *undersampling*

Acurácia	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	63.0401%	60.4992%	62.5972%	67.9896%	74.4977%	75.0719%	77.1395%	76.9221%
Ibk B=TotalA	90.6853%	90.1842%	90.0404%	91.4076%	91.8538%	91.1710%	90.8209%	89.3044%
Multiperceptron B=ARef	41.3737%	44.4775%	49.2196%	60.1440%	64.0699%	65.0319%	63.5810%	57.6923%
Multiperceptron B=TotalA	88.7663%	88.5420%	89.2255%	90.0159%	91.0519%	91.1936%	92.1959%	93.5275%
Random Forest B=ARef	66.0125%	63.3009%	64.4148%	72.5862%	78.4204%	79.8776%	81.4919%	79.4759%
Random Forest B=TotalA	94.3191%	94.0234%	94.3459%	95.3989%	95.2640%	95.1659%	94.1936%	92.7456%

Tabela 18 - Resultados de Precisão do *InfoGainAtributeEval* em *undersampling*

Precisão	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	92.6598%	92.9603%	92.9878%	93.8958%	94.9727%	95.2581%	95.6025%	95.8656%
Ibk B=TotalA	93.9315%	94.0764%	94.3074%	94.8326%	95.1437%	95.2912%	95.5098%	95.6071%
Multiperceptron B=ARef	90.9806%	90.8017%	90.7498%	91.2929%	92.0239%	92.2469%	92.4113%	91.9942%
Multiperceptron B=TotalA	90.2302%	90.7481%	91.7599%	91.9926%	92.6431%	92.7622%	93.4377%	93.9934%
Random Forest B=ARef	93.0985%	93.2271%	93.2979%	94.3414%	95.2859%	95.6186%	96.0378%	96.3086%
Random Forest B=TotalA	94.4183%	94.4846%	94.5917%	95.1787%	95.3785%	95.6415%	95.7024%	95.7870%

Tabela 19 - Resultados de Revocação do *InfoGainAtributeEval l* em *undersampling*

Revocação	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	63.0401%	60.4992%	62.5972%	67.9896%	74.4977%	75.0719%	77.1395%	76.9221%
Ibk B=TotalA	90.6853%	90.1842%	90.0404%	91.4076%	91.8538%	91.1710%	90.8209%	89.3044%
Multiperceptron B=ARef	41.3737%	44.4775%	49.2196%	60.1440%	64.0699%	65.0319%	63.5810%	57.6923%
Multiperceptron B=TotalA	88.7663%	88.5420%	89.2255%	90.0159%	91.0519%	91.1936%	92.1959%	93.5275%
Random Forest B=ARef	66.0125%	63.3009%	64.4148%	72.5862%	78.4204%	79.8776%	81.4919%	79.4759%
Random Forest B=TotalA	94.3191%	94.0234%	94.3459%	95.3989%	95.2640%	95.1659%	94.1936%	92.7456%

Tabela 20 - Resultados de F-Measure do *InfoGainAtributeEval* em *undersampling*

F - Measure	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	73.1047%	71.1477%	72.8002%	77.4416%	82.5016%	82.9880%	84.4689%	84.4021%
Ibk B=TotalA	92.1876%	91.9613%	92.0044%	92.9986%	93.3668%	93.0283%	92.9167%	92.0913%
Multiperceptron B=ARef	50.9602%	54.4089%	59.2551%	69.3090%	72.8003%	73.5999%	72.1843%	67.4432%
Multiperceptron B=TotalA	89.1949%	89.2260%	89.9729%	90.4291%	91.3939%	91.4921%	92.5459%	93.5741%
Random Forest B=ARef	75.9864%	73.9520%	74.7263%	81.1651%	85.2767%	86.2919%	87.4031%	86.1592%
Random Forest B=TotalA	94.3536%	94.2417%	94.4560%	95.2673%	95.3114%	95.3915%	94.8937%	94.1202%

Tabela 21 - Resultados da Acurácia do *InfoGainAtributeEval* em *oversampling*

Acurácia	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	63.0401%	60.4992%	62.5972%	67.9896%	74.4977%	75.0719%	77.1395%	76.9221%
Ibk B=TotalA	90.6853%	90.1835%	90.0404%	91.4076%	92.5178%	93.1703%	93.1558%	92.7086%
Multiperceptron B=ARef	47.3537%	44.4775%	49.2196%	60.1440%	64.0699%	65.0319%	63.5810%	57.6923%
Multiperceptron B=TotalA	88.7663%	87.8747%	88.7052%	90.3379%	91.1362%	92.1852%	93.1841%	93.8752%
Random Forest B=ARef	65.5926%	63.3009%	64.4148%	72.5862%	78.4204%	79.8776%	81.4919%	79.4759%
Random Forest B=TotalA	94.3191%	94.0178%	94.5594%	95.3808%	95.5388%	95.8697%	95.4509%	94.8445%

Tabela 22 - Resultados de Precisão do *InfoGainAtributeEval* em *oversampling*

Precisão	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	92.6598%	92.9603%	92.9878%	93.8958%	94.9727%	95.2581%	95.6025%	95.8656%
Ibk B=TotalA	93.9315%	94.0763%	94.3074%	94.8000%	95.2158%	95.4430%	95.6327%	95.7181%
Multiperceptron B=ARef	90.6502%	90.8017%	90.7498%	91.2929%	92.0239%	92.2469%	92.4113%	91.9942%
Multiperceptron B=TotalA	90.2302%	91.0905%	91.4906%	91.7606%	92.3593%	93.0720%	93.2899%	93.3730%
Random Forest B=ARef	93.0919%	93.2271%	93.2979%	94.3414%	95.2859%	95.6186%	96.0378%	96.3086%
Random Forest B=TotalA	94.4183%	94.4419%	94.6602%	95.2002%	95.4584%	95.7954%	95.8670%	95.9176%

Tabela 23 - Resultados de Revocação do *InfoGainAtributeEval* em *oversampling*

Revocação	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	63.0401%	60.4992%	62.5972%	67.9896%	74.4977%	75.0719%	77.1395%	76.9221%
Ibk B=TotalA	90.6853%	90.1835%	90.0404%	91.4000%	92.5178%	93.1703%	93.1558%	92.7086%
Multiperceptron B=ARef	47.3537%	44.4775%	49.2196%	60.1440%	64.0699%	65.0319%	63.5810%	57.6923%
Multiperceptron B=TotalA	88.7663%	87.8747%	88.7052%	90.3379%	91.1362%	92.1852%	93.1841%	93.8752%
Random Forest B=ARef	65.5926%	63.3009%	64.4148%	72.5862%	78.4204%	79.8776%	81.4919%	79.4759%
Random Forest B=TotalA	94.3191%	94.0178%	94.5594%	95.3808%	95.5388%	95.8697%	95.4509%	94.8445%

Tabela 24 - Resultados de F-Measure do *InfoGainAtributeEval* em *oversampling*

F - Measure	42	44	68	183	456	1384	7926	33125
Ibk B=ARef	73.1047%	71.1477%	72.8002%	77.4416%	82.5016%	82.9880%	84.4689%	84.4021%
Ibk B=TotalA	92.1876%	91.9609%	92.0045%	93.0000%	93.7675%	94.2158%	94.2866%	94.0712%
Multiperceptron B=ARef	57.4078%	54.4089%	59.2551%	69.3090%	72.8003%	73.5999%	72.1843%	67.4432%
Multiperceptron B=TotalA	89.1949%	88.9671%	89.6066%	90.4772%	91.3205%	92.2881%	93.0226%	93.2654%
Random Forest B=ARef	75.6600%	73.9520%	74.7263%	81.1651%	85.2767%	86.2919%	87.4031%	86.1592%
Random Forest B=TotalA	94.3536%	94.2171%	94.5968%	95.2708%	95.4855%	95.8227%	95.6498%	95.3448%

Anexo C

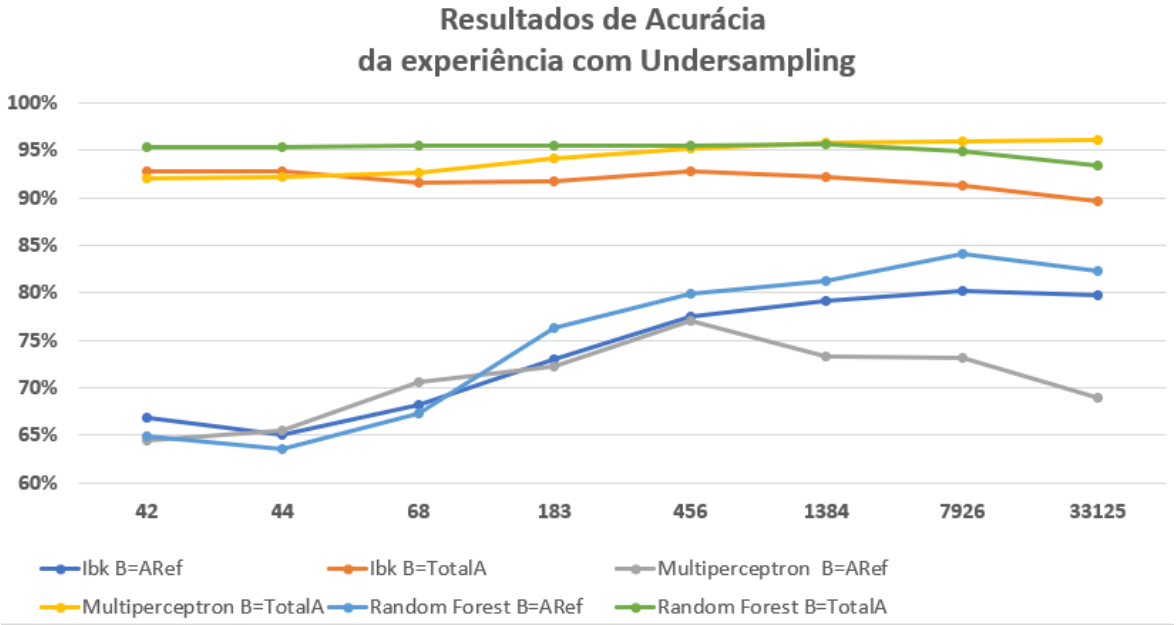

Gráfico 1

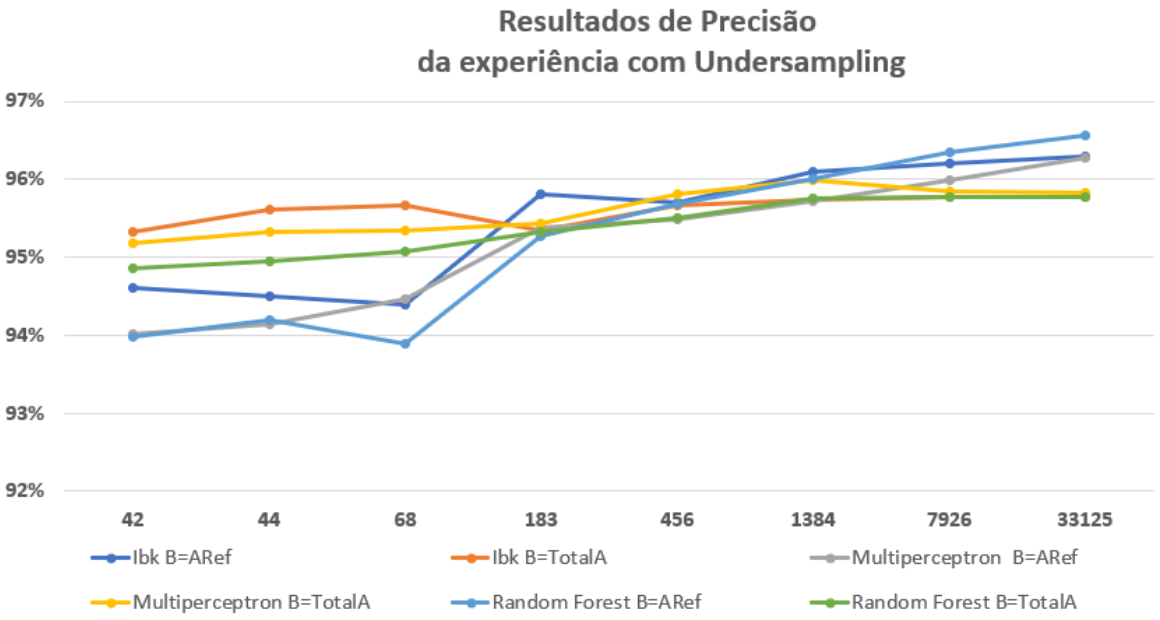

Gráfico 2

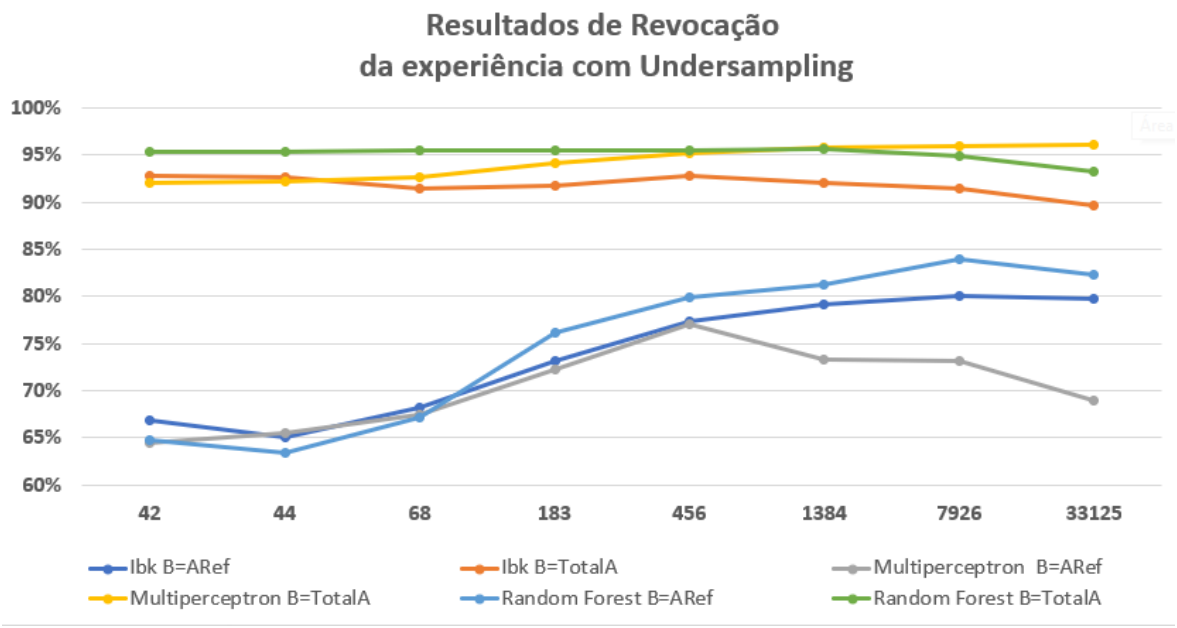

Gráfico 3

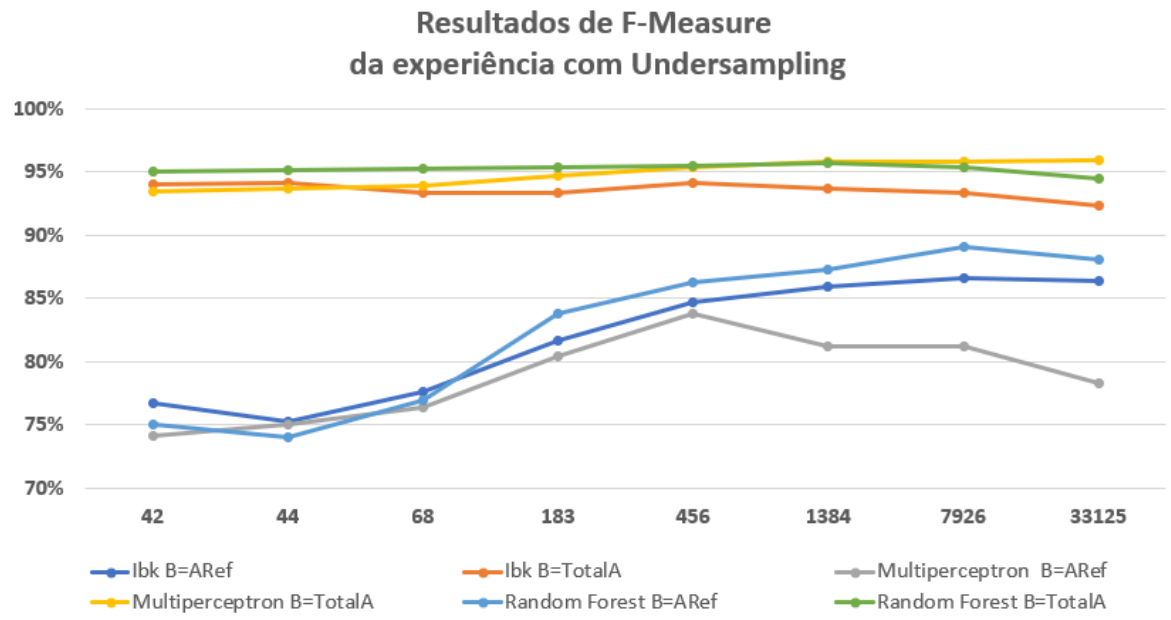

Gráfico 4

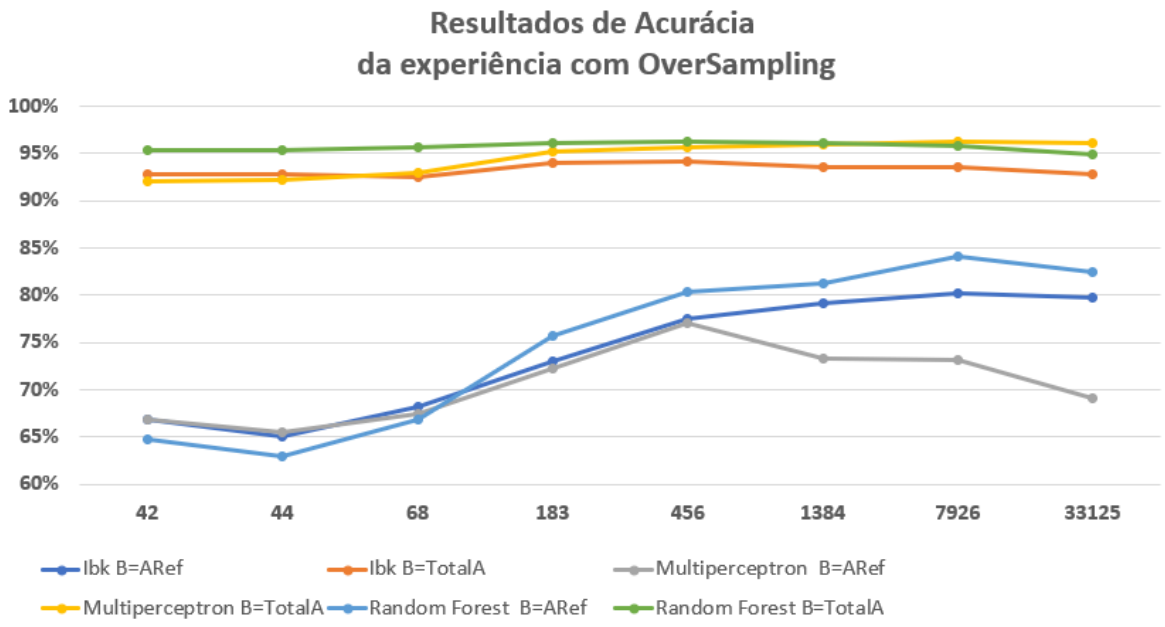

Gráfico 5

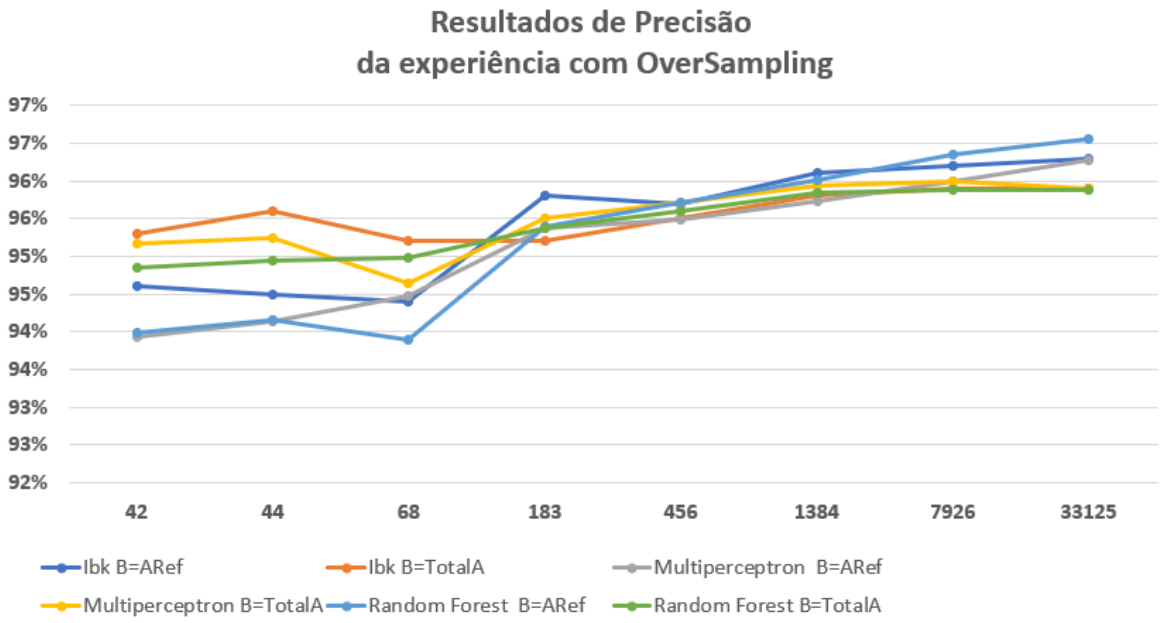

Gráfico 6

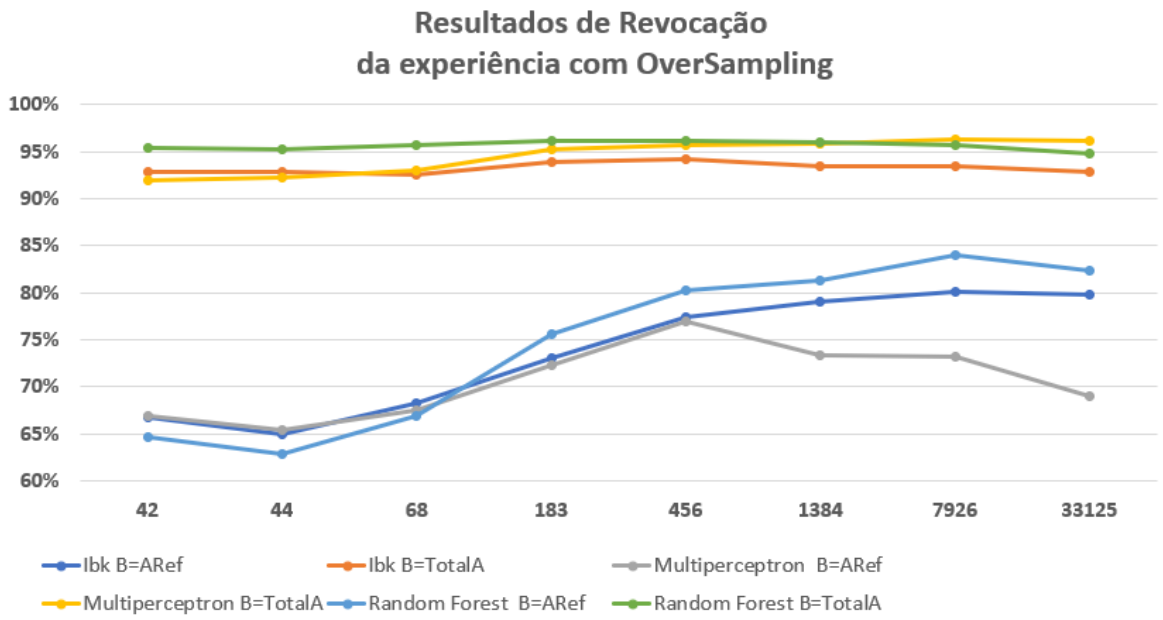

Gráfico 7

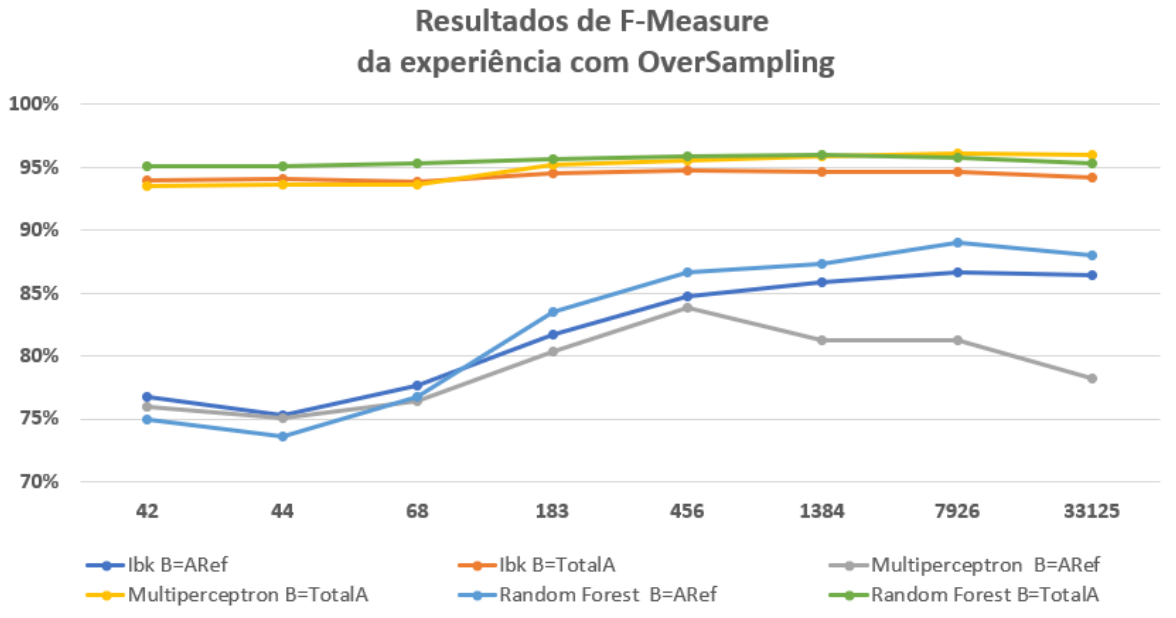

Gráfico 8

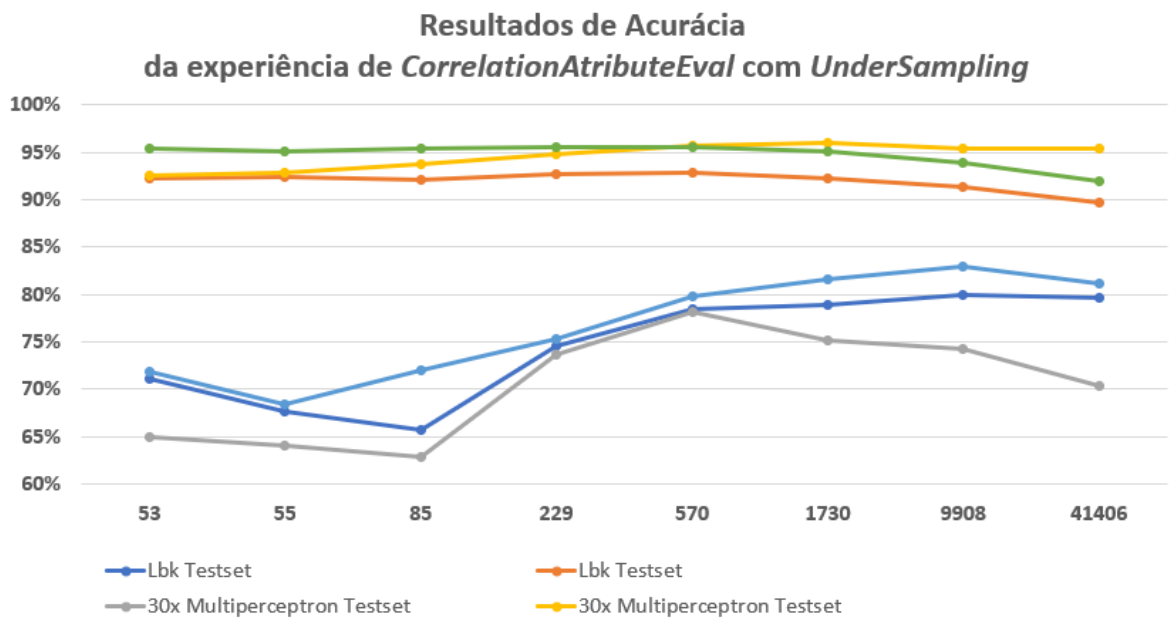

Gráfico 9

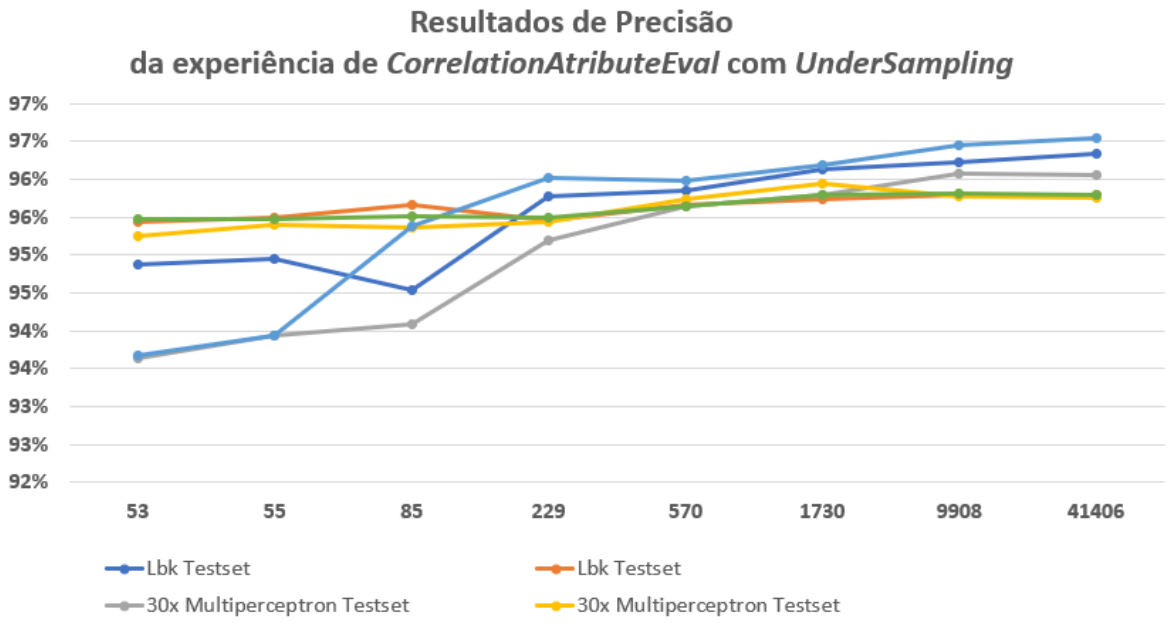

Gráfico 10

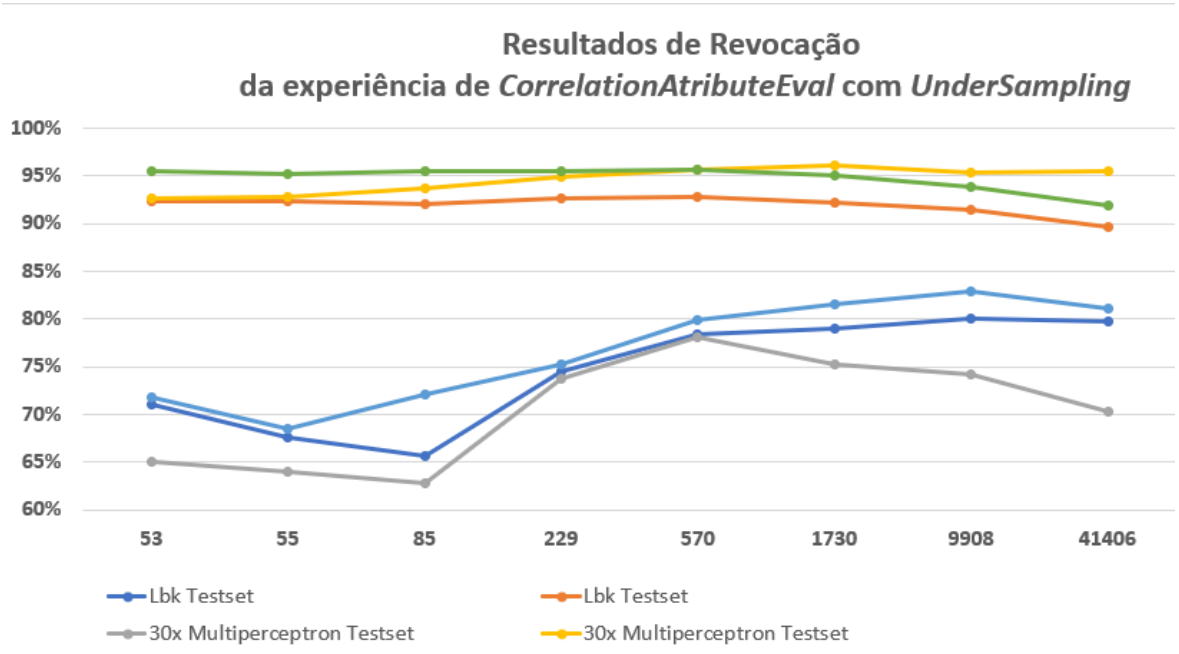

Gráfico 11

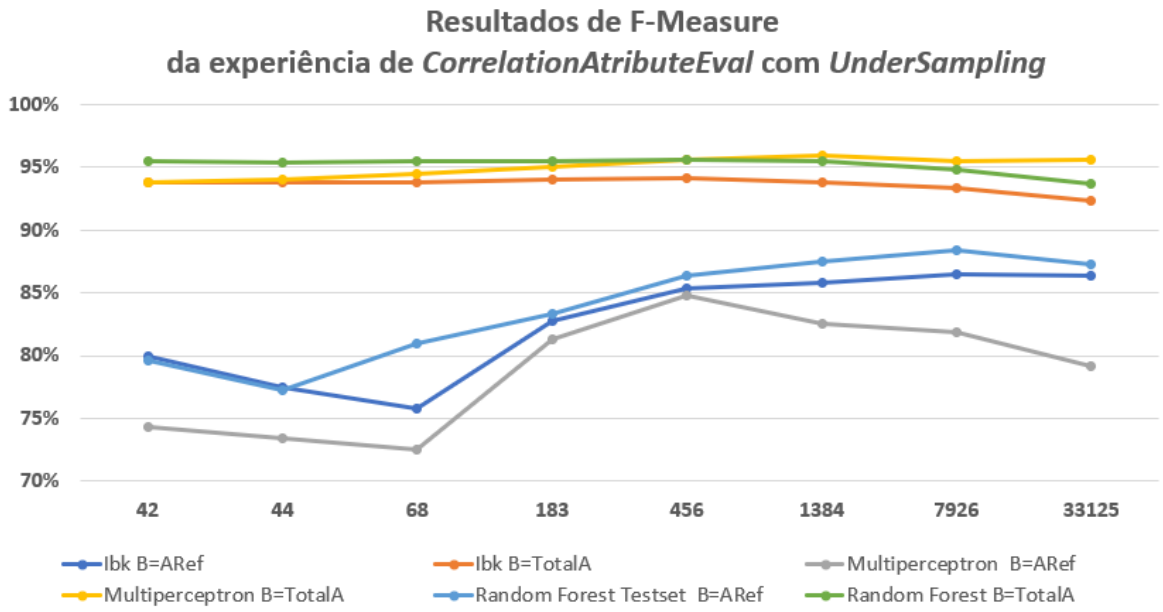

Gráfico 12

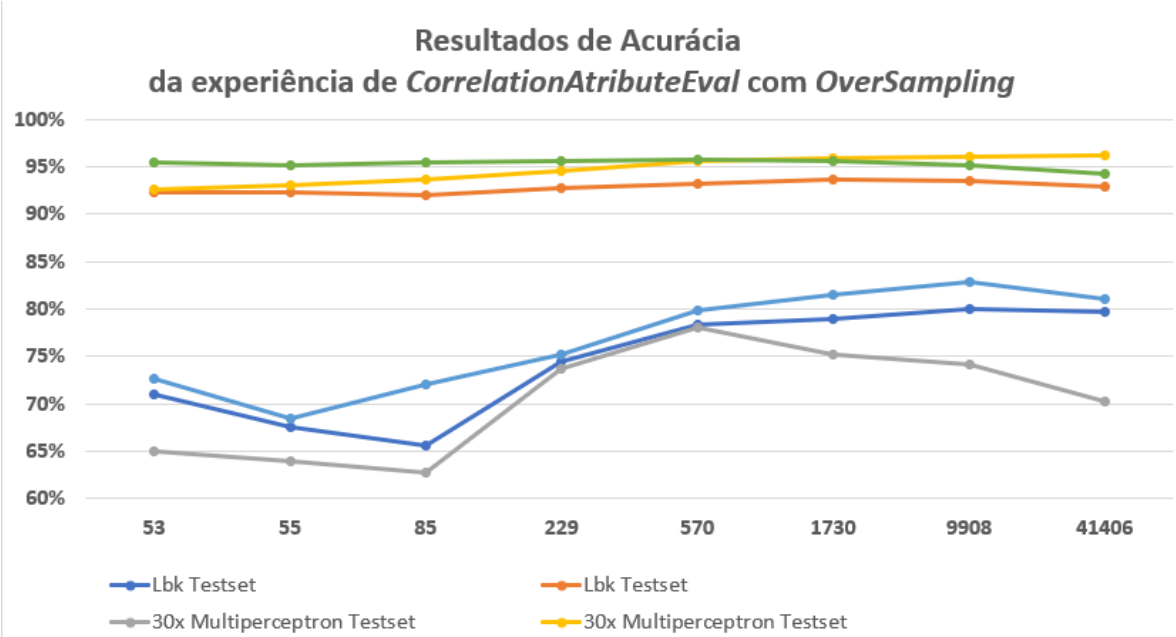

Gráfico 13

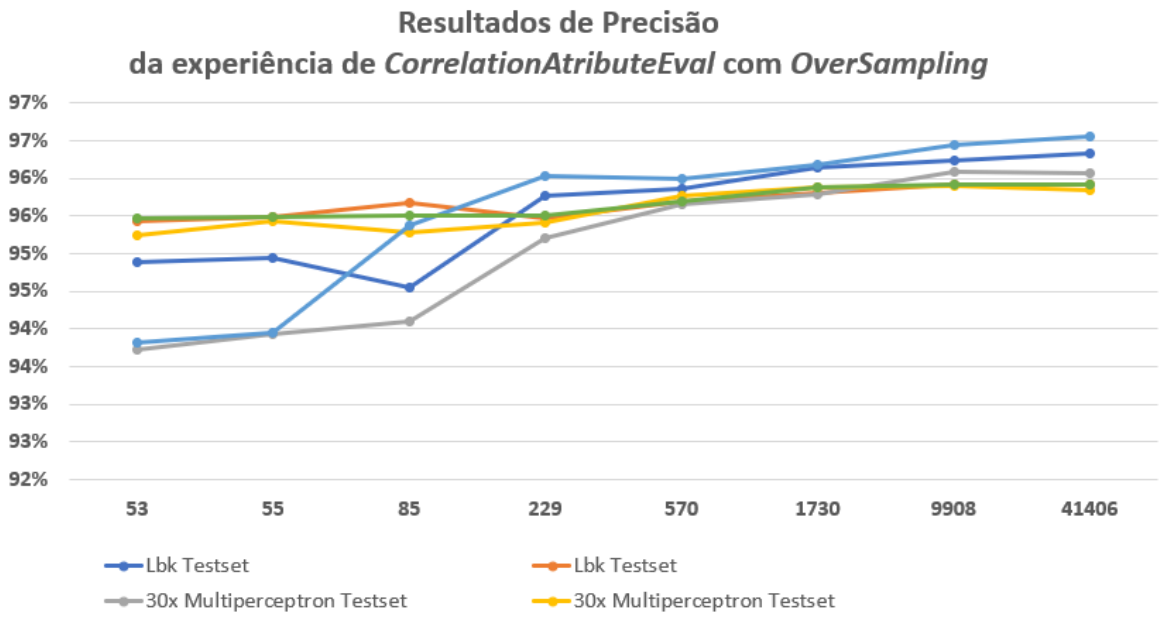

Gráfico 14

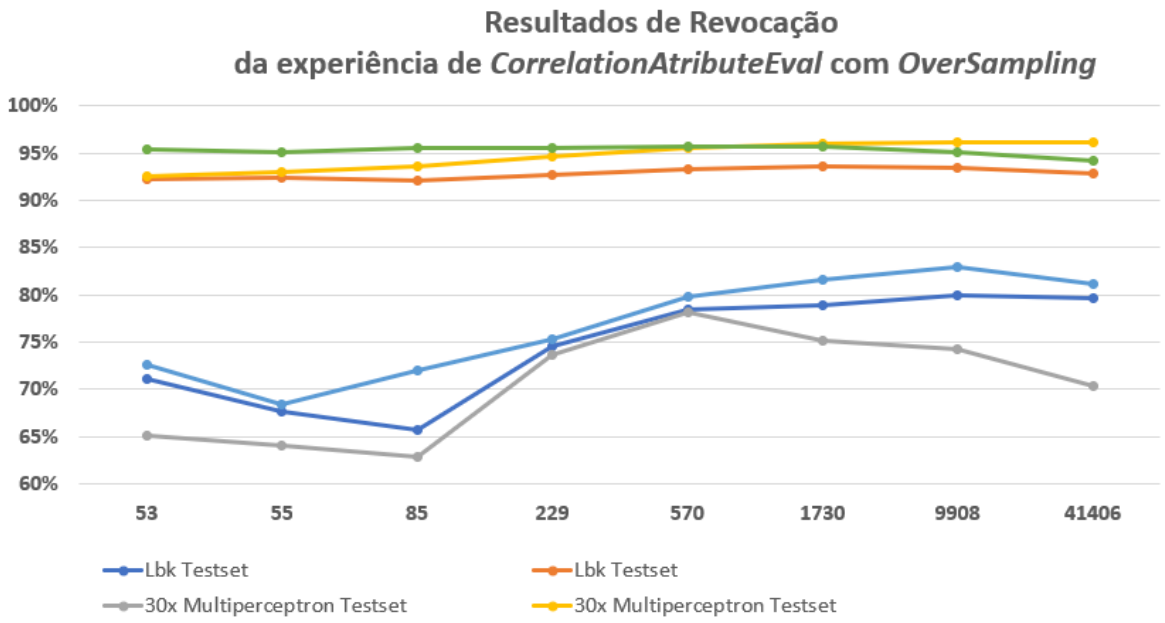

Gráfico 15

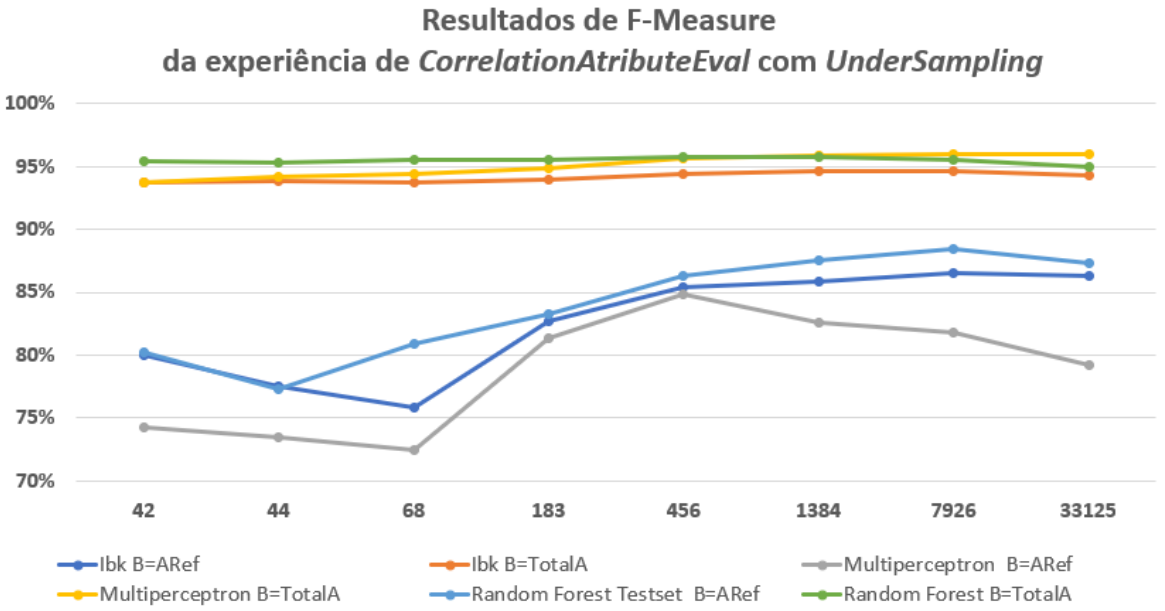

Gráfico 16

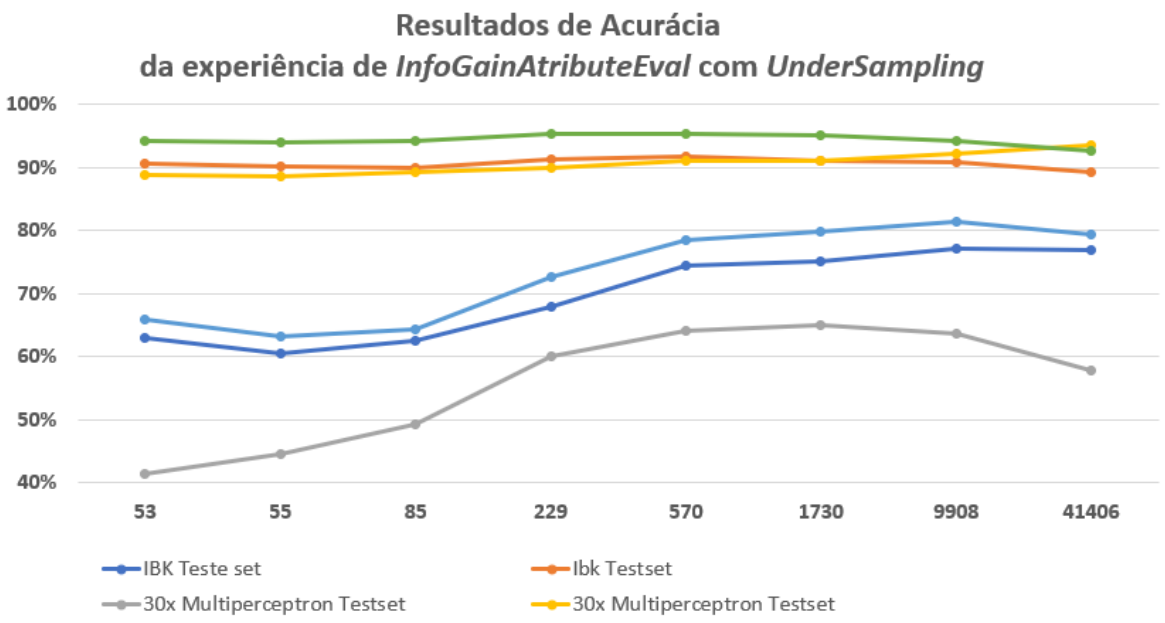

Gráfico 17

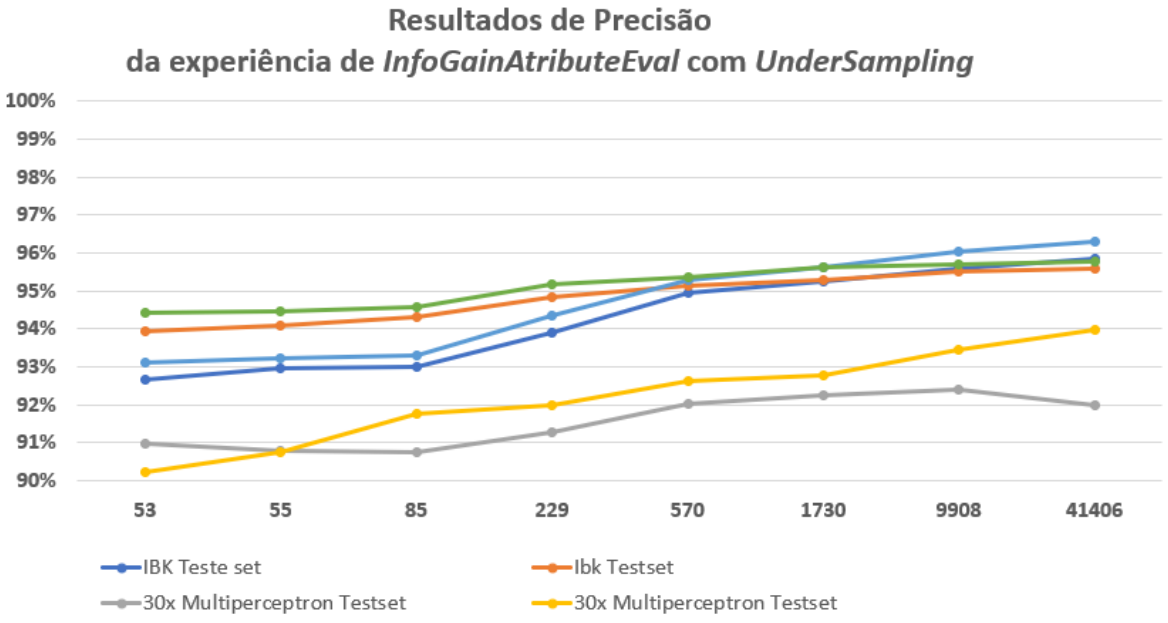

Gráfico 18

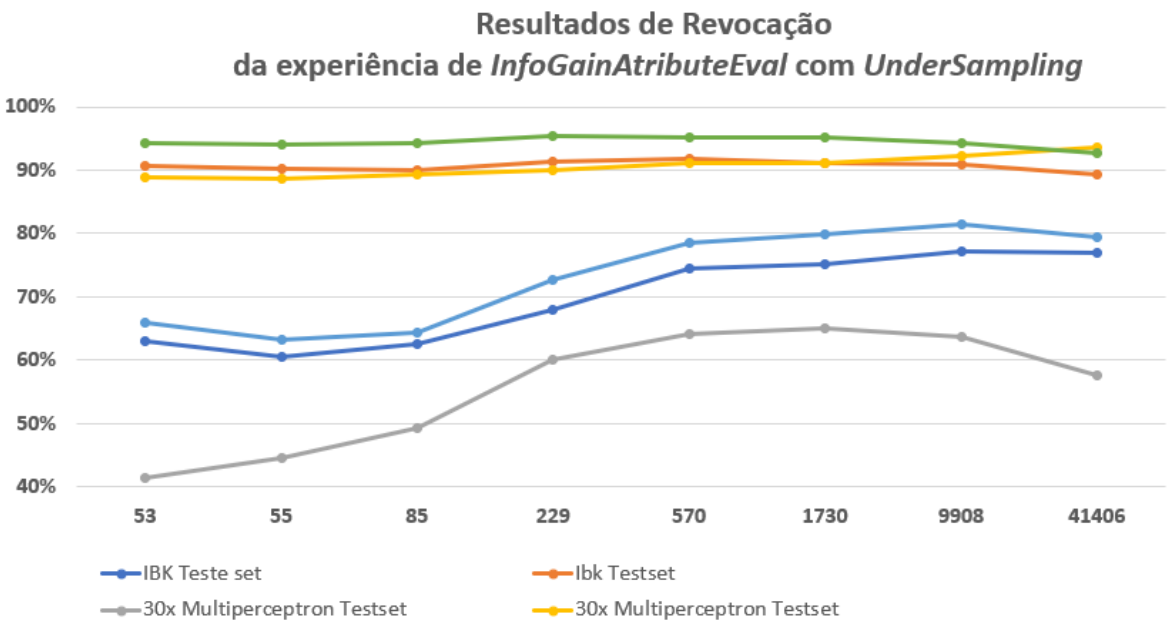

Gráfico 19

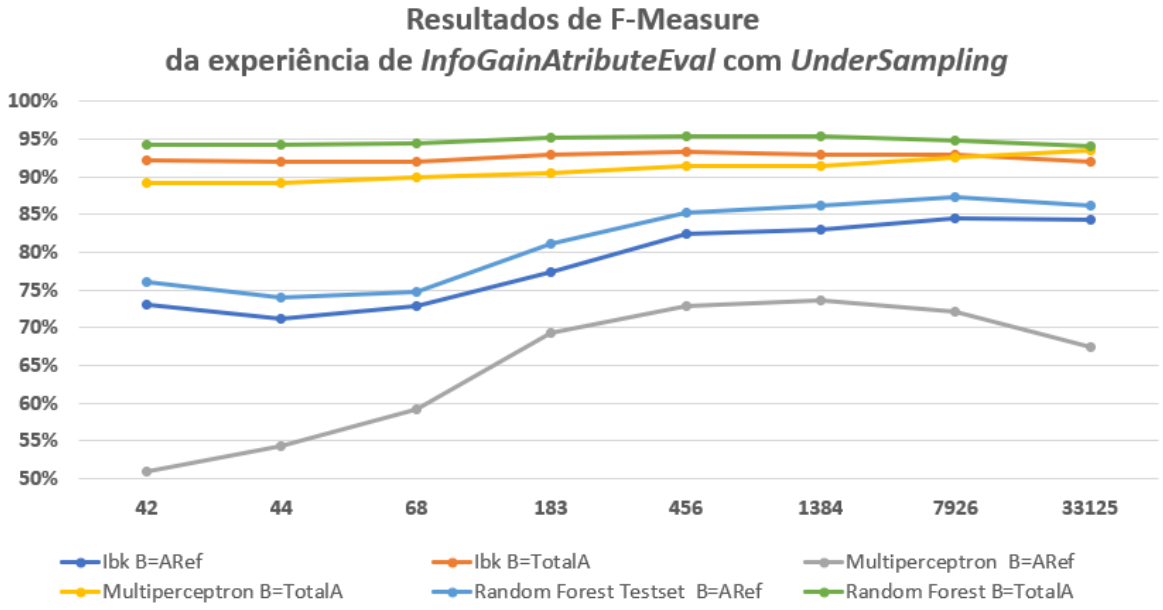

Gráfico 20

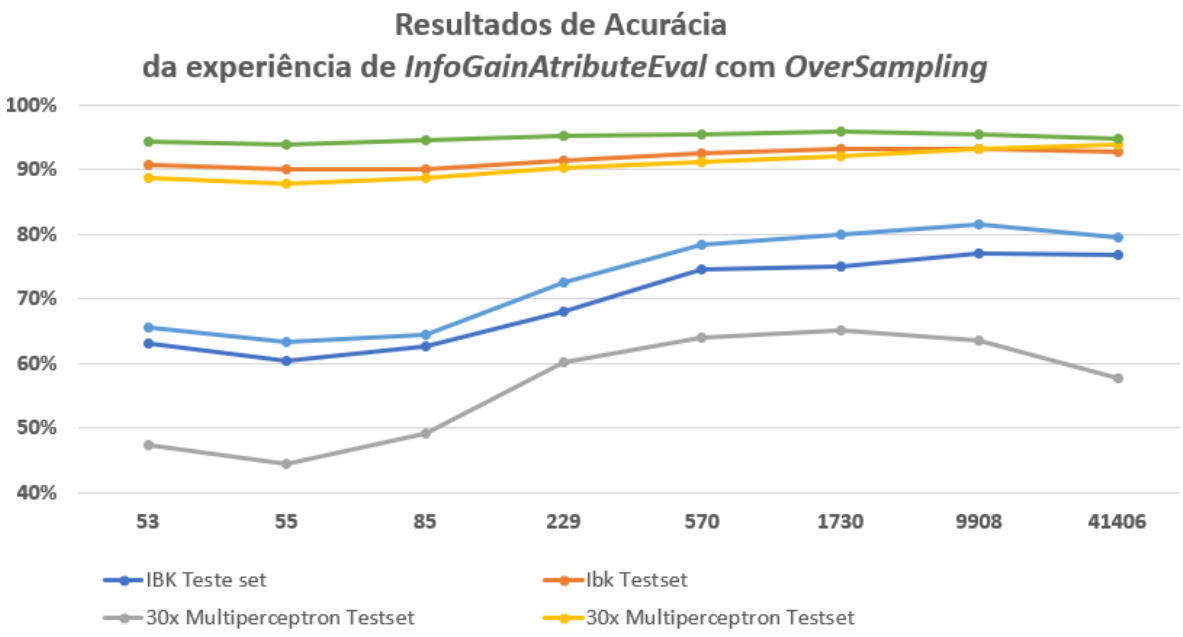

Gráfico 21

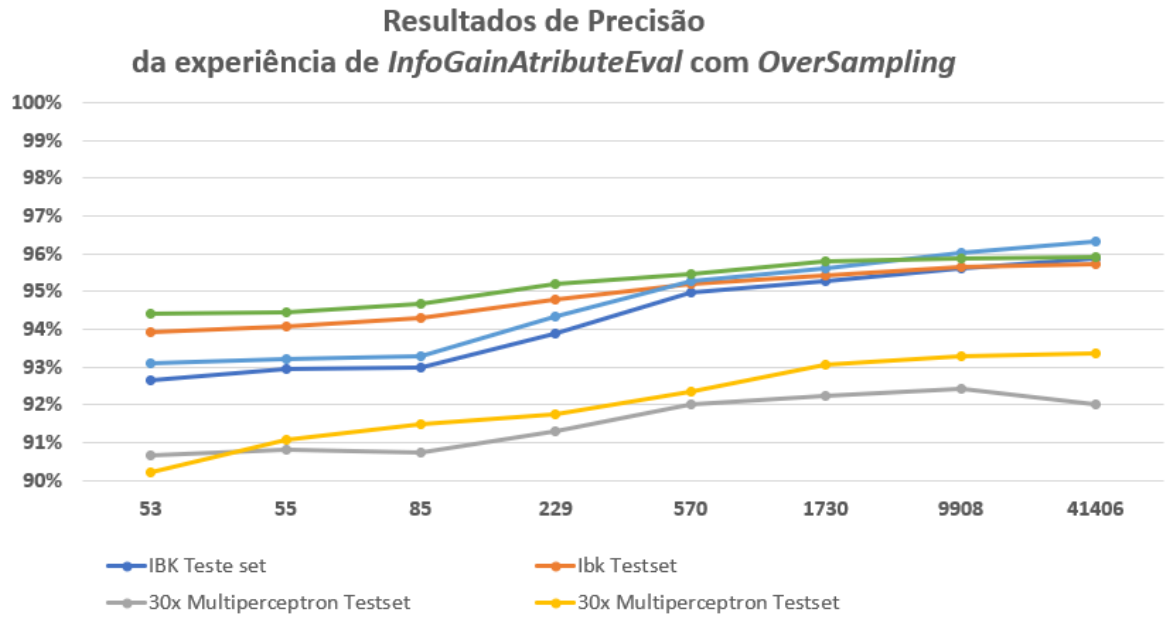

Gráfico 22

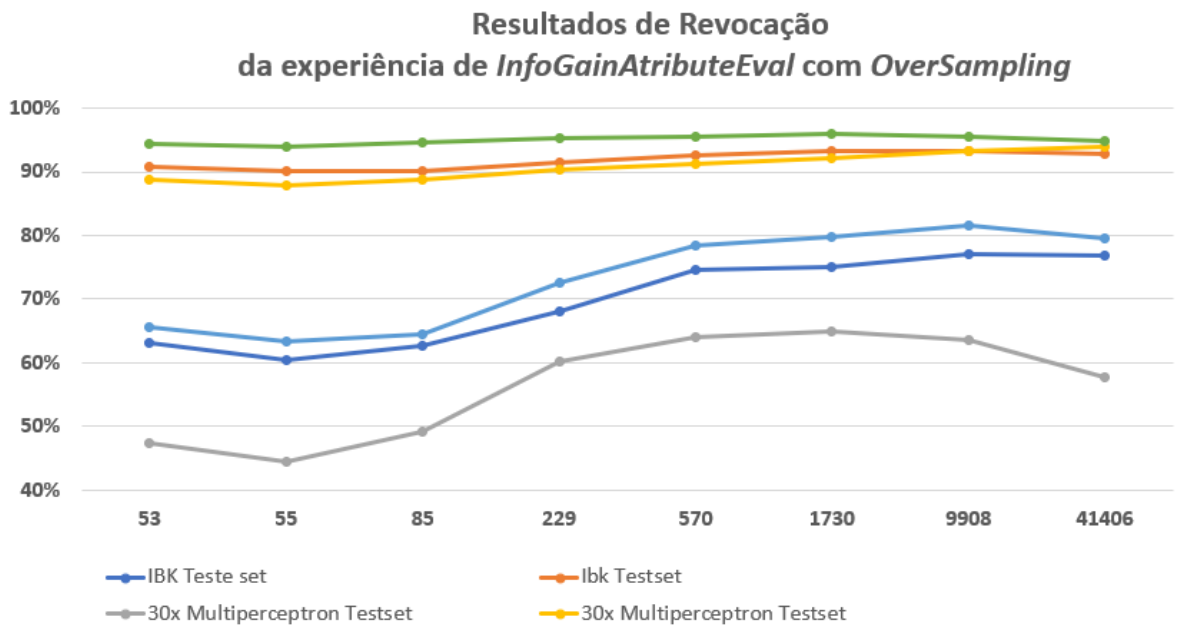

Gráfico 23

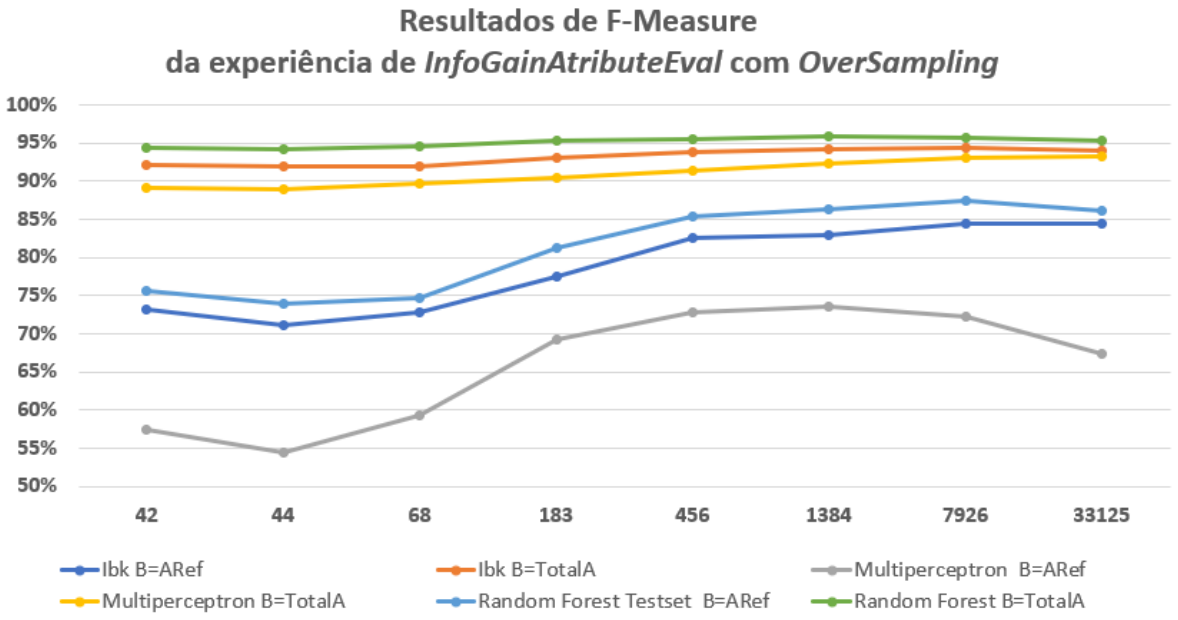

Gráfico 24

Anexo D

Tabela 1 - Conjunto de dados de teste gerado pelo *CorrelationAtributeEval*

Nº	Nº Atributo	Atributo
1	1	Dst Port
2	3	Flow Duration
3	4	Tot Fwd Pkts
4	6	TotLen Fwd Pkts
5	8	Fwd Pkt Len Max
6	10	Fwd Pkt Len Mean
7	11	Fwd Pkt Len Std
8	12	Bwd Pkt Len Max
9	13	Bwd Pkt Len Min
10	22	Fwd IAT Tot
11	28	Bwd IAT Mean
12	30	Bwd IAT Max
13	31	Bwd IAT Min
14	33	Fwd Header Len
15	34	Bwd Header Len
16	38	Pkt Len Max
17	44	RST Flag Cnt
18	45	PSH Flag Cnt
19	48	ECE Flag Cnt
20	49	Down/Up Ratio
21	51	Fwd Seg Size Avg
22	53	Subflow Fwd Pkts
23	54	Subflow Fwd Byts
24	57	Init Fwd Win Byts
25	58	Init Bwd Win Byts
26	59	Fwd Act Data Pkts
27	60	Fwd Seg Size Min
28	99	LABEL

Tabela 2 - Conjunto de dados de teste gerado pelo *InfoGainAtributeEval*

Nº	Nº Atributo	Atributo
1	3	Flow Duration
2	6	TotLen Fwd Pkts
3	8	Fwd Pkt Len Max
4	10	Fwd Pkt Len Mean
5	14	Bwd Pkt Len Mean
6	17	Flow Pkts/s
7	18	Flow IAT Mean
8	20	Flow IAT Max
9	22	Fwd IAT Tot
10	23	Fwd IAT Mean
11	25	Fwd IAT Max
12	33	Fwd Header Len
13	35	Fwd IAT Max
14	36	Bwd Pkts/s
15	38	Pkt Len Max
16	51	Fwd Seg Size Avg
17	52	Bwd Seg Size Avg
18	54	Subflow Fwd Byts
19	57	Init Fwd Win Byts
20	99	LABEL