DISSERTAÇÃO DE MESTRADO PROFISSIONAL

O uso da engenharia de atributos para otimizar o desempenho de modelos de aprendizado de máquina supervisionado aplicados a sistemas de detecção de intrusão

MOISÉS SILVA DE SOUSA

Orientador Prof. Dr. WILLIAM FERREIRA GIOZZA

Coorientador Prof. Dr. ROBSON DE OLIVEIRA ALBUQUERQUE

Programa de Pós-Graduação Profissional em Engenharia Elétrica

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

O uso da engenharia de atributos para otimizar o desempenho de modelos de aprendizado de máquina supervisionado aplicados a sistemas de detecção de intrusão

The use of Feature Engineering to optimize the performance of supervised machine learning models applied to Intrusion Detection Systems

Moisés Silva de Sousa

Orientador: Prof. Dr. William Ferreira Giozza, EnE/UnB

Coorientador: Prof. Dr. Robson de Oliveira Albuquerque, EnE/UnB

PUBLICAÇÃO: PPEE.MP.060

BRASÍLIA-DF

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

DISSERTAÇÃO DE MESTRADO PROFISSIONAL

O uso da engenharia de atributos para otimizar o desempenho de modelos de aprendizado de máquina supervisionado aplicados a sistemas de detecção de intrusão

MOISÉS SILVA DE SOUSA

Orientador Prof. Dr. WILLIAM FERREIRA GIOZZA

Coorientador Prof. Dr. ROBSON DE OLIVEIRA ALBUQUERQUE

Dissertação de Mestrado Profissional submetida ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica

Banca Examinadora

Prof. Dr. William Ferreira Giozza, ENE/UnB _____
Presidente

Prof. Dr. Robson de Oliveira Albuquerque, ENE/UnB _____
Examinador Interno

Prof. Dr. Georges Daniel Amvame Nze, ENE/UnB _____
Examinador Interno

Prof. Dr. Leandro Alves Neves, UNESP _____
(Sigla)
Examinador externo

Prof. Dr. Fábio Lúcio Lopes Mendonça, ENE/UnB _____
Suplente

FICHA CATALOGRÁFICA

SOUSA, MOISÉS SILVA DE
O uso da engenharia de atributos para otimizar o desempenho de modelos de aprendizadode máquina supervisionado aplicados asistemas de detecção de intrusão [Distrito Federal] 2023.
xvi, 89 p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2023).
Dissertação de Mestrado Profissional - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Segurança Cibernética	2. Sistemas de Detecção de Intrusão
3. Aprendizado de Máquina	4. Engenharia de Atributos
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

SOUSA, M.S. (2023). *O uso da engenharia de atributos para otimizar o desempenho de modelos de aprendizadode máquina supervisionado aplicados asistemas de detecção de intrusão* . Dissertação de Mestrado Profissional, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 89 p.

CESSÃO DE DIREITOS

AUTOR: MOISÉS SILVA DE SOUSA
TÍTULO: O uso da engenharia de atributos para otimizar o desempenho de modelos de aprendizadode máquina supervisionado aplicados asistemas de detecção de intrusão .
GRAU: Mestre em Engenharia Elétrica ANO: 2023

MOISÉS SILVA DE SOUSA
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

DEDICATÓRIA

Dedico este trabalho a todos que, direta e indiretamente, permitiram a realização deste sonho.

"Um sonho sonhado sozinho é um sonho.
Um sonho sonhado junto é realidade."

Yoko Ono

AGRADECIMENTOS

Agradeço a Deus, o dom da vida e da saúde, que me proporcionaram vivenciar este momento de realização profissional.

Agradeço aos meus pais pela dedicação e honradez que me educaram desde criança a alcançar os objetivos desejados. Sem eles, essa conquista não seria possível.

Agradeço aos meus professores orientadores, pela prestatividade, paciência, ensino e pelo apoio que confiaram durante estes dois anos de especialização.

Agradeço os meus amigos, especialmente: Antônio Carlos, Eduardo da Costa, Felipe Sousa e Tássila Catarina pelo apoio prestado em meus momentos de angústia e alegria durante esse processo. Contar com vossas conversas e ajudas foram e são grandes motivadores para que este dia chegasse.

Agradeço aos meus irmãos da Igreja, pelas orações e votos de sucesso que todos, sem exceção, feitas em meu nome.

Agradeço aos mais familiares mais próximos, por sempre acreditarem em mim e no meu potencial.

E por fim, mas não menos importante, a minha namorada (e futura esposa). Recordo que iniciei nosso namoro simultaneamente ao mestrado, e seu apoio e palavras de força me permitiram estar onde estou.

RESUMO

Título: O uso da engenharia de atributos para otimizar o desempenho de modelos de aprendizado de máquina supervisionado aplicados a sistemas de detecção de intrusão
Autor: Moisés Silva de Sousa
Orientador: William Ferreira Giozza, Dr.
Coorientador: Robson de Oliveira Albuquerque, Dr.
Programa de Pós-Graduação Profissional em Engenharia Elétrica
Brasília, 20 de dezembro de 2023

O uso de técnicas de aprendizado de máquina (ML) para a construção de sistemas de detecção de intrusão (IDS) vem crescendo a cada ano. Numerosas tecnologias de ML surgiram, permitindo construir modelos preditivos de aprendizado para identificar e detectar anomalias de tráfego de rede. Parte das técnicas de ML é uma abordagem não parametrizada, extraindo dados de grandes conjuntos de dados de forma indiscriminada que inclui dados irrelevantes e redundantes, afetando negativamente o desempenho dos algoritmos de classificação de ML. No entanto, é possível fornecer a uma técnica de ML a capacidade de extrair dados adequadamente do conjunto de dados selecionando um subconjunto apropri-ado de atributos, ou seja, por meio de engenharia de atributos (*FE – feature engineering*), que permite melhorar o desempenho da extração de dados, processos de ML de formação e classificação. Este trabalho discute como a engenharia de atributos pode ser usada para melhorar os processos de ML em sistemas IDS. Em particular, demonstra que com uma seleção adequada de atributos, o processo de treinamento pode ser reduzido, melhorando a velocidade de processamento e mantendo a precisão de classificação desejada. Os experimentos de avaliação de desempenho são baseados na plataforma de software WEKA usando os conjuntos de dados NSL-KDD e CID-IDS, além do *Support-Vector Machine* (SVM) e *Random Forest* como algoritmos de classificação de aprendizado de máquina. Utilizando diferentes razões de divisão teste-treinamento de dados (60-40, 70-30 e 80-20) e técnicas de seleção de atributos (*Information Gain, Correlation* e *Correlation-based Feature Selection* – CFS) este trabalho alcança resultados que permitem entender como a engenharia de atributos pode impactar positivamente o desempenho de um sistema ML-IDS.
Palavras-chave: Cibersegurança, *Feature Engineering, CFS Subset, Information Gain, Correlation*, SVM, Inteligência Artificial, Anomalia de rede.

ABSTRACT

Title: The use of Feature Engineering to optimize the performance of supervised machine learning models applied to Intrusion Detection Systems
Author: Moisés Silva de Sousa
Supervisor: William Ferreira Giozza, Dr.
Co-supervisor: Robson de Oliveira Albuquerque, Dr.
Professional Post-Graduate Program in Electrical Engineering
Brasília, December 20th, 2023
The use of machine learning (ML) techniques for building intrusion detection systems (IDS) has been growing every year. Numerous ML technologies have been emerged allowing to build predictive learning models in order to identify and detect network traffic anomalies using IDS. A part of the ML techniques is a non-parameterized approach, extracting data from large datasets in an undiscriminated way which includes irrelevant and redundant data, affecting adversely the performance of the ML classification al-gorithms. However, it is possible to provide to a ML technique the ability to properly extract data from the dataset by selecting an appropriate subset of attributes, i.e., by means of feature engineering (FE), that allows to improve the performance of the data extraction, training and classification ML processes. This work discusses how feature engineering can be used to improve the ML processes in IDS systems. In particular, it demonstrates that with an appropriate selection of attributes, the training process can be disrupted, improving the processing speed while maintaining the desired classification accuracy. The performance evaluation experiments are based on the WEKA software platform using the dataset NSL-KDD and the Support-Vector Machine (SVM) and Random Forest as machine learning classification algorithm. By using different data test-training division ratios (60- 40, 70-30 and 80-20) and attribute selection techniques (Information Gain, Correlation Gain and Correlation-based Feature Selection – CFS) this work achieves results that allow to understand how feature engineering may impact positively the performance of an ML-IDS system.
Keywords: Cybersecurity, Feature Engineering, CFS Subset, Information Gain, Correlation, SVM, artificial intelligence, network anomaly.

SUMÁRIO

1 INTRODUÇÃO 1
1.1 MOTIVAÇÃO 2
1.2 OBJETIVO 2
1.3 METODOLOGIA 3
1.4 CONTRIBUIÇÕES DO TRABALHO 4
1.5 ORGANIZAÇÃO 4

2 CONCEITOS E TRABALHOS RELACIONADOS 6
2.1 SISTEMAS DE DETECÇÃO DE INTRUSÃO 6
2.2 APRENDIZADO DE MÁQUINA 7
2.2.1 MODELO SUPERVISIONADO 7
2.2.2 MODELO NÃO-SUPERVISIONADO 10
2.3 *DATASET* 10
2.3.1 NSL-KDD 11
2.3.2 CIC-IDS-2017 12
2.4 MÉTRICAS DE DESEMPENHO DE ALGORITMOS ML 13
2.5 ENGENHARIA DE ATRIBUTOS (*FEATURE ENGINEERING*) 15
2.5.1 *INFORMATION GAIN* 17
2.5.2 *CORRELATION* 17
2.5.3 *CORRELATION BASED FEATURE SELECTION SUBSET EVAL* 17
2.6 WEKA 18
2.7 TRABALHOS RELACIONADOS 19
2.7.1 CONSIDERAÇÕES SOBRE A REVISÃO BIBLIOGRÁFICA 20

3 MODELO PROPOSTO 21
3.1 ESTRUTURAÇÃO DOS *DATASETS* 21
3.1.1 NSL-KDD 21
3.1.2 CIC-IDS-2017 23
3.2 PREPARAÇÃO DOS DADOS 24
3.3 PRÉ-PROCESSAMENTO DOS DADOS 24
3.3.1 NSL-KDD / KDDTRAIN+_20PERCENT 25
3.3.2 CIC-IDS-2017 26
3.4 DIVISÃO DOS *DATASETS* 26
3.5 BALANCEAMENTO *UNDERSAMPLING* 27
3.6 BALANCEAMENTO *OVERSAMPLING* 28
3.7 TREINAMENTO E CLASSIFICAÇÃO 28

4 ANÁLISE DOS DADOS E RESULTADOS 30
4.1 *Dataset* NSL-KDD (*KDDTrain+_20Percent)* 31
4.1.1 ALGORITMO CLASSIFICADOR SVM 31
4.1.2 ALGORITMO CLASSIFICADOR *Random Forest* 37

4.2 *Dataset* CIC-IDS-2017 44
4.2.1 ALGORITMO CLASSIFICADOR SVM 44
4.2.2 ALGORITMO CLASSIFICADOR *Random Forest* 50
4.3 CIC-IDS-2017 - COM BALANCEAMENTO *undersampling* 56
4.3.1 ALGORITMO CLASSIFICADOR SVM 56
4.3.2 ALGORITMO CLASSIFICADOR *Random Forest* 63
4.4 CIC-IDS-2017 - COM BALANCEAMENTO *oversampling* 68
4.4.1 ALGORITMO CLASSIFICADOR SVM 68
4.4.2 ALGORITMO CLASSIFICADOR *Random Forest* 75
4.5 COMPARAÇÃO DE RESULTADOS DOS EXPERIMENTOS 81
4.6 COMPARAÇÃO DE RESULTADOS COM TRABALHOS CORRELATOS 82

5 CONCLUSÃO 84

REFERÊNCIAS BIBLIOGRÁFICAS 85

APÊNDICES 89

LISTA DE FIGURAS

1.1 Metodologia de pesquisa 4

2.1 Sistema de Detecção de Intrusão. 7
2.2 Esquematização de aprendizado de máquina supervisionado. 8
2.3 Representação gráfica do funcionamento do SVM. 9
2.4 Representação gráfica do funcionamento do *Random Forest* 9
2.5 Esquematização de aprendizado de máquina não supervisionado 10
2.6 Esquematização do processo de *oversampling* e *undersampling* 11
2.7 Etapas de seleção de atributos 16
2.8 Esquematização do WEKA e seus componentes de interesse para realização do experimento. 18

3.1 Configuração do *framework* FE-ML-IDS implementado na plataforma WEKA, em que $n = 12$ representa o número de rodadas feitas para cada experimento. 22
3.2 Configuração do *dataset* NSL-KDD. 23
3.3 Configuração do *dataset* CIC-IDS-2017 24

4.1 Matriz de confusão do experimento 4.1.1.1. 32
4.2 Matriz de confusão do experimento 4.1.1.2. 33
4.3 Matriz de confusão do experimento 4.1.1.3. 34
4.4 Matriz de confusão do experimento 4.1.1.4. 35
4.5 Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos para o Experimento NSL-KDD/SVM. 36
4.6 Matriz de confusão do experimento 4.2.1.1. 38
4.7 Matriz de confusão do experimento 4.1.2.2. 39
4.8 Matriz de confusão do experimento 4.1.2.3. 40
4.9 Matriz de confusão do experimento 4.1.2.4. 42
4.10 Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento NSL-KDD/*Random Forest* 43
4.11 Matriz de confusão do experimento 4.2.1.1. 45
4.12 Matriz de confusão do experimento 4.2.1.2. 46
4.13 Matriz de confusão do experimento 4.2.1.3. 47
4.14 Matriz de confusão do experimento 4.2.1.4. 48
4.15 Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/SVM. 49
4.16 Matriz de confusão do experimento 4.2.2.1. 51
4.17 Matriz de confusão do experimento 4.2.2.2. 52
4.18 Matriz de confusão do experimento 4.2.2.3. 53
4.19 Matriz de confusão do experimento 4.2.2.4 54
4.20 Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/*Random Forest* 55

4.21 Matriz de confusão do experimento 4.3.1.1 57
4.22 Matriz de confusão do experimento 4.3.1.2 59
4.23 Matriz de confusão do experimento 4.3.1.3 60
4.24 Matriz de confusão do experimento 4.3.1.4 61
4.25 Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/SVM/*Undersampling* 62
4.26 Matriz de confusão do experimento 4.3.2.1 64
4.27 Matriz de confusão do experimento 4.3.2.2 65
4.28 Matriz de confusão do experimento 4.3.2.3 66
4.29 Matriz de confusão do experimento 4.3.2.4 67
4.30 Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/*Random Forest*/*Undersampling* 69
4.31 Matriz de confusão do experimento 4.4.1.1 69
4.32 Matriz de confusão do experimento 4.4.1.2 71
4.33 Matriz de confusão do experimento 4.4.1.3 72
4.34 Matriz de confusão do experimento 4.4.1.4 73
4.35 Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/SVM/*Oversampling* 74
4.36 Matriz de confusão do experimento 4.4.2.3 76
4.37 Matriz de confusão do experimento 4.4.2.1 77
4.38 Matriz de confusão do experimento 4.4.2.2 78
4.39 Matriz de confusão do experimento 4.4.2.4 79
4.40 Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/*Random Forest*/*Oversampling* 80

LISTA DE TABELAS

2.1 *Datasets* do NSL-KDD 12
2.2 Atributos do *dataset* CIC-IDS-2017 13
2.3 Matriz de confusão genérica 15

3.1 Ferramentas utilizadas no experimento 21
3.2 Informações do *dataset* NSL-KDD 23
3.3 Informações do *dataset* CIC-IDS-2017 23
3.4 Atributos selecionados pelo *Information Gain* 25
3.5 Atributos selecionados pelo *Correlation* 25
3.6 Atributos selecionados pelo CFS 26
3.7 Atributos selecionados pelo *Information Gain* 26
3.8 Atributos selecionados pelo *Correlation* 27
3.9 Atributos selecionados pelo CFS 27
3.10 Informações do *dataset* CIC-IDS-2017 28
3.11 Informações do *dataset* CIC-IDS-2017 28

4.1 TP e FP do experimento 4.1.1.1 31
4.2 *Precision*, *Recall* e *F1-Score* do experimento 4.1.1.1 31
4.3 TP e FP do experimento 4.1.1.2 32
4.4 *Precision*, *Recall* e *F1-Score* do experimento 4.1.1.2 32
4.5 TP e FP com o experimento 4.1.1.3 33
4.6 *Precision*, *Recall* e *F1-Score* do experimento 4.1.1.3 34
4.7 TP e FP do experimento 4.1.1.4 34
4.8 *Precision*, *Recall* e *F1-Score* do experimento 4.1.1.4 35
4.9 Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.1.1 37
4.10 TP e FP do experimento 4.1.2.1 37
4.11 *Precision*, *Recall* e *F1-Score* do experimento 4.1.2.1 38
4.12 TP e FP do experimento 4.1.2.2 38
4.13 *Precision*, *Recall* e *F1-Score* do experimento 4.1.2.2 39
4.14 TP e FP do experimento 4.1.2.3 40
4.15 *Precision*, *Recall* e *F1-Score* do experimento 4.1.2.3 40
4.16 TP e FP do experimento 4.1.2.4 41
4.17 *Precision*, *Recall* e *F1-Score* do experimento 4.1.2.4 41
4.18 Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.1.2 42
4.19 TP e FP do experimento 4.2.1.1 44
4.20 *Precision*, *Recall* e *F1-Score* do experimento 4.2.1.1 44
4.21 TP e FP do experimento 4.2.1.2 45

4.22 *Precision*, *Recall* e *F1-Score* do experimento 4.2.1.xii 45
4.23 TP e FP do experimento 4.2.1.3 46
4.24 *Precision*, *Recall* e *F1-Score* do experimento 4.2.1.3 46
4.25 TP e FP do experimento 4.2.1.4 47
4.26 *Precision*, *Recall* e *F1-Score* do experimento 4.2.1.4 47
4.27 Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.27.1 49
4.28 TP e FP do experimento 4.2.2.1 50
4.29 Resultados de: Precisão, *Recall* e *F1-Score* do experimento 4.2.2.1 50
4.30 TP e FP do experimento 4.2.2.2 51
4.31 Resultados de: Precisão, *Recall* e *F1-Score* do experimento 4.2.2.2 51
4.32 TP e FP do experimento 4.2.2.3 52
4.33 Resultados de: Precisão, *Recall* e *F1-Score* do experimento 4.2.2.3 52
4.34 TP e FP do experimento 4.2.2.4 53
4.35 *Precision*, *Recall* e *F1-Score* do experimento 4.2.2.4 54
4.36 Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.2.2 56
4.37 TP e FP do experimento 4.3.1.1 57
4.38 *Precision*, *Recall* e *F1-Score* do experimento 4.3.1.1 57
4.39 TP e FP do experimento 4.3.1.2 58
4.40 *Precision*, *Recall* e *F1-Score* do experimento 4.3.1.2 58
4.41 TP e FP do experimento 4.3.1.3 59
4.42 *Precision*, *Recall* e *F1-Score* do experimento 4.3.1.3 59
4.43 TP e FP do experimento 4.3.1.4 60
4.44 *Precision*, *Recall* e *F1-Score* do experimento 4.3.1.4 60
4.45 Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.3.1 61
4.46 TP e FP do experimento 4.3.2.1 63
4.47 *Precision*, *Recall* e *F1-Score* do experimento 4.3.2.1 63
4.48 TP e FP do experimento 4.3.2.2 64
4.49 *Precision*, *Recall* e *F1-Score* do experimento 4.3.2.2 64
4.50 TP e FP do experimento 4.3.2.3 65
4.51 *Precision*, *Recall* e *F1-Score* do experimento 4.3.2.3 66
4.52 TP e FP do experimento 4.3.2.4 67
4.53 *Precision*, *Recall* e *F1-Score* do experimento 4.3.2.4 67
4.54 Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimentos 4.3.2 68
4.55 TP e FP do experimento 4.4.1.1 70
4.56 *Precision*, *Recall* e *F1-Score* do experimento 4.4.1.1 70
4.57 TP e FP do experimento 4.4.1.2 70
4.58 *Precision*, *Recall* e *F1-Score* do experimento 4.4.1.2 71
4.59 TP e FP do experimento 4.4.1.3 71

4.60 *Precision*, *Recall* e *F1-Score* do experimento 4.4.1.xiii 72
4.61 TP e FP do experimento 4.4.1.4 73
4.62 *Precision*, *Recall* e *F1-Score* do experimento 4.4.1.4 73
4.63 Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.4.1 75
4.64 TP e FP do experimento 4.4.2.3 75
4.65 *Precision*, *Recall* e *F1-Score* do experimento 4.4.2.3 75
4.66 TP e FP do experimento 4.4.2.1 76
4.67 *Precision*, *Recall* e *F1-Score* do experimento 4.4.2.1 77
4.68 TP e FP do experimento 4.4.2.2 77
4.69 *Precision*, *Recall* e *F1-Score* do experimento 4.4.2.2 78
4.70 TP e FP do experimento 4.4.2.4 79
4.71 *Precision*, *Recall* e *F1-Score* do experimento 4.4.2.4 79
4.72 Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.4.2 81
4.73 TP e FP do experimento 4.3.2.4 82
4.74 Comparação com os trabalhos relatados 83
4.75 Comparação com os trabalhos relatados 83

LISTA DE ACRÔNIMOS

Siglas

ARFF	*Attribute-Relation File Format*
CFS	*Correlation based Feature Selection*
CIC	*Canadian Institute for Cybersecurity*
CISTI	*Iberian Conference on Information Systems and Technologies*
CSV	*Comma-separated values*
DNN	*Deep Neural Network*
DoS	*Denial of Service*
FE	*Feature Engineering*
FN	*False Negative*
FP	*False Positive*
HIDS	*Host Intrusion Detection System*
IA	Inteligência Artificial
IG	*Information Gain*
IDS	*Intrusion Detection System*
KNN	*K-Nearest Neighbour*
ML	*Machine Learning*
NIDS	*Network Intrusion Detection System*
NSL-KDD	*Network Security Laboratory - Knowledge Discovery in Databases*
PCAP	*Packet Capture*
TN	*True Negative*
TP	*True Positive*
SVM	*Support Vector Machine*
WEKA	*Waikato Environment for Knowledge Analysis*

1 INTRODUÇÃO

O uso crescente de sistemas focados em tecnologia de informação e comunicação (TIC) tem convivido atualmente com o surgimento de diversos tipos de ataques cibernéticos, como *Denial of Service* (DoS), *malware* e *sniffers*. Isso se deve, em parte, à quantidade de informações que podem ser extraídas desses sistemas bem como à sensibilidade dos dados extraídos ilicitamente [1].

A detecção de ataques cibernéticos é hoje em dia, uma importante ferramenta em sistemas de segurança de redes de comunicação [2]. Em particular, esforços mundiais têm sido concentrados para construir sistemas capazes de detectar intrusões, os chamados *Intrusion Detection System* (IDS) [3]. Como forma de melhorar essa detecção, o uso de técnicas de *Machine Learning* (ML) tornou-se, nos dias atuais, um método comum para lidar com os problemas de segurança que surgem de ataques cibernéticos envolvendo por exemplo o vazamentos de dados.

Sistemas ML-IDS baseados em aprendizado de máquina supervisionado dependem de um treinamento contínuo a partir de uma coleção de informações organizada em datasets, contendo por exemplo referências sobre o tráfego na rede. Os dados podem ser organizados em classes, por exemplo, pacotes resultantes de um ataque ou de um tráfego de pacotes normal [4]. A qualidade dessas informações é imprescindível para a obtenção de resultados robustos e eficientes no que tange a detecção de ataques por sistemas ML-IDS [5].

Além da existência de *datasets* de qualidade, a forma como eles serão usados também é uma etapa importante na criação do ML-IDS. No campo do aprendizado de máquina, modelos supervisionados podem ser utilizados para detectar e classificar pacotes de fluxos de tráfego de rede como normais (sem ataque) ou anômalos (possível ataque).

Para alcançar essa capacidade de identificação, é necessário um processo apropriado de treinamento e construção de modelos de classificação preditiva, conhecido como técnica supervisionada de aprendizado de máquina [6]. Nesse trabalho, o algoritmo aprende a distinguir a classe de cada pacote dos dados de treinamento. Diferentemente dos modelos não supervisionados, que tentam classificar os dados com base na padronização das amostras estudadas, os modelos supervisionados requerem a utilização de um conjunto de dados capaz de rotular cada um dos dados de treinamento. Entre vários outros algoritmos de ML supervisionados, tem-se: árvores de decisão, *Support Vector Machine* (SVM), *Random Forest*, *Linear Regression* como exemplos para este propósito [7, 8].

Uma grande parte das técnicas de ML é feita a partir de uma abordagem não parametrizada, extraindo dados de conjuntos de dados (*datasets*) de forma não discriminada que inclui dados irrelevantes e redundantes, afetando negativamente o desempenho dos algoritmos de ML [9]. No entanto, é possível conceber uma técnica de aprendizado de máquina com a capacidade de extrair dados adequadamente do conjunto de dados, selecionando um subconjunto apropriado de atributos, por meio da engenharia de atributos (FE - *Feature Engineering*).

A engenharia de atributos (FE) permite melhorar o desempenho dos processos de extração de dados, treinamento e classificação do modelo [10]. Usando uma seleção apropriada de atributos alocados a um

conjunto de dados, o que significa menos dados processados durante o treinamento, é possível reduzir o tempo computacional sem afetar os resultados das classificações do algoritmo de ML [8]. O principal desafio da engenharia de atributos envolve a identificação de quais dados não devem ser considerados e, portanto, removidos do processo de treinamento e teste [11].

1.1 MOTIVAÇÃO

Este trabalho foi desenvolvido no intuito de investigar como o uso criterioso dos dados de um determinado *dataset* influencia na melhora, em termos de desempenho, da construção e testagem de modelos de sistemas de detecção de intrusão baseados em aprendizado de máquina supervisionado. Esse estudo torna-se ainda mais relevante, quando aplicado em contextos *real-time* em que são necessárias tomadas de decisão rápidas e precisas, para conter ataques cibernéticos o mais breve possível.

Uma mesma coleção de dados (*dataset*) pode ser utilizada para diferentes abordagens de modo que uma boa parte dos *datasets* carrega dados que podem não ser relevantes para determinados objetivos investigativos. Dessa forma, é interessante analisar como o uso de determinados filtros para seleção de dados em diferentes *datasets* pode contribuir para melhorar o desempenho dos sistemas. Isso porque, cada seletor possui métricas de escolha diferentes uns dos outros e sua eficácia estará atrelada ao tipo de dado contido nos datasets.

Portanto, investigar como o uso da engenharia de atributos (FE) pode, mantendo uma taxa de acertos em relação à classe de um pacote de rede, diminuir o tempo de resposta entre detecção e ação, permite a construção de sistemas de detecção de intrusão que atuem de maneira assertiva e rápida.

Com o intuito de atingir tal objetivo, a aplicação de distintos seletores, juntamente com classificadores, é empregada de modo a determinar se a utilização dessas ferramentas pode ser condensada em procedimentos específicos, ou se cada cenário deve ser meticulosamente analisado, visando a produção de resultados mais robustos.

1.2 OBJETIVO

Neste trabalho, é proposto um modelo de *framework* intitulado *Feature Engineering-Machine Learning Intrusion Detection System* (FE-ML-IDS) para investigar como a engenharia de atributos pode ser usada para melhorar o desempenho da velocidade de processamento em sistemas de detecção de intrusão baseados em aprendizado de máquina supervisionado.

Dessa forma, objetiva-se demonstrar que o uso da engenharia de atributos (FE - *Feature Engineering*) reduz o tempo de construção de um modelo de aprendizado de máquina (ML) supervisionado e, consequentemente, aumentar a velocidade de processamento, mantendo o desempenho de classificação do algoritmo de ML.

1.3 METODOLOGIA

No desenvolvimento deste trabalho foram realizadas as seguintes etapas:

- Adoção da plataforma WEKA (Waikato Environment for Knowledge Analysis) [14];
- Seleção dos datasets (NSL-KDD e CIC-IDS-2017);
- Identificaçãor dosquais filtros de seleção de atributos (i.e., técnicas de FE) compossuem melhor desempenho em termos de eficiência em determinadosconsiderando os datasets escolhidos;.
- Desenvolvimento deer umaa metodologia metodologia que racionalize o uso de um dataset, conforme as necessidades do estudo pretendidotrabalho;.
- Demonstrar queAplicação das técnicas de FE na o uso de Feature Engineering melhora a performance da construção e testagem de modelos ML-IDSsupervisionados;.
- Análise dos resultados obtidos pelo framework FE-ML-IDS com as técnicas de FE escolhidas em cenários com razões de divisão teste-treinamento de dados distintas;
- Análise dos resultados obtidos com o framework FE-ML-IDS atuando sobre um conjunto de dados balanceados e não balanceados;
- Comparação dos resultados obtidos com o frameworkr FE-ML-IDS com outros modelos desenvolvidos e presentes na bibliografia de referência;
- Comparação dos resultados obtidos com o framework FE-ML-IDS com aqueles obtidos sem o uso de FE.

A modelagem do *framework* FE-ML-IDS é baseada na plataforma WEKA (*Waikato Environment for Knowledge Analysis*), uma coleção de algoritmos de aprendizado de máquina para tarefas de mineração de dados, e nos *datasets* NSL-KDD *Network Security Laboratory - Knowledge Discovery in Databases* e CIC-IDS-2017, constituindo-se em um *benchmark* eficaz para a construção e comparação de diferentes tipos de IDS [12, 13, 14].

Três diferentes técnicas de engenharia de recursos são investigadas: *Information Gain*, *Correlation* e *Correlation based feature subset selection* (*CfsSubsetEval*) [15]. Cada uma dessas técnicas de FE tem seu desempenho avaliado usando dois diferentes tipos de algoritmo como classificador em um ambiente com tráfego de pacotes suspeito: *Support-Vector Machine* (SVM) e *Random Forest* [16].

Trata-se uma pesquisa experimental (procedimentos), com uma abordagem qualitativa e de natureza básica. Para o desenvolvimento deste trabalho, algumas etapas foram obedecidas. Elas são descritas abaixo e, de maneira mais visual, apresentada na Figura 1.1.

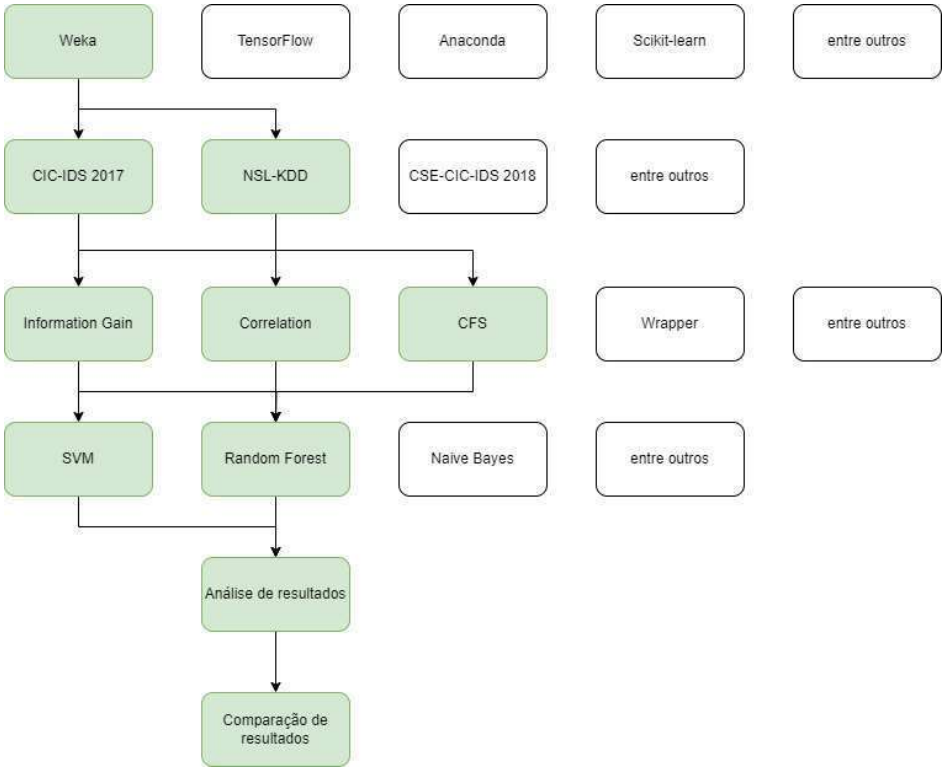


Figura 1.1: Metodologia de pesquisa.

1.4 CONTRIBUIÇÕES DO TRABALHO

O artigo *Information Gain applied to reduce model-building time in decision-tree-based intrusion detection system* [17], foi apresentado no CISTI *Iberian Conference on Information Systems and Technologies* 2022, realizada em Madri, Espanha [18].

Os anais da CISTI 2022 foram publicados na plataforma IEEE Xplore. O IEEE tem grande relevância na área de Engenharia Elétrica em âmbito mundial, tendo suas publicações utilizadas pela comunidade científica para consumo de conteúdo a respeito da área.

Este artigo permitiu introduzir os primeiros passos a respeito do uso de FE para melhoria de performance na construção de modelos baseados em árvores de decisão.

1.5 ORGANIZAÇÃO

O restante deste trabalho está estruturado da seguinte forma. No Capítulo II, são apresentados os conceitos e trabalhos relacionados mais relevantes quanto ao uso de técnicas de mineração de dados e fer-ramentas de aprendizado de máquina para detecção de intrusão. O Capítulo III apresenta a modelagem do *framework* proposto, descrevendo suas funcionalidades e componentes como conjunto de dados, características selecionadas, algoritmo classificador, treinamento e testes, bem como a implementação e as etapas utilizadas para o treinamento dos agentes. O Capítulo IV apresenta, analisa e discute os resultados,

comparando-os com os trabalhos discutidos no Capítulo II. Finalmente, as conclusões são apresentadas no Capítulo V.

2 CONCEITOS E TRABALHOS RELACIONADOS

Neste Capítulo são apresentados os principais conceitos envolvidos no desenvolvimento deste trabalho bem como alguns dos trabalhos relacionados disponíveis na literatura.

2.1 SISTEMAS DE DETECÇÃO DE INTRUSÃO

Atualmente, as redes de comunicação demandam uma grande quantidade de dados. Uma vez que parte dessas informações carrega dados sigilosos ou sensíveis, é fundamental a questão de prover segurança nas redes. Os sistemas de detecção de intrusão (IDS - Intrusion Detection System) desempenham papel significativo nesse processo [19]. Um sistema IDS refere-se aos meios técnicos de descobrir em uma rede se houve acessos autorizados ou não. Com o uso de IDSs, é possível oferecer redes de comunicação mais seguras [20].

Existe uma variedade de IDSs que se comportam de maneira diferente com o mesmo intuito: detectar invasões. Dentre os principais tipos de IDS, tem-se: *Host Intrusion Detection System* (HIDS) e *Network Intrusion Detection System* NIDS [21]. O NIDS avalia o tráfego de rede como um todo. Essa avaliação é feita a partir da análise dos cabeçalhos dos pacotes que transitam pelo sistema, como um scanner de porta entre outros [22]. O NIDS é constituído de duas partes: sensores que monitoram o tráfego propriamente dito, e a estação de gerenciamento que administra remotamente estes sensores. Já o HIDS está atrelado a uma máquina específica. Nessa situação, o computador avaliado arquiva os eventos ocorridos nos logs do sistema para fins de análise.

Sistemas IDS podem ser construídos empregando-se técnicas de aprendizado de máquina (ML - *Machine Learning*) [23]. Os sistemas de detecção de intrusão baseados em aprendizado de máquina (ML-IDS) foram desenvol-vidos para lidar com vários tipos de ataques cibernéticos, como negação de serviço distribuído (DDoS), força bruta, *heartbleed*, *botnet*, ataques da Web e infiltração de rede de dentro, entre outros [24].

O uso da engenharia de recursos (*FE - Feature Engineering*) tem sido recentemente um dos principais focos no desenvolvimento de sistemas ML-IDS [25]. Com a FE, os resultados podem ser aprimorados por meio da busca por um subconjunto de recursos específicos, em vez de pesquisar por todo o conjunto de dados (*dataset*), entregando eficiência e assertividade no processo de identificação de pacotes maliciosos da rede.

O modelo FE-ML-IDS proposto neste trabalho destina-se principalmente à construção de aplicações de detecção de intrusão não em tempo real, identificando tráfego de rede do tipo *Bot*. Ele foi concebido considerando os principais componentes e etapas de processamento de um ML-IDS genérico conforme ilustrado na Figura 2.1.

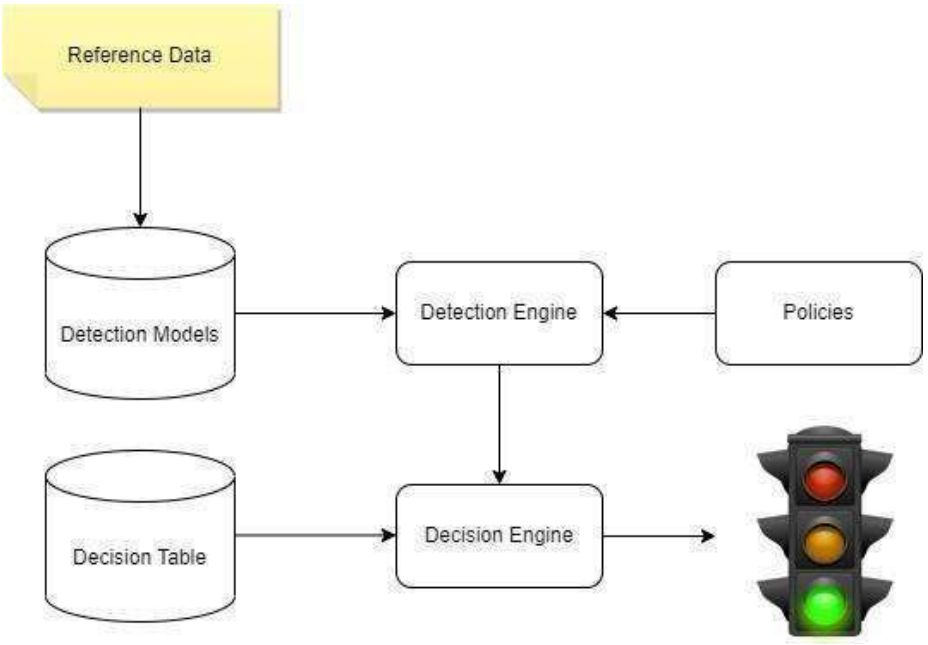


Figura 2.1: Sistema de Detecção de Intrusão.

2.2 APRENDIZADO DE MÁQUINA

A maneira pela qual um agente inteligente consegue determinar as classes de um dado aleatório, por exemplo, pacote de tráfego normal ou pacote corrompido, é feita mediante um aprendizado. Neste processo, o algoritmo coleta dados (de formas diversas), como tentativa de entender os comportamentos esperados e predizer a que classe aquele pacote ou informação pertence.

As técnicas de aprendizado de máquina podem ser classificadas segundo o modelo de aprendizagem: supervisionado, não supervisionado ou híbrido. Discutir-se-á cada um deles a seguir.

2.2.1 MODELO SUPERVISIONADO

Nessa técnica de aprendizado de máquina, o algoritmo é treinado por meio de um *dataset* previamente construído. Esse conjunto de dados utiliza uma combinação rotulada de informações para que o algoritmo seja treinado [25]. As próprias referências indicam que, para uma determinada entrada, existirá uma saída equivalente [26]. A Figura 2.2, indica esse processo. Os dados passados ao algoritmo (Dados para treinamento), são rotulados conforme sua classe. O modelo de treinamento por sua vez, coleta esses dados e os guardam de maneira que, ao passar um dado de teste haja uma predição. A partir disso, o algoritmo classifica o dado para teste recebido, conforme os rótulos fornecidos no início do experimento.

Dessa forma, com base no treinamento feito a partir desses dados, o algoritmo consegue, em outro contexto, predizer qual seria a classificação correta da informação recebida. É importante destacar que, para fins de pesquisa, os próprios fornecedores dividem os *datasets* em subconjuntos, uma parte utilizada no treinamento e a outra parte nas testagens.

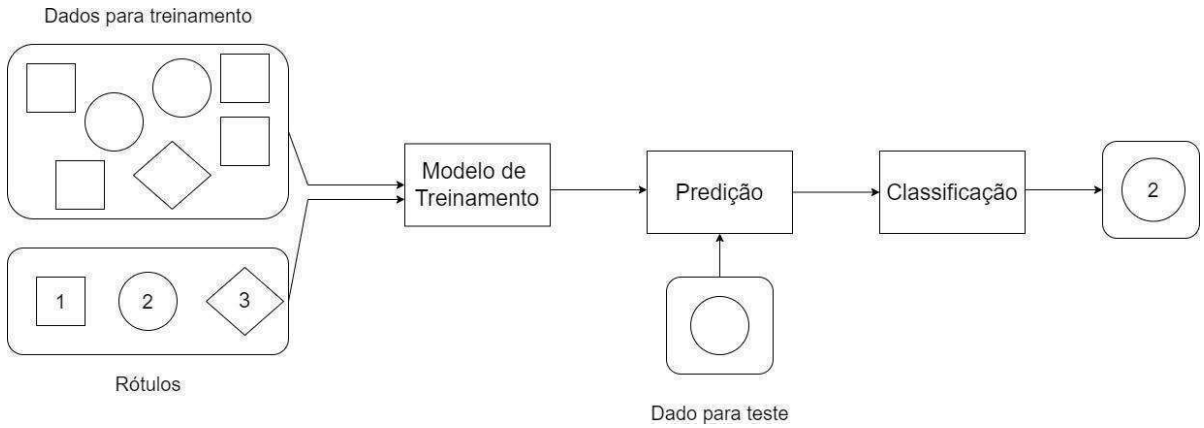


Figura 2.2: Esquematização de aprendizado de máquina supervisionado.

Dentre os algoritmos de classificação existentes, pode-se destacar: *Random Forest* e SVM, ambos utilizados neste trabalho.

O modelo de aprendizado de máquina supervisionado tem sido utilizado preferencialmente para a construção de ML-IDS.

2.2.1.1 *SUPPORT-VECTOR MACHINE*

O *Support Vector Machine* (SVM) é um algoritmo de aprendizado de máquina supervisionado que tem sido amplamente aplicado a tarefas relacionadas por exemplo à classificação de texto, reconhecimento facial e bioinformática [27]. O SVM pode ser linear ou não linear.

No caso do SVM linear as classes são separadas usando o hiperplano que limita a classe a ser descoberta. Esse hiperplano é conhecido como limite e o principal objetivo do algoritmo é aproximar os vetores que possuem clas-ses parecidas, e distanciar-se daquelas que possuem valores distintos [28]. Por isso a sua capacidade de classificação correta é geralmente alta uma vez que, por maximizar as margens entre uma classe e outra, evita que haja sobreposição de dados [28]. A Figura 2.3 representa o hiperplano criado pelo algoritmo SVM linear para classificar o dado recebido.

O SVM, devido à necessidade de separar os vetores, requer uma alta capacidade computacional em relação a outros algoritmos supervisionados, como o Random Forest, por exemplo [29]. Por outro lado, considerando as redes neurais, também muito utilizadas, não há grandes diferenças de desempenho computacional [28].

2.2.1.2 *RANDOM FOREST*

Random Forest é um método de *ensemble learning* para classificação, regressão e outras tarefas que operam através da construção de múltiplas árvores de decisão no momento do treinamento. Para tarefas de classificação, a saída da floresta aleatória é a classe selecionada pela maioria das árvores.

O algoritmo *Random Forest* funciona da seguinte maneira. Continuamente, verifica em cada vértice

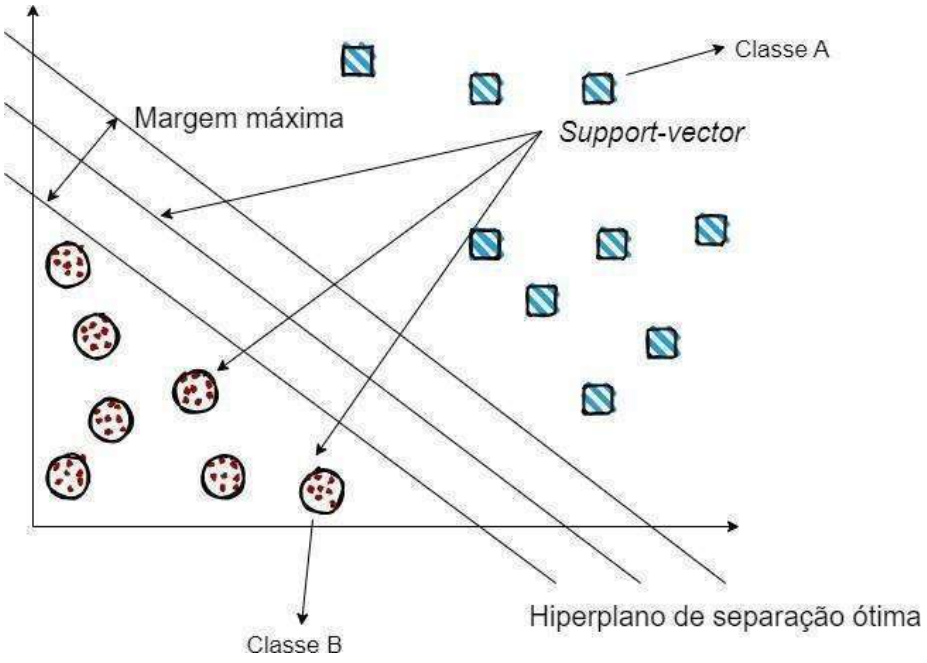


Figura 2.3: Representação gráfica do funcionamento do SVM.

se a classe atende a um determinado critério e constrói aleatoriamente árvores de decisão. Ao final do processo, combina estes resultados para chegar à classificação final.

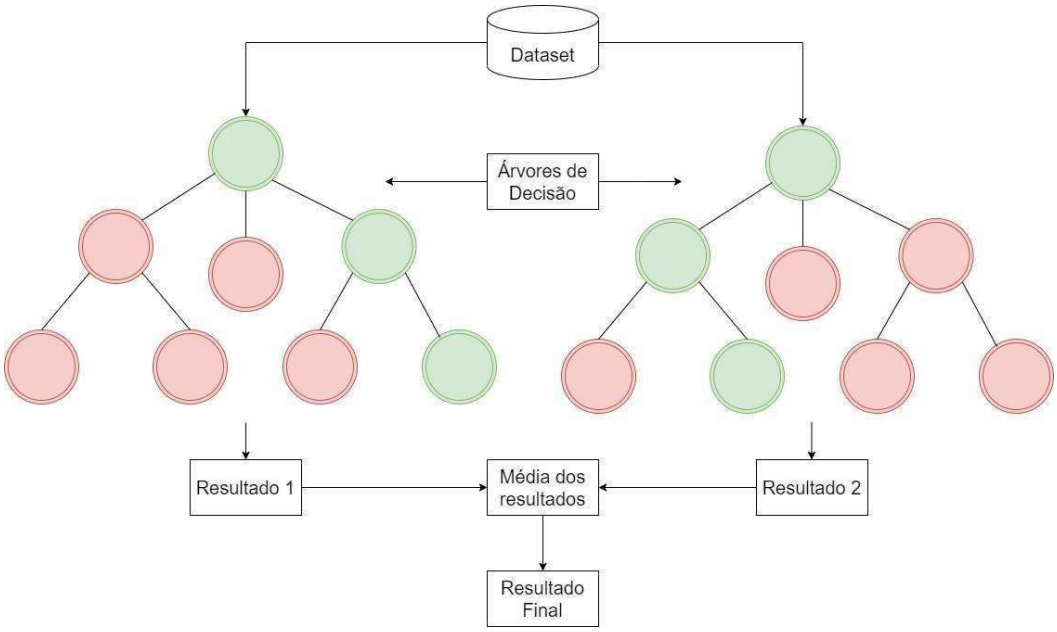


Figura 2.4: Representação gráfica do funcionamento do *Random Forest*.

No algoritmo *Random Forest*, o tempo de treinamento é menor em comparação com os outros algoritmos, como o SVM [30]. Mesmo em *datasets* com uma grande quantidade de dados, o algoritmo *Random Forest* provê uma alta precisão durante as classificações.

2.2.2 MODELO NÃO-SUPERVISIONADO

Na técnica de aprendizado de máquina não supervisionado, há a ausência de um conjunto de dados rotulados a serem passados para o algoritmo. Nesse caso, a entrada ajuda o algoritmo para extrair regras e regulamentos a partir de uma montagem dos padrões observados [25]. Pontos importantes dessas informações são resumidos na tentativa de fornecer algum significado para a entrada passada. Isso é feito, uma vez que não há dados tabulados indicando a que classe pertenceria tal dado. A Figura 2.5, ilustra esse processo.

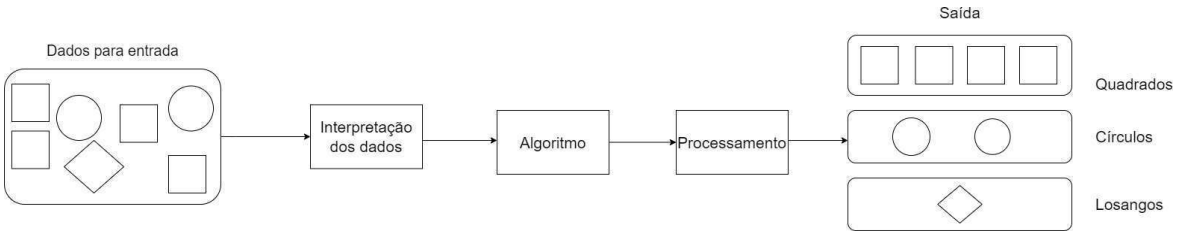


Figura 2.5: Esquematização de aprendizado de máquina não supervisionado.

Uma das técnicas de aprendizado não supervisionado é o *clustering*. descobrir padrões interessantes nas informações recebidas com base em seu comportamento [25]. A aprendizagem profunda, do inglês *Deep Learning*, por exemplo, basea-se em um conjunto de algoritmos que tentam modelar abstrações de alto nível de dados. O termo profundo se deve a utilização de grafos com esta característica, compostas de várias transformações lineares e não lineares. É importante destacar que este trabalho não emprega o uso de modelos não supervisionados, como este mencionado.

Adicionalmente, existem os modelos híbridos de aprendizado de máquina. Neste modelo, é fornecido ao algoritmo um pequeno lote de dados rotulados, tal como ocorre no modelo supervisionado. A partir dessa amostra, um *dataset* muito maior é inserido no contexto de classificação e treinamento, porém com dados não rotulados, como ocorre em modelos não-supervisionados.

2.3 DATASET

Resumidamente, um *dataset* é um compilado de dados. É um dos principais insumos para análise de dados comumente estruturado na forma de uma tabela. Cada linha da tabela representa um registro, enquanto que as colunas, por sua vez, indicam as características daquele dado.

Como visto anteriormente, técnicas de aprendizado de máquina supervisionado exigem um conjunto de dados rotulados para o processo de treinamento. A partir dessa etapa, o modelo pode classificar ou prever a qual classe um determinado dado real pertence. Dessa forma, os *datasets* desempenham uma função essencial no processo de construção de IDS com ML [21].

Encontrar um *dataset* eficaz é um processo primordial para o sucesso do modelo ML e de suas taxas de classificação. Uma característica que deve ser levada em consideração é o quão balanceado esse conjunto de dados é. Define-se como balanceado, aqueles *datasets* que possuem um equilíbrio entre as classes apresentadas.

Quando um conjunto de dados possui essa característica, o processo de treinamento ao qual o modelo ML está inserido também adquire esse equilíbrio. Dessa forma, uma classe não terá privilégios sobre a outra, permitindo que a classificação ou previsão dos dados não seja viesada.

A busca por *datasets* balanceados, ou uso de técnicas que permitem esse balanceamento é portanto bastante importante. Para checar se um conjunto de dados é balanceado ou não, é necessário visualizar como está a distribuição do número de instâncias para cada classe existente. Frisa-se que o número não necessariamente deve ser igual, mas pelo menos próximo.

Depois da etapa de verificação, caso não seja balanceado, é necessário adotar técnicas de balanceamento. Existem duas: *undersampling* e *oversampling* [31].

No *undersampling*, a classe com o maior número de instâncias é dividida em porções menores e, aleatoriamente, esses pedaços são escolhidos de maneira que, ao somá-los, o seu tamanho se aproxime da classe com menos dados.

O funcionamento do *oversampling* é parecido. A classe com o menor número de instâncias é copiada até que o seu tamanho se aproxime da classe com mais dados [24].

A Figura 2.6 esquematiza esse processo de balanceamento.

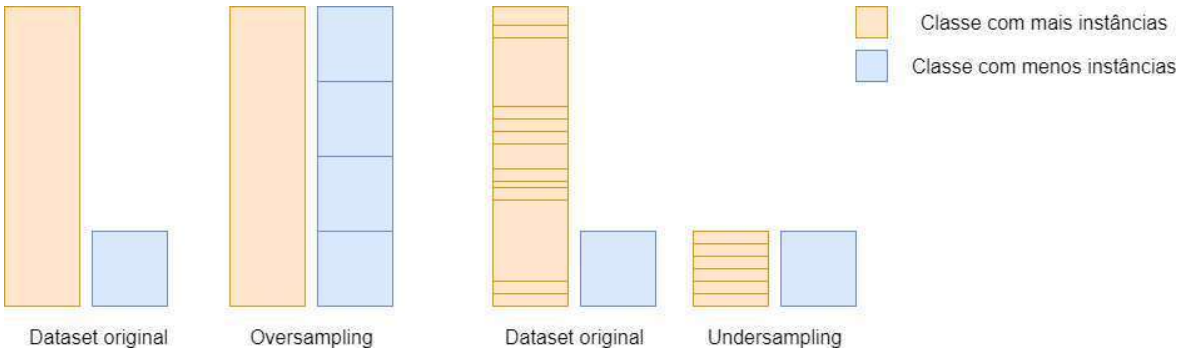


Figura 2.6: Esquematização do processo de *oversampling* e *undersampling*.

Para o desenvolvimento deste trabalho, foram usados *datasets* que computavam dados relacionados a ataques cibernéticos. Nesses *datasets*, cada registro pode conter: endereço IP de origem e destino, protocolo utilizado, tamanho de pacote e muitos outros. A coluna que determinará se o pacote é ou não um ataque a rede receberá o nome de classe neste trabalho. Os dois *datasets* utilizados neste trabalho são apresentados a seguir.

2.3.1 NSL-KDD

O principal *dataset* escolhido para realizar este trabalho é o NSL-KDD [13], fornecido pelo Instituto Canadense de Cibersegurança (CIC) e eventualmente transformado em um *benchmark* eficaz permitindo construir e comparar diferentes tipos de IDS. O NSL-KDD é uma evolução do KDD'99, corrigindo alguns problemas da versão anterior, como a não inclusão de registros redundantes e a não duplicação de registros.

Os conjuntos de dados CIC são usados em todo o mundo por universidades, indústria privada e pes-

Tabela 2.1: *Datasets* do NSL-KDD

KDDTrain+.ARFF - The full NSL-KDD train set with binary labels in ARFF format; KDDTrain+.TXT - The full NSL-KDD train set including attack-type labels and difficulty level in CSV format; KDDTrain+_20Percent.ARFF - A 20% subset of the KDDTrain+.arff file; KDDTrain+_20Percent.TXT - A 20% subset of the KDDTrain+.txt file. KDDTest+.ARFF - The full NSL-KDD test set with binary labels in ARFF format; KDDTest+.TXT - The full NSL-KDD test set including attack-type labels and difficulty level in CSV format; KDDTest-21.ARFF - A subset of the KDDTest+.arff file which does not include records with difficulty level of 21 out of 21; KDDTest-21.TXT - A subset of the KDDTest+.txt file which does not include records with difficulty level of 21 out of 21.

quisadores independentes [14]. Por exemplo, o conjunto de dados de benchmark de detecção de intrusão do CIC foi usado por mais de 800 empresas de segurança em todo o mundo para validar seus resultados de pesquisa ou técnicas. Portanto, ao utilizar o NSL-KDD, este trabalho tem uma rica e ampla escolha de fontes de pesquisa com a finalidade de comparar seus resultados.

O NSL-KDD está organizado da seguinte forma. Ao todo, são oito arquivos disponibilizados, quatro para cada tipo de extensão de arquivo (.TXT e . ARFF). Dois deles são usados para testes e outros dois para treinamento. Possui duas classes diferentes: Ataque e Normal, além de possuir 41 atributos para cada dado forncedido. A Tabela 2.1 resume os tipos de diferentes conjuntos de dados no NSL-KDD.

A disponibilidade de arquivos NSL-KDD no *ATTRIBUTE-Relation File Format* (ARFF) é muito útil porque permite realizar digitalizações de treinamento sem conversão de formato de documento ao usar softwares, como a plataforma de bancada de trabalho WEKA.

Vale ressaltar que, entre os arquivos NSL-KDD, há o original, com todos os dados e uma versão resumida, com 20% do conteúdo principal. Neste trabalho, o KDDTrain+_20percent. O ARFF foi utilizado por limitações de hardware do computador utilizado nos testes.

2.3.2 CIC-IDS-2017

O *dataset* escolhido para validação dos resultados é o CIC-IDS 2017 [14] montado pelo Instituto Canadense de Cibersegurança da Universidade de New Brunswick.

A razão para escolher esse conjunto de dados é porque ele se destina exclusivamente à detecção de intrusão. O CIC-IDS 2017 possui critérios considerados necessários para a construção de conjuntos de dados focados no IDS [32] tais como: configuração completa da rede, tráfego completo, diversidade de ataques, conjunto de dados rotulado, interação completa, captura completa, protocolos disponíveis, heterogeneidade, conjunto de recursos e metadados. Além disso, a organização facilita a preparação de dados para treinamentos. Nesse caso, apenas a seleção de recursos é necessária.

A organização desse *dataset* é a seguinte. O fluxo de rede de uma determinada topologia é capturado

Tabela 2.2: Atributos do *dataset* CIC-IDS-2017

Destination Port, Flow Duration, Total Fwd Packets, Total Backward Packets,Total Length of Fwd Packets, Total Length of Bwd Packets, Fwd Packet Length Max, Fwd Packet Length Min, Fwd Packet Length Mean, Fwd Packet Length Std,Bwd Packet Length Max, Bwd Packet Length Min, Bwd Packet Length Mean, Bwd Packet Length Std, Flow Bytes/s, Flow Packets/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min,Fwd IAT Total, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Max, Fwd IAT Min,Bwd IAT Total, Bwd IAT Mean, Bwd IAT Std, Bwd IAT Max, Bwd IAT Min,Fwd PSH Flags, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Fwd Header Length, Bwd Header Length,Fwd Packets/s, Bwd Packets/s, Min Packet Length, Max Packet Length, Packet Length Mean, Packet Length Std, Packet Length Variance, FIN Flag Count, SYN Flag Count, RST Flag Count, PSH Flag Count, ACK Flag Count, URG Flag Count, CWE Flag Count, ECE Flag Count, Down/Up Ratio, Average Packet Size, Avg Fwd Segment Size, Avg Bwd Segment Size, Fwd Header Length New, Fwd Avg Bytes/Bulk, Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk, Bwd Avg Bulk Rate, Subflow Fwd Packets, Subflow Fwd Bytes, Subflow Bwd Packets, Subflow Bwd Bytes, Init_Win_bytes_forward, Init_Win_bytes_backward,act_data_pkt_fwd, min_seg_size_forward, Active Mean, Active Std, Active Max, Active Min,Idle Mean, Idle Std, Idle Max, Idle Min,class

cinco dias da semana. Para cada dia, há um arquivo CSV com todos os dados coletados.

Para os experimentos deste trabalho, o dia selecionado foi quinta-feira (durante a manhã), já que neste dia foram coletados dados referentes a ataques web como XSS (*Cross-site Scripting*), Força Bruta e SQL *Injection*, configurando assim um fluxo de rede normal e que também está sob ataque. Ao todo, são 78 características para cada uma das 170.366 instâncias existentes nesse conjunto de dados, conforme mostrado na Tabela 2.2.

2.4 MÉTRICAS DE DESEMPENHO DE ALGORITMOS ML

Para medir a eficiência de um algoritmo ML, existem métricas de desempenho. Essas medidas possuem o intuito de aferir a qualidade do modelo desenvolvido.

Quando se fala em predição de classes, existem possíveis resultados quanto a assertividade do algoritmo ML. Nesse espectro, quando o modelo acerta a classe ao qual o dado pertence dá-se o nome de *True Positive* (TP). Infelizmente, nem sempre o algoritmo ML terá uma eficiência em 100%. Caso o modelo infira uma classe diferente ao qual o dado pertence, ou seja, de maneira falsa houve uma classificação, chama-se *False Positive* (FP). Os antagônicos são os *True Negative* e *False Negative*.

Essas métricas básicas são utilizadas para compor as métricas de interesse deste trabalho (*Precision, Recall* e *F1-Score*) e construir as matrizes de confusão como será visto nas subseções a seguir. Vale mencionar que os valores para TP, TN, FP e FN nunca serão maiores que 1, pois eles elucidam valores percentuais dos resultados obtidos pós-classificação.

Outra métrica existente é a *Precision*. Como o próprio nome em inglês sugere, a métrica *Precision* determina o quão preciso o algoritmo foi em determinar quais classes eram positivamente esperadas daquelas

realmente positivas.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Conforme é mostrado na Equação 2.1, o valor da *Precision* é mais próximo de 1 quanto menor for o número de FP, ou seja, quando uma determinada classe foi identificada equivocadamente como sendo de outra classe.

Já a métrica *Recall*, por sua vez, indica quantas classificações positivas esperadas estavam realmente corretas.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Conforme mostra a Equação 2.2, a métrica *Recall* é mais sensível à presença de FN, especialmente quando a presença dessa métrica (FN) é mais prejudicial ao modelo quando comparada a um FP.

Quanto a métrica *F1-Score*, também conhecida como *F-Measure*, possibilita entender (com apenas uma métrica), tanto a *Precision* quanto a *Recall*, já que se trata de uma harmônica entre essas duas métricas, conforme mostra a Equação 2.3.

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.3}$$

Por considerar tanto a *Precision* quanto a *Recall*, valores de *F1-Score* próximos de 1 sugerem que o algoritmo não é eficiente na predição das classes abordadas.

Outra métrica utilizada, é a taxa de acerto. Ela considera as instâncias que foram classificadas corretamente. Ou seja, tomando a classe A como exemplo, a taxa de acerto indica a porcentagem que o classificador classificou tal classe corretamente. Portanto, se houverem 10 classes como estas no *dataset*, e o algoritmo identificou 9, a taxa de acerto é igual a 0,900 ou 90%.

Por fim, existem as matrizes de confusão. Essa representação é uma maneira visual de enxergar quão bem um algoritmo de classificação desempenhou seu papel. Ele compila em uma matriz as estatísticas obtidas no experimento. É interessante visualizar a contagem destes grupos tanto em números absolutos quanto em porcentagens da classe real, já que o número de exemplos em cada classe pode variar.

Para mostrar que o modelo ML construído obteve ótimos resultados, os valores da diagonal indicada pela cor verde devem ser próximos de 1 (ou 100%). Uma vez que é nesta diagonal que os valores de TP e TN são mostrados. Nesse caso, os resultados obtidos com as métricas *Precision* e *Recall*, e consequemente, *F1-Score*, também são bons.

Tabela 2.3: Matriz de confusão genérica

		Valores Reais		
		Positivo	Negativo	Total
Valores previstos	Positivo	TP	FP	$TP + FN$
	Negativo	FN	TN	$FN + TN$
	Total	$TP + FN$	$FP + TN$	N

2.5 ENGENHARIA DE ATRIBUTOS (*FEATURE ENGINEERING*)

A engenharia de atributos (FE – *Feature Engineering*) é uma abordagem comum de pré-processamento de dados no contexto de re-conhecimento de padrões, estatísticas, aprendizado de máquina, mineração de dados, bioinformática e processamento de imagens [33, 34]. Consiste no processo de selecionar, extrair e transformar os recursos mais relevantes dos dados disponíveis para construir modelos de aprendizado de máquina mais precisos e eficientes.

O processo de classificação demanda a utilização de um conjunto de descritores que são submetidos a análises. No entanto, depara-se frequentemente com o desafio da maldição da dimensionalidade [35], em que o número de descritores ultrapassa o número de amostras, resultando em grupos que podem parecer semelhantes para os algoritmos. Para mitigar essa problemática, é recomendável que o tamanho do conjunto de dados seja pelo menos 10 vezes maior que o número de descritores, conforme indicado por Jain et al. [36].

Por outro viés, Hua et al. [37] destacam que, para um conjunto de elementos representado por n, é aconselhável manter no máximo n 1 descritores não correlacionados. Em casos de descritores correlacionados, os autores sugerem um limite próximo a $\sqrt{n}$.

As técnicas de FE envolvem o uso de algoritmos de seleção de atributos que podem ser classificados em três categorias principais: métodos intrínsecos, métodos de filtro e métodos *wrapper*.

No método *wrapper*, os procedimentos consistem na seleção de descritores por meio de um algoritmo de treinamento, visando identificar a combinação mais eficaz entre todos os descritores disponíveis. Essa propriedade singular não é proporcionada por outros métodos. Contudo, a principal limitação dessa abordagem reside no custo computacional associado à análise exaustiva de todas as combinações possíveis.

Por sua vez, na categoria de métodos embutidos, as técnicas incorporam a seleção de descritores ao processo de classificação, buscando realizar o aprendizado dos melhores descritores em cada execução. Entretanto, a desvantagem primordial está no custo computacional decorrente do processo de identificação dos descritores.

Por fim, as técnicas da categoria filtro são aplicadas de maneira independente à etapa de classificação. Nesse contexto, os descritores são selecionados com base em análises estatísticas e de correlação, as quais proporcionam resultados significativos para avaliar a relevância da informação contida no vetor de características.

Este trabalho baseia-se no uso de métodos de filtro para seleção de recursos que geralmente são téc-

nicas de pré-processamento que consideram independentemente cada recurso do conjunto de dados. Ele implementa seu modelo em cada recurso e depois avalia qual pode ser usado para analisar seu impacto em um modelo preditivo. Tais métodos incluem por exemplo técnicas como: *Information Gain, Correlation,* e *Correlation based Feature Selection*(CFS) etc.

O principal objetivo da seleção de atributos em um *dataset* é escolher, dentro de um conjunto de dados completo, os melhores recursos para uma determinada atividade fim. No caso de um IDS, a atividade fim seria a predição de um pacote malicioso ou não. O uso de FE permite uma redução do custo de aprendizado e a otimização de determinadas medidas de desempenho [9].

O processo de seleção de atributos na FE constitui-se de quatro etapas fundamentais: geração de um subconjunto de dados, avaliação do subconjunto de dados, definição de *stopping criteria* e validação dos resultados. Estas etapas são ilustradas na Figura 2.7.

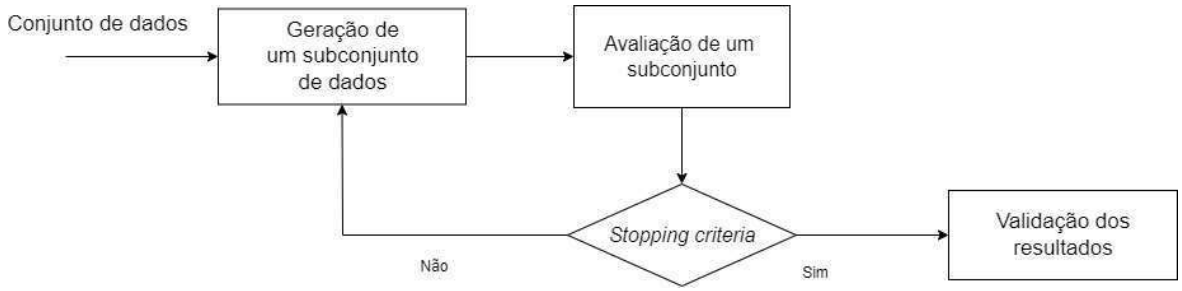


Figura 2.7: Etapas de seleção de atributos.

De acordo com a estratégia de busca utilizada, a primeira etapa da seleção de atributos (geração de subconjunto de recursos) escolhe um subconjunto que será avaliado usando um critério de avaliação [38]. De acordo com o algoritmo escolhido, ele poderá criar um *ranking* dos melhores atributos, como faz o *Information Gain* ou já dar como resultado apenas as variáveis importantes, como ocorre no CFS.

Na segunda etapa, o subconjunto de dados escolhido na etapa anterior é comparado iterativamente com o melhor subconjunto de dados encontrado anteriormente. A depender do resultado desta comparação, este subconjunto é substituído ou não. O processo de geração e avaliação de subconjuntos de dados é repetido até que o critério de parada seja atingido (terceira etapa do processo). Por fim, um método de Inteligência Artificial (IA) é introduzido durante a validação para verificar a eficácia dos subconjuntos de recursos gerados [38].

No entanto, selecionar o subconjunto ideal de recursos é uma tarefa difícil por duas razões principais (34). A primeira delas é que o objeto processado é frequentemente estruturado de maneira complexa. Isso significa dizer que há um grande número de dimensões atreladas aos *datasets*, muitas delas fruto de recursos que são variáveis provenientes de ruído [34].

Neste trabalho, haverá o uso de três classificadores diferentes, abordados logo a seguir. São eles: *Information Gain*, *Correlation* e CFS.

2.5.1 *INFORMATION GAIN*

A métrica *Information Gain* ou ganho de informação, calcula a entropia de cada atributo de maneira separada. A partir da Equação 2.4, pode-se inferir que quanto maior a entropia, mais informações o recurso carrega [39]. Por classificar os atributos de maneira ranqueada, todos aqueles que possuírem maior valor de entropia serão utilizados no processo seguinte ao da mineração de dados, pois isso reduz o ruído existente no conjunto de dados [40].

O cálculo da entropia neste processo de filtragem é visto logo a seguir:

$$I(X) = \sum_{i=1}^{k} P(h_i) log_2 P(h_i) \tag{2.4}$$

em que $P(h_i)$ representa a probabilidade anterior de X.

No caso de algoritmos de árvore de decisão (como por exemplo, *Random Forest*), o ganho de informação para uma divisão é calculado subtraindo as entropias ponderadas de cada ramo da entropia original. Ao treinar uma árvore de decisão usando essas métricas, a melhor divisão é escolhida maximizando o ganho de informação.

2.5.2 *CORRELATION*

Uma métrica bem conhecida para determinar o quão comparáveis duas características aleatórias são é *Correlation* [41]. Ela averigua se dois atributos possuem associação linear bidirecional entre elas. Caso haja essa associação, o valor 1 é apresentado. Caso contrário, coeficiente é determinado pelo valor zero [(42]. Para determinar este coeficiente de correlação, a covariância das duas variáveis (x,y) é dividida pel soma de seus desvios padrão, como mostra a Equação 2.5 [43] para o caso de *n* classes. a

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.5}$$

Este tipo de seletor analisa o valor de um atributo medindo a correlação entre ele e a classe estudada. Os atributos nominais são considerados valor a valor, tratando cada um deles como um indicador. A correlação geral para um determinado atributo nominal é obtida por meio de uma média ponderada.

2.5.3 *CORRELATION BASED FEATURE SELECTION SUBSET EVAL*

O *Correlation Based Feature Selection* (*cfsSubsetEval*) é um algoritmo de filtragem proposto por Hall em 1999 [44]. O algoritmo calcula a correlação média entre cada recurso e classe de rótulo, além de cada recurso em cada subconjunto obtido. Para a validação da escolha do recurso-chave, o CFS escolhe a maior correlação entre o subconjunto de recursos e a coluna de rótulo [45].Quando menor for a redundância

com outros recursos, maior o valor de avaliação obtido. A Equação 2.6 elucida o processo de escolha dos recursos:

$$M_s = \frac{kr_{cf}}{\sqrt{k + (k-1)r_{ff}}} \tag{2.6}$$

em que M_s representa uma avaliação de um subconjunto de S recursos, contendo k itens de recursos; r_{cf} é a média correlação entre diferentes recursos e classes de rótulos, e r_{ff} é a correlação média entre um atributo e outro.

De acordo com a Equação 2.6, o CFS pode, com precisão e eficiência, avaliar a importância de cada subconjunto de recursos nos dados definido e pode ser usado como uma base importante para seleção.

2.6 WEKA

WEKA é uma plataforma de *software* de código aberto construída em Java e utilizado para tarefas envolvendo mineração de dados e inteligência artificial. Por isso, contém ferramentas para preparação de dados, classificação, visualização e muito mais, conforme ilustrado na Figura 2.8 (12).

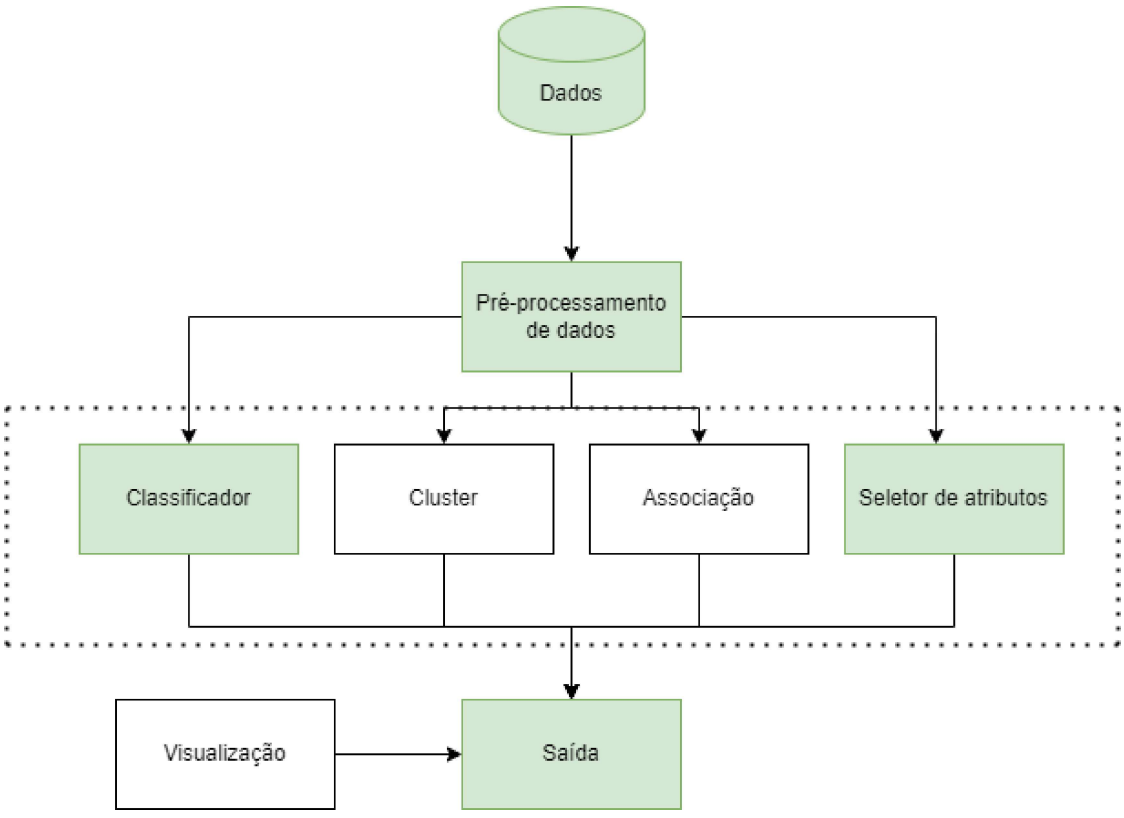


Figura 2.8: Esquematização do WEKA e seus componentes de interesse para realização do experimento.

2.7 TRABALHOS RELACIONADOS

Na literatura, há uma grande coleção de artigos que abordam o uso de algoritmos de aprendizado de máquina para identificar anomalias de rede.

C. Chen *et al*. [46] também utiliza o classificador SVM, mas em linha com dois outros algoritmos (PSO e GWO) capazes de otimizar parâmetros SVM. A utilização da combinação PSOGWO-SVM obteve melhores resultados de acurácia do que aqueles obtidos com SVM em sua versão "pura". No entanto, métricas de desempenho de modelos de construção e teste não são apresentadas, dificultando uma análise sob a perspectiva de ganho ou perda computacional relacionado ao uso desse algoritmo.

Por outro lado, A. Singha *et al*.[47] seleciona atributos do conjunto de dados NSL-KDD usando o teste Qui-quadrado. Os algoritmos de classificação são variados, incluindo o SVM. Contextualizando o cenário, o SVM não obteve os melhores resultados entre os classificadores utilizados. Dos quatro, foi melhor apenas do que *Naive Bayes*. Como o foco deste trabalho é realizar testes utilizando o SVM como único classificador, a abordagem deste trabalho será a utilização de outras técnicas de seleção de atributos, para inferir se o uso do SVM é adequado ao conjunto de dados NSL-KDD.

Na etapa de pré-processamento de dados, G. Yedukondalu *et al*.[48] apenas removeram dados do conjunto de dados que não eram numéricos. De acordo com os autores, esses dados de cadeia de caracteres não agregam nenhum valor no processo de previsão, portanto, poderiam ser ignorados. Mas essa remoção não ocorreu com o uso da Engenharia de Atributos. Isso pode explicar os resultados bem abaixo do esperado com o uso da SVM. A acurácia dos testes com esse tipo de classificador não ultrapassou 50%.

M. D. Rokade e Y. K. Sharma [49] usaram uma versão menor do conjunto de dados NSL-KDD em seu trabalho. O objetivo foi inferir que o uso do VMS é mais apropriado em relação ao *Random Forest*, *Naive Bayes* entre outros. Isso aconteceu. No entanto, vale ressaltar que o número de instâncias utilizadas não foi grande o suficiente para se ter melhores conclusões. Além disso, parte das seções do artigo foi focada em dois conjuntos de dados. Além do NSL-KDD, há também o KDD99, uma versão anterior ao NSL-KDD e que trazia redundância e valores repetidos.

Uma característica comum a todos esses trabalhos referenciados [49,50] é que todos adotam um modelo de aprendizado de máquina supervisionado.

Quanto ao uso de *Random Forest* como algoritmo de classificação de dados, Ahmed e Varol [51] investigaram cinco tecnologias de filtro aplicadas à seleção de atributos. Das 78 características disponíveis no *dataset*, 19 são utilizadas no processo de treinamento e teste do modelo, que usa, além do *Random Forest*, os algoritmos *Naive Bayes* e *BayesNet*.

Este trabalho propõe o *framework* FE-ML-IDS (*Feature Engineering-Machine Learning Intrusion Detection System*) para fins de investigação do desempenho de IDS baseado em modelos de aprendizado de máquina supervisionado e usando técnicas de engenharia de recursos (*feature engineering*). Os *datasets* utilizados são o NSL-KDD e o CIC-IDS-2017. Três técnicas de FE diferentes são usadas para selecionar atributos: *Information Gain, Correlation* e *Correlation based feature subset selection (CfsSubsetEval)*, uma versão simplificada do CFS. Por sua vez, as classificações são feitas através dos algoritmos SVM e *Random Forest*. A plataforma de implementação do *framework* FE-ML-IDS e de configuração dos experi-

mentos de avaliação de desempenho é o *software* WEKA (*Waikato Environment for Knowledge Analysis*) [12].

2.7.1 Considerações sobre a Revisão Bibliográfica

Dentre os trabalhos relacionados, os que se aproximam em termos de metodologia e abordagem são [50], [52] e [53].

S. Thirimanne et al. [50] tem uma abordagem muito semelhante à que é utilizada neste trabalho. Ele usa um dos conjuntos de dados usados neste trabalho (NSL-KDD) e usa o SVM como método classificador, além de expor as métricas de avaliação. A diferença entre as duas produções é a seleção de atributos. Não há menção a essa etapa no trabalho de Thirimanne, uma vez que a preparação de dados apontada por ele remove dados duplicados e transforma dados em valores numéricos.

Um dos principais desafios na construção de IDSs é a confiabilidade e a rapidez de processamento para atuar em tempo real. Esses dois problemas são abordados na obra de A. Ghorbani e S. M. Fakhrahmad [52]. O IDS proposto possui um codificador automático supervisionado que, ao final do processo, gera uma matriz mostrando a importância de cada tarefa na classificação. Os autores alegam que essa supervisão reduz a complexidade do classificador, já que fornece informações mais precisas, o que facilita a solução de problemas decorrentes de um ataque.

A. Thakkar e R. Lohiya [53] usam a arquitetura de *Deep Neural Network* (DNN) para implementar um IDS. É uma estrutura de rede neural multicamadas que realiza transformações matemáticas em dados de entrada para construir padrões para classificação. A seleção de atributos é feita a partir do uso da Seleção de Traços recursiva utilizando Fusão de Desvio Padrão e Diferença Absoluta de Média e Mediana, como proposta de melhoria, bem como o uso de outros seletores, como a Correlação. Além do NSL-KDD, ele usa outros conjuntos de dados, como: UNBW_NB-15 e CIC-IDS-2017.

Outro trabalho importante é do Felipe Barreto [54]. Aborda a seleção de atributos no *dataset* NSL-KDD, com inferência a redução de dimensionalidade do conjunto de dados. A diferença é que o cenário envolve IoT (Internet of Things), e o experimento é realizado em ambiente simulado.

3 MODELO PROPOSTO

Com o intuito de prover um sistema para investigar o desempenho de sistemas de detecção de intrusões construídos a partir de técnicas de *machine learning* e engenharia de atributos, este trabalho propõe o *framework* FE-ML-IDS (*Feature Engineering-Machine Learning Intrusion Detection System*) implementado na plataforma WEKA.

O *framework* FE-ML-IDS permite a construção de IDSs baseado em mineração de dados e aprendizado de máquina utilizando técnicas aprimoradas de *Feature Engineering*, conforme configuração esquemática apresentada na Figura 3.1.

As funcionalidades e componentes principais do *framework* FE-ML-IDS, tais como *dataset*, seleção de recursos, algoritmo classificador, treinamento e teste, são detalhados a seguir.

A tabela 3.1 resume as principais ferramentas usadas para a implementação do *framework* FE-ML-IDS.

Tabela 3.1: Ferramentas utilizadas no experimento

Ferramenta	**Descrição**
Computador	MacBook Air M2
Software	WEKA 3.8.6
Dataset 1	NSL-KDD
Dataset 2	CIC-IDS-2017

3.1 ESTRUTURAÇÃO DOS *DATASETS*

São utilizados dois *datasets* diferentes para a validação do *framework* proposto: NSL-KDD e CIC-IDS-2017. Essa estratégia permite averiguar que a seleção de atributos pode ser realizada em diferentes conjuntos de dados e que sua utilização não está condicionada a apenas um deles.

3.1.1 NSL-KDD

A princípio, o *dataset* NSL-KDD é estruturado como mostra a Figura 3.2. Dois arquivos, todos no formato ARFF, são destinados a treinos, enquanto outros dois são voltados para testes. Vale ressaltar que um dos arquivos de treinamento (KDDTrain+_20Percent) é reduzido a 20% do original. Isso permite que desenvolvedores obtenham resultados mais rápidos (devido à menor quantidade de informações a ser processada), seja por falta de insumos como recursos computacionais ou até mesmo tempo disponível.

O número de instâncias e classes existentes no dataset KDDTrain+_20Percent é descrito na Tabela 3.3. Ao todo, existem mais de 25 mil dados no dataset, que serão divididos durante os treinamentos. Entretanto, embora exista essa grande quantidade de instâncias na coleção estudada, apenas parte desta 19 (entre 20 e

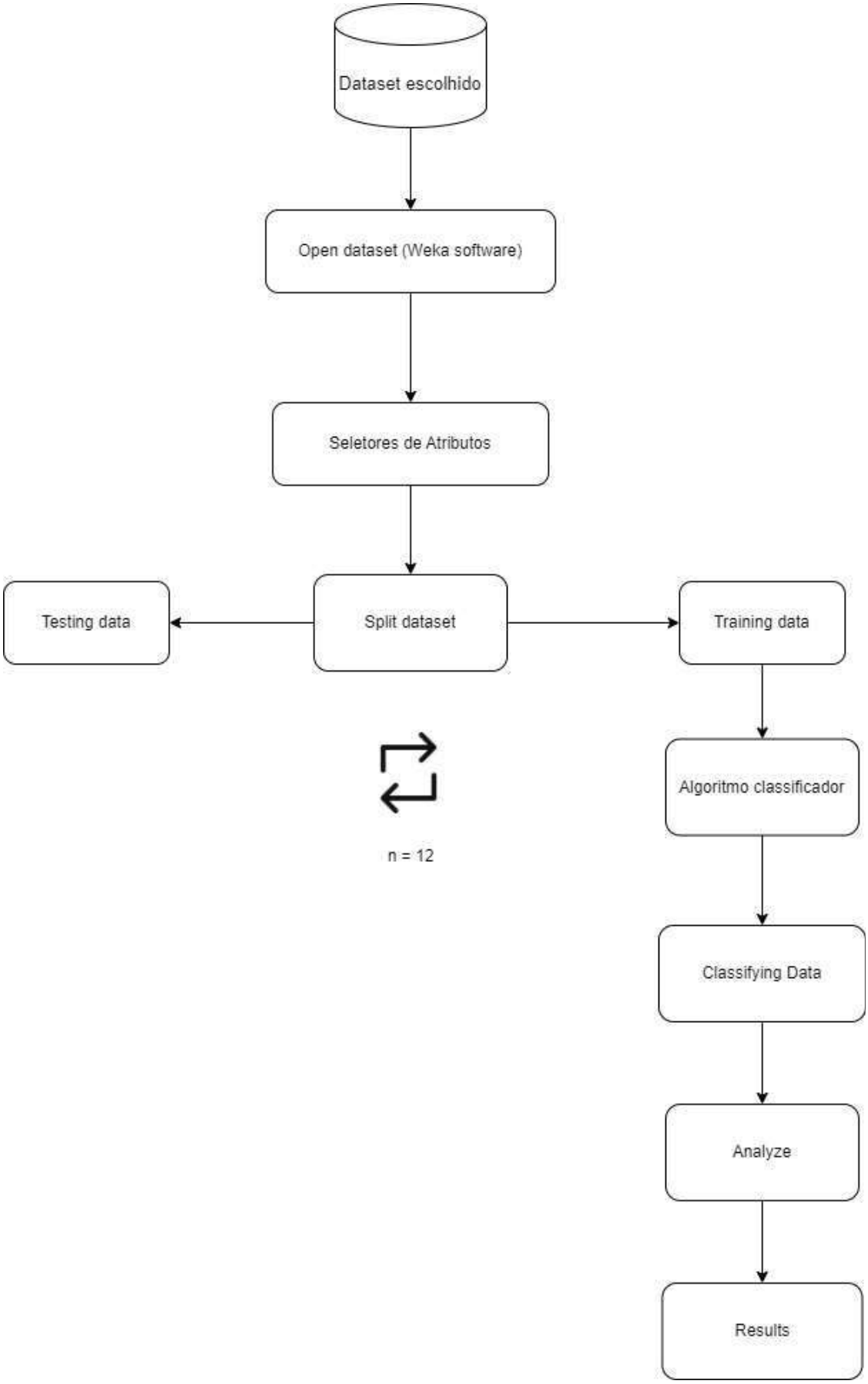


Figura 3.1: Configuração do *framework* FE-ML-IDS implementado na plataforma WEKA, em que $n = 12$ representa o número de rodadas feitas para cada experimento.

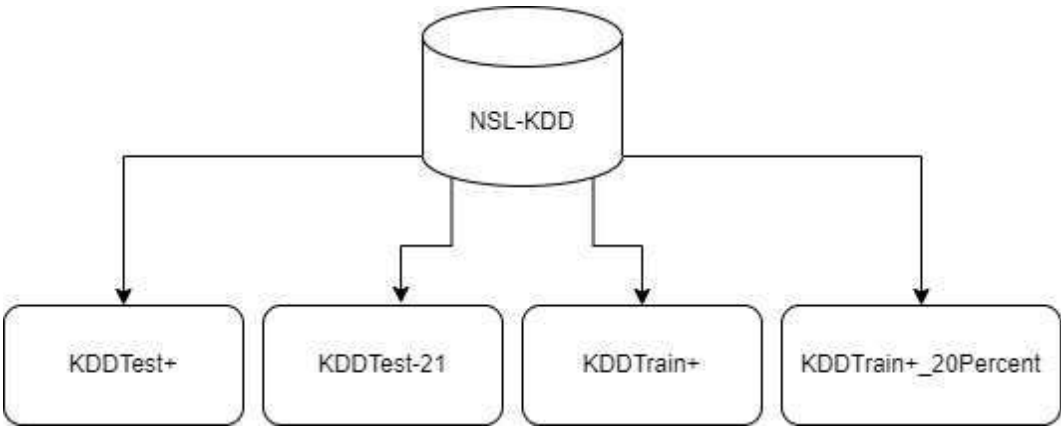


Figura 3.2: Configuração do *dataset* NSL-KDD.

40%) será mostrada nas matrizes de confusão.

Tabela 3.2: Informações do *dataset* NSL-KDD

Rótulo	Número de instâncias
Anomaly	11.743
Normal	13.449

Um aspecto importante relacionado a um *dataset* é o número de colunas existentes nele. No caso do NSL-KDD/KDDTrain+_20Percent, existem 41 atributos diferentes somado a um quadragésimo segundo que classifica o tipo de pacote como sendo fruto de um fluxo anômalo de rede e outro como normal.

3.1.2 CIC-IDS-2017

A princípio, o *dataset* CIC-IDS-2017 é estruturado como mostra a Figura 3.3. Este *dataset* é formado por sub pastas que contém os arquivos com dados tabulados. Estas sub pastas são os dias da semana que os dados foram coletados pelo distribuidor do conjunto de dados.

Os arquivos, separados por período do dia, estão no formato CSV ou *Packet Capture* (PCAP). Para este trabalho, o período da manhã de quinta-feira foi escolhido para o processo de seleção de atributos e treinamento do agente. A Tabela 3.3 mostra como esse arquivo é dividido.

Tabela 3.3: Informações do *dataset* CIC-IDS-2017

Tipo de ataque	Número de instâncias
BENIGN	189.067
Bot	1.966

Um aspecto importante relacionado a um *dataset* é o número de colunas existentes nele. No caso do CIC-IDS-2017 / *Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX 2*, existem 77 atributos dife-

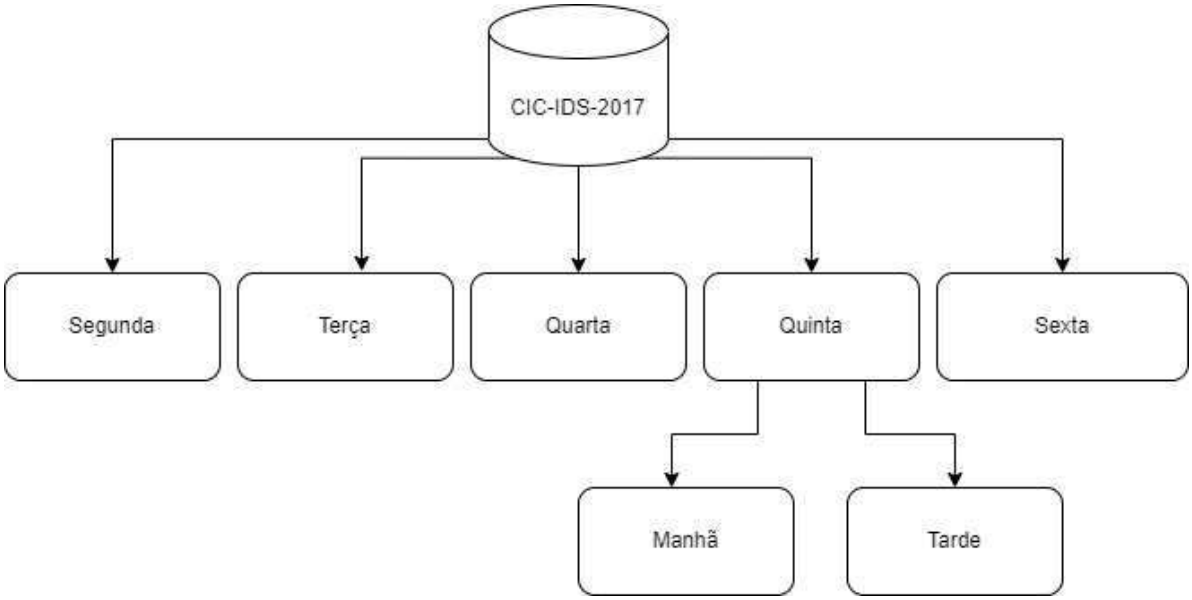


Figura 3.3: Configuração do *dataset* CIC-IDS-2017.

rentes, acrescido da classe ao qual o pacote pertence.

Comparando as Tabelas 3.2 e 3.3, é possível notar que a divisão dos rótulos não é equilibrada. Enquanto que no NSL-KDD/ *KDDTrain+_20Percent*, existe um balanceamento entre o que é considerado anomalia e o que é tráfego normal, o CIC-IDS-2017/*Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX 2* possui uma assimetria no que tange a distribuição de pacotes considerados inofensivos (*BENIGN*) daqueles que representam um ataque real (*Bot*).

3.2 PREPARAÇÃO DOS DADOS

Como dito anteriormente, ambos *datasets* possuem formatos de arquivos diferentes. Como está sendo utilizado o *software* WEKA, é necessário que a entrada desses arquivos esteja na extensão que o programa aceite. No caso, ARFF. Por conta disso, a preparação dos dados com NSL-KDD é mais simples, uma vez que ele já é distribuído neste formato.

Quanto ao CIC-IDS-2017, é necessário fazer a conversão do arquivo. Este processo é feito diretamente diretamente pelo WEKA.

3.3 PRÉ-PROCESSAMENTO DOS DADOS

A etapa de pré-processamento dos dados é feita mediante utilização das técnicas de *Feature Engineering*. Dessa forma, depois de ter aberto o *dataset* no WEKA, é feita uma filtragem nas informações que serão utilizadas no processo de treinamento.

É importante destacar que existem dois modos de seleção de atributos. O primeiro deles é a criação de um *ranking* e o segundo de um classificador. No *ranking*, todas as colunas do *dataset* são mostradas ao final da seleção dos atributos. Estes dados são apresentados de maneira a mostrar aqueles que possuem maior valor de entropia, vide as equações apresentadas no Capítulo 2. O método de ranqueamento é usado tanto pelo *Information Gain* quanto pelo *Correlation*. Para o trabalho, foram consideradas apenas os dados com valor de entropia maior que a média aritmética simples de todas as entropias dos rórulos.

Já a técnica *Correlation based feature subset selection (CfsSubsetEval)* utiliza o método de classificação, onde o resultado final mostra apenas as variáveis importantes. Desse modo, não há necessidade em escolher uma forma de utilizar os atributos, como nos anteriores (onde o critério é valor de entropia maior que a média das entropias). O próprio filtro já devolve o que deve ser considerado.

3.3.1 NSL-KDD / KDDTrain+_20Percent

Como pode ser visto nas Tabelas de 3.4 a 3.6, os 41 atributos foram diminuídos a ponto de ter entre 6 e 10 atributos selecionados. Portanto, durante o pré-processamento, os demais dados serão removidos, no intuito de dinamizar o processo de treinamento e testes. É importante destacar que esses atributos selecionados são utilizados no treinamento, somente.

Tabela 3.4: Atributos selecionados pelo *Information Gain*

#No	Atributo
1	*Service*
2	*Flag*
3	*Src_bytes*
4	*Dst_bytes*
5	*Same_srv_rate*
6	*Diff_srv_rate*
7	*Dst_host_srv_count*
8	*Dst_host_same_srv_rate*

Tabela 3.5: Atributos selecionados pelo *Correlation*

#No	Atributo
1	*Flag*
2	*Logged_in*
3	*Count*
4	*Serror_rate*
5	*Srv_serror_rate*
6	*Same_srv_rate*
7	*Dst_host_srv_count*
8	*Dst_host_same_srv_rate*
9	*Dst_host_srv_serror_rate*
10	*Dst_host_serror_rate*

Tabela 3.6: Atributos selecionados pelo CFS

#No	Atributo
1	*Flag*
2	*Src_bytes*
3	*Dst_bytes*
4	*Logged_in*
5	*Srv_serror_rate*
6	*Diff_srv_rate*

3.3.2 CIC-IDS-2017

No caso do *dataset* CIC-IDS-2017/Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX 2 os 77 atributos são diminuídos a ponto de ter entre 3 e 12 atributos, conforme a técnica de seleção de atributos aplicada, como pode ser visto nas Tabelas 3.7 a 3.9. Portanto, durante o pré-processamento, os demais dados são removidos, no intuito de dinamizar o processo de treinamento e testes.

Tabela 3.7: Atributos selecionados pelo *Information Gain*

#No	Atributo
1	*Init_Win_bytes_backward*
2	*Fwd IAT Min*
3	*Fwd Packets/s*
4	*Fwd Header*
5	*Fwd Header Length*
6	*Flow Packets/s*
7	*Fwd IAT Mean*
8	*Flow IAT Std*
9	*Flow IAT Mean*
10	*Flow IAT Max*
11	*Fwd IAT Std*
12	*Bwd Packets/s*

3.4 DIVISÃO DOS *DATASETS*

Outro ponto importante do trabalho, é entender como a divisão do *dataset* interfere na qualidade das estatísticas obtidas após os testes. Neste caso, optou-se por seccionar em três modelos de divisão diferentes. São elas:

- 60% para treinamento e 40% para testes;
- 70% para treinamento e 30% para testes;

Tabela 3.8: Atributos selecionados pelo *Correlation*

#No	Atributo
1	*Init_Win_bytes_backward*
2	*PSH Flag Count*
3	*Init_Win_bytes_forward*
4	*Down/Up Ratio*
5	*min_seg_size_forward*

Tabela 3.9: Atributos selecionados pelo CFS

#No	Atributo
1	*Fwd Packet Length Mean*
2	*Fwd IAT Min*
3	*Init_Win_bytes_backward*

- 80% para treinamento e 20% para testes;

Quanto maior for a razão, mais dados do *dataset* são utilizados para treinamento. Desse aumento são esperadas duas coisas. A primeira delas é que a taxa de acertos nas classificações aumente, uma vez que mais dados estão sendo incorporados no processo de aprendizagem.

O segundo movimento esperado é que o tempo de construção do modelo aumente, uma vez que há mais dados a serem processados pelo algoritmo. De maneira inversa, o tempo de testagem do modelo diminui, devido a diminuição dos dados destinados a esta finalidade.

3.5 BALANCEAMENTO *UNDERSAMPLING*

Comparando as Tabelas 3.2 e 3.3, é possível perceber que existe um desbalanceamento nos dados inseridos no CIC-IDS-2017/Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX 2, uma vez que a categoria que simboliza o ataque é quase 100 vezes menor que aquela que representa um fluxo de rede normal.

Dessa forma, é necessário aplicar uma das técnicas vistas no referencial teórico deste trabalho (Fig. 2.6), afim de que se mostre a importância de sua utilização no processo de treinamento e classificação de ataques do IDS proposto. Todos os procedimentos são feitos a partir do WEKA, sem necessidade de *software* terceiro.

Após a realização do balanceamento *undersampling* do dataset, o número de instâncias que representam um fluxo normal (*BENIGN*) de rede cai para 1.966, assim como já existia para o ataque *Bot*, como mostra a Tabela 3.10.

Tabela 3.10: Informações do *dataset* CIC-IDS-2017

Tipo de ataque	**Número de instâncias**
BENIGN	1.966
Bot	1.966

3.6 BALANCEAMENTO *OVERSAMPLING*

Além do *undersampling*, existe também o *oversampling* como método de balanceamento de *datasets*. Ao contrário do que ocorre na técnica abordada anteriormente, nesta a classe minoritária se estende de maneira a ter número de instâncias parecido com a classe majorirária. Todos os procedimentos são feitos a partir do WEKA, sem necessidade de *software* terceiro.

Após a realização do balanceamento *oversampling* do *dataset*, o número de instâncias que representam um fluxo anormal de rede aumenta para 125.824. Esse número é aproximadamente igual ao número de um fluxo normal (*BENIGN*) de rede. Esses dados podem ser vistos na Tabela 3.11.

Tabela 3.11: Informações do *dataset* CIC-IDS-2017

Tipo de ataque	**Número de instâncias**
BENIGN	189.067
Bot	125.824

Ao empregar a estratégia de *oversampling*, que consiste em ampliar a quantidade de instâncias de uma classe minoritária em conjuntos de dados desequilibrados, é imperativo considerar os potenciais impactos na descaracterização do conjunto de dados. No âmbito deste estudo, uma vez que alguns experimentos foram conduzidos sem a utilização dessa técnica de balanceamento, a realização dessas análises proporcionou insights relevantes sobre como a mencionada técnica influencia a distribuição e as características originais do conjunto de dados. Esses *insights*, por sua vez, desempenharam um papel fundamental na tomada de decisões fundamentadas acerca da aplicação dessa estratégia. Para tal avaliação, procedeu-se à comparação dos resultados de classificação, bem como das métricas de desempenho previamente abordadas neste trabalho.

3.7 TREINAMENTO E CLASSIFICAÇÃO

Definidas as divisões do *dataset*, bem como a filtragem dos atributos considerados necessários pelos algoritmos de seleção, a próxima etapa é de treinamento e classificação.

Por conta do uso do WEKA, esse processo é feito de maneira sequencial, ou seja, a partir do momento que o algoritmo de classificação é acionado, a saída de todo esse processo já contém os dados de tempo de construção e testagem do modelo, bem como os resultados estatísticos do experimento, como: *Precision*,

Recall, *F1-Score* e matrizes de confusão.

A fim de obter resultados com um nível de confiança satisfatório, cada teste foi conduzido em 12 instâncias. A repetição dos experimentos desempenha um papel crucial na atenuação de viés relacionado a eventos específicos que podem ter ocorrido em uma única execução. Assegurar que os resultados permaneçam consistentes ao longo de múltiplas repetições contribui para diminuir o impacto de eventos fortuitos. Os resultados apresentados neste estudo refletem a média aritmética simples dessas iterações. Devido à extensiva quantidade de repetições, subdivisões do *dataset*, diferentes conjuntos de dados e filtros selecionados, foram realizadas 1152 rodadas de testes.

À semelhança da coleta de métricas de desempenho em cada experimento, procedeu-se da mesma maneira com os tempos de construção e teste dos modelos gerados nesse processo. Todos os resultados são derivados do *software* WEKA e a média é calculada para a apresentação dos resultados na próxima seção.

4 ANÁLISE DOS DADOS E RESULTADOS

Os resultados de desempenho obtidos com o *framework* FE-ML-IDS em termos de eficiência de classificação de pacotes e das métricas de desempenho usuais, como: *Precision*, *Recall* e *F1-Score* (16), são comparados com aqueles obtidos com a estratégia de classificação usando todos os atributos contidos no conjunto de dados escolhido (*Network Security Laboratory - Knowledge Discovery in Databases* NSL-KDD ou CIC-IDS-2017).

A configuração de experimentação para avaliação de desempenho de FE foi baseada no software *workbench* WEKA (versão 3.8.6 com tamanho de heap de 8086 MB) com os *datasets* NSL-KDD/*KDDTrain+_20* e CIC-IDS-2017/*Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX 2* com as classes e instâncias apresentadas nas Tabelas 3.2 e 3.3.

Os dois algoritmos de classificação ML estudados são: *Support Vector Machine* (SVM) e *Random Forest*. Como o SVM não é um algoritmo nativo do WEKA, foi necessário baixar previamente a biblioteca LibSVM para poder importar o SVM para o programa.

Os três seletores de atributos estudados são: *Information Gain*, *Correlation* e *cfssubseteval*, cujas características de seleção de atributos foram apresentadas nas Tabelas 3.4 a 3.9. Além disso, três diferentes divisões do *dataset* foram utilizados, onde os números indicam, em números percentuais, a quantidade do banco usada respectivamente para treinamento e teste (60-40, 70-30 e 80-20).

Os quatro processos de treinamento e teste (incluindo aquele que carrega todos os recursos), cada um explorando todas as três taxas de divisão teste-treinamento, foram repetidos 12 vezes em sequência para mitigar resultados inesperados de desempenho devido a variações não controláveis no processo de ML. Em suma, 1152 simulações foram realizadas durante os experimentos. Para fins de análise, será considerada o intervalo de confiança para nível de confiança igual a 95%.

Os resultados de desempenho geral foram calculados por meio de uma média aritmética simples dos resultados parciais de cada rodada de treinamento e teste. As métricas utilizadas para a avaliação de desempenho foram:

- Tempo para construção do modelo;
- Tempo para testagem do modelo;
- Métricas de desempenho de classificação: *Precision, Recall* e *F1-Score* e Taxa de Acerto.

Todos os experimentos de simulação foram executados em um computador com as seguintes especificações: processador Apple M2 3.5 GHz, 8 GB de RAM, sistema operacional macOS Ventura.

4.1 *DATASET* NSL-KDD (*KDDTRAIN+_20PERCENT)*

Para o NSL-KDD, o primeiro algoritmo de classificação utilizado foi o SVM. Por ser um método que utiliza recursos computacionais em demasia, o que requer um tempo maior para a construção de modelos e testes, o arquivo utilizado neste processo é o reduzido (KDDTrain+_20Percent), que possui 20% do tamanho original do conjunto de dados. Isso permite reduzir os tempos de simulação sem no entanto comprometer a generalização dos resultados uma vez que todas as técnicas de engenharia de atributos são aplicadas ao mesmo *dataset*.

4.1.1 Algoritmo classificador SVM

4.1.1.1 Experimento NSL-KDD/SVM/*Information Gain*

Em relação ao experimento com o uso do *Information Gain* como seletor de atributos, dos 41 atributos do dataset KDDTrain+_20Percent, 8 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na Tabela 3.4. Para esta configuração de experimento, os resultados de desempenho de classificação em termos das métricas de TP e FP são mostrados na Tabela 4.1 e, em termos das métricas *Precision, Recall* e *F1-Score*, na Tabela 4.2.

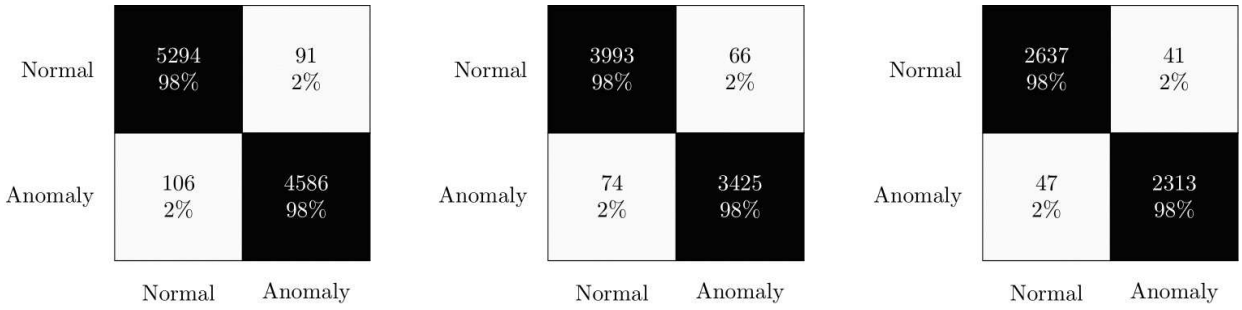
Tabela 4.1: TP e FP do experimento 4.1.1.1

Split	Class	*True Positive*	*False Positive*
60	Normal	0,983	0,023
	Anomalia	0,977	0,017
70	Normal	0,984	0,021
	Anomalia	0,979	0,016
80	Normal	0,985	0,020
	Anomalia	0,980	0,015

Tabela 4.2: *Precision*, *Recall* e *F1-Score* do experimento 4.1.1.1

Split	Class	*Precision*	*Recall*	*F1-Score*
60	Normal	0,980	0,981	0,983
	Anomalia	0,981	0,977	0,979
70	Normal	0,982	0,984	0,983
	Anomalia	0,981	0,979	0,980
80	Normal	0,982	0,985	0,984
	Anomalia	0,983	0,980	0,981

As matrizes de confusão obtidas com o experimento NSL-KDD/SVM/*Information Gain* são apresentadas na Figura 4.1. É possível observar que o experimento apresentou bons resultados, uma vez que a diagonal principal mostra índices de TP e TN próximos de 100%. Observa-se também que os diferentes modelos de divisão treinamento-teste não impactam os resultados de desempenho de classificação.



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.1: Matriz de confusão do experimento 4.1.1.1.

4.1.1.2 Experimento NSL-KDD/SVM/*Correlation*

Em relação ao experimento NSL-KDD/SVM/*Correlation*, dos 41 atributos do dataset KDDTrain+_-20Percent, 10 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na Tabela 3.5. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos em termos das métricas de TP e FP são mostrados na Tabela 4.3 e, em termos das métricas *Precision, Recall* e *F1-Score*, na Tabela 4.4.

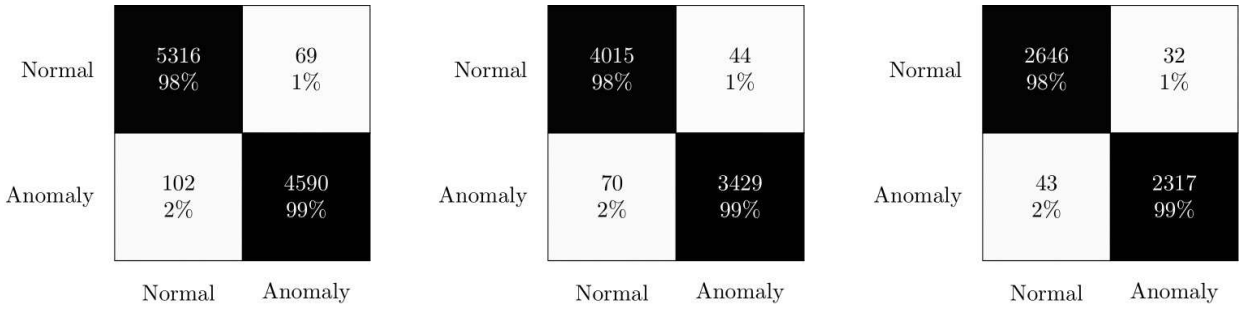
Tabela 4.3: TP e FP do experimento 4.1.1.2

Split	Class	*True Positive*	*False Positive*
60	Normal	0,969	0,097
	Anomalia	0,903	0,031
70	Normal	0,971	0,088
	Anomalia	0,912	0,029
80	Normal	0,975	0,092
	Anomalia	0,908	0,025

Tabela 4.4: *Precision*, *Recall* e *F1-Score* do experimento 4.1.1.2

Split	Class	*Precision*	*Recall*	*F1-Score*
60	Normal	0,919	0,969	0,943
	Anomalia	0,962	0,903	0,931
70	Normal	0,927	0,971	0,949
	Anomalia	0,964	0,912	0,937
80	Normal	0,924	0,975	0,948
	Anomalia	0,969	0,908	0,938

As matrizes de confusão obtidas com o experimento NSL-KDD/SVM/*Correlation* são apresentadas na Figura 4.2. É possível, através dela, mensurar que houve bons resultados com os experimentos dessa rodada de testes, uma vez que a diagnoal principal está em destaque, indicando que os índices de TP e TN foram satisfatórias.



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.2: Matriz de confusão do experimento 4.1.1.2.

A diferença nos resultados entre TP e FP entre o uso do *Information Gain* e *Correlation* não foi significativa.

O fato das diferentes razões de divisão treinamento-teste não impactarem os resultados de desempenho de classificação sugere que o uso de razões mais balanceadas (e.g., 60-40) é mais adequado quando se busca reduzir o tempo de construção de modelos ML.

4.1.1.3 Experimento NSL-KDD/SVM/*cfssubseteval*

Em relação ao experimento com o uso do NSL-KDD/SVM/*cfssubseteval* como seletor de atributos, dos 41 atributos do conjunto de dados, 6 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na Tabela 3.6. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.5.

Tabela 4.5: TP e FP com o experimento 4.1.1.3

Split	Class	*True Positive*	*False Positive*
60	Normal	0,953	0,007
	Anomalia	0,993	0,047
70	Normal	0,957	0,007
	Anomalia	0,993	0,043
80	Normal	0,955	0,007
	Anomalia	0,993	0,045

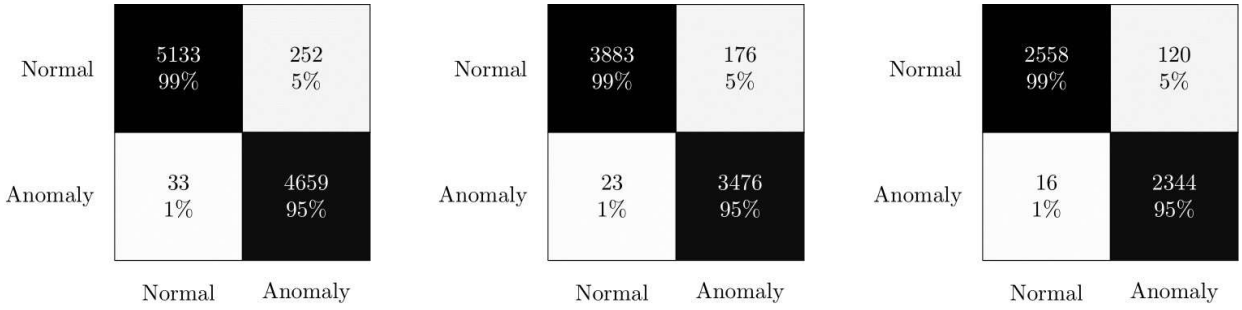
As matrizes de confusão são apresentadas na Figura 4.3.

Analisando as tabelas, nota-se que os resultados para o *F1-Score* deste experimento são muito próximos daquele que utilizou o *Information Gain*, mesmo adotando menos atributos (6 contra 8).

Comparando com os dois seletores anteriores, o *cfssubseteval* foi o único que de alguma forma melhorou os resultados de classificação de TP e FP.

Tabela 4.6: *Precision*, *Recall* e *F1-Score* do experimento 4.1.1.3

Split	Class	*Precision*	*Recall*	*F1-Score*
60	Normal	0,994	0,953	0,973
	Anomalia	0,949	0,993	0,970
70	Normal	0,994	0,957	0,975
	Anomalia	0,952	0,993	0,972
80	Normal	0,994	0,955	0,974
	Anomalia	0,951	0,993	0,972



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.3: Matriz de confusão do experimento 4.1.1.3.

Dessa forma, valendo-se do tempo de construção de modelos, a escolha pelo *cfssubseteval* é mais adequada, pois esse processo é mais rápido sem afetar de maneira drástica os valores para *F1-Score*.

4.1.1.4 Experimento NSL-KDD/SVM/Sem filtragem

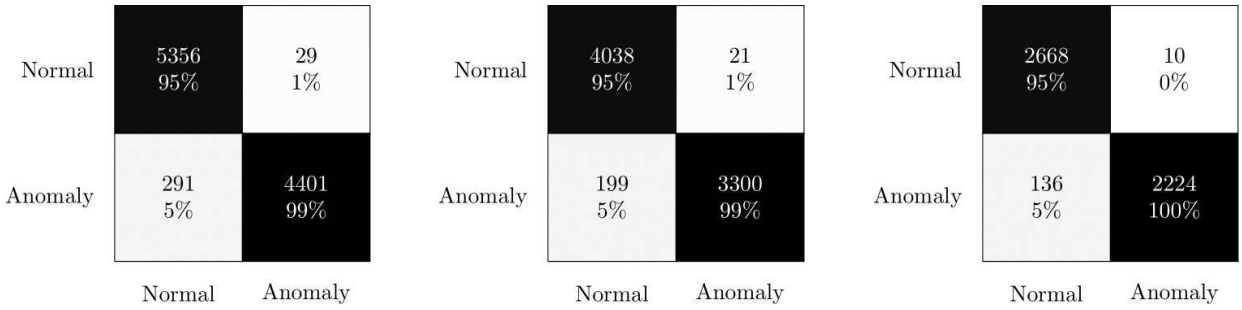
O último experimento NSL-KDD/SVM/ Sem Seleção de Atributos (*All Features*) não teve seleção de atributos, ou seja, o dataset é utilizado integral-mente. Como esperado, a taxa de acerto foi alta, mas não o suficiente para tornar a métrica *F1-Score* superior àquela obtida com o seletor CFS, que funcionou com apenas 6 dos 41 atributos. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na 4.7 e, em termos das métricas *Precision, Recall* e *F1-Score*, na Tabela 4.8.

Tabela 4.7: TP e FP do experimento 4.1.1.4

Split	Class	*True Positive*	*False Positive*
60	Normal	0,995	0,062
	Anomalia	0,938	0,005
70	Normal	0,995	0,057
	Anomalia	0,943	0,005
80	Normal	0,996	0,058
	Anomalia	0,942	0,004

Tabela 4.8: *Precision*, *Recall* e *F1-Score* do experimento 4.1.1.4

Split	Class	*Precision*	*Recall*	*F1-Score*
60	Normal	0,948	0,995	0,971
	Anomalia	0,993	0,938	0,965
70	Normal	0,953	0,995	0,973
	Anomalia	0,994	0,943	0,968
80	Normal	0,951	0,996	0,973
	Anomalia	0,996	0,942	0,968



(a) *Split* de 60-40 (b) *Split* de 70-30 (c) *Split* de 80-20

Figura 4.4: Matriz de confusão do experimento 4.1.1.4.

As matrizes de confusão deste experimento são apresentadas na Figura 4.4.

É possível observar que o experimento NSL-KDD/SVM/Sem Seleção de Atributos apresentou resultados relativos ao índice TP inferiores (95%) quando comparados por exemplo com aqueles obtidos com uso de seletores, Information Gain (98%), Correlation (98%) e CFS (99%).

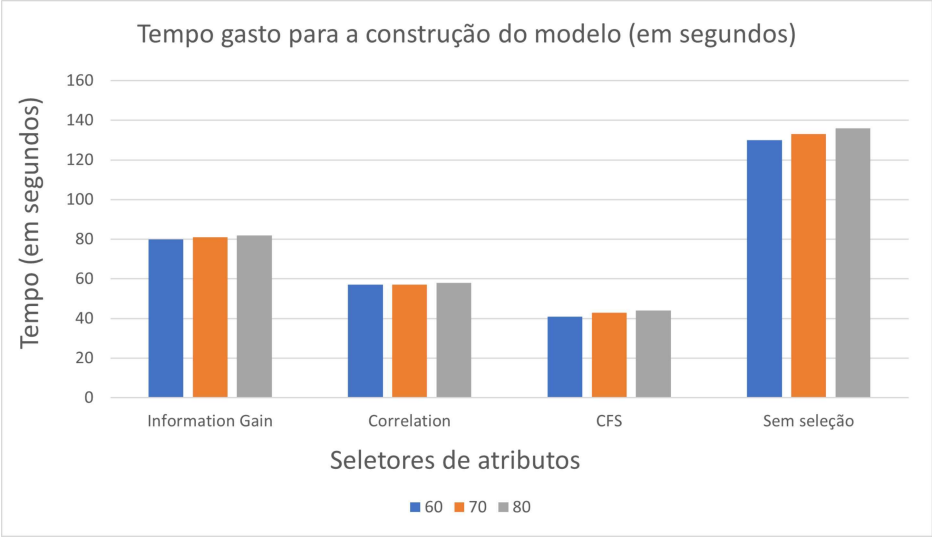
Por outro lado, os resultados em termos do índice FP (99%) mostraram-se iguais ou superiores aos obtidos com o uso do seletor Information Gain (98%), Correlation (99%) e CFS (95%).

Observa-se também que a razão de divisão treinamento-teste 80-20 tende a melhorar um pouco os resultados de desempenho de classificação.

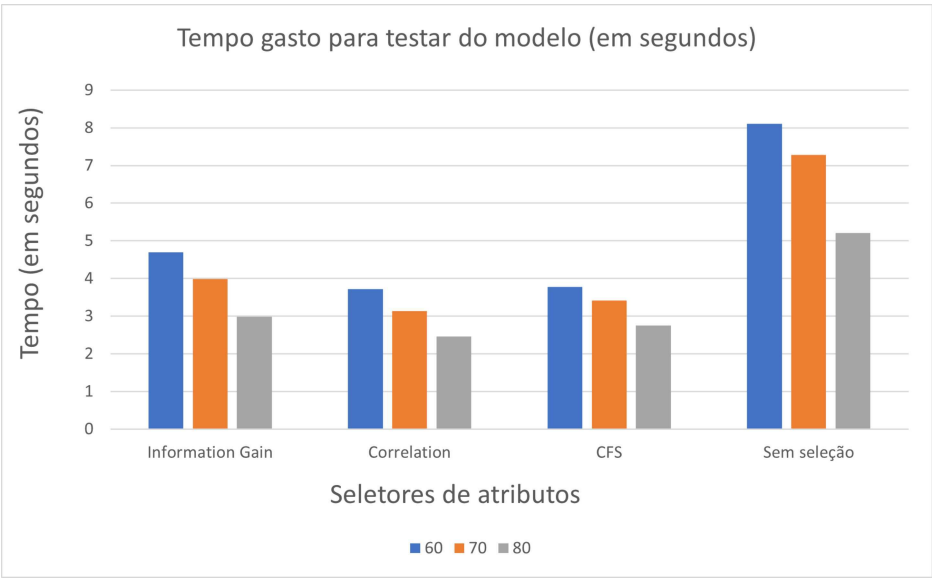
4.1.1.5 Tempos de processamento dos modelos nos experimentos NSL-KDD/SVM

Outra forma de avaliar o desempenho da aplicação da engenharia de atributos no *framework* FE-ML-IDS é enxergar como isso impacta o tempo de processamento na construção e testagem dos modelos. Observando a Figura 4.5, é possível entender como a redução de atributos foi positiva no que tange ao tempo de processamento para construção e testagem dos modelos.

Em cada cenário com o mesmo seletor de atributos nota-se que a razão de divisão treinamento-teste altera o tempo de processamento para constru-ção de modelos ML. Diminuindo a quantidade de dados a serem computados no treinamento, implica em um menor tempo de construção dos modelos ML. O mesmo ocorre na testagem. O tempo gasto para testar o modelo também diminui à medida que a quantidade de



(a) Tempo gasto para a construção do modelo (em segundos).



(b) Tempo gasto para o teste do modelo (em segundos)

Figura 4.5: Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos para o Experimento NSL-KDD/SVM.

dados voltados para testes decai.

Outra forma de avaliar os experimentos é enxergar como a taxa de erros e acertos nas classificações se comporta de um modo geral. A Tabela 4.9 ilustra esse nível de acertos, que é o diagnóstico correto da classe a que o pacote pertence. Além disso, é possível inferir pela Figura 4.5, que a diminuição do número de features, leva também a uma diminuição do tempo de processamento na construção e testagem dos modelos. E que neste caso, sugere que o uso do CFS, mesmo que não tenha obtido os melhores resultados de classificação (vide Tabela 4.9), dado o menor tempo de processamento, algo imprescindível no contexto de mitigação de ataques à rede.

Tabela 4.9: Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.1.1

Split	***Information Gain***	***Correlation***	***cfssubseteval***	***All features***
60	0,980	0,937	0,971	0,968
70	0,981	0,943	0,973	0,970
80	0,982	0,943	0,973	0,971

4.1.2 Algoritmo classificador *Random Forest*

Assim como no experimento feito a partir da técnica de SVM, o uso do *Random Forest* neste contexto também está atrelado as mesmas configurações vistas na seção anterior. Logo, os mesmos parâmetros são utilizados nesta rodada, seja os filtos utilizados ou das divisões do *dataset*.

4.1.2.1 Experimento NSL-KDD/*Random Forest*/*Information Gain*

Em relação ao experimento NSL-KDD/Random Forest/*Information Gain* como seletor de atributos, dos 41 atributos do conjunto de dados, 10 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na Tabela 3.5. Para esta configuração de experimento, os resultados de desempenho de classificação em termos das métricas de TP e FP são mostrados na Tabela 4.10 e, em termos das métricas Precision, Recall e F1-Score, na Tabela 4.11.

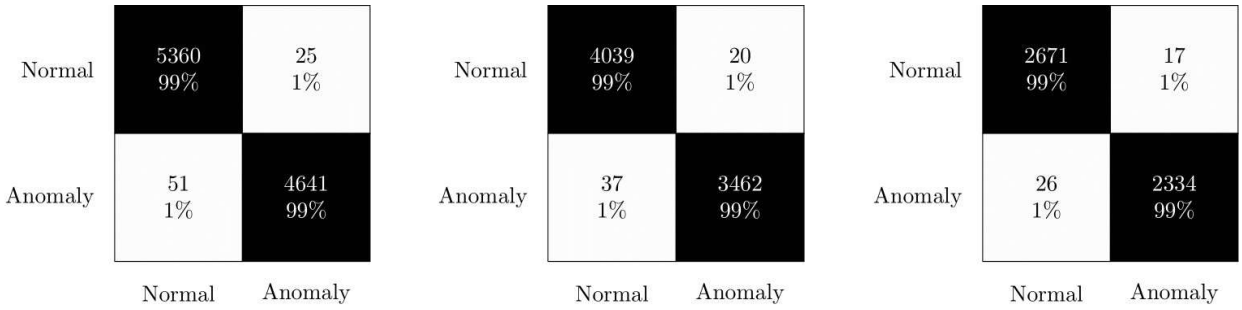
Tabela 4.10: TP e FP do experimento 4.1.2.1

Split	Class	*True Positive*	*False Positive*
60	Normal	0,995	0,005
	Anomalia	0,995	0,005
70	Normal	0,998	0,005
	Anomalia	0,995	0,002
80	Normal	0,996	0,003
	Anomalia	0,997	0,004

As matrizes de confusão são apresentadas na Figura 4.6. É possível, através dela, mensurar que houve bons resultados com os experimentos dessa rodada de testes, uma vez que a diagnoal principal está em

Tabela 4.11: *Precision*, *Recall* e *F1-Score* do experimento 4.1.2.1

Split	Class	*Precision*	*Recall*	*F1-Score*
60	Normal	0,996	0,995	0,995
	Anomalia	0,994	0,995	0,995
70	Normal	0,996	0,998	0,997
	Anomalia	0,997	0,995	0,996
80	Normal	0,997	0,996	0,997
	Anomalia	0,996	0,997	0,996



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.6: Matriz de confusão do experimento 4.2.1.1.

destaque, indicando que os índices de TP e TN foram satisfatórias.

4.1.2.2 Experimento NSL-KDD/*Random Forest*/*Correlation*

Em relação ao experimento NSL-KDD/Random Forest/*Correlation* como seletor de atributos, dos 41 atributos do conjunto de dados, 10 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na Tabela 3.4. Para esta configuração de experimento, os resultados de desempenho de classificação em termos das métricas de TP e FP são mostrados na Tabela 4.12 e, em termos das métricas Precision, Recall e F1-Score, na Tabela 4.13.

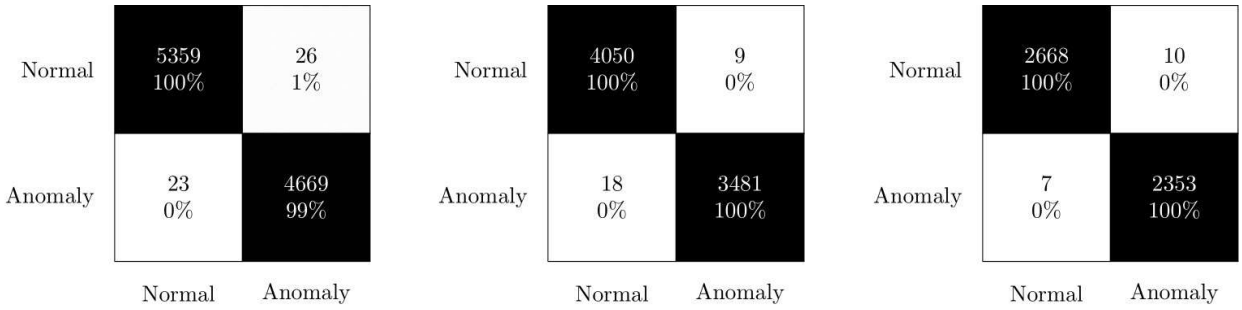
Tabela 4.12: TP e FP do experimento 4.1.2.2

Split	Class	*True Positive*	*False Positive*
60	Normal	0,975	0,040
	Anomalia	0,960	0,025
70	Normal	0,974	0,039
	Anomalia	0,961	0,026
80	Normal	0,976	0,039
	Anomalia	0,961	0,024

Observa-se das Tabelas 4.10 e 4.12 que a diferença nos resultados de TP e FP no uso dos seletores

Tabela 4.13: *Precision*, *Recall* e *F1-Score* do experimento 4.1.2.2

Split	Class	*Precision*	*Recall*	*F1-Score*
60	Normal	0,965	0,975	0,970
	Anomalia	0,970	0,960	0,965
70	Normal	0,967	0,974	0,970
	Anomalia	0,970	0,961	0,965
80	Normal	0,966	0,976	0,971
	Anomalia	0,973	0,961	0,967



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.7: Matriz de confusão do experimento 4.1.2.2.

Information Gain e Correlation foi rela-tivamente significativa. Por exemplo, no caso de split 60-40, os índices TP obtidos para tráfego normal foram 99,5% e 97,5%, já os índices FP foram 0,5% e 4%, respectivamente. Em ambos os casos, a razão de divisão dos datasets não implicou em melhorias significativas nos resultados.

As matrizes de confusão são apresentadas na Figura 4.7. É possível, através dela, mensurar que o experimento obteve bons resultados, uma vez que a diagonal principal mostra índices de TP e TN próximos de 100%. Observa-se também que os diferentes modelos de divisão treinamento-teste não impactam os resultados de desempenho de classificação.

4.1.2.3 Experimento NSL-KDD/*Random Forest*/cfssubseteval

Em relação ao experimento com o uso do cfssubseteval como seletor de atributos, dos 41 atributos do conjunto de dados, 6 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.6. Não houve destaque das colunas, uma vez que os resultados foram muito parecidos para qualquer divisão do *dataset*.

Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.14.

Analisando as tabelas 4.14 e 4.15 é possível inferir que os resultados são muito próximos, independentemente da razão de divisão do dataset escolhida. Nesses casos, a opção por modelos que requerem

Tabela 4.14: TP e FP do experimento 4.1.2.3

Split	Class	*True Positive*	*False Positive*
60	Normal	0,995	0,011
	Anomalia	0,989	0,005
70	Normal	0,995	0,011
	Anomalia	0,989	0,005
80	Normal	0,994	0,011
	Anomalia	0,989	0,006

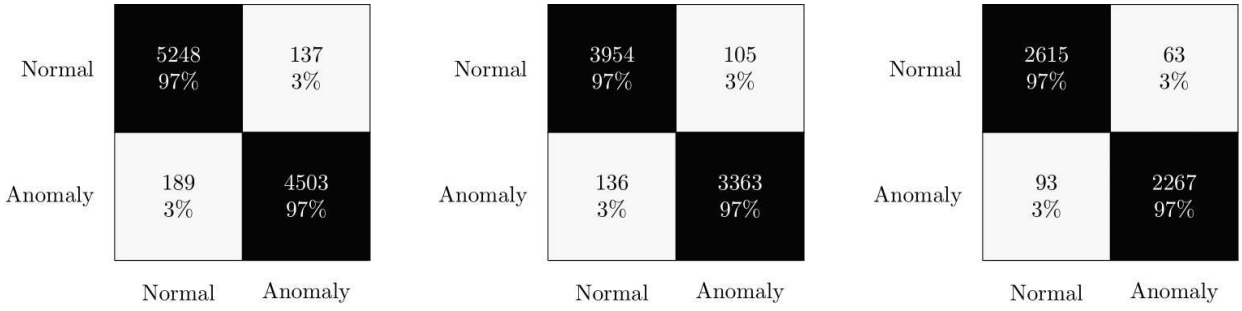
Tabela 4.15: *Precision*, *Recall* e *F1-Score* do experimento 4.1.2.3

Split	Class	*Precision*	*Recall*	*F1-Score*
60	Normal	0,991	0,995	0,993
	Anomalia	0,995	0,989	0,993
70	Normal	0,991	0,995	0,993
	Anomalia	0,994	0,989	0,992
80	Normal	0,994	0,995	0,994
	Anomalia	0,951	0,993	0,972

menos tempo de processamento tende a ser mais vantajosa.

Dentre os três experimentos NSL-KDD/Random Forest usando seletores de atributos, os resultados para o *F1-Score* são melhores com o *Information Gain*, mas com uma diferença relativamente pequena ao do CFS. Para esses casos, novamente, a escolha entre menor tempo de construção acaba sendo mais decisiva.

As matrizes de confusão são apresentadas na Figura 4.8. É possível observar que o experimento NSL-KDD/*Random Forest*/*Information Gain* apresentou bons resultados, uma vez que a diagonal principal mostra índices de TP e TN da ordem de 97%. Observa-se também que os diferentes modelos de divisão treinamento-teste não impactam significativamente os resultados de desempenho de classificação.



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.8: Matriz de confusão do experimento 4.1.2.3.

4.1.2.4 Experimento NSL-KDD/*Random Forest*/Sem filtragem

não teve seleção de atributos, ou seja, o dataset foi utilizado integral-mente. Como esperado, a taxa de acertos foi alta, mas não o suficiente para tornar a métrica *F1-Score* superior àquela obtida com o seletor CFS, que funcionou com apenas 6 dos 41 atributos. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.16. e, em termos das métricas *Precision, Recall* e *F1-Score*, na Tabela 4.17. As colunas não foram destacadas, haja vista a similaridade e equivalências entre as divisões.

Tabela 4.16: TP e FP do experimento 4.1.2.4

Split	Class	*True Positive*	*False Positive*
60	Normal	0,998	0,004
	Anomalia	0,996	0,002
70	Normal	0,999	0,003
	Anomalia	0,997	0,001
80	Normal	0,999	0,003
	Anomalia	0,997	0,001

Tabela 4.17: *Precision*, *Recall* e *F1-Score* do experimento 4.1.2.4

Split	Class	*Precision*	*Recall*	*F1-Score*
60	Normal	0,997	0,998	0,997
	Anomalia	0,997	0,996	0,997
70	Normal	0,997	0,999	0,998
	Anomalia	0,999	0,997	0,998
80	Normal	0,998	0,999	0,999
	Anomalia	0,999	0,997	0,998

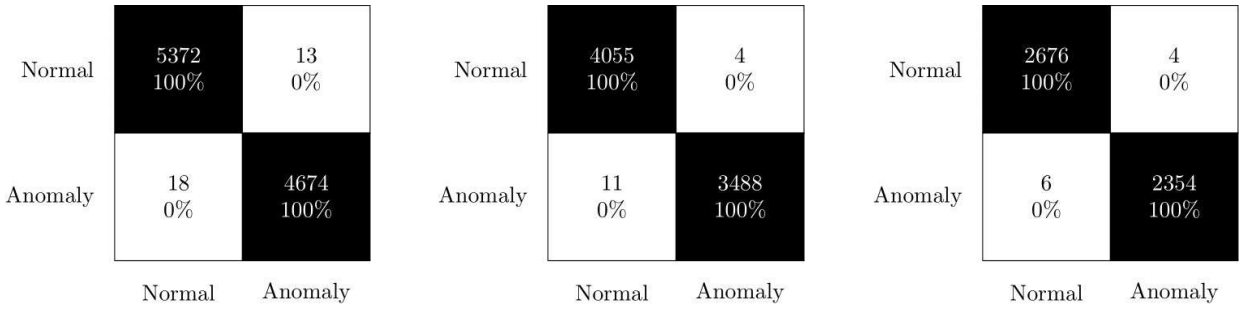
Por conta dos resultados próximos, não há necessidade de destacar as linhas com melhores resultados, uma vez que são muito próximos uns dos outros.

As matrizes de confusão são apresentadas na Figura 4.9. É possível observar que o experimento NSL-KDD/Random Forest/Sem Seleção de Atributos apresentou bons resultados, uma vez que a diagonal principal mostra índices de TP e TN da ordem de 100%. Observa-se também que os diferentes modelos de divisão treinamento-teste não impactam significativamente os resultados de desempenho de classificação.

4.1.2.5 Tempos de processamento dos modelos nos experimentos NSL-KDD/*Random Forest*

Outra forma de avaliar o desempenho da aplicação da engenharia de atributos no framework FE-ML-IDS é enxergar como isso impacta o tempo de processamento na construção e testagem dos modelos.

Os resultados mostrados na Figura 4.10, permitem entender como a redução de atributos afeta os tempos de processamento para a construção e testagem dos modelos.



(a) *Split* de 60-40 (b) *Split* de 70-30 (c) *Split* de 80-20

Figura 4.9: Matriz de confusão do experimento 4.1.2.4.

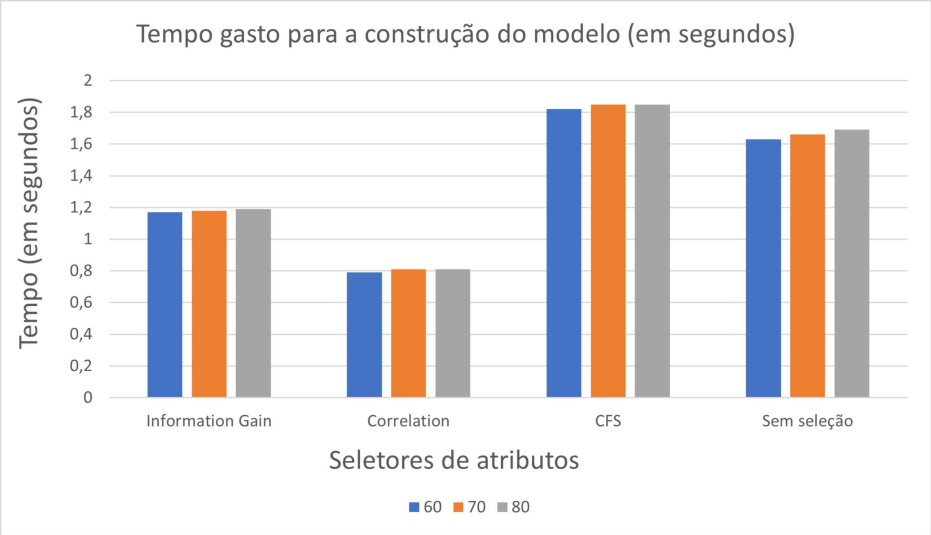
Observa-se por exemplo que o cenário com o seletor Correlation apresentou os menores tempos gastos tanto na construção (Fig. 4.10a) como nos testes dos modelos (Fig. 4.10b). Por outro lado, o cenário com o seletor CFS apresentou um desempenho inferior ao cenário Sem Seleção de Atributos em termos de tempo de construção dos modelos (Fig. 4.10a). Além disso, o cenário com CFS também apresentou um desempenho inferior ao cenário Sem Seleção de Atributos em termos de tempo gasto com testes dos modelos (Fig. 4.10b). Um outro cenário que apresentou tempos de teste superiores aos do cenário com Sem Seleção de Atributos foi o cenário com Information Gain (Fig. 4.10b).

Nota-se também na Figura 4.10 que, para um mesmo cenário de seletor de atributos, a razão de divisão dos datasets altera o processo de construção de modelos, aumentando o tempo de construção à medida que aumentam os dados de treinamento. Por outro lado, em termos da testagem (Fig, 4,10b), observa-se que o tempo gasto para teste do modelo diminui à medida que o número de dados voltados para testes decai.

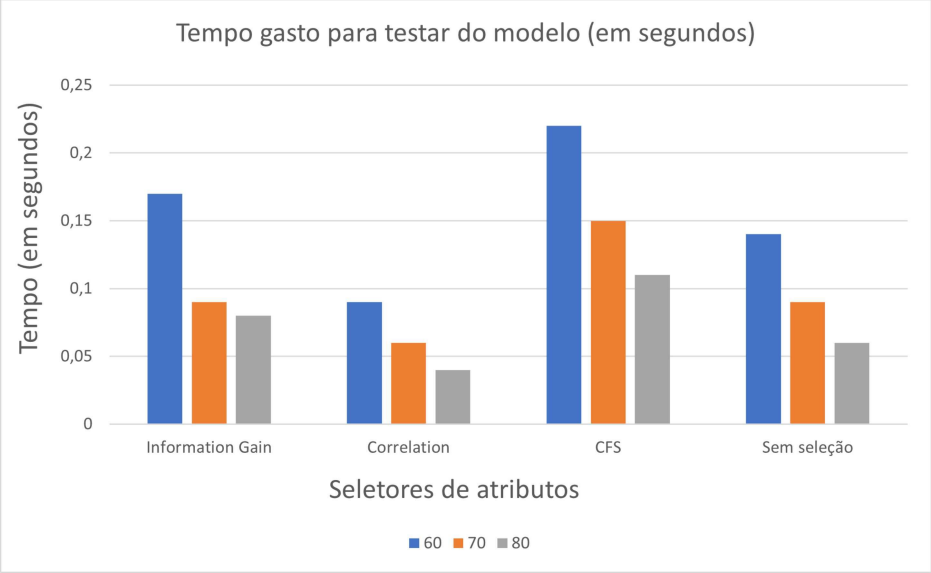
Outra forma de acompanhar os experimentos é enxergar a taxa de erros e acertos nas classificações de um modo geral. A taxa de acertos é definida como instâncias que foram classificadas corretamente, ou seja, se uma classe chamada *A* foi classificada como tal pelo algoritmo, no caso, *Random Forest*. A Tabela 4.18 ilustra esse nível de acertos.

Tabela 4.18: Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.1.2

Split	***Information Gain***	***Correlation***	***cfssubseteval***	***All features***
60	0,992	0,995	0,967	0,996
70	0,992	0,996	0,968	0,998
80	0,991	0,996	0,969	0,998



(a) Tempo gasto para a construção do modelo (em segundos).



(b) Tempo gasto para o teste do modelo (em segundos)

Figura 4.10: Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento NSL-KDD/*Random Forest*.

4.2 *DATASET* CIC-IDS-2017

4.2.1 Algoritmo classificador SVM

4.2.1.1 Experimento CIC-IDS/SVM/*Information Gain*

Em relação ao experimento CIC-IDS/SVM com o *Information Gain* como seletor de atributos, dos 77 atributos do conjunto de dados, 12 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.7. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.19. Já *Precision, Recall* e *F1-Score* são mostrados na Tabela 4.20.

Tabela 4.19: TP e FP do experimento 4.2.1.1

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	1,000	0.640
	Bot	0.360	0,000
70	*BENIGN*	1,000	0.616
	Bot	0.384	0,000
80	*BENIGN*	1,000	0.571
	Bot	0.429	0,000

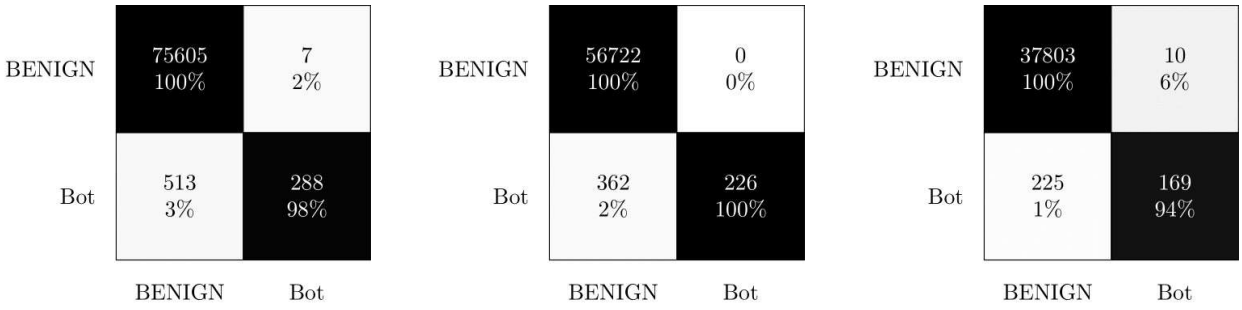
Tabela 4.20: *Precision*, *Recall* e *F1-Score* do experimento 4.2.1.1

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,993	1,000	0,997
	Bot	0,976	0.360	0.526
70	*BENIGN*	0,994	1,000	0,997
	Bot	1,000	0.384	0.555
80	*BENIGN*	0,994	1,000	0,997
	Bot	0,944	0.429	0.590

É possível, das tabelas 4.19 e 4.20, perceber como os resultados obtidos para o ataque são muito baixos, mesmo com a adoção dos filtros de seleção de atributos. Parte desse problema pode ser explicado pelo desbalanceamento do *dataset*, que viesa o resultado. As matrizes de confusão deste experimento são apresentadas na Figura 4.11.

4.2.1.2 Experimento CIC-IDS/SVM/*Correlation*

Em relação ao experimento com o uso do *Correlation* como seletor de atributos, dos 77 atributos do conjunto de dados, 5 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.8. Para esta configuração de experimento, os resultados de desempenho de



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.11: Matriz de confusão do experimento 4.2.1.1.

classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.21.

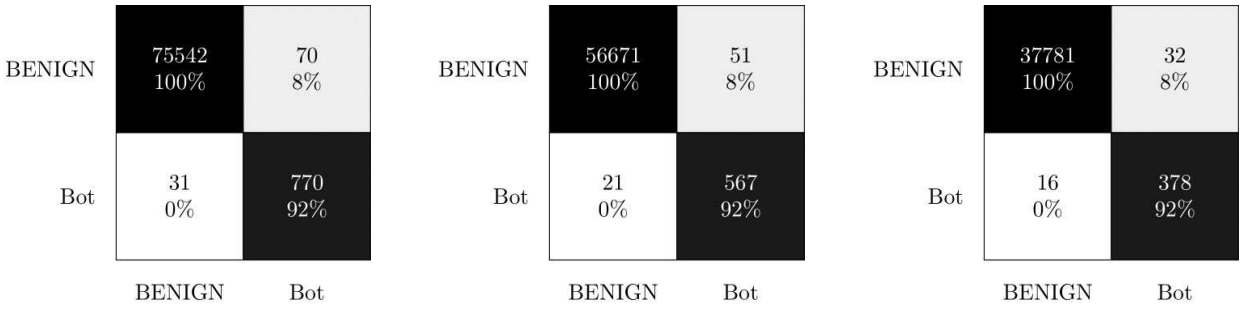
Tabela 4.21: TP e FP do experimento 4.2.1.2

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,999	0,039
	Bot	0,961	0,001
70	*BENIGN*	0,999	0,036
	Bot	0,964	0,001
80	*BENIGN*	0,999	0,041
	Bot	0,959	0,001

Tabela 4.22: *Precision*, *Recall* e *F1-Score* do experimento 4.2.1.2

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	1,000	0,999	0,999
	Bot	0,917	0,961	0,938
70	*BENIGN*	1,000	0,999	0,999
	Bot	0,917	0,964	0,940
80	*BENIGN*	1,000	0,999	0,999
	Bot	0,922	0,959	0,940

A diferença nos resultados entre TP e FP entre o uso do *Information Gain* e *Correlation* foi significativa, principalmente quando se compara os valores de TP para a detecção de anomalias na rede (0,360 contra 0,961). Isso mostra como o uso do filtro correto a depender do tipo de *dataset* utilizado é essencial no processo de classificação. As matrizes de confusão são apresentadas na Figura 4.12. Diferentemente do que ocorreu com o *Information Gain*, a soma de cada uma das colunas deu 100%. Isso se deve ao fato de que parte os resultados não foram viesados a uma determinada classe, como ocorreu com o *Information Gain*.



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.12: Matriz de confusão do experimento 4.2.1.2.

4.2.1.3 Experimento CIC-IDS/SVM/cfssubseteval

Em relação ao experimento com o uso do cfssubseteval como seletor de atributos, dos 77 atributos do conjunto de dados, 3 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.9. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.23 e métricas de *Precision, Recall* e *F1-Score* na Tabela 4.24.

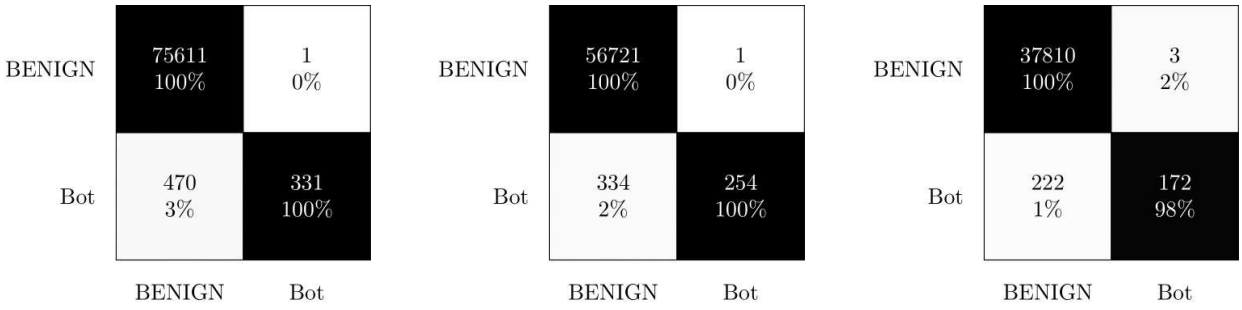
Tabela 4.23: TP e FP do experimento 4.2.1.3

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	1,000	0.587
	Bot	0.413	0,000
70	*BENIGN*	1,000	0.568
	Bot	0.432	0,000
80	*BENIGN*	1,000	0.563
	Bot	0.437	0,000

Tabela 4.24: *Precision*, *Recall* e *F1-Score* do experimento 4.2.1.3

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,994	1,000	0,997
	Bot	0,997	0.413	0.584
70	*BENIGN*	0,994	1,000	0,997
	Bot	0,996	0.432	0.603
80	*BENIGN*	0,994	0,955	0,974
	Bot	0,951	0,993	0,972

Analisando as tabelas, nota-se que os melhores resultados para o *F1-Score* são para os testes feitos com o *Correlation*. Assim como o *Information Gain*, o cfssubseteval não obteve resultados bons quando usado o SVM no *dataset* CIC. Especialmente quando se observa as classificações para os ataques recebidos.



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.13: Matriz de confusão do experimento 4.2.1.3.

É possível, das tabelas 4.23 e 4.24, perceber como os resultados obtidos para o ataque são muito baixos, mesmo com a adoção dos filtros de seleção de atributos. Parte desse problema pode ser explicado pelo desbalanceamento do *dataset*, que viesa o resultado. As matrizes de confusão são apresentadas na Figura 4.13.

4.2.1.4 Experimento CIC-IDS/SVM/Sem seleção

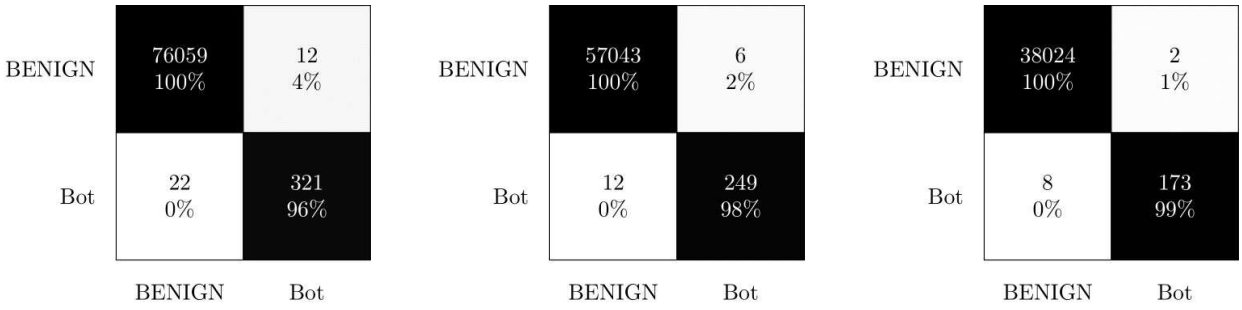
Em relação ao experimento sem seleção alguma de atributos como seletor de atributos, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.25. Já *Precision*, *Recall* e *F1-Score* são colocados na Tabela 4.26.

Tabela 4.25: TP e FP do experimento 4.2.1.4

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,999	0,037
	Bot	0,963	0,001
70	*BENIGN*	0,999	0,024
	Bot	0,976	0,001
80	*BENIGN*	0,999	0,012
	Bot	0,988	0,001

Tabela 4.26: *Precision*, *Recall* e *F1-Score* do experimento 4.2.1.4

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,999	0,999	1,000
	Bot	0,935	0,963	0,957
70	*BENIGN*	0,999	0,999	1,000
	Bot	0,954	0,976	0,963
80	*BENIGN*	0,999	0,999	1,000
	Bot	0,956	0,988	0,972



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.14: Matriz de confusão do experimento 4.2.1.4.

Analisando as tabelas acima, observa-se que quando não houve seleção, o comportamento dos resultados se melhorou a medida que as divisões do *dataset* aumentava. Comparativamente aos outros experimentos, este obteve as melhores métricas para *F1-Score*. Já as matrizes de confusão são mostradas na Figura 4.14.

4.2.1.5 Tempos de processamento dos modelos nos experimentos CIC-IDS-2017/SVM

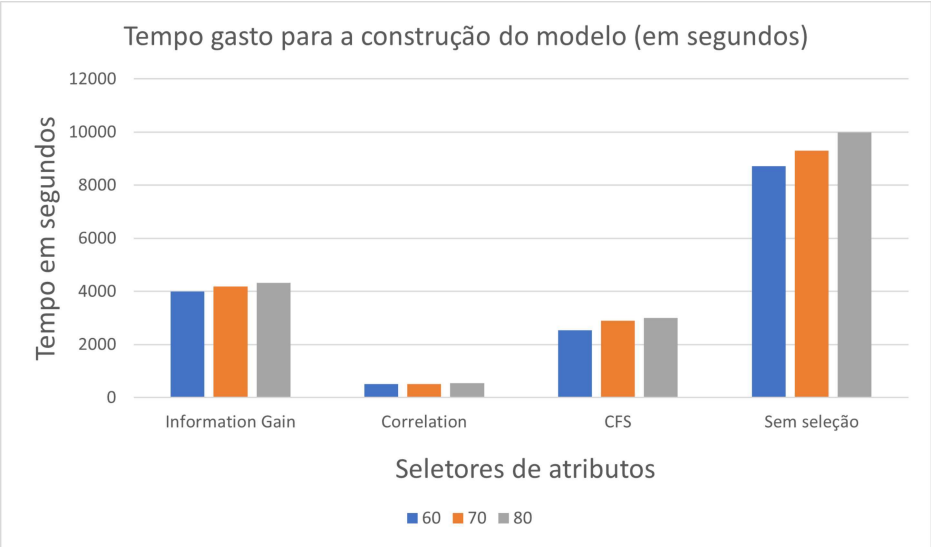
Outras formas de mensurar a qualidade da seleção dos atributos, é enxergar como isso impacta no processo de treinamento e testagem dos modelos. Afinal, caso haja queda nos índices estatísticos estudados, ela deve ser contornada com uma maior agilidade na predição das classes de cada.

Os resultados, porém, indicam apenas que o *Correlation* seria candidato a uso, dada as baixas acertividades com os demais filtros em detectar ataques. Curiosamente, observando as Figuras 4.15a e 4.15b, o *Correlation* obteve os melhores índices perfomáticos que os demais filtros, mesmo obtendo taxas de acerto muito maiores.

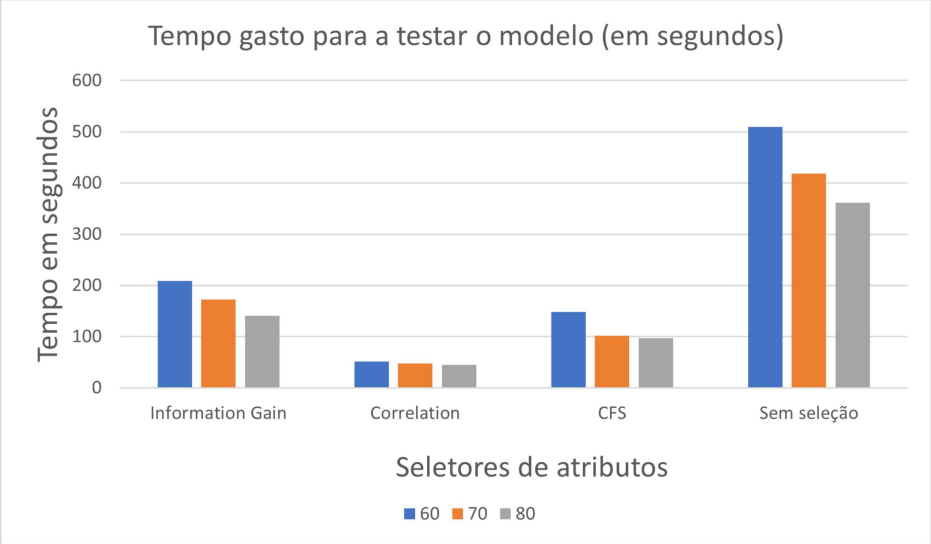
É possível notar também que, a divisão dos *datasets* altera o processo de construção de modelos (no cenário com o mesmo seletor de atributos), ao passo que na testagem esse tempo também diminui a medida que o número de dados voltados para testes decai. Mais uma vez, os índices obtidos pelo *Correlation* foram melhores.

Outra forma de acompanhar os experimentos é enxergar a taxa de erros e acertos nas classificações de um modo geral. A Tabela 4.27 ilustra esse nível de acertos.

Apesar da taxa alta observada, isso denota o impacto de usar um *dataset* desbalanceado. Por ter muitos elementos de uma mesma classe, caso o algoritmo interprete todos os dados como normais, a taxa de acerto continuaria alta, mesmo que os acertos para detecção de ataque fossem zerados, já que o número de instâncias com essa classe é irrisória. Observando esses números isoladamente, entende-se que o uso dos filtros foi sempre muito positiva, o que não foi confirmada com os dados mostrados previamente.



(a) Tempo gasto para a construção do modelo (em segundos).



(b) Tempo gasto para o teste do modelo (em segundos)

Figura 4.15: Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/SVM.

Tabela 4.27: Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.2.1

Split	*Information Gain*	*Correlation*	*cfssubseteval*	*All features*
60	0,993	0,998	0,993	0,999
70	0,993	0,998	0,994	0,999
80	0,993	0,998	0,994	0,999

4.2.2 Algoritmo classificador *Random Forest*

4.2.2.1 Experimento CIC-IDS/*Random Forest*/*Information Gain*

Em relação ao experimento com o uso do *Information Gain* como seletor de atributos, dos 77 atributos do conjunto de dados, 12 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.7. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.28.

Tabela 4.28: TP e FP do experimento 4.2.2.1

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,999	0.245
	Bot	0,755	0,001
70	*BENIGN*	0,999	0.260
	Bot	0,740	0,001
80	*BENIGN*	0,999	0.272
	Bot	0,728	0,001

Tabela 4.29: Resultados de: Precisão, *Recall* e *F1-Score* do experimento 4.2.2.1

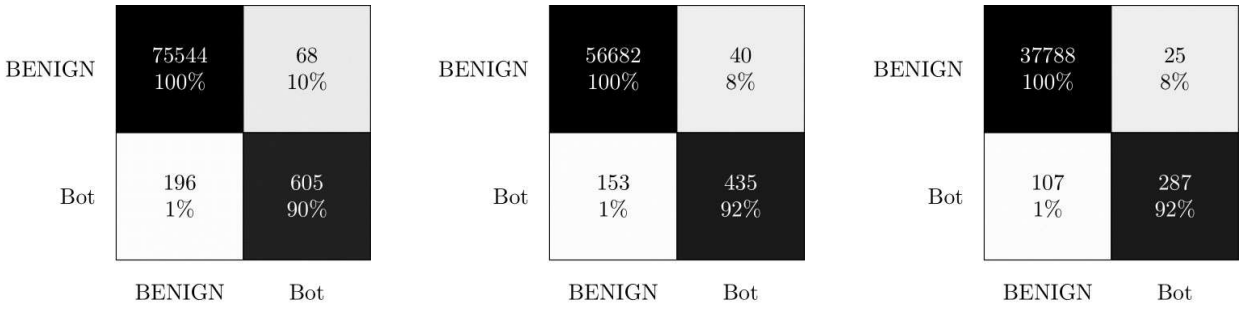
Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,997	0,999	0,998
	Bot	0,899	0,755	0,821
70	*BENIGN*	0,997	0,999	0,998
	Bot	0,916	0,740	0,818
80	*BENIGN*	0,997	0,999	0,998
	Bot	0,920	0,728	0,813

Comparando os resultados vistos com o uso do SVM, o *Random Forest* obteve melhores índices de acertos, mesmo que longes do ideal.

As matrizes de confusão deste experimento são apresentadas na Figura 4.16. É possível observar que o experimento CIC-IDS-2017/*Random Forest*/*Information Gain* apresentou resultados ótimos (100%) relativos ao índice TP. Por outro lado, os resultados de desempenho de classificação relativos ao índice TN foram menores (90% para 60-40), melhorando um pouco (92%) quando a razão de divisão treinamento-teste aumenta para 70-30 e 80-20.

4.2.2.2 Experimento CIC-IDS/*Random Forest*/*Correlation*

Em relação ao experimento com o uso do *Correlation* como seletor de atributos, dos 77 atributos do conjunto de dados, 5 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.8. Para esta configuração de experimento, os resultados em termos das métricas de



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.16: Matriz de confusão do experimento 4.2.2.1.

TP e FP são mostrados na Tabela 4.30.

Tabela 4.30: TP e FP do experimento 4.2.2.2

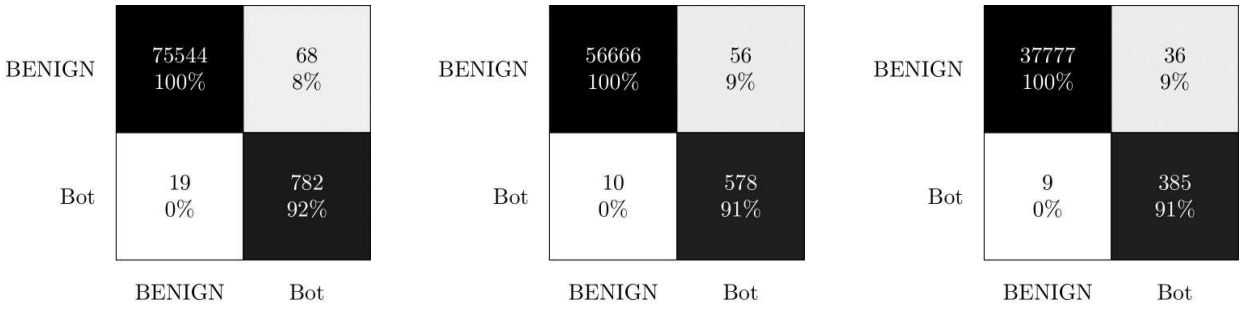
Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,999	0,024
	Bot	0,976	0,001
70	*BENIGN*	0,999	0,017
	Bot	0,983	0,001
80	*BENIGN*	0,999	0,023
	Bot	0,977	0,001

Tabela 4.31: Resultados de: Precisão, *Recall* e *F1-Score* do experimento 4.2.2.2

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	1,000	0,999	0,993
	Bot	0,920	0,976	0,947
70	*BENIGN*	1,000	0,999	0,999
	Bot	0,912	0,983	0,946
80	*BENIGN*	1,000	0,999	0,999
	Bot	0,914	0,977	0,945

A diferença nos resultados entre TP e FP entre o uso do *Information Gain* e *Correlation* foi significativa. Por exemplo, considerando o tráfego NORMAL e a razão de divisão 60-40, obteve-se com o uso do seletor Information Gain um índice FP igual a 24,5% (Tab. 4.28) enquanto que no caso do Correlation o índice FP foi 2,4% (Tab. 4.30). Por outro lado, considerando a mesma razão de divisão 60-40 mas com o tráfego Anômalo, obteve-se com o uso do seletor Information Gain um índice TP igual a 75,5% (Tab. 4.28) enquanto que no caso do Correlation o índice TP foi de 97,6% (Tab. 4.28). Assim, observa-se que o uso de mais variáveis pelo Information Gain não implica em uma melhoria de desempenho

Novamente, os índices se manteram ótimos, como na seção anterior quando o SVM fora utilizado, não sendo um caso isolado. Além disso, comparando com o Information Gain, houve estatísticas de acerto



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.17: Matriz de confusão do experimento 4.2.2.2.

mais positivas. As matrizes de confusão do experimento CIC-IDS-2017/Random Forest/Correlation são apresentadas na Figura 4.16. É possível, através dela, mensurar que houve bons resultados com índices de TP ótimos (100%) e índices TN da ordem de 92%.

4.2.2.3 Experimento CIC-IDS/*Random Forest*/cfssubseteval

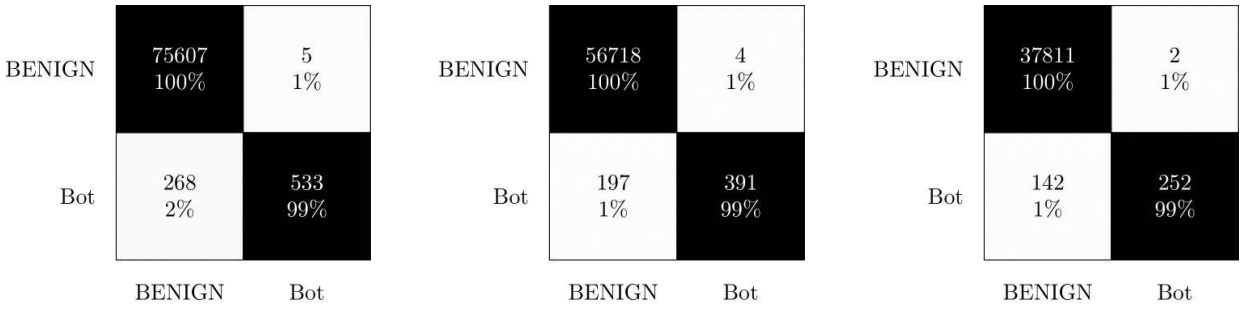
Em relação ao experimento com o uso do cfssubseteval como seletor de atributos, dos 77 atributos do conjunto de dados, 3 foram classificados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.9. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.32.

Tabela 4.32: TP e FP do experimento 4.2.2.3

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	1,000	0,335
	Bot	0.665	0,000
70	*BENIGN*	1,000	0,335
	Bot	0.665	0,000
80	*BENIGN*	1,000	0,360
	Bot	0.640	0,000

Tabela 4.33: Resultados de: Precisão, *Recall* e *F1-Score* do experimento 4.2.2.3

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,996	1,000	0,998
	Bot	0,991	0.665	0,796
70	*BENIGN*	0,997	1,000	0,9998
	Bot	0,990	0.665	0,796
80	*BENIGN*	0,996	1,000	0,998
	Bot	0,992	0.640	0,778



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.18: Matriz de confusão do experimento 4.2.2.3.

É possível, das Tabelas 4.32 e 4.33, perceber como os resultados obtidos para o ataque (Anomalia) são muito baixos, mesmo com a adoção dos filtros de seleção de atributos. Parte desse problema pode ser explicado pelo desbalanceamento do dataset, que viesa o resultado. As matrizes de confusão do experimento CIC-IDS-2017/Random Forest/CFS são apresentadas na Figura 4.18., onde é possível observar os bons resultados obtidos, uma vez que a diagonal principal mostra índices de TP e TN da ordem de 100% e 99%, respectivamente. Observa-se também que os diferentes modelos de divisão treinamento-teste não impactam significativamente os resultados de desempenho de classificação.

4.2.2.4 Experimento CIC-IDS/*Random Forest*/Sem filtragem

O último experimento não teve seleção de atributos, ou seja, o conjunto de dados é utilizado integralmente. Como esperado, a taxa de acerto foi alta, e significativamente melhor se comparada com os experimentos em que o *Information Gain* e cfssubseteval foram usados, conforme é percebido na Tabela 4.34.

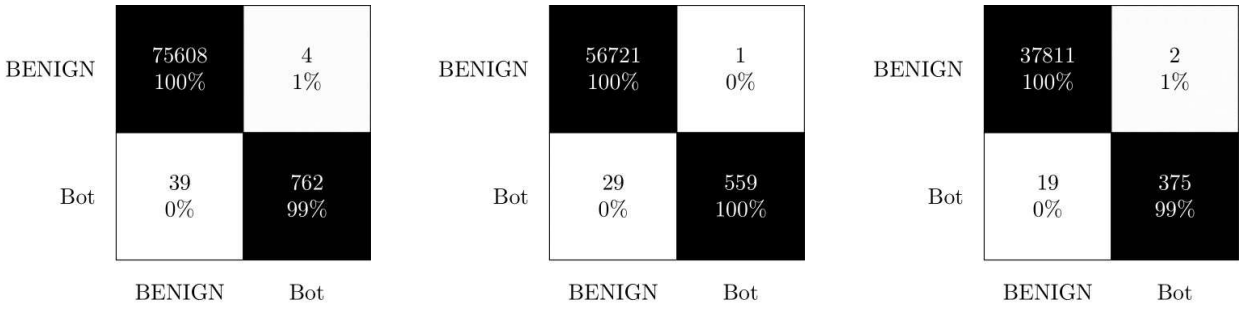
Tabela 4.34: TP e FP do experimento 4.2.2.4

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	1,000	0,049
	Bot	0,951	0,000
70	*BENIGN*	1,000	0,049
	Bot	0,951	0,005
80	*BENIGN*	1,000	0,048
	Bot	0,952	0,000

Já quando se compara com o *Correlation*, a opção por diminuir de 78 para 5 atributos é considerada positiva (melhoria das métricas), já que mesmo com a retirada de atributos, houve um aumentos na taxa de acertos das classes de cada pacote. Por exemplo,... considerando a razão 60-40 e o tráfego malicioso (Anomalia), os índice TP foram 97,6% com o uso do Correlation (Tab. 4.30) e 95,1% no caso sem seleção de atributos (Tab.4.34). Por outro lado, considerando a mesma razão e o tráfego normal (BENIGN), os índices FP foram, respectivamente, 2,4% (Tab. 4.30) e 4,9% (Tab. 4.34).

Tabela 4.35: *Precision*, *Recall* e *F1-Score* do experimento 4.2.2.4

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,999	1,000	1,000
	Bot	0,995	0,951	0,973
70	*BENIGN*	0,999	1,000	1,000
	Bot	0,998	0,951	0,974
80	*BENIGN*	0,999	1,000	1,000
	Bot	0,995	0,952	0,973



(a) *Split* de 60-40 (b) *Split* de 70-30 (c) *Split* de 80-20

Figura 4.19: Matriz de confusão do experimento 4.2.2.4

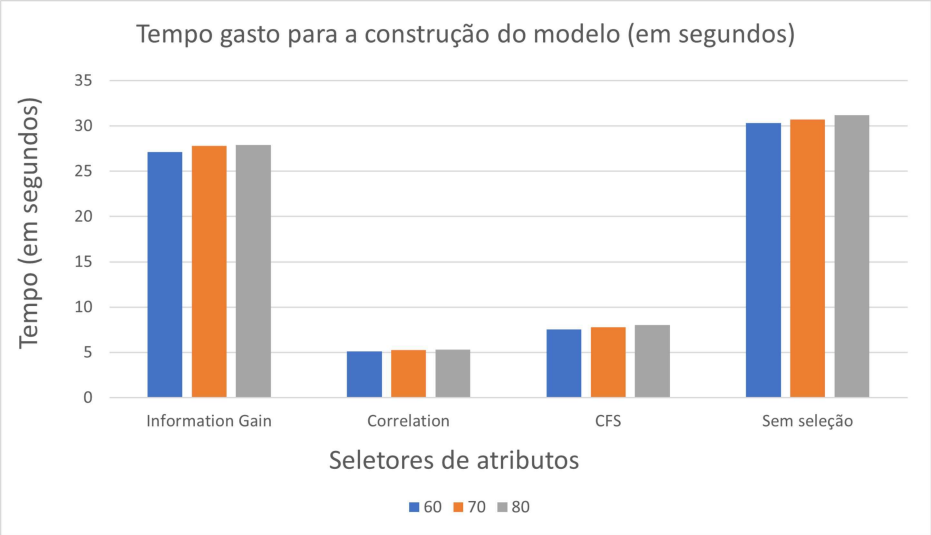
As matrizes de confusão são apresentadas na Figura 4.19. É possível, através dela, mensurar que houve bons resultados com os experimentos dessa rodada de testes, uma vez que a diagnoal principal está em destaque, indicando que os índices de TP e TN foram satisfatórias. Observa-se também que os diferentes modelos de divisão treinamento-teste não impactam significativamente os resultados de desempenho de classificação.

4.2.2.5 Tempos de processamento dos modelos nos experimentos CIC-IDS-2017/*Random Forest*

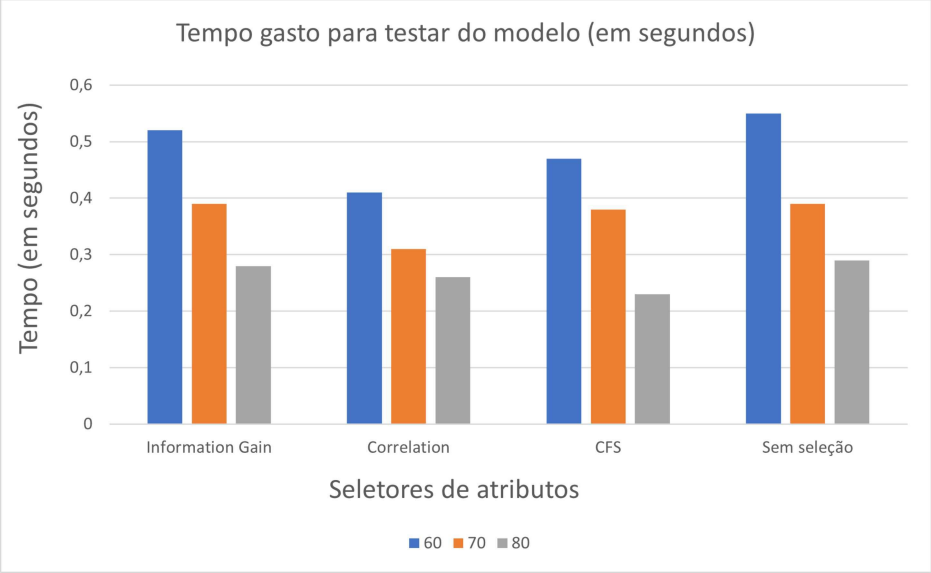
Outras formas de mensurar a qualidade da seleção dos atributos, é enxergar como isso impacta no processo de treinamento e testagem dos modelos.

Observando a Figura 4.20, é possível entender como a redução de atributos foi negativa positiva para a redução do tempo de processamento para construção e testagem dos modelos principalmente com a filtragem *Correlation* e CfsSubsetEval. Em particular, destaca-se que o uso do seletor *Correlation* se mostrou bastante positivo neste cenário cujo dataset (CIC-IDS-2017) é desbalanceado. Além de ter melhorado os índices de acerto com relação ao cenário sem seleção de atributos, o uso deste seletor resultou em tempos de construção de modelo menores do que os obtidos com em relação aos demais seletores.

Nota-se também (Fig. 4.20) que, a razão de divisão dos datasets não altera significativamente o tempo de construção dos modelos (no cenário com o mesmo seletor de atributos), ao passo que na testagem o tempo diminui à medida que o número de dados voltados para testes decai.



(a) Tempo gasto para a construção do modelo (em segundos).



(b) Tempo gasto para o teste do modelo (em segundos)

Figura 4.20: Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/*Random Forest*.

Outra forma de acompanhar os experimentos é enxergar a taxa de erros e acertos nas classificações de um modo geral. A Tabela 4.36 ilustra esse nível de acertos. Como esperado, utilizar todos os atributos permite melhores taxas de acerto. Porém, não muito distante, o uso do *Correlation* possibilitou bons índices de classificação.

Tabela 4.36: Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.2.2

Split	***Information Gain***	***Correlation***	***cfssubseteval***	***All features***
60	0,996	0,998	0,996	0,999
70	0,996	0,998	0,996	0,999
80	0,996	0,998	0,996	0,999

No caso dos experimentos com o dataset CIC-IDS-2017 observou-se que, de um modo geral, a efetividade da engenharia de atributos ficou comprometida pelo desbalanceamento do dataset que possui uma grande quantidade de dados da classe de tráfego normal (BENIGN), enquanto que a classe do ataque estudado (e.g., Anomalia) possui poucos dados.

4.3 CIC-IDS-2017 - COM BALANCEAMENTO *UNDERSAMPLING*

Anteriormente, viu-se que o uso de técnicas de filtragem em *datasets* desbalanceados não foi positiva em todos os experimento. A partir do uso da técnica de balanceamento do *dataset*, os resultados obtidos são expostos a seguir, no intuito de entender se a técnica permite melhora de resulados. O experimento, em si, permanece o mesmo, seja pelas divisões do conjunto de dados, seja pela utilização dos filtros já mencionados no trabalho.

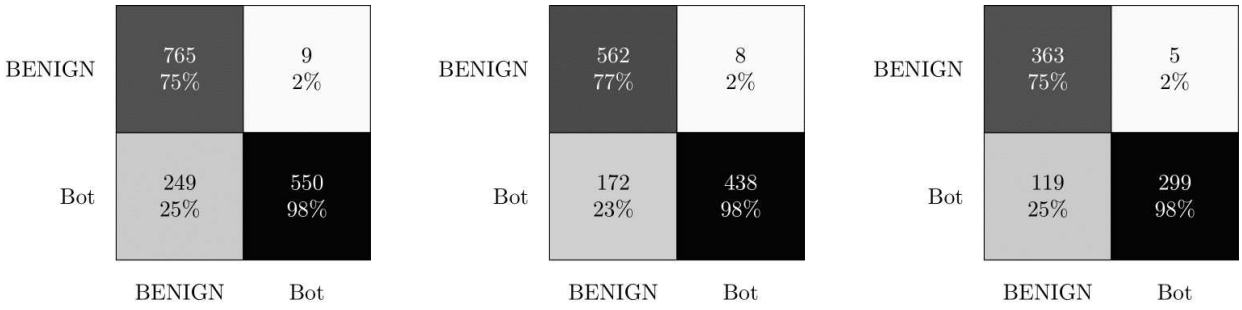
4.3.1 Algoritmo classificador SVM

Na Seção 4.2 viu-se que o uso de técnicas de filtragem de atributos em dataset desbalanceado não foi positivo em todos os experimentos. Assim sendo buscou-se investigar se a aplicação de técnicas de balanceamento no dataset CIC-IDS-2017 (Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX 2) poderia mitigar os efeitos adversos encontrados e permitir uma melhora de resultados com o uso da engenharia de atributos.

Nesta seção são apresentados os resultados da aplicação da técnica de balanceamento Undersamplig no dataset CIC-IDS-2017. As configurações dos experimentos continuam as mesmas, seja pelas divisões do conjunto de dados, algoritmos classificadores ou seletores de atributos.

4.3.1.1 Experimento CIC-IDS/*Undersampling*/SVM/*Information Gain*

Em relação ao experimento com o uso do *Information Gain* como seletor de atributos, dos 77 atributos do conjunto de dados, 12 foram selecionados para o treinamento e teste do algoritmo. Estas características



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.21: Matriz de confusão do experimento 4.3.1.1.

são mostradas na tabela 3.7. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.37.

Tabela 4.37: TP e FP do experimento 4.3.1.1

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,988	0.312
	Bot	0.688	0,012
70	*BENIGN*	0,986	0.282
	Bot	0,718	0,014
80	*BENIGN*	0,986	0.285
	Bot	0,715	0,014

Tabela 4.38: *Precision*, *Recall* e *F1-Score* do experimento 4.3.1.1

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,754	0,988	0,856
	Bot	0,984	0.688	0,810
70	*BENIGN*	0,766	0,986	0,862
	Bot	0,982	0,718	0,830
80	*BENIGN*	0,753	0,986	0,854
	Bot	0,984	0,715	0,828

As matrizes de confusão são apresentadas na Figura 4.21. É possível enxergar como os índices ruins apresentados nas duas tabelas anteriores constroem matrizes não ótimas. Ainda assim, comparando-se os resultados do experimento anterior, com *dataset* desbalanceado, a técnica de *undersampling* foi bastante positiva no contexto do *Information Gain*. Por exemplo, o índice de TP saltou 36% para para 68,8%, e o *F1-Score* para detecção de anomalia na rede, passou de 52,6% para 81%. Estes números são para o *split* 60-40, mas a melhoria é observada também nas demais divisões.

4.3.1.2 Experimento CIC-IDS/*Undersampling*/SVM/*Correlation*

Em relação ao experimento com o uso do *Correlation* como seletor de atributos, dos 77 atributos do conjunto de dados, 5 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.8. Para esta configuração de experimento, os resultados em termos das métricas de TP e FP são mostrados na Tabela 4.39.

Tabela 4.39: TP e FP do experimento 4.3.1.2

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,996	0,025
	Bot	0,975	0,004
70	*BENIGN*	0,996	0,025
	Bot	0,975	0,004
80	*BENIGN*	0,997	0,026
	Bot	0,974	0,003

Tabela 4.40: *Precision*, *Recall* e *F1-Score* do experimento 4.3.1.2

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,975	0,996	0,985
	Bot	0,996	0,975	0,985
70	*BENIGN*	0,974	0,996	0,971
	Bot	0,997	0,986	0,971
80	*BENIGN*	0,971	0,997	0,984
	Bot	0,998	0,974	0,985

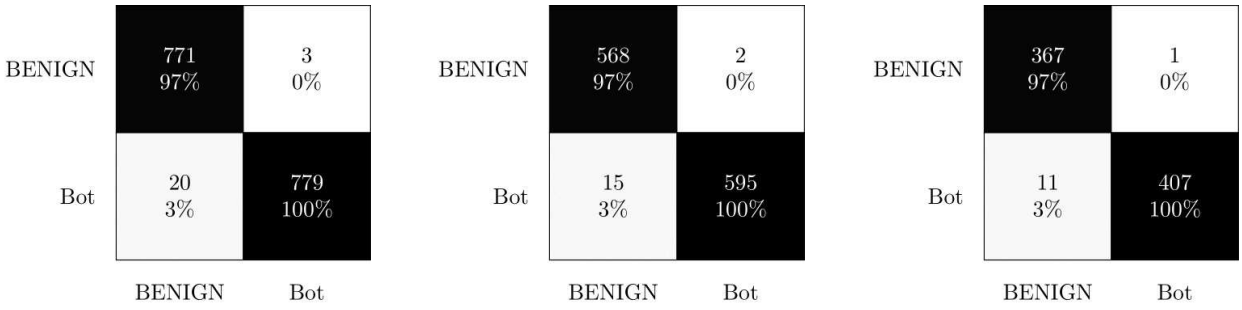
A diferença nos resultados entre TP e FP entre o uso do *Information Gain* e *Correlation* continuou significativa.

As matrizes de confusão são apresentadas na Figura 4.22. É possível, através dela, mensurar que houve bons resultados com os experimentos dessa rodada de testes, uma vez que a diagnoal principal está em destaque, indicando que os índices de TP e TN foram satisfatórias.

Por outro lado, ao se comparar os resultados com o *dataset* desbalanceado, não houve melhoria significativa das métricas. Exemplo: utilizando a divisão 80-20, os índices de *F1-Score* para o cenário desbalanceado é de 94,0%, enquanto que neste experimento foi de 98,5%.

4.3.1.3 Experimento CIC-IDS/*Undersampling*/SVM/cfssubseteval

Em relação ao experimento com o uso do cfssubseteval como seletor de atributos, dos 77 atributos do conjunto de dados, 3 foram classificados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.9. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.41.



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.22: Matriz de confusão do experimento 4.3.1.2.

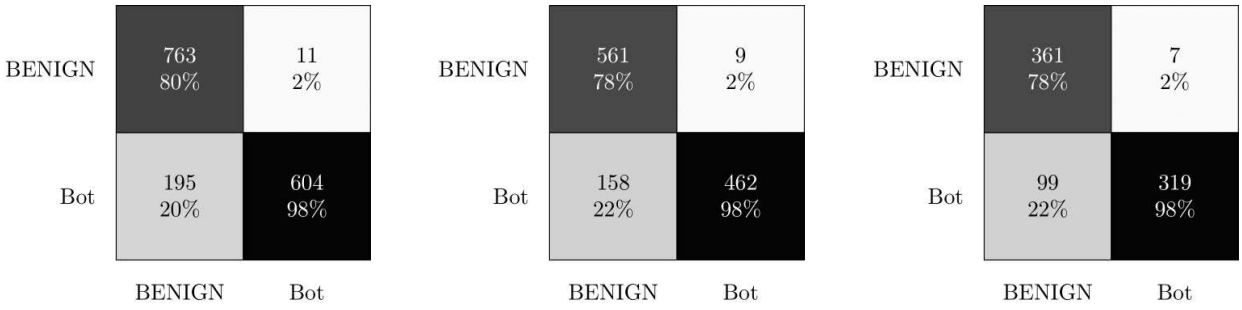
Tabela 4.41: TP e FP do experimento 4.3.1.3

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,986	0.244
	Bot	0,756	0,014
70	*BENIGN*	0,984	0.243
	Bot	0,757	0,016
80	*BENIGN*	0,981	0.237
	Bot	0,763	0,019

Analisando as tabelas, observa-se que o algoritmo de seleção cfssubseteval obteve o mesmo comportamento que o *Information Gain*. Porém, comparando com o cenário de *dataset* desbalanceado, o uso do *undersampling* melhorou os índices, mesmo que não suficiente para torná-lo um possível candidato de uso. Os índices de detecção de ataque foram medianos. Consequentemente os resultados de precisão, *recall* e *F1-Score* saíram prejudicados, como elucida a Tabela 4.42. As matrizes de confusão são apresentadas na Figura 4.23. Nela, é possível notar que no caso dos índices de TP (80% e 78%) os resultados ficaram distantes do ótimo enquanto que os índices TN ficaram próximos do ótimo (98%).

Tabela 4.42: *Precision*, *Recall* e *F1-Score* do experimento 4.3.1.3

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,796	0,986	0,881
	Bot	0,982	0,756	0,854
70	*BENIGN*	0,791	0,984	0,877
	Bot	0,981	0,757	0,855
80	*BENIGN*	0,785	0,981	0,872
	Bot	0,979	0,763	0,858



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.23: Matriz de confusão do experimento 4.3.1.3.

4.3.1.4 Experimento CIC-IDS/*Undersampling*/SVM/Sem seleção

Em relação ao experimento sem seleção alguma de atributos como seletor de atributos, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.43. Já *Precision*, *Recall* e *F1-Score* são colocados na Tabela 4.44.

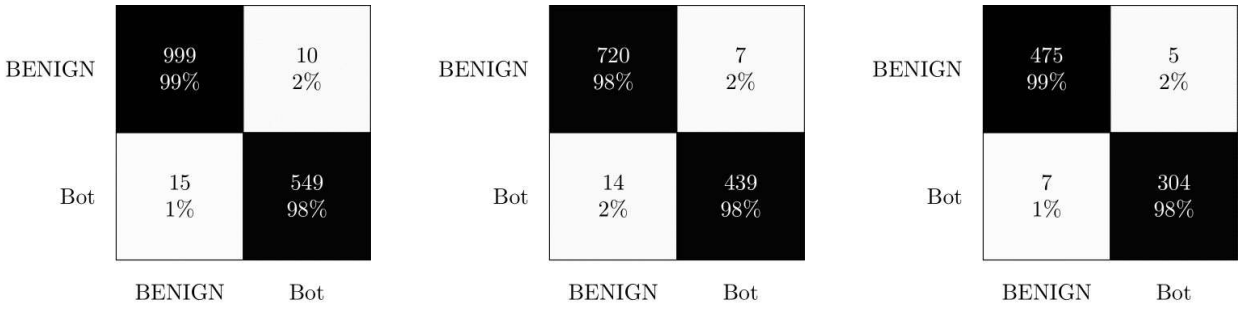
Tabela 4.43: TP e FP do experimento 4.3.1.4

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,985	0,018
	Bot	0,982	0,015
70	*BENIGN*	0,980	0,023
	Bot	0,977	0,020
80	*BENIGN*	0,985	0,017
	Bot	0,983	0,015

Tabela 4.44: *Precision*, *Recall* e *F1-Score* do experimento 4.3.1.4

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,990	0,985	0,990
	Bot	0,973	0,982	0,980
70	*BENIGN*	0,990	0,980	0,987
	Bot	0,969	0,984	0,980
80	*BENIGN*	0,989	0,985	0,990
	Bot	0,977	0,983	0,981

Analisando as tabelas acima, observa-se que quando não houve seleção, o comportamento dos resultados se manteve estável em quaisquer divisões. Comparativamente aos outros experimentos, este obteve as melhores métricas para *F1-Score*. Já as matrizes de confusão são mostradas na Figura 4.24.



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.24: Matriz de confusão do experimento 4.3.1.4.

4.3.1.5 Tempos de processamento dos modelos nos experimentos CIC-IDS-2017/*Undersampling*/SVM

Outras formas de mensurar a qualidade da seleção dos atributos, é enxergar como isso impacta no processo de treinamento e testagem dos modelos. Afinal, caso haja queda nos índices estatísticos estudados, ela deve ser contornada com uma maior agilidade na predição das classes de cada. Observando a Figura 4.25. De um modo geral pode-se notar que os seletores com menor quantidade de atributos apresentam menores tempos de processamento tanto na construção quanto na testagem,

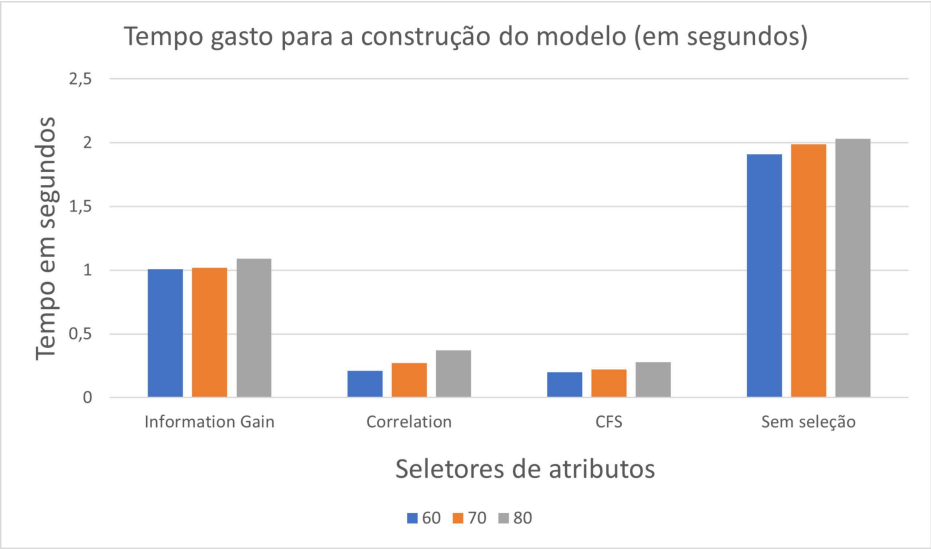
É possível notar também que, a razão de divisão do *dataset* para um mesmo seletor de atributos altera muito pouco o processo de cons-trução dos modelos com tempos de processamento inalterados no caso do *Information Gain* e com pequenos aumentos no caso dos seletores CFS e *Correlation*. Por outro lado, na testagem o tempo de processamento diminui para todos os seletores à medida que o número de dados voltados para testes decai.

Outra forma de acompanhar os experimentos é enxergar a taxa de erros e acertos nas classificações de um modo geral. A Tabela 4.45 ilustra esse nível de acertos. No caso de um *dataset* balanceado é mais visível como os índices estatísticos ruins refletem também nas taxas de acerto vistos abaixo.

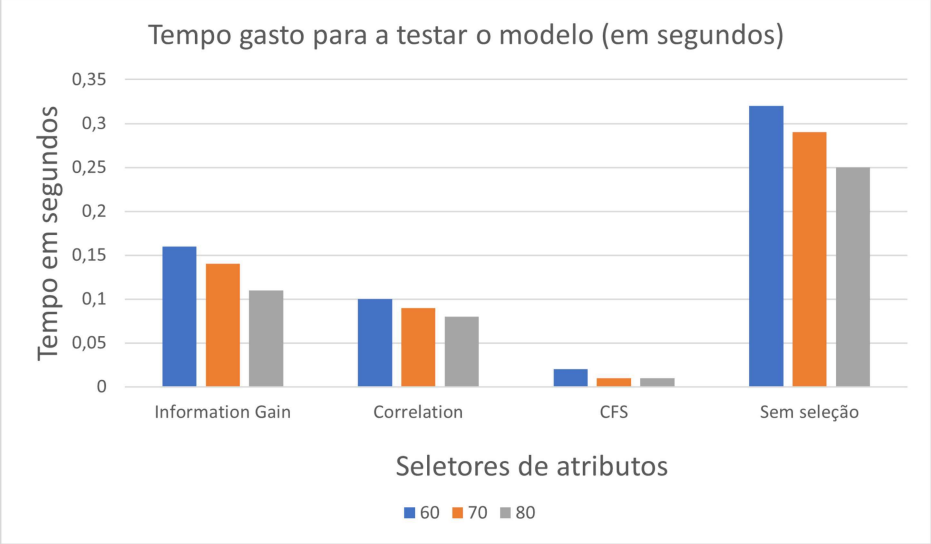
Tabela 4.45: Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.3.1

Split	***Information Gain***	***Correlation***	***cfssubseteval***	***All features***
60	0,835	0,985	0,867	0,994
70	0,847	0,985	0,867	0,994
80	0,842	0,984	0,865	0,994

Como pode ser visto, o uso de algumas técnicas de (cfssubseteval e *Information Gain*) não foi satisfatória no contexto de conjuntos de dados CIC-IDS-2017 mais balanceado, mesmo que haja uma melhora nos índices quando comparado ao *dataset* sem *undersampling*, dado que o os índices de TP para detecção de ataque ficou abaixo de 70% nos experimentos.



(a) Tempo gasto para a construção do modelo (em segundos).



(b) Tempo gasto para o teste do modelo (em segundos)

Figura 4.25: Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/SVM/*Undersampling*.

4.3.2 Algoritmo classificador *Random Forest*

4.3.2.1 Experimento CIC-IDS/*Undersampling*/*Random Forest*/*Information Gain*

Em relação ao experimento com o uso do *Information Gain* como seletor de atributos, dos 77 atributos do conjunto de dados, 12 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.7. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.46.

Tabela 4.46: TP e FP do experimento 4.3.2.1

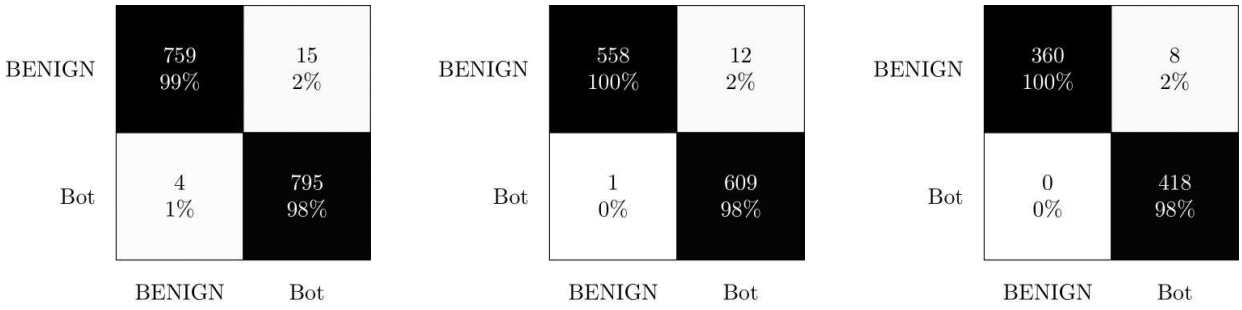
Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,981	0,005
	Bot	0,995	0,019
70	*BENIGN*	0,979	0,002
	Bot	0,998	0,021
80	*BENIGN*	1,000	0,022
	Bot	0,978	0,000

Tabela 4.47: *Precision*, *Recall* e *F1-Score* do experimento 4.3.2.1

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,995	0,981	0,988
	Bot	0,981	0,995	0,988
70	*BENIGN*	0,998	0,979	0,988
	Bot	0,981	0,998	0,989
80	*BENIGN*	1,000	0,978	0,989
	Bot	0,981	1,000	0,991

As matrizes de confusão do experimento CIC-IDS-2017/Undersampling/Random Forest/Information Gain são apresentadas na Figura 4.26. É possível, através da Figura 4.26, perceber que houve bons resultados com índices de TP e TN próximos do ótimo (99%). No caso do índice TP os resultados são semelhantes àqueles obtidos com mesmo experimento sem o balanceamento do dataset CIC-IDS-2017 (Fig. 4.18). Entretanto, no caso dos índices TN, observa-se que o balanceamento do dataset resultou em uma melhora significativa aumentando de 92% para 99%.

Em conjuntos de dados altamente desequilibrados, modelos de machine learning podem desenvolver um viés em direção à classe majoritária, devido à sua predominância. Ao realizar o undersampling, o modelo é forçado a considerar ambas as classes de maneira mais equilibrada, mitigando esse viés.



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.26: Matriz de confusão do experimento 4.3.2.1.

4.3.2.2 Experimento CIC-IDS/*Undersampling*/*Random Forest*/*Correlation*

Em relação ao experimento com o uso do *Correlation* como seletor de atributos, dos 77 atributos do conjunto de dados, 5 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.8. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.48.

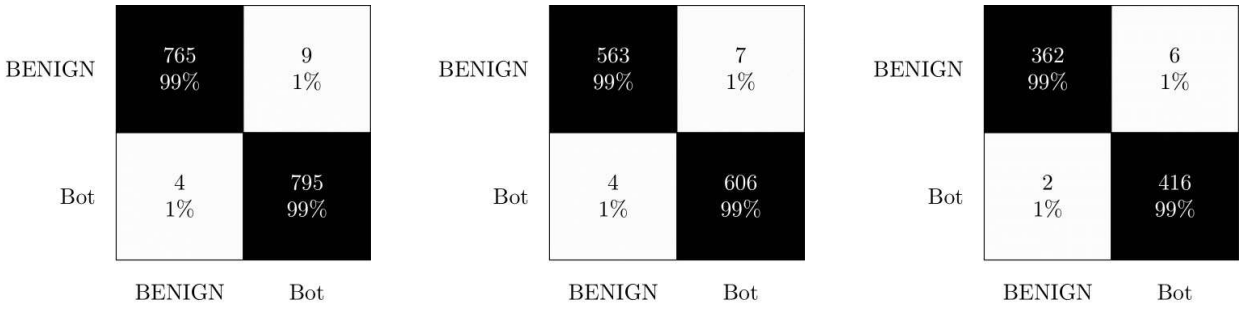
Tabela 4.48: TP e FP do experimento 4.3.2.2

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,988	0,005
	Bot	0,995	0,012
70	*BENIGN*	0,988	0,007
	Bot	0,993	0,012
80	*BENIGN*	0,984	0,005
	Bot	0,995	0,016

Tabela 4.49: *Precision*, *Recall* e *F1-Score* do experimento 4.3.2.2

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,995	0,998	0,992
	Bot	0,989	0,995	0,992
70	*BENIGN*	0,993	0,988	0,990
	Bot	0,989	0,993	0,991
80	*BENIGN*	0,995	0,984	0,989
	Bot	0,986	0,995	0,990

A diferença nos resultados entre TP e FP entre o uso do *Information Gain* e *Correlation* não foi significativa. Por exemplo, para tráfego BENIGN, com o uso de razão de divisão de dados treinamento-teste mais balanceada, como 60-40, o desempenho de classificação em termos de TP é da ordem de 98,1% para o *Information Gain* e 98,8% para o *Correlation*, e em termos de FP, é da ordem de 0,5% tanto para o



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.27: Matriz de confusão do experimento 4.3.2.2.

Information Gain quanto para o Correlation. Já para o tráfego Bot, com a razão 60-40, ambos os seletores apresentaram índices de TP e FP iguais, respectivamente, 99,5% e 1,2%. Por outro lado, o tempo para construir modelos com o *Correlation* tende a ser menor do que com o *Information Gain* em função da menor quantidade de atributos.

As matrizes de confusão do experimento CIC-IDS-2017/Undersampling/Random Forest/Correlation são apresentadas na 4.27. É possível, através dela, mensurar que houve bons resultados, com índices de TP e TN próximos do ótimo (99%). No caso do índice TP os resultados são semelhantes àqueles obtidos com mesmo experimento sem o balanceamento do dataset CIC-IDS-2017 (Fig. 4.16). Entretanto, no caso dos índices TN, o balanceamento do dataset resultou em uma melhora significativa aumentando de 92% para 99%.

4.3.2.3 Experimento CIC-IDS/*Undersampling*/*Random Forest*/cfssubseteval

Em relação ao experimento com o uso do cfssubseteval como seletor de atributos, dos 41 atributos do conjunto de dados, 6 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.6. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.50.

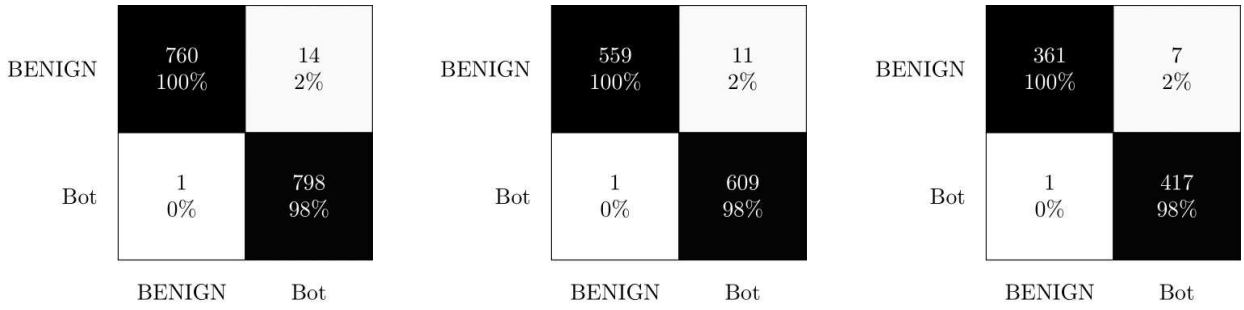
Tabela 4.50: TP e FP do experimento 4.3.2.3

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,982	0,001
	Bot	0,999	0,018
70	*BENIGN*	0,981	0,002
	Bot	0,998	0,019
80	*BENIGN*	0,981	0,002
	Bot	0,998	0,019

Analisando as tabelas, nota-se que os resultados para o F1-Score são bastante semelhantes com uma pequena vantagem para os seletores Correlation (99,2%) e CFS (99%) com relação ao Information Gain

Tabela 4.51: *Precision*, *Recall* e *F1-Score* do experimento 4.3.2.3

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,999	0,982	0,990
	Bot	0,983	0,999	0,991
70	*BENIGN*	0,998	0,981	0,990
	Bot	0,982	0,998	0,990
80	*BENIGN*	0,997	0,981	0,989
	Bot	0,983	0,998	0,990



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.28: Matriz de confusão do experimento 4.3.2.3.

(98,8%). Comparando com os dois seletores anteriores, o cfssubseteval foi o único que de alguma forma melhorou significativamente os resultados de classificação de TP e FP.

Analisando os resultados da Tabela 4.51, nota-se que os melhores resultados para o *F1-Score* são para os testes feitos com o subconjunto cfssubseteval, que menos requer atributos do conjunto de dados. Para esses casos, a escolha entre menor tempo de construção dos modelos ou melhores índices acaba sendo mais decisiva.

As matrizes de confusão do experimento CIC-IDS-2017/Undersampling/Random Forest/CFS são apresentadas na Figura 4.28. É possível, através dela, mensurar que houve bons resultados, com índices de TP e TN próximos do ótimo (100% e 98%). Esses resultados são semelhantes àqueles obtidos com a mesma configuração de experimento sem o balanceamento do dataset CIC-IDS-2017 (Fig. 4.17).

4.3.2.4 Experimento CIC-IDS/*Undersampling*/*Random Forest*/Sem filtragem

O último experimento não teve seleção de atributos, ou seja, o conjunto de dados é utilizado integralmente. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.52.

As matrizes de confusão são apresentadas na Figura 4.29. É possível, através dela, observar que houve bons resultados, com índices de TP e TN próximos do ótimo (100% e 99%). Esses resultados são semelhantes àqueles obtidos com a mesma configuração de experimento sem o balanceamento do dataset

Tabela 4.52: TP e FP do experimento 4.3.2.4

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,988	0,000
	Bot	1,000	0,012
70	*BENIGN*	0,989	0,000
	Bot	1,000	0,011
80	*BENIGN*	0,989	0,000
	Bot	1,000	0,011

Tabela 4.53: *Precision*, *Recall* e *F1-Score* do experimento 4.3.2.4

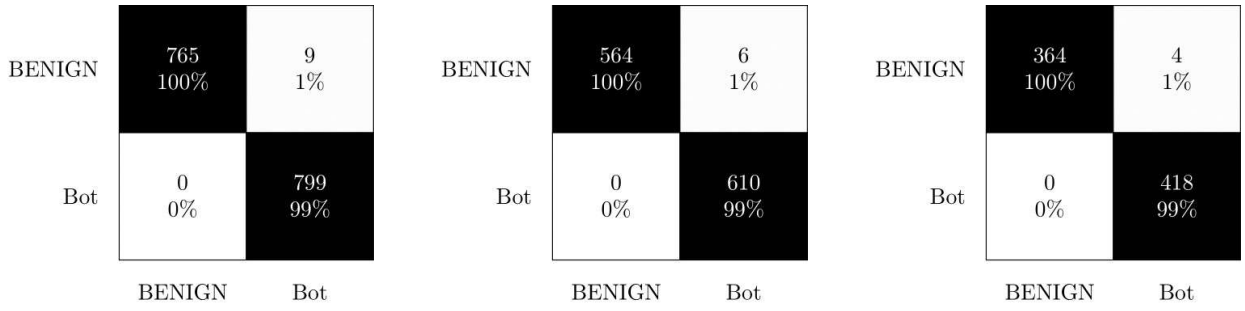
Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	1,000	0,988	0,994
	Bot	0,989	1,000	0,994
70	*BENIGN*	1,000	0,989	0,995
	Bot	0,990	0,943	0,968
80	*BENIGN*	1,000	0,989	0,995
	Bot	0,991	1,000	0,995

CIC-IDS-2017 (Fig. 4.18).

4.3.2.5 Tempos de processamento dos modelos nos experimentos CIC-IDS-2017/*Undersampling*/*Random Forest*

Observando a Figura 4.30, é possível entender como a redução de atributos foi positiva no que tange o tempo de processamento para construção e testagem dos modelos.

É possível notar também que, um aumento da razão de divisão dos dataset aumenta o tempo de construção de modelos no cenário com o mesmo seletor de atributos. Por outro lado, no caso da testagem o tempo de processamento tende a diminuir à medida que o número de dados voltados para testes decai.



(a) *Split* de 60-40 (b) *Split* de 70-30 (c) *Split* de 80-20

Figura 4.29: Matriz de confusão do experimento 4.3.2.4

Entretanto, com exceção dos experimentos com o seletor *Information Gain* esse comportamento não se verifica plenamente com os outros experimentos (*Correlation*, CFS e Sem Seleção de Atributos).

É possível notar também que, a divisão dos *datasets* não altera significativamente o processo de construção de modelos (no cenário com o mesmo seletor de atributos), ao passo que na testagem esse tempo também diminui a medida que o número de dados voltados para testes decai.

Outra forma de acompanhar os experimentos é enxergar a taxa de erros e acertos nas classificações de um modo geral. A Tabela 4.54 ilustra esse nível de acertos.

Tabela 4.54: Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimentos 4.3.2

Split	***Information Gain***	***Correlation***	***cfssubseteval***	***All features***
60	0,980	0,937	0,971	0,968
70	0,981	0,943	0,973	0,970
80	0,982	0,943	0,973	0,971

4.4 CIC-IDS-2017 - COM BALANCEAMENTO *OVERSAMPLING*

Nesta seção são apresentados os resultados da aplicação da técnica de balanceamento Oversamplig no dataset CIC-IDS-2017. Nesta técnica, a classe com menos instâncias é repetida "n" vezes, de modo que seu número se aproxime da classe com maior número de dados. As configurações dos experimentos continuam as mesmas, seja pelas divisões do conjunto de dados, algoritmos classificadores ou seletores de atributos.

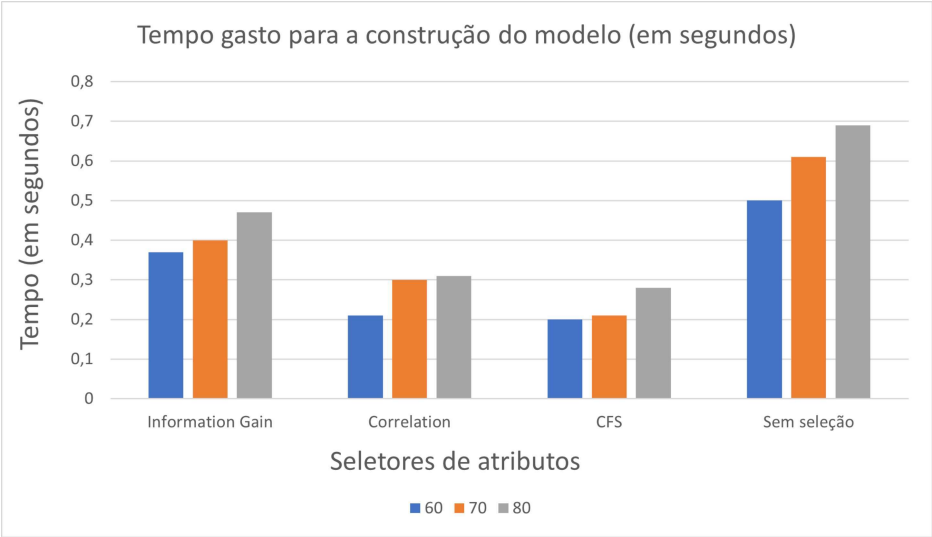
4.4.1 Algoritmo classificador SVM

4.4.1.1 Experimento CIC-IDS/*Oversampling*/SVM/*Information Gain*

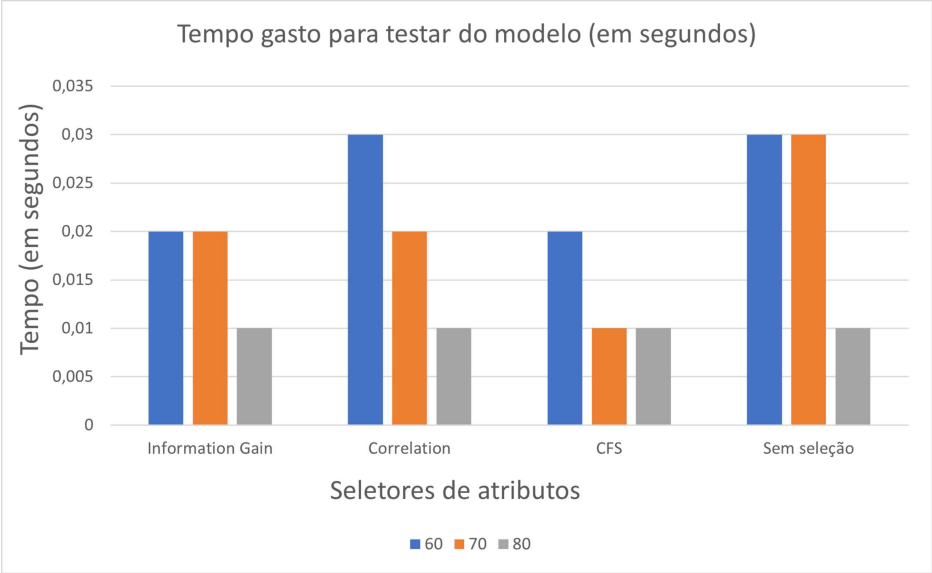
Em relação ao experimento com o uso do *Information Gain* como seletor de atributos, dos 77 atributos do conjunto de dados, 12 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.7. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.55.

As matrizes de confusão são apresentadas na Figura 4.31. É possível enxergar como os índices ruins apresentados nas duas tabelas anteriores constroem matrizes ótimas, com diagonais principais bem visíveis.

Ainda assim, comparando-se os resultados do experimento anterior, com *dataset* desbalanceado, a técnica de *undersampling* foi bastante positiva no contexto do *Information Gain*.

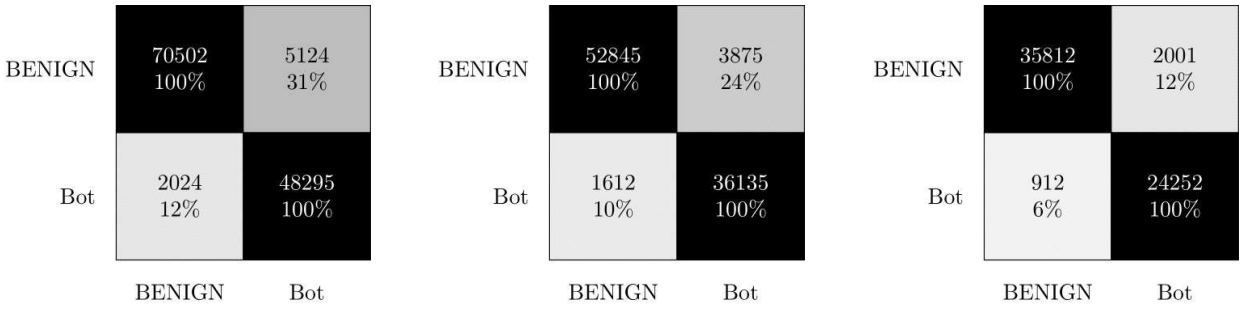


(a) Tempo gasto para a construção do modelo (em segundos).



(b) Tempo gasto para o teste do modelo (em segundos)

Figura 4.30: Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/*Random Forest*/*Undersampling*.



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.31: Matriz de confusão do experimento 4.4.1.1.

Tabela 4.55: TP e FP do experimento 4.4.1.1

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,932	0,041
	Bot	0,959	0,068
70	*BENIGN*	0,970	0,043
	Bot	0,957	0,030
80	*BENIGN*	0,975	0,077
	Bot	0,923	0,025

Tabela 4.56: *Precision*, *Recall* e *F1-Score* do experimento 4.4.1.1

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,932	0,972	0,951
	Bot	0,959	0,904	0,930
70	*BENIGN*	0,931	0,970	0,950
	Bot	0,957	0,903	0,929
80	*BENIGN*	0,995	0,975	0,984
	Bot	0,993	0,923	0,956

4.4.1.2 Experimento CIC-IDS/*Oversampling*/SVM/*Correlation*

Em relação ao experimento com o uso do *Correlation* como seletor de atributos, dos 77 atributos do conjunto de dados, 5 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.8. Para esta configuração de experimento, os resultados em termos das métricas de TP e FP são mostrados na Tabela 4.57.

Tabela 4.57: TP e FP do experimento 4.4.1.2

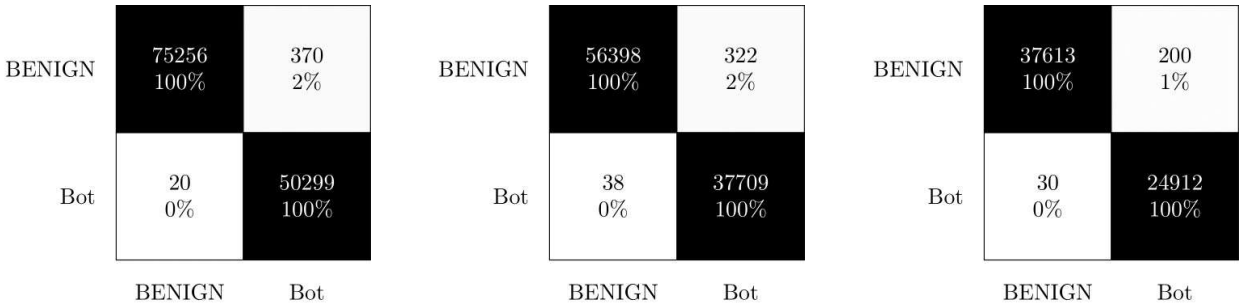
Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,999	0,008
	Bot	0,992	0,001
70	*BENIGN*	0,999	0,009
	Bot	0,991	0,001
80	*BENIGN*	0,999	0,008
	Bot	0,992	0,001

A diferença nos resultados entre TP e FP entre o uso do *Information Gain* e *Correlation* continuou significativa.

As matrizes de confusão são apresentadas na Figura 4.32. Nela, é possível observar que houve bons resultados, com índices de TP e TN ótimos (100%). Esses resultados são iguais (TP) ou superiores (TN) àqueles obtidos com a mesma configuração de experimento sem o balanceamento do dataset CIC-IDS-2017 (Fig. 4.12).

Tabela 4.58: *Precision*, *Recall* e *F1-Score* do experimento 4.4.1.2

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,995	0,999	0,996
	Bot	0,999	0,992	0,985
70	*BENIGN*	0,994	0,999	0,996
	Bot	0,998	0,991	0,971
80	*BENIGN*	0,995	0,995	1,000
	Bot	0,993	0,992	0,985



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.32: Matriz de confusão do experimento 4.4.1.2.

4.4.1.3 Experimento CIC-IDS/*Oversampling*/SVM/cfssubseteval

Em relação ao experimento com o uso do cfssubseteval como seletor de atributos, dos 77 atributos do conjunto de dados, 3 foram classificados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.9. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.59.

Tabela 4.59: TP e FP do experimento 4.4.1.3

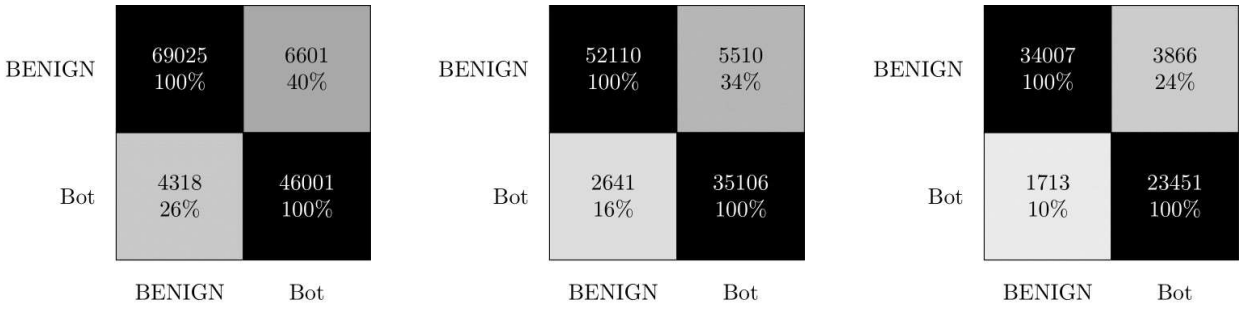
Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,941	0.226
	Bot	0,874	0,059
70	*BENIGN*	0,951	0.236
	Bot	0,864	0,049
80	*BENIGN*	0,952	0.240
	Bot	0,860	0,048

Analisando as tabelas, observa-se que o algoritmo de seleção *cfssubseteval* obteve o mesmo comportamento que o *Information Gain*.

Porém, comparando com o cenário de *dataset* desbalanceado, o uso do *undersampling* melhorou os

Tabela 4.60: *Precision*, *Recall* e *F1-Score* do experimento 4.4.1.3

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,912	0,941	0,926
	Bot	0,914	0,874	0,893
70	*BENIGN*	0,904	0,951	0,926
	Bot	0,930	0,864	0,895
80	*BENIGN*	0,900	0,952	0,925
	Bot	0,931	0,860	0,894



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.33: Matriz de confusão do experimento 4.4.1.3.

índices, mesmo que não suficiente para torná-lo um possível candidato de uso. Os índices de detecção de ataque foram medianos. Consequentemente os resultados de precisão, *recall* e *F1-Score* saíram prejudicados, como elucida a Tabela 4.60. As matrizes de confusão são apresentadas na Figura 4.33.

As matrizes de confusão do experimento CIC-IDS-2017/Oversampling/SVM/CFS são apresentadas na Figura 4.31., onde é possível observar que houve bons resultados, com índices de TP e TN ótimos (100%). Esses resultados são iguais àqueles obtidos com a mesma configuração de experimento sem o balanceamento do dataset CIC-IDS-2017 (Fig. 4.13).

4.4.1.4 Experimento CIC-IDS/*Oversampling*/SVM/Sem filtragem

O último experimento não teve seleção de atributos, ou seja, o conjunto de dados é utilizado integralmente. Como esperado, a taxa de acerto foi alta, mas não o suficiente para tornar a métrica *F1-Score* superior ao seletor cfssubseteval, que funcionou com apenas 6 dos 41 atributos. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.61.

As matrizes de confusão são apresentadas na Figura 4.34, onde é possível observar que houve bons resultados, com índices de TP e TN ótimos (100

Tabela 4.61: TP e FP do experimento 4.4.1.4

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,999	0,001
	Bot	0,999	0,001
70	*BENIGN*	0,999	0,001
	Bot	0,999	0,001
80	*BENIGN*	0,999	0,001
	Bot	0,999	0,001

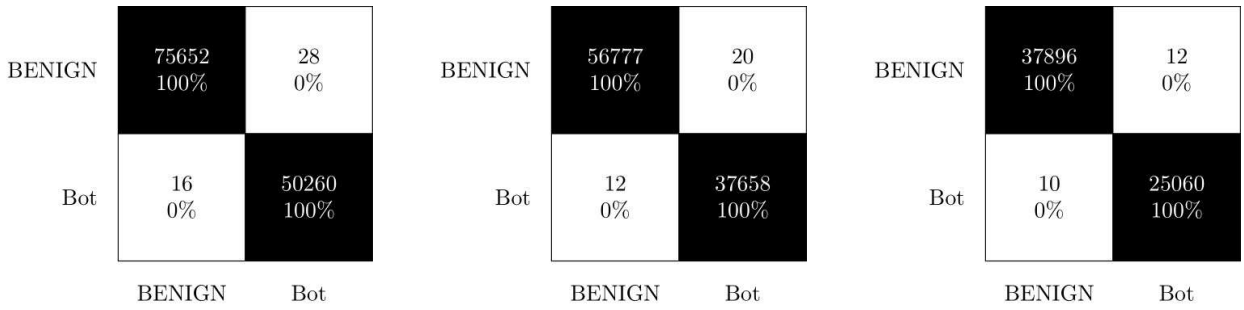
Tabela 4.62: *Precision*, *Recall* e *F1-Score* do experimento 4.4.1.4

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,999	0,999	0,999
	Bot	0,999	0,999	0,999
70	*BENIGN*	0,999	0,999	0,999
	Bot	0,999	0,999	0,999
80	*BENIGN*	0,999	0,999	0,999
	Bot	0,999	0,999	0,999

4.4.1.5 Tempos de processamento dos modelos nos experimentos CIC-IDS-2017/*Oversampling*/SVM

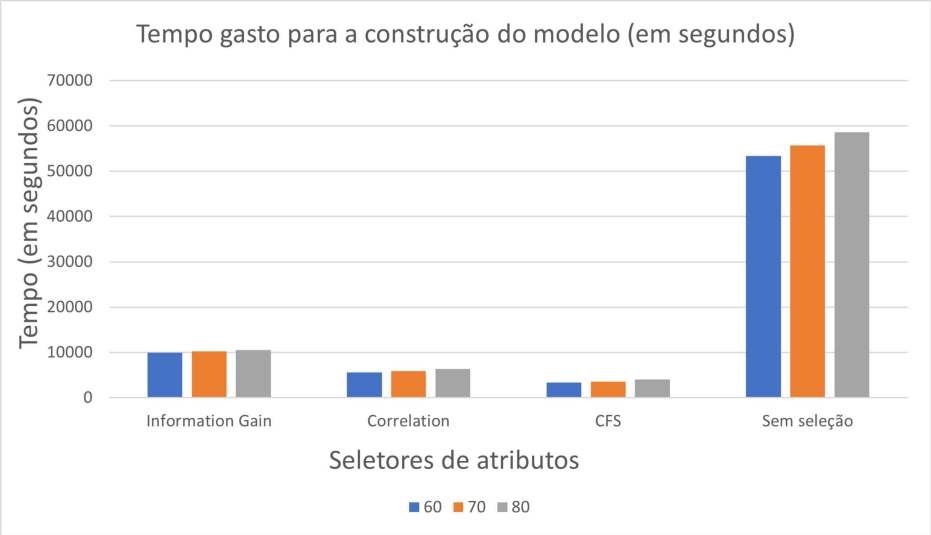
Observando a Figura 4.35, é possível observar que os tempos de construção com os seletores de atributos foram bem inferiores aos obtidos no experimento Sem Seleção de Attributos. Além disso, nota-se também que os tempos de construção e de teste dos modelos com o seletor CFS foram os melhores (i.e., menores), seguidos dos obtidos com o Correlation e, por último, os obtidos com o *Information Gain*.

É possível notar também que, a divisão dos *datasets* altera o processo de construção dos modelos (no cenário com o mesmo seletor de atributos) aumentando o tempo gasto à medida que os dados de treinamento aumentam. Por outro lado, na testagem esse tempo diminui à medida que o número de dados voltados para testes decai.

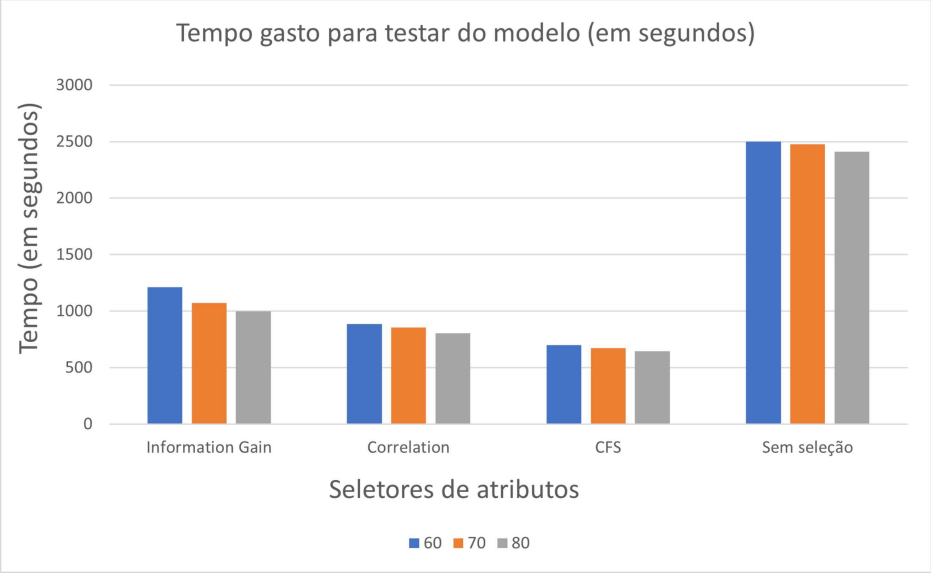


(a) *Split* de 60-40 (b) *Split* de 70-30 (c) *Split* de 80-20

Figura 4.34: Matriz de confusão do experimento 4.4.1.4



(a) Tempo gasto para a construção do modelo (em segundos).



(b) Tempo gasto para o teste do modelo (em segundos)

Figura 4.35: Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/SVM/*Oversampling*.

Outra forma de acompanhar os experimentos é enxergar a taxa de erros e acertos nas classificações de um modo geral. A Tabela 4.63 ilustra esse nível de acertos. No caso de um *dataset* balanceado é mais visível como os índices estatísticos ruins refletem também nas taxas de acerto vistos abaixo.

Tabela 4.63: Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.4.1

Split	***Information Gain***	***Correlation***	***cfssubseteval***	***All features***
60	0,943	0,996	0,913	0,999
70	0,941	0,996	0,914	0,999
80	0,943	0,996	0,912	0,999

4.4.2 Algoritmo classificador *Random Forest*

4.4.2.1 Experimento CIC-IDS/*Oversampling*/*Random Forest*/*Information Gain*

Em relação ao experimento com o uso do *Information Gain* como seletor de atributos, dos 77 atributos do conjunto de dados, 12 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.7. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.64.

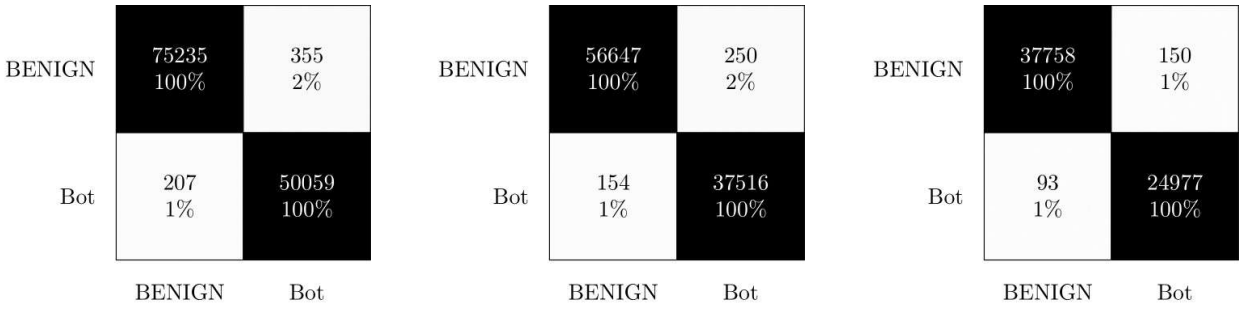
Tabela 4.64: TP e FP do experimento 4.4.2.3

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,995	0,004
	Bot	0,996	0,005
70	*BENIGN*	0,996	0,004
	Bot	0,996	0,004
80	*BENIGN*	0,996	0,004
	Bot	0,996	0,004

Tabela 4.65: *Precision*, *Recall* e *F1-Score* do experimento 4.4.2.3

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,997	0,995	0,996
	Bot	0,993	0,996	0,994
70	*BENIGN*	0,997	0,996	0,996
	Bot	0,993	0,996	0,995
80	*BENIGN*	0,996	0,998	0,997
	Bot	0,994	0,996	0,995

As matrizes de confusão são apresentadas na Figura 4.36. É possível, através dela, mensurar que houve bons resultados com os experimentos dessa rodada de testes, uma vez que a diagnoal principal está



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.36: Matriz de confusão do experimento 4.4.2.3.

em destaque, indicando que os índices de TP e TN foram ótimos.

Os resultados obtidos com a mesma configuração de experimento sem o balanceamento do dataset CIC-IDS-2017 (Fig. 4.15) foram 100% e 90%, respectivamente.

4.4.2.2 Experimento CIC-IDS/*Oversampling*/*Random Forest*/*Correlation*

Em relação ao experimento com o uso do *Correlation* como seletor de atributos, dos 77 atributos do conjunto de dados, 5 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.8. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.66.

Tabela 4.66: TP e FP do experimento 4.4.2.1

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,960	0,001
	Bot	0,999	0,040
70	*BENIGN*	0,960	0,001
	Bot	0,999	0,040
80	*BENIGN*	0,959	0,001
	Bot	0,999	0,041

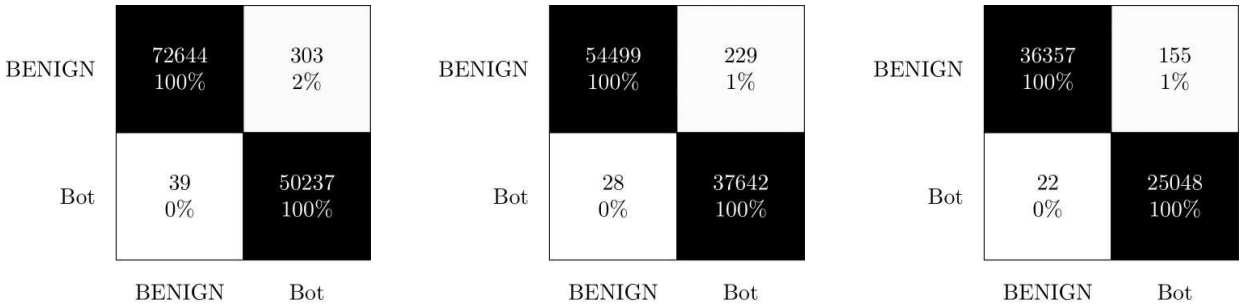
A diferença nos resultados entre TP e FP entre o uso do *Information Gain* e *Correlation* não foi significativa. O uso de razão de divisão de dados treinamento-teste mais balanceada, como 60-40, não implica perda de desempenho de classificação. Por outro lado, o tempo para construir modelos neste cenário mais equilibrado (60-40) tende a ser menor.

As matrizes de confusão do experimento CIC-IDS-2017/Oversampling/Random Forest/Correlation são apresentadas na Figura 4.37. onde é possível observar que houve bons resultados, com índices de TP e TN ótimos (100%).

Os resultados obtidos com a mesma configuração de experimento sem o balanceamento do dataset

Tabela 4.67: *Precision*, *Recall* e *F1-Score* do experimento 4.4.2.1

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,999	0,960	0,979
	Bot	0,943	0,999	0,970
70	*BENIGN*	0,999	0,960	0,979
	Bot	0,942	0,999	0,970
80	*BENIGN*	0,999	0,959	0,979
	Bot	0,942	0,999	0,970



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.37: Matriz de confusão do experimento 4.4.2.1.

CIC-IDS-2017 (Fig. 4.16) foram 100% e 92%, respectivamente.

4.4.2.3 Experimento CIC-IDS/*Oversampling*/*Random Forest*/cfssubseteval

Em relação ao experimento com o uso do cfssubseteval como seletor de atributos, dos 77 atributos do conjunto de dados, 3 foram selecionados para o treinamento e teste do algoritmo. Estas características são mostradas na tabela 3.6. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.68.

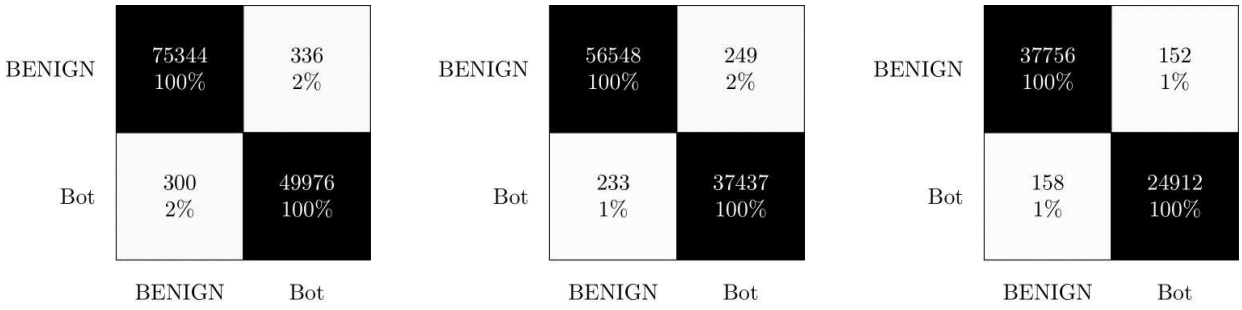
Tabela 4.68: TP e FP do experimento 4.4.2.2

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	0,996	0,006
	Bot	0,994	0,004
70	*BENIGN*	0,996	0,006
	Bot	0,994	0,004
80	*BENIGN*	0,981	0,002
	Bot	0,998	0,019

Analisando as tabelas, nota-se que os melhores resultados para o *F1-Score* são para os testes feitos

Tabela 4.69: *Precision*, *Recall* e *F1-Score* do experimento 4.4.2.2

Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	0,996	0,996	0,996
	Bot	0,993	0,994	0,994
70	*BENIGN*	0,996	0,996	0,996
	Bot	0,993	0,994	0,994
80	*BENIGN*	0,996	0,996	0,996
	Bot	0,994	0,994	0,994



(a) *Split 80-20* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.38: Matriz de confusão do experimento 4.4.2.2.

com o Subconjunto cfssubseteval, que menos requer atributos do conjunto de dados. Das 77 existentes, restavam apenas 3. Comparando com os dois seletores anteriores, o cfssubseteval foi o único que de alguma forma melhorou significativamente os resultados de classificação de TP e FP. Analisando os resultados da Tabela 4.6, nota-se que os melhores resultados para o *F1-Score* são para os testes feitos com o subconjunto cfssubseteval, que menos requer atributos do conjunto de dados. Para esses casos, a escolha entre menor tempo de construção dos modelos ou melhores índices acaba sendo mais decisiva.

As matrizes de confusão são apresentadas na Figura 4.38. É possível, através dela, mensurar que houve bons resultados com os experimentos dessa rodada de testes, uma vez que a diagnoal principal está em destaque, indicando que os índices de TP e TN foram satisfatórias.

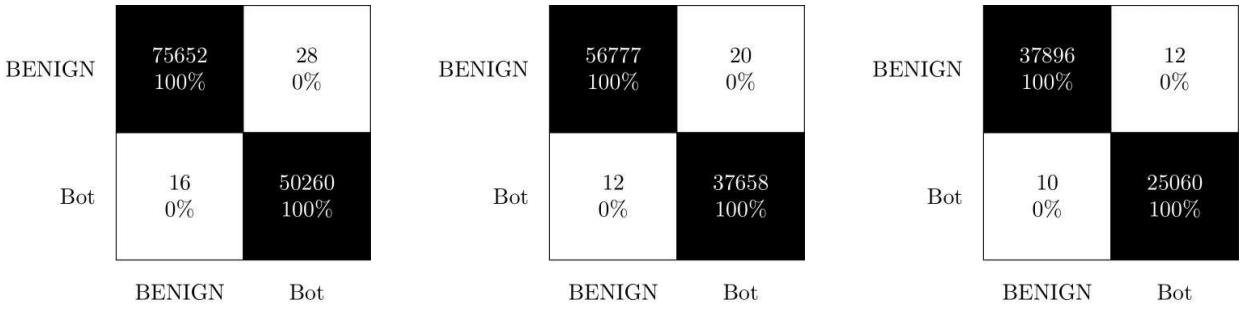
4.4.2.4 Experimento CIC-IDS/*Oversampling*/*Random Forest*/Sem filtragem

O último experimento não teve seleção de atributos, ou seja, o conjunto de dados é utilizado integralmente. Como esperado, a taxa de acerto foi alta, mas não o suficiente para tornar a métrica F-Score superior ao seletor CFS, que funcionou com apenas 3 dos 77 atributos. Para esta configuração de experimento, os resultados de desempenho de classificação obtidos do experimento utilizando em termos das métricas de TP e FP são mostrados na Tabela 4.70.

Já os valores de *Precision*, *Recall* e *F1-Score* são mostrados na Tabela 4.71. As matrizes de confusão são apresentadas na Figura 4.39. É possível, através dela, mensurar que houve bons resultados com os experimentos dessa rodada de testes, uma vez que a diagnoal principal está em destaque, indicando que os

Tabela 4.70: TP e FP do experimento 4.4.2.4

Split	Class	*True Positive*	*False Positive*
60	*BENIGN*	1,000	0,000
	Bot	1,000	0,000
70	*BENIGN*	1,000	0,000
	Bot	1,000	0,000
80	*BENIGN*	1,000	0,000
	Bot	1,000	0,000



(a) *Split 60-40* (b) *Split 70-30* (c) *Split 80-20*

Figura 4.39: Matriz de confusão do experimento 4.4.2.4.

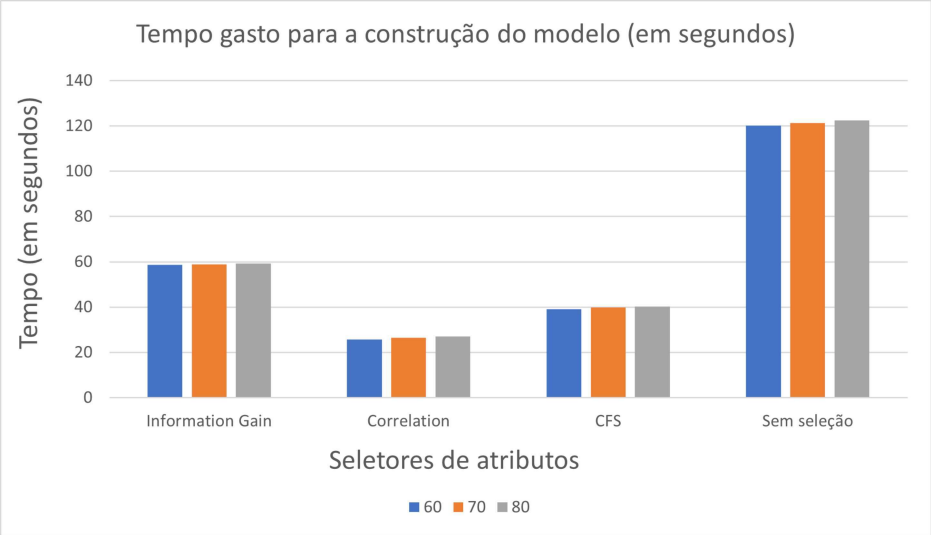
índices de TP e TN foram satisfatórias.

Tabela 4.71: *Precision*, *Recall* e *F1-Score* do experimento 4.4.2.4

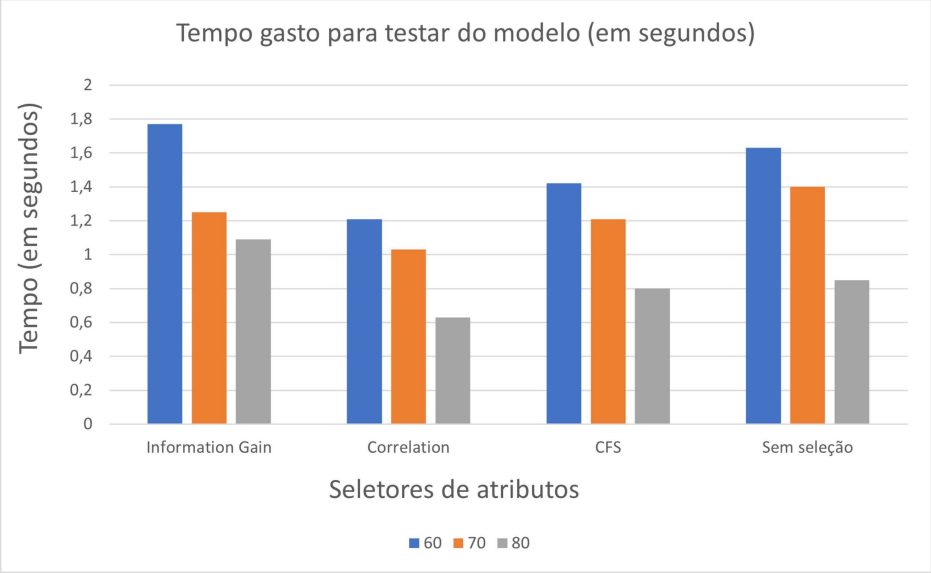
Split	Class	*Precision*	*Recall*	*F1-Score*
60	*BENIGN*	1,000	1,000	1,000
	Bot	0,999	1,000	1,000
70	*BENIGN*	1,000	1,000	1,000
	Bot	0,999	1,000	1,000
80	*BENIGN*	1,000	1,000	1,000
	Bot	0,999	1,000	1,000

4.4.2.5 Tempos de processamento dos modelos no experimento CIC-IDS/*Oversampling/Random Forest*

Outras formas de mensurar a qualidade da seleção dos atributos, é enxergar como isso impacta no processo de treinamento e testagem dos modelos. Afinal, caso haja queda nos índices estatísticos estudados, ela deve ser contornada com uma maior agilidade na predição das classes de cada. Observando a Figura 4.40, é possível entender como a redução de atributos foi positiva no que tange o processamento para construção e testagem dos modelos.



(a) Tempo gasto para a construção do modelo (em segundos).



(b) Tempo gasto para o teste do modelo (em segundos)

Figura 4.40: Gráficos de desempenho no tempo de processamento de construção e testagem dos modelos do experimento CIC-IDS/*Random Forest*/*Oversampling*.

É possível notar também que, a divisão dos *datasets* não altera significativamente o processo de construção de modelos (no cenário com o mesmo seletor de atributos), ao passo que na testagem esse tempo também diminui a medida que o número de dados voltados para testes decai.

Outra forma de acompanhar os experimentos é enxergar a taxa de erros e acertos nas classificações de um modo geral. A Tabela 4.72 ilustra esse nível de acertos.

Tabela 4.72: Instâncias classificadas corretamente por algoritmo de seleção de atributos do experimento 4.4.2

Split	*Information Gain*	*Correlation*	*cfssubseteval*	*All features*
60	0,994	0,975	0,995	0,999
70	0,995	0,975	0,995	0,999
80	0,995	0,976	0,995	0,999

4.5 COMPARAÇÃO DE RESULTADOS DOS EXPERIMENTOS

Primeiramente, ao observar os resultados com o *dataset* NSL-KDD, é possível inferir que o uso de técnicas de *Feature Engineering* é suficiente para melhorar resultados de classificação em processos de treino e teste. Um motivo que explica a adoção dessa metodologia é a redução do tempo de geração dos modelos. Em um conjunto de dados com quase 25.000 instâncias, a redução pode parecer insignificante, mas, dada a geração de pacotes em uma rede comum, menor poder de processamento e classificação de dados mais rápida certamente é necessária.

Porém, para validar essa informação, o uso do CIC-IDS-2017 foi necessária. Sua estrutura diferente, desbalanceada, poderia confirmar ainda mais que apenas o uso do *Feature Engineering* era suficiente. Isso não foi percebido. Apenas o *Correlation* obteve resultados satisfatórios, até melhores quando se comparado ao uso de todos os atributos.

Como o NSL-KDD era relativamente balanceado, a tentativa de entender se o uso dos filtros se dava melhor em ambientes como este surgiu. Assim, com a técnica de *undersampling*, os índices observados no *Information Gain* e *cfssubseteval* mais que dobraram, mostrando que a eficácia desses seletores está condicionado à estrutura do *dataset* utilizado.

Outro ponto que cabe destaque, é a importância de que quanto mais balanceado for um *dataset*, melhores são os resultados expostos, no ponto de vista de qualidade de classificação, mediante análise do *F1-Score* de cada experimento. Em cenários desbalanceados, foi comum notar taxas de acerto globais altas, porém com medidas de *F1-Score* baixas (especialmente no contexto de detecção de ataque).

Quanto aos algoritmos classificadores, é notório o tempo maior para construção e testagem dos modelos ao se usar o SVM. Isso pode ser explicado pela forma como este algoritmo funciona, onde cada vetor é utilizado no cálculo para a classificação, ao contrário do *Random Forest*, que constrói sua árvore de decisão aleatoriamente.

Ao observar os resultados, percebe-se que a divisão do dataset em 60-40, 70-30 e 80-20, não influencia

de maneira significativa, os resultados. Além disso, percebe-se que o uso de algoritmos que selecionam atributos relevantes no contexto do conjunto de dados é importante, haja vista a melhora em termo de métricas desempenho e performance em muitos dos cenários analisados.

Já considerando as métricas de desempenho, especialmente o *F1-Score* que é uma harmônica entre as métricas de *Precision* e *Recall*. Neste ponto, considerando os intervalos de confiança para essa métrica, a um nível de confiança de 95%, os melhores resultados foram para o uso do *Correlation*, em quaisquer divisão de *dataset*: 60-40, 70-30 e 80-20. Veja a Tabela 4.73:

Tabela 4.73: TP e FP do experimento 4.3.2.4

Split	**Intervalo de confiança para o F1-Score**
60	Entre 0,918 e 0,987
70	Entre 0,921 e 0,986
80	Entre 0,933 e 0,991

Todos os valores podem ser encontrados no Apêndica I.a, I.b e I.c, ao final deste documento.

Dos oito cenários diferentes (NSL-KDD com SVM e *Random Forest*, CIC-IDS com três balanceamentos diferentes e SVM e *Random Forest*), apenas em um houve vantagem real no uso de todos os atributos (sem seleção). Nas demais, seja o *Correlation* seja o *Information Gain* se mostraram escolhas mais inteligentes na prototipagem de um IDS real. Já o cfssubseteval possuiu o maior número de resultados fora do intervalo de confiança.

Comparando com o exposto na Tabela 4.1 - 4.8, o uso dos 41 atributos influencia melhores resultados de acertos de classe, mas não o suficiente para que seu uso seja determinante em processo de treinamento e teste, pois a diferença entre as demais não ultrapassa 0,05 pontos percentuais.

Na verdade, a indicação errada de uma classe, verificada pelo segundo maior índice de falsos positivos do experimento, sugere que alguns dos dados do conjunto de dados são irrelevantes para o processo de treinamento.

Outro aspecto relevante consiste na observação de que a redução do número de características (*features*) nem sempre resulta em tempos menores de processamento ou indica ineficiência do seletor. Portanto, não é possível inferir, com base apenas na quantidade de atributos selecionados, quais resultados são esperados, uma vez que outros fatores intervenientes no processo, tais como o algoritmo classificador utilizado e as características dos dados contidos em cada uma das colunas, devem ser considerados.

4.6 COMPARAÇÃO DE RESULTADOS COM TRABALHOS CORRELATOS

Ao comparar os resultados da avaliação de desempenho obtidos neste trabalho com os relatados na Seção II deste documento (3.8), pode-se observar que há uma vantagem na utilização da engenharia de atributos, pois o *F1-Score* foi maior em relação aos outros. Vale ressaltar que a métrica de desempenho que representa este trabalho refere-se aos resultados obtidos com o uso do subconjunto cfssubseteval, com

uma razão de divisão do conjunto de dados de 60-40.

Tabela 4.74: Comparação com os trabalhos relatados

	Precision	Recall	*F1-Measure*
S. Thirimanne et al. (50)	0,925	0,720	0,809
A. Ghorbani and S. Fakhrahmad (52)	0,938	0,862	0,898
Este trabalho	0,973	0,972	0,972

Como A. Thakkar e R. Lohiya (53) também utilizaram a Correlação como método de seleção de atributos em seus trabalhos, a comparação apresentada na tabela valida o uso do SVM contra o DNN, apesar da pequena margem existente entre o F- Pontuações obtidas por ambos.

Tabela 4.75: Comparação com os trabalhos relatados

	Precision	Recall	*F1-Measure*
A. Thakkar and R. Lohiya (53)	0,995	0,912	0,952
Este trabalho	0,974	0,973	0,973

Para experimentos futuros, pretende-se alimentar os classificadores em tempo real, para simular verdadeiramente um ambiente de produção, e mitigar ou impedir possíveis ataques à rede estudada.

Além disso, faz-se necessário aprofundar as pesquisas no que tange o uso de técnicas de seleção de atributos em *datasets* não balanceados, e porque seu comportamente se difere do contexto onde o conjunto de dados é considerado mais balanceado.

5 CONCLUSÃO

Em conclusão, este trabalho explora as diversas técnicas de seleção de atributos de dois *datasets* diferentes, com estruturas distintas (balanceado e desbalanceado), atrelado a classificadores também diferentes e com uso do conjunto de dados em três formas: 60-40, 70-30 e 80-20.

O objetivo proposto desse trabalho foi alcançado, uma vez que o uso de técnicas de *Feature Engineering* reduziu os tempos de construção e testagem dos modelos criados pelos classificadores.

Com isso, a partir da análise dos resultados obtidos, pode-se concluir que a adoção de técnicas de filtragem de atributos no conjunto de dados foi satisfatória. Isso pode ser explicado pelo fato de as métricas de desempenho terem se mantido semelhantes e, em alguns casos, até melhores quando comparadas ao uso integral do *dataset*.

Outra conclusão importante, é que os modelos de seleção de atributos não podem ser utilizados sem quaisquer discriminações. Notou-se que o uso em *datasets* desbalanceados não foi satisfatória em um dos três filtros utilizados. Ou seja, por mais que a solução de diminuir a dimensionalidade desses conjuntos de dados pareça ser trivial, deve-se levar em conta outros aspectos para o seu uso racional.

Além disso, para os *datasets* escolhidos, a utilização do classificador *Random Forest* foi mais indicada, já que suas métricas de desempenho foram relativamente melhores, com tempos de construção e testagem mais rápidas.

De modo resumido, o trabalho contribuiu para uma análise mais acertiva em relação ao uso dos *datasets*, uma vez que, mesmo com a remoção de dados, houve manutenção ou melhoria das métricas de desempenho, se compararmos com os experimento que houveram o uso integral do conjunto de dados.

REFERÊNCIAS BIBLIOGRÁFICAS

1 FAIZ, O. S. M. N.; MUHAMMAD, A. W. Machine learning-based feature engineering to detect ddos attacks. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi*, v. 11, p. 176–182, 2022.

2 TOKER, O. Performance bounds for cyberattack detectors using multiple observations. In: *SoutheastCon 2022*. [S.l.: s.n.], 2022. p. 104–109.

3 ULLAH, S.; AHMAD, J.; KHAN, M. A.; ALKHAMMASH, E. H.; HADJOUNI, M.; GHADI, Y. Y.; SAEED, F.; PITROPAKIS, N. A new intrusion detection system for the internet of things via deep convolutional neural network and feature engineering. *Sensors*, v. 22, n. 10, 2022. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/22/10/3607>.

4 GUEZZAZ, A.; AZROUR, M.; BENKIRANE, S.; MOHY-EDDINE, M.; ATTOU, H.; DOUIBA, M. A lightweight hybrid intrusion detection framework using machine learning for edge-based iiot security. *International Arab Journal of Information Technology*, v. 19, 09 2022.

5 AHMADI, A.; NABIPOUR, M.; TAHERI, S.; MOHAMMADI-IVATLOO, B.; VAHIDINASAB, V. A new false data injection attack detection model for cyberattack resilient energy forecasting. *IEEE Transactions on Industrial Informatics*, v. 19, n. 1, p. 371–381, 2023.

6 TAWIL, A. A.; SABRI, K. E. A feature selection algorithm for intrusion detection system based on moth flame optimization. In: *2021 International Conference on Information Technology (ICIT)*. [S.l.: s.n.], 2021. p. 377–381.

7 KHODASKAR, M.; MEDHANE, D.; INGLE, R.; BUCHADE, A.; KHODASKAR, A. Feature-based intrusion detection system with support vector machine. In: *2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*. [S.l.: s.n.], 2022. p. 1–7.

8 KHODASKAR, M.; MEDHANE, D.; INGLE, R.; BUCHADE, A.; KHODASKAR, A. Feature-based intrusion detection system with support vector machine. In: *2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*. [S.l.: s.n.], 2022. p. 1–7.

9 SONG, X.-F.; ZHANG, Y.; GONG, D.-W.; GAO, X.-Z. A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data. *IEEE Transactions on Cybernetics*, v. 52, n. 9, p. 9573–9586, 2022.

10 SANG, B.; CHEN, H.; YANG, L.; LI, T.; XU, W. Incremental feature selection using a conditional entropy based on fuzzy dominance neighborhood rough sets. *IEEE Transactions on Fuzzy Systems*, v. 30, n. 6, p. 1683–1697, 2022.

11 YADAV, R.; SREEDEVI, I.; GUPTA, D. Augmentation in performance and security of wsns for iot applications using feature selection and classification techniques. *Alexandria Engineering Journal*, 2022. ISSN 1110-0168. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1110016822006871>.

12 FRANK, M. A. H. E.; WITTEN, I. H. Data mining: Practical machine learning tools and techniques. *Morgan Kaufmann*, 2016.

13 TAVALLAEE E. BAGHERI, W. L. M.; GHORBANI, A. A. A detailed analysis of the kdd cup 99 data set. *2nd IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

14 SHARAFALDIN ARASH HABIBI LASHKARI, S. H. I.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018.

15 STIAWAN, D.; HERYANTO, A.; BARDADI, A.; RINI, D. P.; SUBROTO, I. M. I.; KURNIABUDI; IDRIS, M. Y. B.; ABDULLAH, A. H.; KERIM, B.; BUDIARTO, R. An approach for optimizing ensemble intrusion detection systems. *IEEE Access*, v. 9, p. 6930–6947, 2021.

16 ISLAM, M. R.; LIMA, A. A.; DAS, S. C.; MRIDHA, M. F.; PRODEEP, A. R.; WATANOBE, Y. A comprehensive survey on the process, methods, evaluation, and challenges of feature selection. *IEEE Access*, v. 10, p. 99595–99632, 2022.

17 SOUSA, M. S. D.; VEIGA, C. E. L.; ALBUQUERQUE, R. D. O.; GIOZZA, W. F. Information gain applied to reduce model-building time in decision-tree-based intrusion detection system. In: *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2022. p. 1–6.

18 CISTI 2022 Cover Page. In: 2022 17th Iberian Conference on Information Systems and Technologies (CISTI). [S.l.: s.n.], 2022. p. c1–c1.

19 AL-A'ARAJI, N. H.; AL-MAMORY, S. O.; AL-SHAKARCHI, A. H. Classification and clustering based ensemble techniques for intrusion detection systems: A survey. *Journal of Physics: Conference Series*, IOP Publishing, v. 1818, n. 1, p. 012106, mar 2021. Disponível em: <https://dx.doi.org/10.1088/1742-6596/1818/1/012106>.

20 CHAHAL, J. K.; GANDHI, V.; KAUSHAL, P.; RAMKUMAR, K.; KAUR, A.; MITTAL, S. Kas-ids: A machine learning based intrusion detection system. In: *2021 6th International Conference on Signal Processing, Computing and Control (ISPCC)*. [S.l.: s.n.], 2021. p. 90–95.

21 DALBHANJAN, R.; CHATTERJEE, S.; GOGOI, R.; PATHAK, T.; SAHAY, S. 3 implementation of honeypot, nids, and hids technologies in soc environment. In: ____. *Implementing Enterprise Cybersecurity with Open-Source Software and Standard Architecture*. [S.l.: s.n.], 2021. p. 51–66.

22 FERNANDES, M. *IDS e IPS: detecção e bloqueio de ameaças!* 2022. Urlhttps://blog.starti.com.br/ids-ips/.

23 RASHID, M.; KAMRUZZAMAN, J.; HASSAN, M.; IMAM, T.; GORDON, S. Cyberattacks detection in iot-based smart city applications using machine learning techniques. *International Journal of Environmental Research and Public Health*, v. 17, p. 9347, 12 2020.

24 OBAID, W.; NASSIF, A. B. The effects of resampling on classifying imbalanced datasets. In: *2022 Advances in Science and Engineering Technology International Conferences (ASET)*. [S.l.: s.n.], 2022. p. 1–6.

25 DALAL, K. R. Analysing the role of supervised and unsupervised machine learning in iot. In: *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. [S.l.: s.n.], 2020. p. 75–79.

26 SHARMA, A.; KAUR, A.; SEMWAL, A. Supervised and unsupervised prediction application of machine learning. In: *2022 International Conference on Cyber Resilience (ICCR)*. [S.l.: s.n.], 2022. p. 1–5.

27 YE, Q.; HUANG, P.; ZHANG, Z.; ZHENG, Y.; FU, L.; YANG, W. Multiview learning with robust double-sided twin svm. *IEEE Transactions on Cybernetics*, v. 52, n. 12, p. 12745–12758, 2022.

28 KUYORO, A. O.; ALIMI, S.; AWODELE, O. Comparative analysis of the performance of various support vector machine kernels. In: *2022 5th Information Technology for Education and Development (ITED)*. [S.l.: s.n.], 2022. p. 1–7.

29 DAN, Z.; YI, C. Realization and simulation analysis of intelligent data fusion algorithm based on support vector machine. In: *2022 International Conference on Information System, Computing and Educational Technology (ICISCET)*. [S.l.: s.n.], 2022. p. 323–327.

30 PAVITHRA, M.; GEETHA, B. Prediction of chronic kidney cancer using rbf support vector machine compared with random forest for better accuracy. In: *2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*. [S.l.: s.n.], 2022. p. 1–5.

31 GOPALAN, S. S.; RAVIKUMAR, D.; LINEKAR, D.; RAZA, A.; HASIB, M. Balancing approaches towards ml for ids: A survey for the cse-cic ids dataset. In: *2020 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*. [S.l.: s.n.], 2021. p. 1–6.

32 SHAUKAT, S.; ALI, A.; BATOOL, A.; ALQAHTANI, F.; KHAN, J. S.; ARSHAD; AHMAD, J. Intrusion detection and attack classification leveraging machine learning technique. In: *2020 14th International Conference on Innovations in Information Technology (IIT)*. [S.l.: s.n.], 2020. p. 198–202.

33 SANG, B.; CHEN, H.; YANG, L.; LI, T.; XU, W. Incremental feature selection using a conditional entropy based on fuzzy dominance neighborhood rough sets. *IEEE Transactions on Fuzzy Systems*, v. 30, n. 6, p. 1683–1697, 2022.

34 LI, X.; WANG, Y.; RUIZ, R. A survey on sparse learning models for feature selection. *IEEE Transactions on Cybernetics*, v. 52, n. 3, p. 1642–1660, 2022.

35 WATANABE, S. *Pattern Recognition: Human and Mechanical First Edition*. [S.l.]: Wiley, 1985.

36 JAIN, A.; DUIN, R.; MAO, J. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 1, p. 4–37, 2000.

37 HUA, J.; XIONG, Z.; LOWEY, J.; SUH, E.; DOUGHERTY, E. R. Optimal number of features as a function of sample size for various classification rules. *Bioinformatics*, v. 21, n. 8, p. 1509–1515, 11 2004. ISSN 1367-4803. Disponível em: <https://doi.org/10.1093/bioinformatics/bti171>.

38 QIAO, Q.; YUNUSA-KALTUNGO, A.; EDWARDS, R. E. Feature selection strategy for machine learning methods in building energy consumption prediction. *Energy Reports*, v. 8, p. 13621–13654, 2022. ISSN 2352-4847. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2352484722020601>.

39 MANDALA, S.; RAMADHAN, A. I.; ROSALINDA, M.; YAFOOZ, W. M.; KHOHAR, R. H. Ddos detection by using information gain-naïve bayes. In: *2022 2nd International Conference on Intelligent Cybernetics Technology Applications (ICICyTA)*. [S.l.: s.n.], 2022. p. 283–288.

40 KURNIABUDI; STIAWAN, D.; DARMAWIJOYO; IDRIS, M. Y. B.; BAMHDI, A. M.; BUDIARTO, R. Cicids-2017 dataset feature analysis with information gain for anomaly detection. *IEEE Access*, v. 8, p. 132911–132921, 2020.

41 S, A. R. M.; R, N. C.; B, C. B.; RAFI, M.; R, S. B. Online feature selection using pearson correlation technique. In: *2022 IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*. [S.l.: s.n.], 2022. v. 7, p. 172–177.

42 CLASS CorrelationAttributeEval. <https://weka.sourceforge.io/doc.dev/weka/attributeSelection/CorrelationAttributeEval.html>. Accessed: 2023-04-23.

43 KUMAR, S.; BHUSHAN, B.; BHAMBHU, L.; THAKUR, M.; MOHAPATRA, U. M.; CHOUBEY, D. K. Medical datasets classification using a hybrid genetic algorithm for feature selection based on pearson correlation coefficient. In: *2022 International Conference on Machine Learning, Computer Systems and Security (MLCSS)*. [S.l.: s.n.], 2022. p. 214–218.

44 ZHAO, R.; MU, Y.; ZOU, L.; WEN, X. A hybrid intrusion detection system based on feature selection and weighted stacking classifier. *IEEE Access*, v. 10, p. 71414–71426, 2022.

45 CLASS CfsSubsetEval. <https://weka.sourceforge.io/doc.dev/weka/attributeSelection/CfsSubsetEval.html>. Accessed: 2023-04-23.

46 CHEN, C.; SONG, L.; BO, C.; SHUO, W. A support vector machine with particle swarm optimization grey wolf optimizer for network intrusion detection. In: *2021 International Conference on Big Data Analysis and Computer Science (BDACS)*. [S.l.: s.n.], 2021. p. 199–204.

47 SINGHAL, A.; MAAN, A.; CHAUDHARY, D.; VISHWAKARMA, D. A hybrid machine learning and data mining based approach to network intrusion detection. In: *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*. [S.l.: s.n.], 2021. p. 312–318.

48 YEDUKONDALU, G.; BINDU, G. H.; PAVAN, J.; VENKATESH, G.; SAITEJA, A. Intrusion detection system framework using machine learning. In: *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*. [S.l.: s.n.], 2021. p. 1224–1230.

49 ROKADE, M. D.; SHARMA, Y. K. Mlids: A machine learning approach for intrusion detection for real time network dataset. In: *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*. [S.l.: s.n.], 2021. p. 533–536.

50 THIRIMANNE, S.; JAYAWARDANA, L.; LIYANAARACHCHI, P.; YASAKETHU, L. Comparative algorithm analysis for machine learning based intrusion detection system. In: *2021 10th International Conference on Information and Automation for Sustainability (ICIAfS)*. [S.l.: s.n.], 2021. p. 191–196.

51 AHMED, O. I.; VAROL, C. Detection of web attacks via part classifier. In: *2021 9th International Symposium on Digital Forensics and Security (ISDFS)*. [S.l.: s.n.], 2021. p. 1–4.

52 GHORBANI, A.; FAKHRAHMAD, S. M. A deep learning approach to network intrusion detection using a proposed supervised sparse auto-encoder and svm. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, v. 46, n. 3, p. 829–846, Sep 2022. ISSN 2364-1827. Disponível em: <https://doi.org/10.1007/s40998-022-00498-1>.

53 THAKKAR, A.; LOHIYA, R. Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system. *Information Fusion*, v. 90, p. 353–363, 2023. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1566253522001646>.

54 OLIVEIRA, F. B. de. FRAMEWORK PARA DETECÇÃO DE ATAQUES DOS EM DISPOSITIVOS IOT, UTILIZANDO ABORDAGENS DE APRENDIZADO DE MÁQUINAS. *Programa de Mestrado Profissional em Engenharia Elétrica da Universidade de Brasília*, 06 2023.

I.1.1 Resultados para os experimentos com divisão 60-40

	Experimento			*Dataset*	Balanceamento			Algoritmo classificador		Seletor de atributos				Métricas de Desempenho					Tempos em segundos	
	Seção	Tabelas	Figuras		Sem	*Under*	*Over*	SVM	RF	*Information Gain*	*Correlation*	CFS	Sem seletor	TP	FP	*Precision*	*Recall*	*F1-Score*	Construção	Teste
1	4.1.1.1	4.1, 4.2	4.1	NSL-KDD	X			X		X				0.980	0.020	0.981	9.979	0.981	80.1	4.83
2	4.1.1.2	4.3, 4.4	4.2	NSL-KDD	X			X			X			0.936	0.064	0.940	0.936	0.937	57.2	3.91
3	4.1.1.3	4.5, 4.6	4.3	NSL-KDD	X			X				X		0.973	0.026	0.973	0.973	0.972	40.9	3.97
4	4.1.1.4	4.7, 4.8	4.4	NSL-KDD	X			X					X	0.966	0.034	0.970	0.967	0.968	129.8	8.12
5	4.1.2.1	4.10, 4.11	4.6	NSL-KDD	X				X	X				0.995	0.005	0.995	0.995	0.995	1.18	0.17
6	4.1.2.2	4.12, 4.13	4.7	NSL-KDD	X				X		X			0.967	0.032	0.967	0.967	0.967	0.79	0.09
7	4.1.2.3	4.14, 4.15	4.8	NSL-KDD	X				X			X		0.992	0.008	0.993	0.992	0.993	1.81	0.22
8	4.1.2.4	4.16, 4.17	4.9	NSL-KDD	X				X				X	0.997	0.003	0.997	0.997	0.997	1.62	0.14
9	4.2.1.1	4.19, 4.20	4.11	CIC-IDS	X			X		X				0.680	0.320	0.983	0.680	0.761	4009.12	209.6
10	4.2.1.2	4.21, 4.22	4.12	CIC-IDS	X			X			X			0.980	0.020	0.958	0.980	0.968	787.33	54.57
11	4.2.1.3	4.23, 4.24	4.13	CIC-IDS	X			X				X		0.706	0.293	0.996	0.706	0.790	2232.46	167.85
12	4.2.1.4	4.25, 4.26	4.14	CIC-IDS	X			X					X	0.981	0.019	0.967	0.981	0.972	8641.39	509.34
13	4.2.2.1	4.28, 4.29	4.16	CIC-IDS	X				X	X				0.877	0.123	0.948	0.866	0.909	27.75	0.51
14	4.2.2.2	4.30, 4.31	4.17	CIC-IDS	X				X		X			0.987	0.012	0.960	0.987	0.970	5.09	0.42
15	4.2.2.3	4.32, 4.33	4.18	CIC-IDS	X				X			X		0.833	0.168	0.993	0.833	0.897	8.56	0.48
16	4.2.2.4	4.34, 4.35	4.19	CIC-IDS	X				X				X	0.975	0.025	0.997	0.975	0.986	30.45	0.57
17	4.3.1.1	4.37, 4.38	4.21	CIC-IDS		X		X		X				0.838	0.162	0.859	0.838	0.833	1.09	0.16
18	4.3.1.2	4.39, 4.40	4.22	CIC-IDS		X		X			X			0.986	0.014	0.985	0.985	0.985	0.23	0.11
19	4.3.1.3	4.41, 4.42	4.23	CIC-IDS		X		X				X		0.871	0.129	0.889	0.871	0.867	0.21	0.03
20	4.3.1.4	4.43, 4.44	4.24	CIC-IDS		X		X					X	0.983	0.016	0.981	0.983	0.985	1.88	0.34
21	4.3.2.1	4.46, 4.47	4.26	CIC-IDS		X			X	X				0.987	0.012	0.988	0.988	0.988	0.39	0.02
22	4.3.2.2	4.48, 4.49	4.27	CIC-IDS		X			X		X			0.992	0.008	0.992	0.997	0.992	0.22	0.03
23	4.3.2.3	4.50, 4.51	4.28	CIC-IDS		X			X			X		0.989	0.010	0.991	0.991	0.991	0.21	0.02
24	4.3.2.4	4.52, 4.53	4.29	CIC-IDS		X			X				X	0.994	0.006	0.994	0.994	0.994	0.50	0.03
25	4.4.1.1	4.55, 4.56	4.31	CIC-IDS			X	X		X				0.945	0.054	0.945	0.938	0.945	10012.32	1253.33
26	4.4.1.2	4.57, 4.58	4.32	CIC-IDS			X	X			X			0.995	0.005	0.997	0.995	0.990	6763.49	953.14
27	4.4.1.3	4.59, 4.60	4.33	CIC-IDS			X	X				X		0.907	0.142	0.913	0.907	0.909	2694.02	673.00
28	4.4.1.4	4.61, 4.62	4.34	CIC-IDS			X	X					X	0.999	0.001	0.999	0.999	0.999	52864.03	2499.56
29	4.4.2.1	4.64, 4.65	4.36	CIC-IDS			X		X	X				0.995	0.004	0.995	0.995	0.995	59.11	1.79
30	4.4.2.2	4.66, 4.67	4.37	CIC-IDS			X		X		X			0.971	0.020	0.971	0.979	0.975	22.45	1.21
31	4.4.2.3	4.68, 4.69	4.38	CIC-IDS			X		X			X		0.995	0.005	0.995	0.995	0.995	39.94	1.42
32	4.4.2.4	4.70, 4.71	4.39	CIC-IDS			X		X				X	1.000	0.000	0.999	0.999	0.999	120.67	1.61

I.1.2 Resultados para os experimentos com divisão 70-30

	Experimento			***Dataset***	**Balanceamento**			**Algoritmo classificador**		**Seletor de atributos**				**Métricas de Desempenho**					**Tempos em segundos**	
	Seção	Tabelas	Figuras		Sem	*Under*	*Over*	SVM	RF	*Information Gain*	*Correlation*	CFS	Sem seletor	TP	FP	*Precision*	*Recall*	*F1-Score*	Construção	Teste
1	4.1.1.1	4.1, 4.2	4.1	NSL-KDD	X			X		X				0.981	0.018	0.982	0.982	0.981	81.17	4.83
2	4.1.1.2	4.3, 4.4	4.2	NSL-KDD	X			X			X			0.946	0.059	0.946	0.941	0.943	58.32	3.08
3	4.1.1.3	4.5, 4.6	4.3	NSL-KDD	X			X				X		0.975	0.024	0.973	0.975	0.973	42.63	3.34
4	4.1.1.4	4.7, 4.8	4.4	NSL-KDD	X			X					X	0.969	0.031	0.973	0.969	0.970	132.43	7.17
5	4.1.2.1	4.10, 4.11	4.6	NSL-KDD	X				X	X				0.996	0.004	0.996	0.996	0.997	1.19	0.09
6	4.1.2.2	4.12, 4.13	4.7	NSL-KDD	X				X		X			0.967	0.032	0.968	0.967	0.967	0.80	0.06
7	4.1.2.3	4.14, 4.15	4.8	NSL-KDD	X				X			X		0.992	0.008	0.993	0.992	0.993	1.82	0.15
8	4.1.2.4	4.16, 4.17	4.9	NSL-KDD	X				X				X	0.998	0.002	0.998	0.998	0.998	1.63	0.09
9	4.2.1.1	4.19, 4.20	4.11	CIC-IDS	X			X		X				0.692	0.308	0.997	0.692	0.776	4381.91	188.40
10	4.2.1.2	4.21, 4.22	4.12	CIC-IDS	X			X			X			0.981	0.018	0.958	0.981	0.970	809.99	51.14
11	4.2.1.3	4.23, 4.24	4.13	CIC-IDS	X			X				X		0.716	0.256	0.995	0.716	0.800	2778.76	104.07
12	4.2.1.4	4.25, 4.26	4.14	CIC-IDS	X			X					X	0.987	0.012	0.976	0.987	0.981	9123.19	402.09
13	4.2.2.1	4.28, 4.29	4.16	CIC-IDS	X				X	X				0.869	0.130	0.956	0.869	0.908	28.13	0.41
14	4.2.2.2	4.30, 4.31	4.17	CIC-IDS	X				X		X			0.991	0.009	0.956	0.991	0.970	5.17	0.31
15	4.2.2.3	4.32, 4.33	4.18	CIC-IDS	X				X			X		0.833	0.168	0.993	0.833	0.897	8.89	0.38
16	4.2.2.4	4.34, 4.35	4.19	CIC-IDS	X				X				X	0.975	0.025	0.997	0.975	0.986	31.14	0.39
17	4.3.1.1	4.37, 4.38	4.21	CIC-IDS		X		X		X				0.852	0.148	0.874	0.852	0.846	1.12	0.14
18	4.3.1.2	4.39, 4.40	4.22	CIC-IDS		X		X			X			0.986	0.014	0.985	0.985	0.985	0.32	0.08
19	4.3.1.3	4.41, 4.42	4.23	CIC-IDS		X		X				X		0.870	0.129	0.886	0.870	0.865	0.28	0.02
20	4.3.1.4	4.43, 4.44	4.24	CIC-IDS		X		X					X	0.978	0.021	0.979	0.982	0.983	1.98	0.29
21	4.3.2.1	4.46, 4.47	4.26	CIC-IDS		X			X	X				0.989	0.012	0.989	0.989	0.989	0.40	0.02
22	4.3.2.2	4.48, 4.49	4.27	CIC-IDS		X			X		X			0.992	0.008	0.991	0.991	0.991	0.30	0.02
23	4.3.2.3	4.50, 4.51	4.28	CIC-IDS		X			X			X		0.989	0.010	0.990	0.990	0.990	0.21	0.01
24	4.3.2.4	4.52, 4.53	4.29	CIC-IDS		X			X				X	0.994	0.006	0.995	0.966	0.981	0.62	0.03
25	4.4.1.1	4.55, 4.56	4.31	CIC-IDS			X	X		X				0.964	0.036	0.944	0.936	0.940	10127.27	1107.87
26	4.4.1.2	4.57, 4.58	4.32	CIC-IDS			X	X			X			0.995	0.005	0.996	0.995	0.983	6838.09	900.75
27	4.4.1.3	4.59, 4.60	4.33	CIC-IDS			X	X				X		0.907	0.142	0.917	0.907	0.910	3624.99	616.09
28	4.4.1.4	4.61, 4.62	4.34	CIC-IDS			X	X					X	0.999	0.001	0.999	0.999	0.999	57704.83	2418.62
29	4.4.2.1	4.64, 4.65	4.36	CIC-IDS			X		X	X				0.995	0.004	0.995	0.995	0.995	59.56	1.21
30	4.4.2.2	4.66, 4.67	4.37	CIC-IDS			X		X		X			0.971	0.020	0.971	0.979	0.975	23.53	1.03
31	4.4.2.3	4.68, 4.69	4.38	CIC-IDS			X		X			X		0.995	0.005	0.995	0.995	0.995	40.04	1.20
32	4.4.2.4	4.70, 4.71	4.39	CIC-IDS			X		X				X	1.000	0.000	0.999	0.999	0.999	123.70	1.40

I.1.3 Resultados para os experimentos com divisão 80-20

	Experimento			***Dataset***	**Balanceamento**			**Algoritmo classificador**		**Seletor de atributos**				**Métricas de Desempenho**					**Tempos em segundos**	
	Seção	Tabelas	Figuras		Sem	*Under*	*Over*	SVM	RF	*Information Gain*	*Correlation*	CFS	Sem seletor	TP	FP	*Precision*	*Recall*	*F1-Score*	Construção	Teste
1	4.1.1.1	4.1, 4.2	4.1	NSL-KDD	X			X		X				0.982	0.017	0.982	0.982	0.983	81.7	2.99
2	4.1.1.2	4.3, 4.4	4.2	NSL-KDD	X			X			X			0.942	0.059	0.947	0.943	0.943	59.3	2.33
3	4.1.1.3	4.5, 4.6	4.3	NSL-KDD	X			X				X		0.974	0.027	0.972	0.974	0.973	44.1	2.92
4	4.1.1.4	4.7, 4.8	4.4	NSL-KDD	X			X					X	0.969	0.031	0.974	0.969	0.973	138.3	5.23
5	4.1.2.1	4.10, 4.11	4.6	NSL-KDD	X				X	X				0.996	0.003	0.996	0.996	0.996	1.21	0.07
6	4.1.2.2	4.12, 4.13	4.7	NSL-KDD	X				X		X			0.996	0.003	0.996	0.996	0.996	0.81	0.04
7	4.1.2.3	4.14, 4.15	4.8	NSL-KDD	X				X			X		0.992	0.008	0.994	0.992	0.993	1.85	0.11
8	4.1.2.4	4.16, 4.17	4.9	NSL-KDD	X				X				X	0.998	0.002	0.998	0.998	0.998	1.66	0.06
9	4.2.1.1	4.19, 4.20	4.11	CIC-IDS	X			X		X				0.714	0.285	0.969	0.714	0.794	4381.91	153.14
10	4.2.1.2	4.21, 4.22	4.12	CIC-IDS	X			X			X			0.970	0.021	0.961	0.979	0.961	841.78	50.03
11	4.2.1.3	4.23, 4.24	4.13	CIC-IDS	X			X				X		0.718	0.281	0.972	0.974	0.973	2991.22	100.57
12	4.2.1.4	4.25, 4.26	4.14	CIC-IDS	X			X					X	0.993	0.006	0.977	0.993	0.986	10043.09	374.09
13	4.2.2.1	4.28, 4.29	4.16	CIC-IDS	X				X	X				0.863	0.136	0.958	0.863	0.905	28.55	0.28
14	4.2.2.2	4.30, 4.31	4.17	CIC-IDS	X				X		X			0.988	0.012	0.957	0.988	0.972	5.19	0.26
15	4.2.2.3	4.32, 4.33	4.18	CIC-IDS	X				X			X		0.820	0.180	0.993	0.820	0.898	9.01	0.22
16	4.2.2.4	4.34, 4.35	4.19	CIC-IDS	X				X				X	0.975	0.024	0.997	0.975	0.986	32.45	0.29
17	4.3.1.1	4.37, 4.38	4.21	CIC-IDS		X		X		X				0.850	0.149	0.868	0.850	0.841	1.26	0.11
18	4.3.1.2	4.39, 4.40	4.22	CIC-IDS		X		X			X			0.986	0.014	0.985	0.985	0.985	0.45	0.07
19	4.3.1.3	4.41, 4.42	4.23	CIC-IDS		X		X				X		0.872	0.129	0.882	0.872	0.865	0.31	0.02
20	4.3.1.4	4.43, 4.44	4.24	CIC-IDS		X		X					X	0.984	0.016	0.983	0.984	0.986	2.04	0.25
21	4.3.2.1	4.46, 4.47	4.26	CIC-IDS		X			X	X				0.989	0.011	0.990	0.989	0.990	0.48	0.01
22	4.3.2.2	4.48, 4.49	4.27	CIC-IDS		X			X		X			0.992	0.008	0.990	0.989	0.989	0.32	0.01
23	4.3.2.3	4.50, 4.51	4.28	CIC-IDS		X			X			X		0.989	0.010	0.990	0.990	0.990	0.28	0.01
24	4.3.2.4	4.52, 4.53	4.29	CIC-IDS		X			X				X	0.994	0.006	0.996	0.995	0.995	0.69	0.01
25	4.4.1.1	4.55, 4.56	4.31	CIC-IDS			X	X		X				0.949	0.051	0.994	0.949	0.970	10199.30	998.73
26	4.4.1.2	4.57, 4.58	4.32	CIC-IDS			X	X			X			0.995	0.005	0.994	0.994	0.992	7004.81	893.12
27	4.4.1.3	4.59, 4.60	4.33	CIC-IDS			X	X				X		0.906	0.144	0.915	0.906	0.909	4023.95	576.71
28	4.4.1.4	4.61, 4.62	4.34	CIC-IDS			X	X					X	0.999	0.001	0.999	0.999	0.999	59064.39	2376.17
29	4.4.2.1	4.64, 4.65	4.36	CIC-IDS			X		X	X				0.995	0.004	0.995	0.995	0.995	59.93	1.06
30	4.4.2.2	4.66, 4.67	4.37	CIC-IDS			X		X		X			0.971	0.020	0.971	0.979	0.975	23.59	0.62
31	4.4.2.3	4.68, 4.69	4.38	CIC-IDS			X		X			X		0.989	0.011	0.995	0.995	0.995	40.49	0.81
32	4.4.2.4	4.70, 4.71	4.39	CIC-IDS			X		X				X	1.000	0.000	0.999	0.999	0.999	124.07	0.83