MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

1 Ten REBECA CALAZANS DE BRITO
1 Ten YURI FAÇANHA BEZERRA

COMITÊS DE CLASSIFICADORES APLICADOS À DETECÇÃO DE PADRÕES MALICIOSOS NO TRÁFEGO DE REDES DE COMPUTADORES

Rio de Janeiro
2018

INSTITUTO MILITAR DE ENGENHARIA

1 Ten REBECA CALAZANS DE BRITO
1 Ten YURI FAÇANHA BEZERRA

COMITÊS DE CLASSIFICADORES APLICADOS À DETECÇÃO DE PADRÕES MALICIOSOS NO TRÁFEGO DE REDES DE COMPUTADORES

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: Prof. Ronaldo Ribeiro Goldschmidt - D.Sc.
Co-Orientador: Prof. Sérgio dos Santos Cardoso Silva - M.Sc.

Rio de Janeiro
2018

c2018

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 - Praia Vermelha
Rio de Janeiro - RJ CEP 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

005 de Brito, Rebeca Calazans
B862c Comitês de Classificadores Aplicados à Detecção de Padrões Maliciosos no Tráfego de Redes de Computadores / Rebeca Calazans de Brito, Yuri Façanha Bezerra, orientado por Ronaldo Ribeiro Goldschmidt e Sérgio dos Santos Cardoso Silva - Rio de Janeiro: Instituto Militar de Engenharia, 2018.

38p.: il.

Projeto de Fim de Curso (graduação) - Instituto Militar de Engenharia, Rio de Janeiro, 2018.

1. Curso de Graduação em Engenharia de Computação - projeto de fim de curso. 1. Inteligência Artivicial. 2. Defesa Cibernética. 3. Classificação. 4. Comitê de Classificadores. I. Goldschmidt, Ronaldo Ribeiro. II. Silva, S.rgio dos Santos Cardoso . III. Título. IV. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

1 Ten REBECA CALAZANS DE BRITO
1 Ten YURI FAÇANHA BEZERRA

COMITÊS DE CLASSIFICADORES APLICADOS À DETECÇÃO DE PADRÕES MALICIOSOS NO TRÁFEGO DE REDES DE COMPUTADORES

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: Prof. Ronaldo Ribeiro Goldschmidt - D.Sc.

Co-Orientador: Prof. Sérgio dos Santos Cardoso Silva - M.Sc.

Aprovado em 24 de setembro de 2018 pela seguinte Banca Examinadora:

Prof. Ronaldo Ribeiro Goldschmidt - D.Sc. do IME - Presidente

Prof. Sérgio dos Santos Cardoso Silva - M.Sc. do IME

Prof. Humberto Henriques de Arruda - M.Sc. do IME

Mauricio Costa Reis - M.Sc. do IME

Rio de Janeiro
2018

AGRADECIMENTOS

Primeiramente gostaríamos de agradecer a todos os que contribuíram, diretamente ou indiretamente, com o desenvolvimento desse trabalho.

Aos professores Ronaldo Goldschmidt, Sérgio Cardoso que nos orientaram e acompanharam de perto todo o desenvolvimento deste trabalho.

Ao Capitão Tosta, cuja ajuda foi indispensável para a conclusão desse projeto.

Ao Maurício Reis e ao Major Humberto, que aceitaram gentilmente fazer parte dessa banca e que contribuiram bastante com o desenvolvimento do projeto com seus conselhos durante as avaliações parciais.

Por fim, aos familiares e amigos que deram apoio e suporte para que pudéssemos superar todos os desafios.

“DON’T PANIC ”

DOUGLAS ADAM

SUMÁRIO

LISTA DE ILUSTRAÇÕES 7

LISTA DE TABELAS 8

1 INTRODUÇÃO 12
1.1 EB-CyberDef 12
1.2 Objetivo 14
1.3 Metodologia 15
1.4 Estrutura do Texto 15

2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 Aprendizado de máquina 16
2.2 Modelos preditivos e Classificação 16
2.3 Árvore de decisão 18
2.3.1 Formato de uma árvore de decisão 18
2.3.2 Regras para criação de partições 19
2.3.2.1 Entropia 20
2.3.2.2 Índice de Gini 20
2.3.3 CART 20
2.4 Algoritmo K-NN 21
2.4.1 Escolha de parâmetros 21
2.4.1.1 Escolha do valor de K 21
2.4.1.2 Métrica para cálculo de distâncias 22
2.4.2 Funcionamento 22
2.5 Support Vector Machine(SVM) 22
2.5.1 Hiperplanos 22
2.5.2 SVMs lineares 23
2.5.2.1 SVMs lineares de margem rígida 24
2.5.2.2 SVMs lineares de margem suave 24
2.5.3 SVMs não lineares 25
2.6 Naive Bayes 26
2.7 Perceptron Multicamadas 27
2.8 Comitês de classificadores 28

2.8.1 *Stacking* 28

3 PROTÓTIPO, EXPERIMENTOS E RESULTADOS 30

3.1 Protótipo desenvolvido 30

3.2 Base de dados utilizada 32

3.3 Desenvolvimento dos experimentos 32

3.4 Resultados 35

3.4.1 Árbitros 35

4 CONSIDERAÇÕES FINAIS 37

LISTA DE ILUSTRAÇÕES

FIG.1.1 Visão Macro-Funcional da Arquitetura da Fase de Produção/Operação do EB-CyberDef 14

FIG.2.1 Fases de uma classificação 17
FIG.2.2 Rodadas de uma validação cruzada 18
FIG.2.3 Exemplo de Árvore de Decisão e suas regiões no espaço de objetos 19
FIG.2.4 Divisão do espaço por meio de um hiperplano 23
FIG.2.5 Representação de hiperplano com margens rígidas 24
FIG.2.6 Representação de hiperplano com margens suaves 25
FIG.2.7 Exemplo de mapeamento em novo espaço 25
FIG.2.8 Modelo Computacional de Neurônio 27
FIG.2.9 Funcionamento do Perceptron Multicamadas 27
FIG.2.10 Comitê de classificadores em dois níveis 29

FIG.3.1 Programa rodando em *background* por meio do terminal 30
FIG.3.2 Interface desenvolvida para manipulação do comitê 31
FIG.3.3 Fluxo dos processos de classificação 34

LISTA DE TABELAS

TAB.3.1 Distribuição inicial dos rótulos para cada cenário do *dataset* 32
TAB.3.2 Avaliação individual dos classificadores. 35
TAB.3.3 Avaliação dos árbitros antes e após a junção com dados de entrada. 36

RESUMO

Com o aumento de uso das redes e do número de serviços disponíveis, tanto governamentais quanto privados, houve um grande crescimento da importância da internet na vida das pessoas. Como consequência, a Internet se tornou alvo em potencial para atividades ilícitas, surgindo a necessidade de desenvolver soluções de inteligência e segurança cibernética. Dessa forma o Exército Brasileiro viu a necessidade de desenvolver uma solução própria em defesa cibérnética. Dentro das possíveis soluções, aquelas que utilizam inteligência artificial têm sido amplamente estudadas com o objetivo de detectar padrões maliciosos. O presente projeto de final de curso faz parte de um projeto maior do Instituto Militar de Engenharia de desenvolver uma ferramenta própria de defesa cibernética capaz de analisar fluxos de redes visando a classificação de fluxos maliciosos. Como parte dessa ferramenta ele buscou desenvolver uma ferramenta para a criação de comitês de classificadores capaz de classificar fluxos gerados por Botnets. Para cumprir com tal objetivo, foram implementados diversos algoritmos de classificação que foram testados separadamente e em comitê. Os resultados dos testes mostram pouca diferença entre os resultados encontrados para os classificadores e os encontrados após se utilizar comitês implementados com o mesmo algorítmo. Além da implementação dos algorítmos, foi desenvilvida uma ferramenta que permite a criação de comitês através da seleção dos classificadores e do tipo de comitê utilizado.

ABSTRACT

With the increasing use of networks and the number of available services, both governmental and private, there has been a great increase in the importance of the internet in people's lives. As a result, the Internet has become a potential target for illicit activities, with the need to develop cyber security and intelligence solutions. In this way the Brazilian Army saw the need to develop its own solution in cybernetic defense. Among the possible solutions, those using artificial intelligence have been extensively studied in order to detect malicious patterns. This year-end project is part of a larger project of the Military Engineering Institute to develop its own cyber defense tool capable of analyzing network flows aiming at the classification of malicious flows. As part of this tool sought to develop a tool for the creation of classificatory committees capable of classifying flows generated by botnets. To fulfill this objective, several classification algorithms were implemented and they were tested separately and in committee. The results of the tests showed little difference between the results found for the classifiers and those found after using committees that use the same algorithm. In addition to the implementation of the algorithms, it has been developed a tool that allows the creation of committees through the selection of classifiers and the type of committee used.

1 INTRODUÇÃO

O número de ataques cibernéticos tem aumentado exponencialmente no mundo. É cada vez maior o risco de ameaças avançadas que focam em invadir ou destruir sistemas de infraestruturas críticas, com bases de dados governamentais e militares. Com isso, cresce também a demanda por soluções de inteligência e segurança cibernética [1]. Nesse contexto surgiu a necessidade do Exército Brasileiro criar um ambiente integrado de defesa cibérnética próprio, para apoio a detecção e ao combate a comportamentos maliciosos no tráfego de redes de computadores. A esse projeto deu-se o nome EB-CyberDef.

Além disso, inteligência Artificial (IA) e o Aprendizado de Máquina são tendências bastante utilizadas atualmente em vendas de tecnologias de segurança da informação [2]. Essas tecnologias podem basicamente ajudar aos computadores a agirem sozinhos contra ameaças. Assim, a partir de informações recebidas, a arquitetura consegue analisar o cenário do possível ataque e reagir da melhor forma possível para proteger as aplicações [2].

1.1 EB-CYBERDEF

O projeto EB-CyberDef é um projeto do Instituto Militar de Engenharia que apresenta como objetivo geral de desenvolver um protótipo de um ambiente computacional, denominado EB-CyberDef, para dar apoio à detecção e ao combate de comportamentos maliciosos no tráfego de redes de computadores.

O projeto se justifica do contexto do Exército Brasileiro(EB) no sentido de buscar uma solução própria para segurança da informação, na necessidade de se registrar e compartilhar soluções, na dificuldade de lidar com experimentos (*workflows* científicos) e dados em larga escala (*big data*) além da necessidade de se acompanhar o desempenho das soluções de monitoramento de tráfego malicioso adotadas dentro das instituições.

A arquitetura do projeto EB-CyberDef está dividida nos seguintes módulos, como ilustrado na figura 1.1.

- Coleta de Dados: O módulo de coleta dos dados da rede deve ser capaz de extrair informações de diferentes fontes de dados (traces, logs e fluxos) em diferentes formatos.

- Pré-processamento: Este módulo compreende operações de tratamento e padronização dos dados de redes coletados, de forma a prepará-los para serem processados pelo sistema.

- Detecção de Maliciosos Conhecidos: Módulo responsável pela identificação dos dados maliciosos conhecidos (vermelhos) através de sua assinatura, isto é, características físicas que podem ser percebidas sem a necessidade de mecanismos baseados em Inteligência Artificial e Aprendizado de Máquina.

- Defesa/Proteção: Confirmado que determinado fluxo é malicioso, este módulo deve acionar medidas de defesa e/ou proteção para mitigar o problema. Tais medidas são denominadas Funções de Proteção. Estas funções podem ser especializadas para firewalls, servidores e roteadores. Um exemplo de medida preventiva que pode ser tomada é a adição de uma regra em um firewall com o objetivo de bloquear, exclusivamente, o fluxo malicioso.

- Pré-processamento e Análise de Legitimidade: Este módulo compreende a execução de algoritmos inteligentes de análise de legitimidade dos dados, responsáveis por avaliar os padrões de dados, procurando classificá-los quanto ao seu potencial em se constituir um padrão malicioso. Cabe ressaltar que a execução dos algoritmos de análise de legitimidade pode ser precedida de diferentes processamentos que irão adequar (pré-processar) os dados conforme a necessidade de cada algoritmo.

- Conjugação de Pareceres sobre Legitimidade: Este módulo tem como propósito integrar os pareceres emitidos pelos algoritmos de legitimidade executados pelo módulo anterior a fim de obter um parecer único que reflita a impressão do sistema quanto à classificação do padrão analisado. Desta forma, o presente módulo deve classificar os dados em (i) certamente não maliciosos (verde), (ii) certamente maliciosos (vermelho) e (iii) suspeito. A última classificação deve refletir situações em que o sistema não consiga identificar com clareza se o dado é malicioso ou não (amarelo).

- Painel de Apoio: O módulo de painel de apoio tem como objetivo apresentar a um analista humano (oráculo) os resultados da conjugação de pareceres de legitimidade dos dados suspeitos. Diante dos dados apresentados, o oráculo deverá opinar sobre a situação do padrão apresentado e indicar como tal padrão deverá ser tratado pelo sistema. Caso o oráculo deseje, o painel de apoio poderá apresentar dados externos ao sistema que auxiliem no processo decisório.

- Complementação de Dados: Módulo responsável por obter dados externos ao sistema que serão apresentados pelo Painel de Apoio. Conforme seu nome sugere, o módulo de complementação dos dados deve realizar buscas inteligentes cujo resultado possa ser útil para a avaliação do oráculo.

- Consultas Gerenciais: Este módulo compreende a exibição de relatórios e consultas gerenciais sobre o comportamento do sistema, permitindo monitorar a qualidade dos resultados encontrados e assim facilitar um eventual ajuste dos parâmetros do sistema a ser realizado na fase de experimentação/configuração.

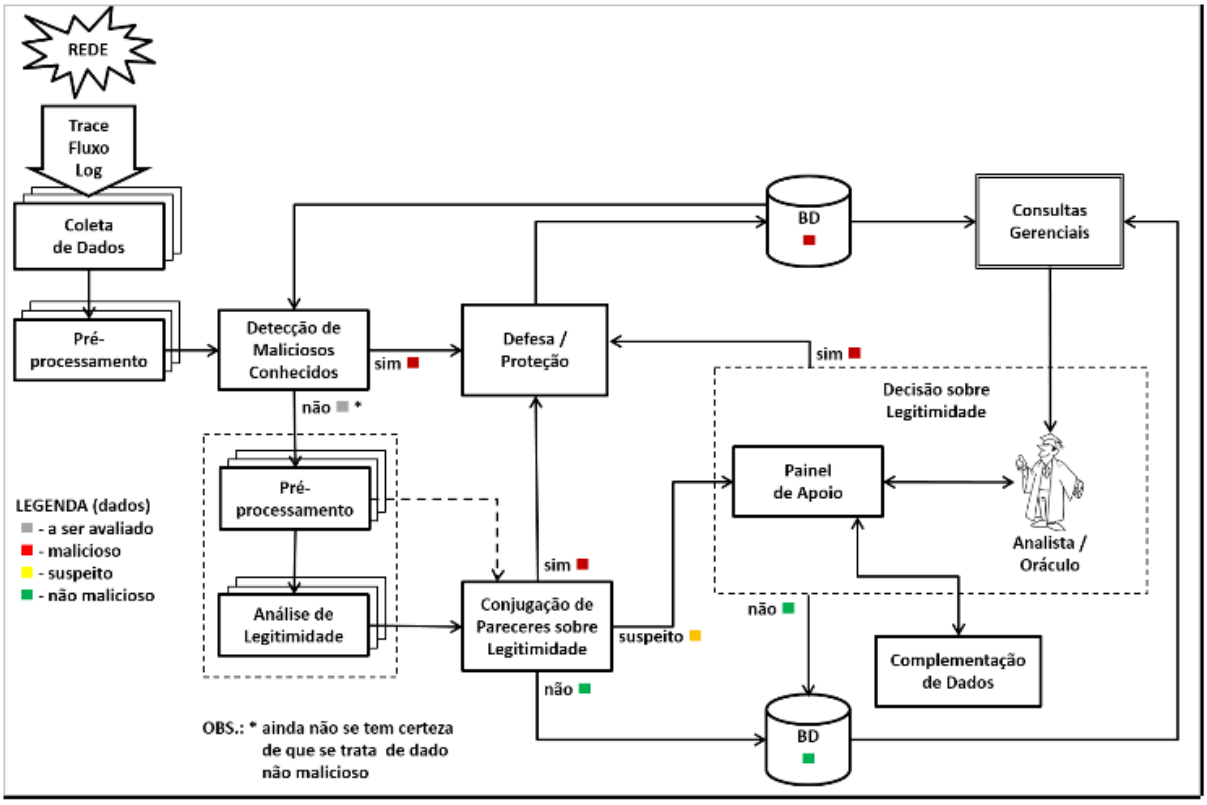

FIG. 1.1: Visão Macro-Funcional da Arquitetura da Fase de Produção/Operação do EB-CyberDef

1.2 OBJETIVO

Este Projeto de Final de Curso é parte do projeto EB-CyberDef, assim sendo, o presente trabalho abrange os módulos de pré-processamento, análise de legitimidade e conjugação de pareceres sobre legitimidade, tendo o objetivo de desenvolver e aplicar experimentalmente uma ferramenta que permita a criação de comitês de classificadores voltados à

identificação de padrões maliciosos no tráfego de redes de computadores, conjugando os resultados dos classificadores para obter uma classificação mais confiável.

1.3 METODOLOGIA

Para realizar o trabalho, foram implementados 5 algoritmos de classificação estudados para utilização. Foi utilizado um dataset já rotulado para treinar e testar o desempenho e para os testes foi utilizado o método de validação cruzada. Na fase seguinte, foi implementada uma ferramenta para criação de comitês de classificadores e utilizou-se a mesma metodologia para testes e validação.

1.4 ESTRUTURA DO TEXTO

No capítulo 2 será apresentada a fundamentação teórica acerca dos algoritmos de classificação a serem implementados e do conceito de comitês de classificação. No capítulo 3 seão apresentados o protótipo da ferramente desenvolvuda, os experimentos feitos e os resultados encontrados. No capítulo 4 serão apresentadas as considerações finais e no capítulo 6 apresentaremos as referências bibliográficas utilizadas na confecção do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 APRENDIZADO DE MÁQUINA

Durante muito tempo, a área de inteligência artificial (IA) foi vista como algo distante de ser alcançado, sendo temas de filmes de ficção científica, revistas de curiosidades e, no máximo, alvo de estudos simples e bastante teóricos, porém a aplicação prática desse conhecimento parecia ser algo inatingível a curto prazo.

O objetivo do uso de IA é construir sistemas computacionais que consigam resolver problemas por meio de raciocínio e uso de conhecimentos. Nas últimas décadas, as técnicas de IA foram sendo aperfeiçoadas e começaram a ser implantadas em diversas áreas como, por exemplo, na medicina, adquirindo conhecimento de especialistas em diversas áreas.

Um dos ramos da inteligência artificial procura utilizar conhecimentos previamente adquiridos (aprendidos) para induzir hipóteses. Esse ramo é conhecido como Aprendizado de Máquina (AM). O AM permite que máquinas armazenem informações sobre um conjunto de objetos conhecidos e possam tomar decisões sobre objetos desconhecidos. O aprendizado pode ser feito de duas formas: Supervisionado e Não Supervisionado. Nos supervisionados, os objetos possuem, além de informações sobre si próprio, informações sobre o que o os objetos representam no conjunto. Já nos não supervisionados, não se possui informações prévias sobre o que os objetos representam no conjunto [3].

Existem diversas formas para obter conclusões sobre um conjunto de dados, podendo ser utilizado regressão, agrupamento de dados ou classificação. Nas seções seguintes, serão abordados alguns dos classificadores mais comuns.

2.2 MODELOS PREDITIVOS E CLASSIFICAÇÃO

Dependendo do critério ou ponto de vista, os algoritmos de AM podem ser organizados de diversas formas. O formato de aprendizado, divide os algoritmos de AM em [3]:

- Preditivos;
- Descritivos

Em tarefas preditivas, o conjunto de treinamento deve possuir informações de entrada e saída. Então, acha-se uma função responsável por prever o rótulo ou valor de novos exemplos, analisando apenas suas informações de entrada. Dessa forma, segue-se

o paradigma de aprendizado supervisionado. Já nas descritivas, o objetivo é descrever os dados, visto que não há atributos de saída previamente conhecidos no conjunto (o paradigma nesse caso é o não supervisionado).

Os algoritmos denominados de classificadores são métodos preditivos. Eles rotulam (classificam) objetos por meio de seus atributos de entrada. A classificação ocorre em duas etapas: aprendizagem e reconhecimento. Na fase de aprendizagem, usa-se algum algoritmo de classificação (também conhecidos como algoritmos de indução) sobre um conjunto de dados utilizados para treinamento para criar-se um modelo de classificação. Na segunda fase, novos objetos são submetidos ao modelo criado, sendo então rotulados com alguma das classes segundo o modelo treinado [4]. As fases de uma classificaçao estão ilustradas na figura 2.1.

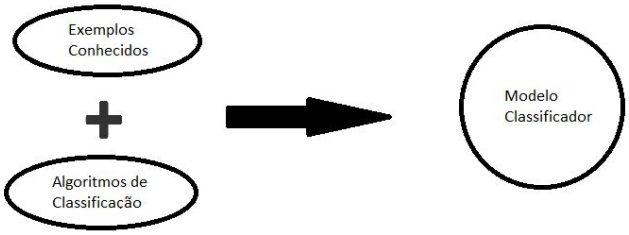

FIG. 2.1: Fases de uma classificação

Os algoritmos de classificação são submetidos a um conjunto de treinamento com o objetivo de gerar um modelo classificador. Porém é essencial verificar a eficiência do classificador criado. Esse processo se dá por meio da divisão dos exemplos conhecidos em dados e treino e dados de teste. Os dados de treinamento são responsáveis por estimar o classificador, enquanto os dados de teste que também pertencem ao conjunto de treinamento (possuem saída conhecida) são responsáveis por verificar a eficiência do modelo criado.

É intuitivo que, dado um conjunto de treinamento, existem diferentes combinações entre dados de teste e de treinamento e que disposições diferentes podem gerar resultados diferentes quanto à eficiência dos modelos. O método da validação cruzada permite realizar a divisão do conjunto de treinamento em n conjuntos, dos quais n-1 são dados de treinamento e o conjunto restante é composto pelos dados de teste. Os conjuntos são permutados em n rodadas distintas, de forma que todos os conjuntos passem uma vez pela situação de teste. Em cada rodada é obtida uma nota (avaliação) quanto à eficiência do classificador e ao final é feito uma média dos resultados para, só então, obter uma

avaliação sobre o modelo classificador gerado [5].

Na figura 2.2 observa-se um exemplo prático de validação cruzada. Nele, o suposto conjunto de dados foi dividido em 4. Em cada rodada foram usados 3 para treino e 1 para teste. Como mencionado anteriormente , a média das 4 rodadas fornece a eficiência do suposto classificador treinado por esse conjunto de treinamento.

Exemplo de validação curzada(4 conjuntos)				
original	rodada1	rodada2	rodada3	rodada4
A	A	B	C	D
B	B	C	D	A
C	C	D	A	B
D	D	A	B	C

dados de treino

dados de teste

FIG. 2.2: Rodadas de uma validação cruzada

Nas seções seguintes serão abordados alguns dos algoritmos de classificação (também conhecidos como indutores) mais utilizados atualmente e que serão utilizados na fase prática deste trabalho.

2.3 ÁRVORE DE DECISÃO

Analisar um espaço de soluções é uma das formas de se tomar decisões na área de aprendizado de máquina. Árvores de decisões se encaixam nesse formato, utilizando a metodologia "dividir para conquistar" com o intuito de dividir o problema maior em problemas mais simples e dessa forma alcançar uma solução otimizada para o problema de classificar e prever dados.

2.3.1 FORMATO DE UMA ÁRVORE DE DECISÃO

O modelo consiste em dividir o espaço de instâncias em subespaços de forma que a árvore possua dois possíveis tipos de nós:

- Nó folha: Nó que já não possui mais filhos, representando uma decisão final ou uma classificação.
- Nó de divisão: Possui um teste condicional baseado nos atributos do objeto a ser classificado.

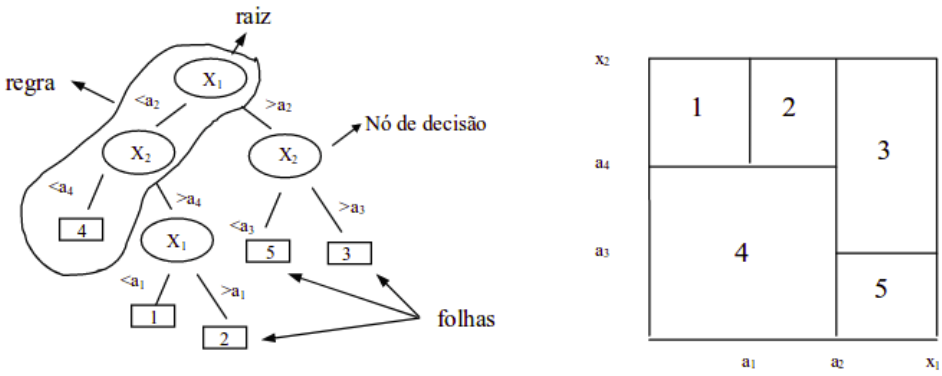

FIG. 2.3: Exemplo de Árvore de Decisão e suas regiões no espaço de objetos

Os atributos dos objetos representados pela árvore da figura 2.3 são x1 e x2. Eles dividem o espaço de objetos em regiões de decisão (classificação) numeradas de 1 a 5. Estas regiões devem abranger todo espaço de instâncias dos objetos de forma que, para qualquer instância de entrada, haja uma região de saída específica. As regiões devem ser excludentes, isto é, não deve haver interseção entre elas.

2.3.2 REGRAS PARA CRIAÇÃO DE PARTIÇÕES

Para escolha do atributo a ser utilizado na criação das partições leva-se em conta o ganho de informação que ele fornece, uma medida mais conhecida como "goodness of split" que indica a eficiência de um atributo em separar as classes [3]. Há diversos algoritmos cuja função é a formação de árvores de decisão otimizadas, sendo o de CART um dos mais conhecidos. Antes de abordar o algoritmo em si, é importante expor um breve conhecimento sobre dois conceitos importantes na área de ganho de informação.

2.3.2.1 ENTROPIA

O conceito de entropia é bem difundido na estatística. Ele mede a aleatoriedade de uma variável aleatória e pode ser utilizada no cálculo do ganho de informação, caracterizando o grau de impureza (falta de homogeneidade) dos dados. Dada uma variável S de domínio $\{s_1, s_2, s_3...s_n\}$, podemos medir sua entropia conforme indicado na eq. 2.1:

$$H(S) = \sum_i -p_i \times \log_2 p_i \tag{2.1}$$

Onde p_i é a probabilidade de ocorrência de s_i. O ganho de informação é baseado na diminuição de entropia ao se usar um atributo s_k na partição do conjunto de dados. O ganho também pode ser representado como a diferença entre a entropia do conjunto menos a soma ponderada das entropias das partições [3]. A construção da árvore está ligada, em geral, a 3 principais objetivos:

- Diminuir a Entropia (ganho de homogeneidade)
- Consistência de dados
- Reduzir o número de nós

2.3.2.2 ÍNDICE DE GINI

O índice de Gini também é uma medida para mensurar o grau de heterogeneidade, podendo ser utilizado para verificar a impureza de um nó. Segue os mesmos moldes da entropia, porém p_i passa a significar a frequência relativa de cada classe nos nós:

$$IG = 1 - \sum_i p_i^2 \tag{2.2}$$

Dessa forma, quando o índice tende a zero o nó tende a ser puro e quanto maior o índice maior será a heterogeneidade (impureza do nó).

Na prática usa-se igualmente os dois índices na construção de árvores de decisão, sendo poucas as diferenças entre o IG(índice de Gini) e a H (entropia). Uma delas reside no fato de que a IG tende a isolar a classe mais frequente em um ramo, enquanto a entropia produz árvores mais balanceadas [6].

2.3.3 CART

O algoritmo CART (*Classification* And *Regression Tree*) é o mais usado na criação de árvores de decisão [6]. Apesar de muito eficiente, seu funcionamento é bem simples:

primeiro ocorre a divisão do conjunto de dados de treinamento em dois subconjuntos usando um único atributo k e um limiar l_k (por exemplo, $l_k \leq t$). Mas qual k e l_k deve-se escolher? A resposta é simples, o par (k, l_k) que torne os subconjuntos o mais puro possível .

A ideia continua sendo usada recursivamente com os subconjuntos que vão sendo gerados. O algoritmo para quando atinge a profundidade máxima ou quando a melhor divisão a ser feita não reduz a impureza. A classe com maior probabilidade nos nós folha, serão eleitas para serem atreladas como resultado da entrada que levou àquele nó.

Vale ressaltar que CART é um algoritmo guloso e dessa forma atinge soluções razoavelmente boas, porém sem garantia alguma de que serão soluções ótimas [6].

2.4 ALGORITMO K-NN

O K-NN (*K- Nearest neighbourds*) é um dos métodos mais simples de aprendizado. É um método baseado em distâncias, isto é, objetos pertencentes às mesmas classes são semelhantes entre si [3].

Dessa forma, o K-NN classifica seus objetos de acordo com a proximidade dos exemplos do conjunto de treinamento. É considerado um algoritmo preguiçoso, pois não forma um modelo compacto com os dados de treinamento, apenas memoriza-os. As métricas que avaliam a eficiência do método são:

- Número de vizinhos considerados para decisão;
- Métrica usada para cálculo de distâncias.

2.4.1 ESCOLHA DE PARÂMETROS

2.4.1.1 ESCOLHA DO VALOR DE K

A escolha da quantidade K de vizinhos mais próximos a serem considerados é uma tarefa que requer bastante atenção, pois irá depender da disposição dos dados em questão. Se ocorrer a escolha de um K muito pequeno a classificação fica exposta a pontos de ruídos, porém se o K escolhido for muito grande pode haver uma inclusão de muitos elementos de outras classes, prejudicando na hora da votação. Vale ressaltar que o K deve ser preferencialmente ímpar, de modo a evitar empates [3].

Existem diversas medidas que visam otimizar a escolha da quantidade de vizinhos. Estimar K por validação cruzada [3] é uma alternativa a ser considerada, bem como associar pesos de contribuição aos vizinhos (inversamente proporcional à distância).

2.4.1.2 MÉTRICA PARA CÁLCULO DE DISTÂNCIAS

Uma vez definida a métrica para cálculo de distâncias, calcula-se a distância entres dois pontos do espaço considerado. A escolha dessa métrica depende de muitos fatores como, por exemplo, se os dados são quantitativos (escalas intervalais) ou qualitativos (escalas nominais ou ordinais). Usualmente opta-se por distância euclidiana. A escala a ser utilizada também é muito importante e, para minimizar os efeitos colaterais dessa escolha, os dados são usualmente normalizados.

2.4.2 FUNCIONAMENTO

Dados um conjunto de dados de treinamento, uma métrica para cálculo de distâncias e um valor de K apropriado para a situação podemos então classificar quaisquer objetos desconhecidos de acordo com os rótulos de classe dos seus vizinhos.

Para isso, basta calcular a distância entre o exemplo desconhecido e o conjunto de treinamento, determinando em seguida os k objetos mais próximos e utilizar o rótulo dos vizinhos mais próximos para por meio de uma votação majoritária determinar a classe a qual o elemento pertence.

A vantagem de se usar esse algoritmo reside na facilidade de implementação, trazendo bons resultados apesar da simplicidade da técnica. Entretanto, classificar os exemplos desconhecidos pode ser bastante custoso (calcular distâncias para cada elemento do conjunto de treinamento) e a presença de características irrelevantes pode atrapalhar o processo.

2.5 SUPPORT VECTOR MACHINE(SVM)

O método SVM consiste em encontrar fronteiras lineares que separem objetos de classes diferentes. Adequam-se muito bem em casos linearmente separáveis e, a partir dos conceitos obtidos nos casos lineares, formulam-se os casos de fronteiras não lineares. Está entre os mais populares métodos utilizados no aprendizado de máquina [6]. A seguir será dada uma abordagem geral sobre o uso e funcionamento da SVM para classificações binárias, sabendo que os conceitos podem ser estendidos para problemas multiclasse.

2.5.1 HIPERPLANOS

O entendimento do funcionamento de SVMs está atrelado a uma noção básica sobre o que são hiperplanos. Um hiperplano pode ser representado pela equação 2.3 no domínio

X [3]:

$$h(x) = w \bullet x + b \tag{2.3}$$

Onde $w \bullet x$ é o produto escalar entre os vetores w e x, de forma que $w \epsilon X$ é o vetor normal ao hiperplano e $b \epsilon \mathbb{R}$, tal que $\frac{b}{\|w\|}$ representa o módulo da distância desde a origem até o hiperplano.

2.5.2 SVMS LINEARES

Um conjunto linearmente separável é composto por objetos cujas classes podem ser separadas por hiperplanos [7]. SVMs lineares buscam hiperplanos ótimos para dividir essas classes em questão, de forma que as distâncias entre os objetos das classes e o hiperplano seja a maior possível.

A definição de hiperplano nos permite dividir o espaço em duas classes de forma que a equação w •x+b > 0 represente uma das classes e w •x+b < 0 represente a outra classe (ilustrado na figura 2.4). Fica evidente que, em geral, diversos hiperplanos podem

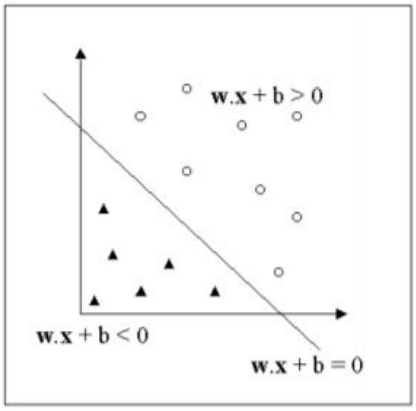

FIG. 2.4: Divisão do espaço por meio de um hiperplano

separar um conjunto linearmente separável, mas nem todos os hiperplanos dão um retorno satisfatório no âmbito de classificadores. Para contornar esse problema (determinação de um hiperplano ótimo), um aspecto muito importante está na determinação da margem de distância entre os objetos e o hiperplano, de modo que o risco de classificação diminua, ou seja, que novos objetos fora do conjunto de treinamento possam ser classificados com uma proporção de acertos razoável [7]. Assim, podemos dividir as SVMs lineares em duas classes: SVMs de margem rígidas e de margem suaves.

2.5.2.1 SVMS LINEARES DE MARGEM RÍGIDA

Define-se como hiperplano canônico, aquele em que w e b são escolhidos de forma que os objetos mais próximos do hiperplano satisfaçam a equação [3]:

$$|w \bullet x + b| = 1 \tag{2.4}$$

A restrição acima divide as classes em duas regiões $|w \bullet x + b| \geq 1$ e $|w \bullet x + b| \leq -1$ (margens).

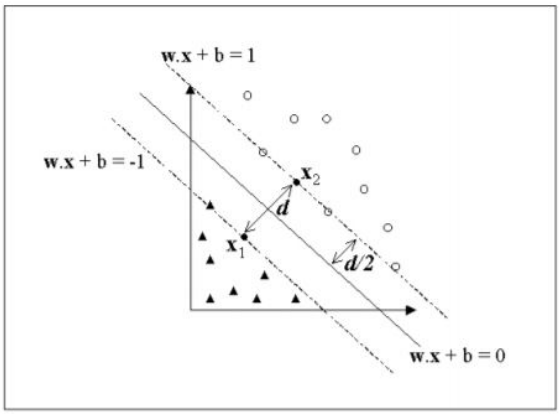

FIG. 2.5: Representação de hiperplano com margens rígidas

Os dados de treinamento são então separados de forma que não há objetos entre as margens e o hiperplano. Essa condição faz com que a SVM seja caracterizada como de margem rígida (vide figura 2.5).

2.5.2.2 SVMS LINEARES DE MARGEM SUAVE

A condição imposta para separar as classes por meio de margens rígidas requer classes bem definidas e que não haja ruídos nos dados de treinamento, o que não maioria dos casos não acontece. Nesse intuito, introduz-se o conceito de variáveis de relaxamento.

Variáveis de relaxamento (ς) tornam as restrições necessárias para um hiperplano ótimo mais suaves, permitindo alguns erros de classificação. Essas variáveis medem onde se encontram os exemplos em relação ao hiperplano [7].

De acordo com a figura 2.6 , as variáveis ς podem assumir valores 0 (está fora da região entre as margens), menor que 1 (mede a distância até o hiperplano), maior que 1 (dado classificado de forma errada). Deve-se admitir um valor mínimo de ς para todo o conjunto de treinamento, pois a quantidade de erros deve ser minimizada. O uso dessas variáveis gera o que chamamos de "suavização de margens" ou margens suaves [7].

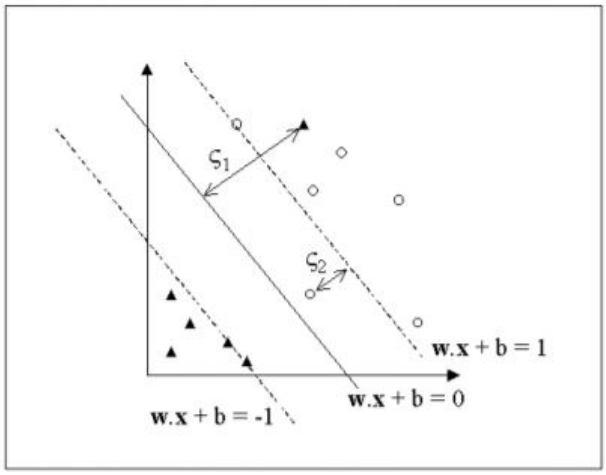

FIG. 2.6: Representação de hiperplano com margens suaves

Fica claro desde o propósito de uso das margens suaves que estas se adequam melhor aos casos reais em que dificilmente há uma uniformidade de dados e ausência de ruídos. Porém em muitos casos a suavização não é capaz de minimizar o percentual de erros de classificação no conjunto de treinamento. Esses casos necessitam de uma fronteira não linear e serão abordados mais à frente.

2.5.3 SVMS NÃO LINEARES

Em muitas situações não é possível dividir satisfatoriamente o conjunto de treinamento por meio de um hiperplano (conjunto não é linearmente separável). Nesses casos, a utilização de fronteiras curvas para divisão do conjunto de dados é o mais adequado. Para contornar essa situação, são utilizadas funções reais φ que mapeiam o conjunto de dados para um novo espaço.

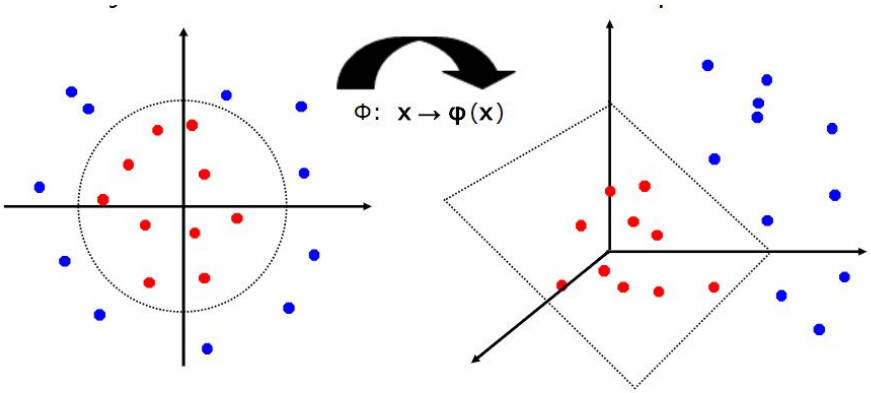

FIG. 2.7: Exemplo de mapeamento em novo espaço

O objetivo da transformação ϕ é fazer com que no novo espaço, os dados possam ser separados por um hiperplano e então, todas as propriedades utilizadas para SVMs lineares poderão ser aplicadas [7]. A equação do plano passará a ser:

$$w \bullet \varphi(x) + b = 0 \tag{2.5}$$

Um exemplo prático de demonstração do uso dessa técnica está representado na figura 2.7. A fronteira no caso é uma circunferência e pode-se utilizar sua equação clássica, $x^2 + y^2 = r^2$, para gerar o seguinte artifício: dado um ponto $p = (x, y)$, tomam-se $\varphi(p) = (x^2, xy, y^2)$, $w = (1, 0, 1)$, $b = r^2$. Dessa forma, é trivial ver que no novo espaço formado (tridimensional) será possível, por meio de um hiperplano, separar os pontos interiores e exteriores à circunferência, voltando ao caso linear (como ilustrado na figura 2.7).

2.6 NAIVE BAYES

O algorítmo Naive Bayers é uma técnica de classificação probabilística baseada no teorema de Bayes, que pode ser expresso matemáticamente com a seguinte equação:

$$P(A|B) = \frac{P(A|B) * P(A)}{P(B)}$$

A principal característica do algoritmo, o que justifica o nome *naive*, assume que todas as *features* são independentes, visto que para 2 eventos A e B independentes temos que: $P(A|B) = P(A)$, dessa forma, seja uma classe y dependente das *features* $\{x_1, x_2, ..., x_n\}$, temos:

$$P(y|x_1, x_2, ..., x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n}$$

ou seja,

$$P(y|x_1, x_2, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

assim podemos escolher a classe cuja probabilidade dados os valores das *features* é maior:

$$\hat{y} = arg \max_{y} P(y) \prod_{i=1}^{n} P(x_i|y)$$

2.7 PERCEPTRON MULTICAMADAS

O Perceptron Multicamadas é uma rede neural. Seu funcionamento é baseado no modelo computacional de neurônio ilustrado na figura 2.8

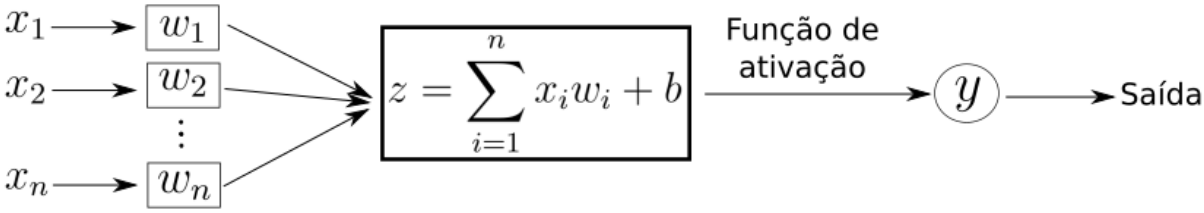

FIG. 2.8: Modelo Computacional de Neurônio

O neurônio recebe os parâmetros de entrada, atribui um peso a cada parâmetro e computa os valores das entradas multiplicadas pelos seus pesos somados a uma constante, como pode ser observado na figura 2.8. Dessa forma, é calculado o valor z de saída da forma:

$$z = \sum_{i=1}^{n} x_i w_i + b$$

Em que o valor b corresponde ao *bias*, que provê um grau de liberdade a mais. O valor z em seguida passa por uma função de ativação, que deve ser não linear, responsável por limitar o valor da saída.

Esses neurônios são, então, combinados em uma estrutura de camadas, de forma que as características do objeto que desejamos rotular sejam as entradas da primeira camada, as saídas da primeira camada e das camadas intermediárias sejam as entradas para outras camadas intermediárias e a saída da última camada corresponda ao rótulo, definindo, então, o modelo de classificação ilustrado na figura 2.9.

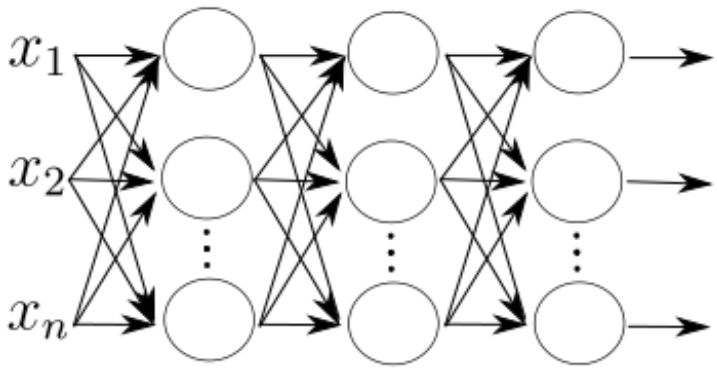

FIG. 2.9: Funcionamento do Perceptron Multicamadas

2.8 COMITÊS DE CLASSIFICADORES

Para se entender melhor o conceito e o uso de comitês de classificadores, deve-se levar em consideração que o erro de um algoritmo é composto de viés (tendência de um algoritmo favorecer determinada condição dos dados) e variância. Dessa forma, quanto mais simples o modelo, menor será sua variância e maior será seu viés. Em contrapartida, quanto maior a complexidade de um algoritmo de classificação, menor será seu viés e maior será sua variância. A formação de comitês visa diminuir esses dois efeitos negativos, procurando analisar diversos classificadores para a tomada de decisão. É importante ressaltar que só faz sentido a utilização de comitês caso os classificadores tenham um nível substancial de desacordo [3].

Segundo [3], os métodos mais usados atualmente para realizar a combinação dos classificadores são os métodos de votação. Neles, o voto de cada classificador deve ser ponderado com a probabilidade de aquela hipótese estar certa. Uma das variações desses métodos é a votação uniforme, onde todos os classificadores contribuem igualmente com seus votos. Uma boa melhoria desses métodos, é alcançada quando se sabe a probabilidade de cada classe em vez de apenas o rótulo final de classificação, pois as probabilidades alcançadas nos diversos classificadores podem ser combinadas de forma a otimizar a previsão final (métodos de seriação).

Os comitês podem ainda ser divididos em homogêneos (classificadores do mesmo tipo) e heterogêneos (classificadores de tipos diferentes). Os exemplos mais conhecidos segundo essa divisão são: Bagging e boosting (homogêneos), stacking (heterogêneo). No escopo deste trabalho abordar-se-á apenas o método stacking [8].

2.8.1 *STACKING*

Também conhecido como Generalização em pilha, esse método consiste em usar classificadores em níveis mais altos que sejam responsáveis por aprender com os erros dos classificadores de níveis mais baixos, ou seja, aprender como os classificadores anteriores cometeram erros.

A fase de treinamento desse método pode ser facilmente explicada em seu formato mais simples (vide figura 2.10): duas camadas, a primeira (nível 0) que recebe dados originais e, a segunda (nível 1) que recebe os dados oriundos do nível 0. Ao receber os objetos de exemplo, o nível 0 promove o treinamento dos seus classificadores por meio de validação cruzada, gerando como saída um vetor informando a probabilidade predita por cada classificador de aquele exemplo ser de cada classe disponível, bem como o resultado

real do dado de treinamento. Ou seja, dado um exemplo de objeto de entrada no nível 0, o novo exemplo de entrada no nível 1 terá como atributos as probabilidades estimadas pelos classificadores para cada classe disponível e a classe já conhecido do objeto. Só então o árbitro realiza a previsão final. É possível a utilização do prório dado de entrada do nível 0 para que além de aprender com os resultados dos classificadores anteriores, o Nível 1 também aprenda com os dados de entrada, reforçando os dados.

Uma vez treinado, a predição continua no mesmo formato. O objeto passa pelos dois níveis de comitê e de forma transparente ocorre a rotulação para o exemplo em questão.

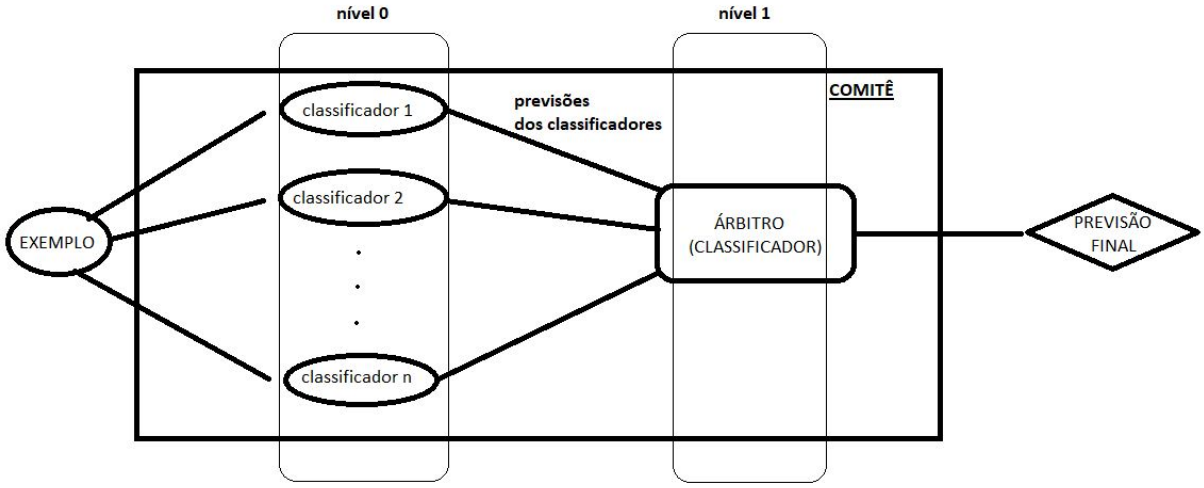

FIG. 2.10: Comitê de classificadores em dois níveis

3 PROTÓTIPO, EXPERIMENTOS E RESULTADOS

3.1 PROTÓTIPO DESENVOLVIDO

O protótipo desenvolvido considera como entrada o *dataset* rotulado e permite a seleção dos algoritmos de classificação a ser utilizados, bem como selecionar o tipo de comitê a ser utilizado e selecionar o algoritmo a ser utilizado pelo mesmo, quando se tratar de um árbitro classificador. Além disso, a ferramenta permite a seleção da quantidade de conjuntos a ser utilizada na validação cruzada.

Para melhor adaptação às necessidades do usuário, o protótipo permite tanto a utilização de interface gráfica quanto o uso direto com comandos no terminal, como ilustrado nas figuras 3.1 e 3.2.

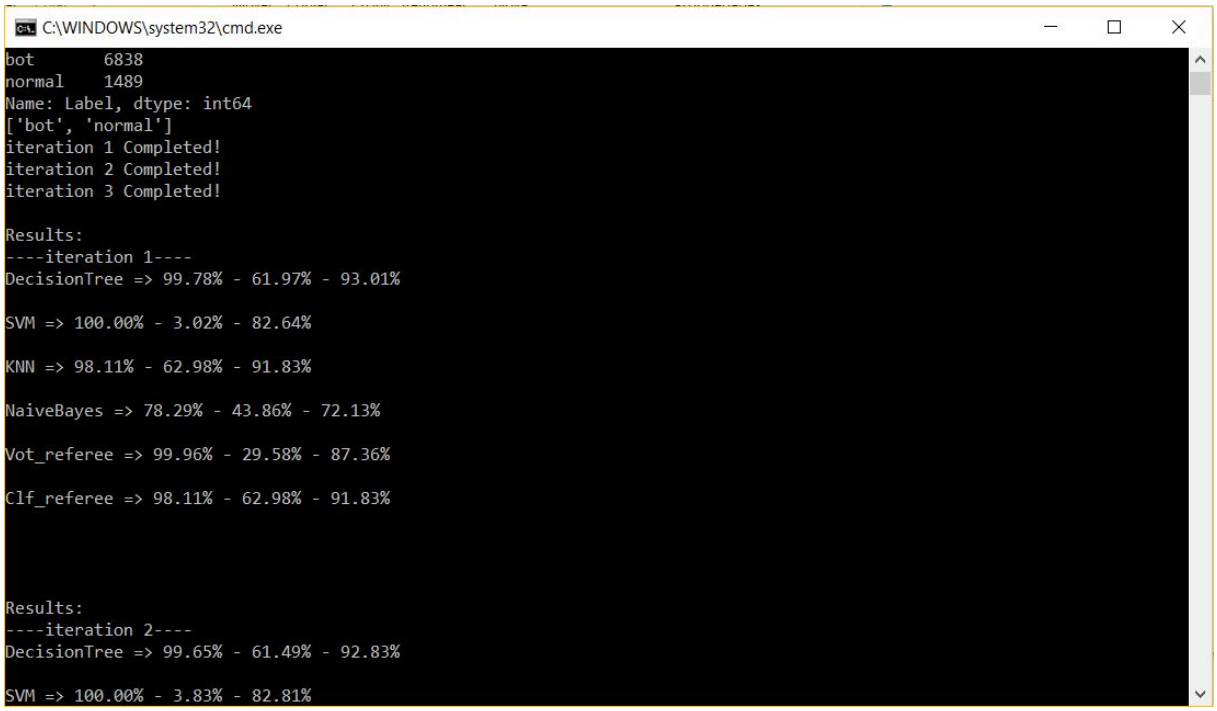

FIG. 3.1: Programa rodando em *background* por meio do terminal

A biblioteca permite tanto o desenvolvimento gráfico da interface como a manipulação dos seus eventos. Como mostrado na figura 3.2(a), foi permitindo ao usuário escolher três parâmetros dinamicamente: A lista de classificadores do comitê, o modelo do juiz classificador, o número de conjuntos para validação cruzada.

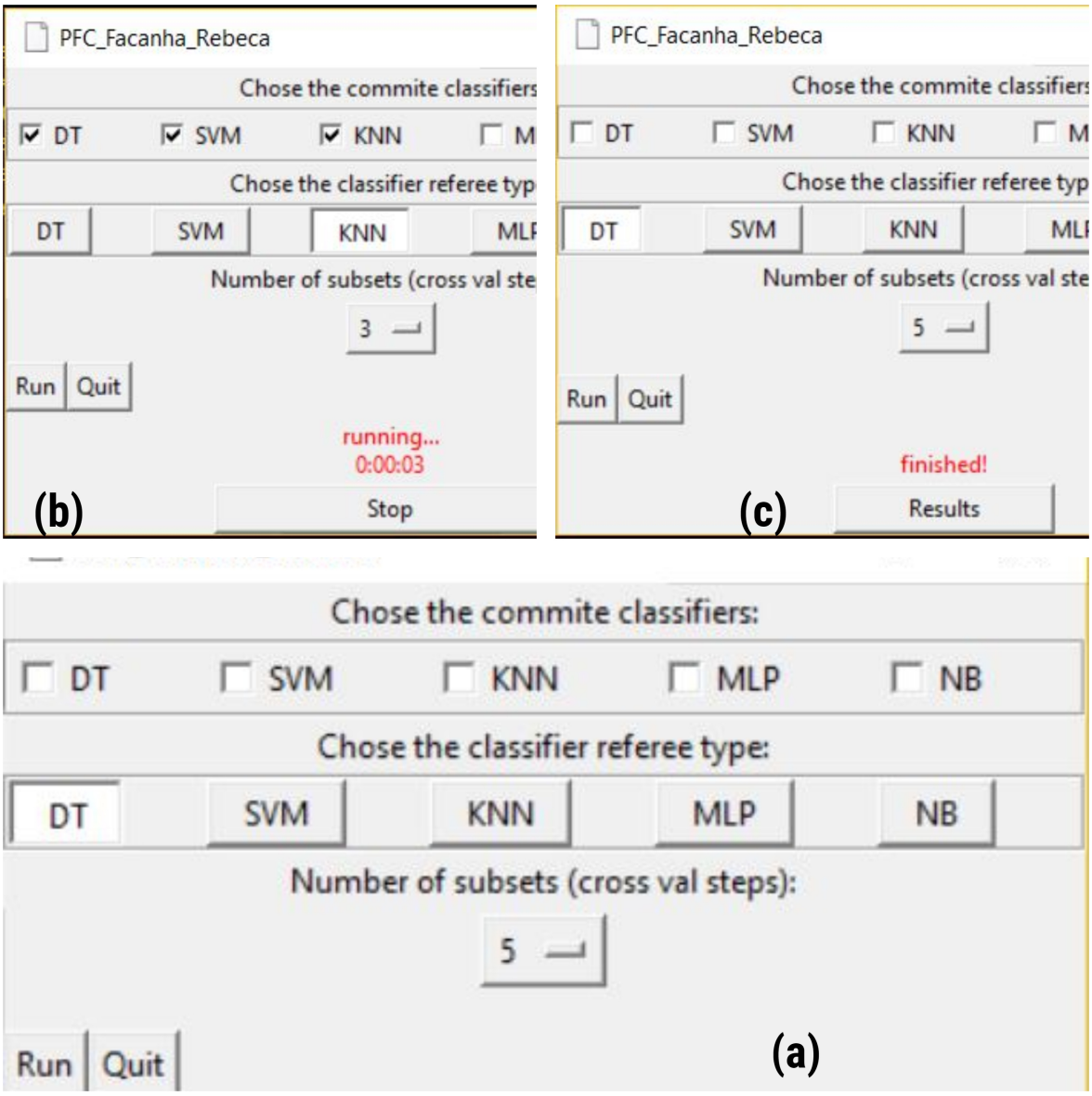

FIG. 3.2: Interface desenvolvida para manipulação do comitê

Após a escolha dos parâmetros, dá-se início ao processo por meio do botão "RUN". Conforme a figura 3.2(b) o tempo decorrido desde o início é exibido para o usuário que a qualquer momento pode encerrar o procedimento forçadamente. Após um término bem-sucedido (figura 3.2(b)), uma mensagem de sucesso é enviada ao usuário e a interface permite o acesso à pasta com os resultados do procedimento.

3.2 BASE DE DADOS UTILIZADA

Para a realização dos testes, foi utilizada a base de dados CTU-13, capturada na universidade CTU (República Tcheca) em 2011[1].

Esta base consiste em um conjunto de 13 cenários construídos da seguinte forma: Malwares foram executados, gerando fluxos contaminados em situações de *botnet*. Os cenários foram transformados para um conjunto de arquivos Netflow e, em seguida foram concatenados em um único arquivo no formato csv (usado como ponto de partida para o projeto em questão). Os fluxos estão classificados em 3 rótulos: *Botnet*, Normal e Background. Foi feito um processo de rotulação de forma que inicialmente todo o tráfego fosse classificado como background. Em seguida, computadores controlados e conhecidos foram classificados como normais (computadores laboratoriais, proxies, switches, etc.). Por último, foi assinalado o rótulo *botnet* aos IPs infectados. Partindo desse princípio, dividiu-se apenas os fluxos em normais e *botnet*, pois os background foram assim assinalados por não se ter informação nenhuma sobre a rotulação das instâncias, sendo dados não úteis. A divisão inicial dos rótulos está representada na tabela 3.1 [9].

TAB. 3.1: Distribuição inicial dos rótulos para cada cenário do *dataset*

ID	Background	Botnet	Normal	Total
1	10124854(95,41%)	94972(0,89%)	392433(3,70%)	10612259
2	6071419(95,60%)	54433(0,86%)	225336(3,55%)	6351188
3	14381899(94,60%)	75891(0,50%)	744270(4,90%)	15202060
4	3895469(91,92%)	6466(0,15%)	336103(7,93%)	4238038
5	416267(91,38%)	2129(0,47%)	37144(8,15%)	455540
6	2031967(94,13%)	4927(0,23%)	121854(5,64%)	2158748
7	425611(93,71%)	293(0,06%)	28270(6,22%)	454174
8	11451205(95,47%)	12063(0,10%)	530666(4,42%)	11993934
9	6881228(90,22%)	383215(5,02%)	362594(4,75%)	7627037
10	4535493(87,54%)	323441(6,24%)	321917(6,21%)	5180851
11	119933(29,34%)	277892(67,97%)	11010(2,69%)	408835
12	119933(29,34%)	277892(67,97%)	11010(2,69%)	408835
13	1218140(93,77%)	21760(1,68%)	59190(4,56%)	1299090

3.3 DESENVOLVIMENTO DOS EXPERIMENTOS

Para fins de treinamento, a classe background não foi utilizada, visto que essa classe representa os computadores não rotulados. Portanto, o fluxo de trabalho inicia com o

[1]Essa base de dados se encontra disponível em <https://mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html>.

arquivo concantenado contendo apenas duas classes: botnet e normal.

A etapa de pré-processamento de dados inicia-se com a exclusão de alguns campos desnecessários à análise, permanecendo os seguintes campos: Duration, Direction, Protocol , SToS, DToS, Total Packets, Total Bytes e o próprio Label de cada fluxo. Após o pré-prcessamento, foram corrigidos alguns problemas potenciais num processo de refinamento dos dados, ocorrendo um simples tratamento para consertar ou excluir amostras com informações mal formatadas.

Em seguida foi utilizada uma técnica bastante conhecida, "One hot encoder", que consiste em transformar campos discretos (categóricos) em um formato mais eficiente para a utilização em regressões ou classificações. A necessidade dessa transformação surge quando um campo das amostras pode assumir várias categorias que não podem ser comparadas quantitativamente, ou seja, ao se transformar as categorias em números pode-se cair no erro de compará-las numericamente. Um exemplo da aplicação dessa técnica no presente trabalho é o campo protocolo, que está dividido em categorias como Tcp, Udp, Arp, etc. Não há sentido em falar que Tcp corresponde ao valor 0, Arp ao valor 1 e que Udp corresponde ao valor 2 uma vez que não existe relação Tcp $<$ Arp $<$ Udp. Dessa forma, foram criadas colunas booleanas para os possíveis valores, de forma a se fazer uma análise independente entre as diversas categorias. Além do parâmetro Protocol, a técnica foi aplicada no parâmetro Direction.

Após o tratamento descrito, um gerador de conjuntos (representado em [0] na figura 3.3) age de forma a separar as amostras em k conjuntos aleatórios para realização da validação cruzada. Em seguida, as k separações (k-1 treinos e 1 teste) são alternadas e repetem o processo de treinamento representado em [1]. Em cada um dos k processos, os dados de teste são utilizados tanto para fornecer a eficiência dos classificadores individualmente quanto para utilizar-se as previsões como dado de entrada para os árbitros e posterior fornecimento da acurácia dos mesmos.

Os 5 classificadores ilustrados na figura 3.3 foram implementados com o auxílio da biblioteca scikit-Learn (Python). O árbitro como classificador possui suporte para assumir qualquer desses modelos.

Os 5 classificadores ilustrados na figura 3.3 foram implementados com o auxílio da biblioteca scikit-Learn (Python). O árbitro como classificador possui suporte para assumir qualquer desses modelos.

A princípio foram usados os parâmetros default da biblioteca em python para os classificadores. Os classificadores KNN e Decision Tree reagiram bem ao *dataset* e puderam se manter dessa forma, porém os outros obtiveram resultados discrepantes e passaram a

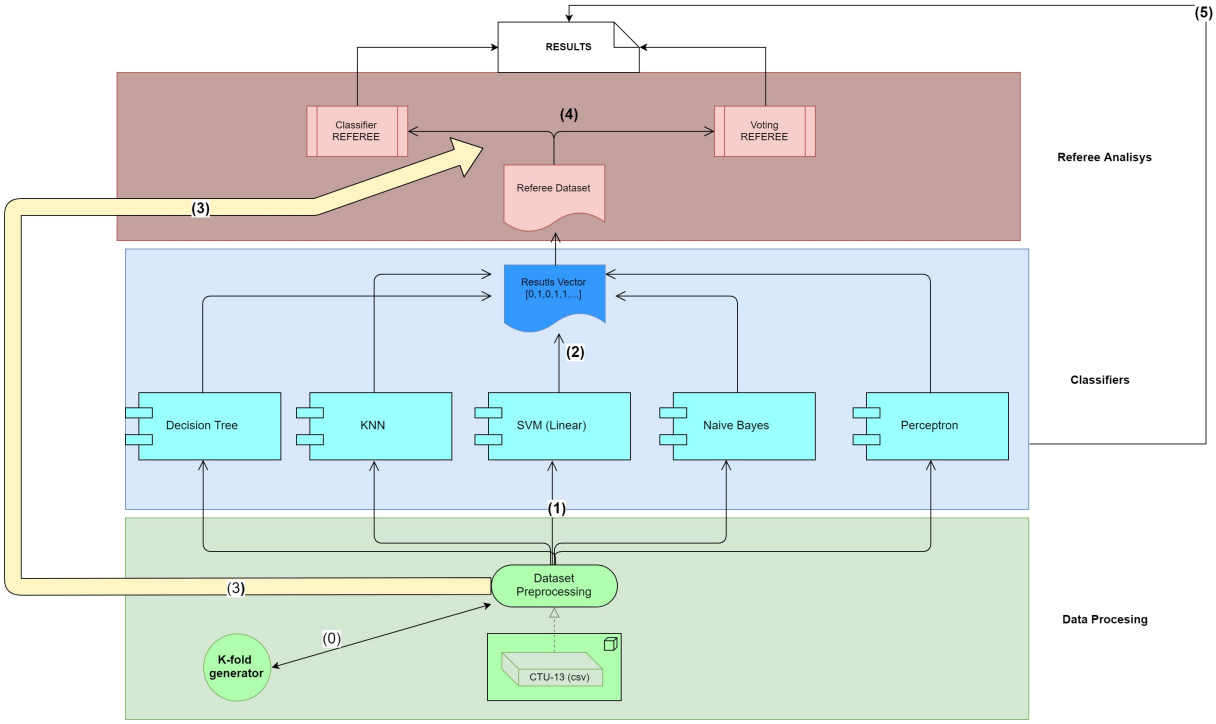

FIG. 3.3: Fluxo dos processos de classificação

ser analisados separadamente.

Além disso, inicialmente os resultados foram analisados tomando a média de acertos dos classificadores sobre todo o conjunto. Durante o processo de análise percebeu-se que alguns classificadores estavam com *bias* muito alto, recaindo numa espécie de *underfitting* (situação oposta ao *overfitting*, gerando um modelo muito geral) devida a pequena proporção de elementos da classe *botnet* no *dataset*. Então passou-se usar a métrica *recall*, que analisa a acurácia geral aliada a acurácia para as duas classes separadamente, ou seja, o quanto de cada classe o modelo acerta na fase de teste.

Com relação ao NB, há três tipos de implementação possíveis para a biblioteca usada: O Bernoulli, Gaussiano, Multinomial. Os três tipos foram testados com seus parâmetros *default* e o tipo vencedor: Naive Bayes Gaussiano, foi o que apresentou o melhor resultado.

O modelo utilizado para implementar o SVM inicialmente estava obtendo índices de acertos baixíssimos(abaixo de 1% de acerto) para a classe normal. Estes valores foram melhorados de acordo com mudanças nos parâmetros de criação, obtendo um modelo otimizado em relação ao anterior. A busca dos melhores parâmetros foi feita de forma experimental através de testes com diversas combinações entre os parâmetros C, kernel,

gamma [2] [3]. Primeiramente foi feita uma busca pela melhor média total de acertos, em seguida foram feitos novos testes visando a obtenção de resultados melhores para a classe bot.

Os parâmetros para o modelo Perceptron também foram escolhidos empiricamente. Devido a alta quantidade de possíveis combinações de parâmetros nesse modelo, inicialmente foi feita uma pré filtragem dos valores para o tamanho da camada intermediária, com os outros parâmetros com valor *default*. Após essa pré filtragem foi feito o mesmo processo do SVM, testando-se as combinações[4] dos valores para os parâmetros hidden_layer_sizes, solver e alpha [5]. Os parâmetros que melhor se adequaram para esse modelo foram: solver='adam', hidden_layer_sizes=(8), alpha = 100 e random_state=1.

3.4 RESULTADOS

O resultado obtido após o melhor ajuste dos parâmetros pode ser observado na tabela 3.2.

TAB. 3.2: Avaliação individual dos classificadores.

Classificador	Acurácia	Botnet	Normal
Decision Tree	93.89%	96.24%	83.07%
SVM	85.11%	98.79%	22.30%
KNN	91.17%	95.96%	69.17%
MLP	69.53%	79.97%	21.55%
Naive Bayes	89.23%	98.96%	44.52%

3.4.1 ÁRBITROS

Após obtermos resultados satisfatórios para os classificadores individuais, passou-se a analisar a atuação dos mesmos juntamente com os árbitros. O árbitro de votos é de simples implementação, pegando-se apenas a maioria dos votos. Já o outro árbitro inicialmente foi testado tendo como dados de treinamento as saídas dos demais classificadores além

[2] Os parâmetros C, kernel e gamma são explicados em <http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

[3] Foram testadas todas as combinações dos parâmetros C, gamma e kernel nos conjuntos {0.1, 1, 10, 100, 1000}, { 1, 0.1, 0.01, 0.001, 0.0001}, {'rbf', 'linear'}

[4] Foram testadas todas as combinações dos parâmetros solver, alpha nos conjuntos {'lbfgs', 'sgd', 'adam'}, {1000, 100, 10, 1, 0.1, 0.01, 0.001} e tamanhos da camada intermediária de 1 a 100.

[5] Os parâmetros hidden_layer_sizes, solver e alpha são explicados em <http://scikit-learn.org/stable/modules/generated /sklearn.neural_network.MLPClassifier.html>

dos dados da base. A tabela 3.3 mostra os resultados obtidos para o árbitro de votação e para o árbitro classificador ao se utilizar cada um dos 5 algoritmos.

TAB. 3.3: Avaliação dos árbitros antes e após a junção com dados de entrada.

árbitro	Acurácia	Botnet	Normal
Votos	91.80%	99.75%	55.32%
Classificador (Decision Tree)	91.46%	96.74%	57.23%
Classificador (SVM)	86.75%	98.19%	34.24%
Classificador (KNN)	91.52%	95.96%	69.17%
Classificador (MLP)	69.53%	79.99%	20.20%
Classificador (Naive Bayes)	89.23%	98.96%	44.52%

Pode observar-se na tabela 3.2 que a maior parte dos resultados não se distancia muito dos valores encontrados ao utilizar no algoritmo quando testado individualmente. Portanto, pode-se sugerir que no treinamento os dados do *dataset* acabaram adquirindo um peso maior na avaliação.

4 CONSIDERAÇÕES FINAIS

A ferramenta desenvolvida nesse trabalho permite a utilização de forma simples de comitês de classificadores. Além disso, é possível adicionar ao código novos classificadores ou alterar no mesmo os parâmetros dos algoritmos implementados de forma que o mesmo se adeque à situação desejada. Além disso a ferramenta se adequa às necessidades do usuário, podendo ser utilizada tanto pelo terminal quanto pela interface gráfica, podendo também ser utilizado por outros programas pelo acesso a comandos do sistema.

Os resultados dos testes mostraram pouca diferença entre os árbitros classificadores e os classificadores de mesmo algorítmo, o que sugere que os dados do dataset assumiram um peso maior que o resultado dos outros classificadores durante o treinamento.

Fica como principal sugestão para trabalhos futuros de pesquisa, o estudo de adequação dos parâmetros de forma a resolver da melhor forma o problema de classificação de tráfego malicioso na rede, além do desenvolvimento de algoritmos mais complexos que possam ser incluídos na ferramenta desenvolvida para obtenção de melhores resultados na classificação.

Além de estudos mais aprofundados de como os algoritmos podem funcionar, seriam interessantes trabalhos futuros com o desenvolvimento de ferramentas integradas à solução aqui apresentada que possibilitem a manipulação e filtragem dos dados juntamente com o a seleção dos algoritmos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] H. B. Sene, "Aumento de ameaças cibernéticas aquece setor de tecnologia de inteligência e segurança cibernética," 2017, 2018-09-21. [Online]. Available: https://www.symmetry.com.br/2017/05/02/aumento-de-ameacas-ciberneticas-aquece-setor-de-tecnologia-de-inteligencia-e-seguranca-cibernetica/

[2] J. Couto, "Inteligência artificial na segurança da informação: o que realmente considerar para sua empresa," 2018, 2018-09-20. [Online]. Available: http://tiinside.com.br/tiinside/seguranca/artigos-seguranca/09/01/2018/inteligencia-artificial-na-seguranca-da-informacao-o-que-realmente-considerar-para-sua-empresa/

[3] J. G. Katti Faceli, Ana Carolina Lorena and A. C. P. L. F. de Carvalho, *Inteligência Artificial. Uma Abordagem de Aprendizado de Máquina.* Rio de Janeiro: Campus, 2011.

[4] L. E. A. dos Santos Santana, "Otimização em comitês de classificadores: Uma abordage baseada em filtro para seleção de subconjuntos," Ph.D. dissertation, Universidade Federal do Rio Grande do Norte, Natal-Rn, 2012, [Online; accessed 06-Maio-2018].

[5] A. C. B. K. Luciano Drosda M. dos Santos, Renê Mikami, "Procedimentos de validação cruzada em mineração de dados para ambiente de computação paralela," in *Escola Regional de Alto Rendimento RS*, ser. Sessão de Iniciaciação Científica, no. 1, 2009, pp. 233–236.

[6] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, 2017.

[7] A. C. P. L. F. d. C. Ana Carolina Lorena, "Introdução às máquinas de vetores suporte," Universidade de Sao Paulo, São Paulo-Sp, Tech. Rep., 2003, [Online; accessed 07-Maio-2018]. [Online]. Available: http://conteudo.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_RT_192.pdf

[8] D. WOLPERT, *"Stacked Generalization"*, 1992, pp. 241–260.

[9] Z. A. Garcia Sebastian, "An empirical comparison of botnet detection methods," *Elsevier Computers & Security Journal*, vol. 7, no. 5, pp. 878–903, 2013.