Henrique Matheus Silva Arouca

UMA PROPOSTA DE DETECÇÃO DE ANOMALIAS NA REDE COM USO DE ESTATÍSTICA E INTELIGÊNCIA COMPUTACIONAL

Formiga - MG

2018

Henrique Matheus Silva Arouca

UMA PROPOSTA DE DETECÇÃO DE ANOMALIAS NA REDE COM USO DE ESTATÍSTICA E INTELIGÊNCIA COMPUTACIONAL

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Campus Formiga

Ciência da Computação

Orientador: Prof.º Diego Mello da Silva

Coorientador: Prof.º Everthon Valadão dos Santos

Formiga - MG

2018

Arouca, Henrique Matheus Silva.
004 Uma proposta de detecção de anomalias na rede com uso de Estatística e inteligência computacional / Henrique Matheus Silva Arouca. -- Formiga : IFMG, 2018.
90p. : il.

Orientador: Prof. MSc. Diego Mello da Silva
Coorientador: Prof. MSc. Everthon Valadão dos Santos
Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Chebyshev. 2. Random Forest. 3. Detecção de Anomalias. 4. Comitê de Votações. 5. Inteligência computacional. I. Título.

CDD 004

Ficha catalográfica elaborada pela Bibliotecária MSc. Naliana Dias Leandro CRB6-1347

HENRIQUE MATHEUS SILVA AROUCA

UMA PROPOSTA DE DETECÇÃO DE ANOMALIAS NA REDE COM USO DE ESTATÍSTICA E INTELIGÊNCIA COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais-Campus Formiga, como Requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Aprovado em: 21 de Novembro de 2018.

BANCA EXAMINADORA

Prof.º Diego Mello da Silva

Prof.º Everthon Valadão dos Santos

Prof.º Raí Caetano de Jesus

Prof.º Walace de Almeida Rodrigues

Agradecimentos

Agradeço primeiramente a Deus, guia desta presente etapa da minha vida que se encerra com este trabalho, Ele que me proporcionou, saúde, discernimento e principalmente a dádiva da paciência, essa última muito necessária em tempos difíceis.

Agradeço também aos meus pais que com toda sua humildade e sacrifício, fizeram de tudo para que eu pudesse me tornar alguém com estudo, e poder ter uma vida melhor e com mais seguraça. Sem eles eu nunca seria capaz de atingir esses objetivos, pois sempre que uma dificuldade era encontrada meu porto seguro estava garantido juntos a eles.

Agradeço aos meus amigos de sala e de repúblicas que sempre estiveram junto a mim, enfrentando e se divertindo em cima de qualquer dificuldade, está irreverência e maneira de enxergar as coisas, sempre procurando a ironia em tudo, foi de suma importância para suavização dos problemas. Isso também vale para o agradecimento a minha namorada que sempre esteve ao meu lado dando apoio e tranquiladade.

Agradeço ao meu coorientador Everthon Valadão, um ótimo professor muito importante na confecção deste trabalho e para minha formação acadêmica. Agradeço em especial ao professor Diego Mello que além de um ótimo professor, guia e orientador, se tornou um amigo próximo, sempre com paciência, compreensão, durante todo o perído do curso de Ciência da Computação, sempre com profissionalismo e amor ao trabalho desenvolvido, se tornando não só um herói, mas um exemplo de vida, para mim e para muitos em um curso tão complicado e árduo.

"Cada um pensa naquilo que lhe faz falta."
Roberto Bolaños

Resumo

O presente trabalho apresenta uma proposta para detecção de suspeitas anomalias em redes com o uso de estatística e inteligência computacional, mais especificamente com uso do conceito de desigualdade de Chebyshev e do algoritmo *Random Forest*. A metodologia proposta utiliza-se da estratégia baseada em comitês de votações com decisão feita a partir do consenso de maioria simples de votos aplicada em três níveis: nas variáveis dos conjuntos de dados utilizados para determinar uma possível assinatura de anomalia; nas árvores de decisão que compõem o treinamento de uma floresta aleatória; e entre as diferentes florestas aleatórias para classificar novas observações desconhecidas. A técnica foi aplicada sobre conjuntos de dados reais, que contêm cerca de 100 mil medições feitas sobre o uso da CPU, uso de memória, carga da CPU acumuladas em janelas com os últimos 1, 5 e 15 minutos, taxa de *download* e taxa de *upload*, obtidas a partir de 20 máquinas distribuídas ao longo do globo e coletadas via plataforma PlanetLab, e disponibilizadas no trabalho de Valadão (2009). Sobre este conjunto de dados foram realizados experimentos variando-se o tamanho de cada comitê *ensemble* de florestas aleatórias com tamanhos 3, 4 e 5. Os resultados obtidos são preliminares, relevantes para um *insight* inicial, mas que indicam que novas investigações devem ser conduzidas para aperfeiçoar a metodologia proposta, refinar seus parâmetros ou mesmo colocar sua validade à prova.

Palavras-chave: Detecção de Anomalias, Chebyshev, *Random Forest*, Comitê de Votações.

Abstract

This work presents a proposal for detection of suspect of anomalies in computer networks that uses both statistics and computational intelligence. It uses the concept of Chebushev inequality and Random Forest algorithms merged with a voting strategy in three different levels: a committee to decide with tuples are suspect acoording the Chebushev inequality; a commitee to of decision trees that are part of Random Forest algorithms; and a commitee to suggest which new tuples are suspect of anomalies on unknown datsets, considering the votes of several Random Forest. The datasets used to trainning and check have about 100,000 measurements each one, and consider variables as CPU Usage, Memory Usage, CPU Load, Download Rate and Upload Rate, obtained from 20 machines spread along the globe, from Planetlab platform, available from the work of Valadão (2009). In the experiments tests were done varying the size of the third committe. The results obtained are preliminary, but relevant to first insight. To be more conclusive its necessary further investigations to improve and validate the proposed methodology and, perform it more effective.

Keywords: *Anomaly Detection*, Chebyshev, *Random Forest*, *Voting Committees*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

TI	Técnologia da Informação
ITIL	*Information Technology Infrastructure Library*
IC	Inteligência Computacional
RN	Rede Neural
TCP	*Transmission Control Protocol*
SVM	*Support Vector Machine*
Html	*Hypertext Markup Language*
CSV	*Comma-separated values*
API	*Application Programming Interface*
UTC	*Universal Time Coordinated*

Lista de símbolos

μ	Média amostral
σ	Desvio-Padrão

Sumário

1 INTRODUÇÃO

Uma reportagem feita em 2014 pelo Jornal A Folha de São Paulo [1] mostrou uma pesquisa feita pela ONU (Organização das Nações Unidas) afirmando que quase três bilhões de pessoas hoje no mundo possuem acesso a Internet, ou seja, atualmente a maioria das residências, instituições ou empresas têm à sua disposição um ou mais dispositivos com acesso a Internet que provê conectividade a todos os públicos. Prover tal conectividade admite brechas de segurança que, por sua vez, introduz vulnerabilidades na rede que devem ser sanadas ou, ao menos, detectadas.

Devido a esta disponibilidade crescente de tecnologias ligadas à redes de computadores e seus derivados, como dito na reportagem, a detecção de ataques e anomalias em redes de computadores torna-se um tema importante cada vez mais estudado. Existem ainda diversas oportunidades e visões diferentes para sua exploração e contribuição, seja no estado da arte, seja na proposição de novas formas de detecção. É neste contexto que se encaixa o presente trabalho.

Segundo Bhuyan (2013) uma tentativa de invasão ou ameaça causadora de anomalias na rede é definida como: tentativa deliberada e não autorizada de: (i) Acessar informações; (ii) manipular informações, ou (iii) tornar um sistema pouco confiável ou inutilizável. Fries (2017) em um artigo publicado, comenta sobre o uso de datasets conhecidos para descoberta de anomalias e discorre sobre um dos mais famosos estudos desenvolvidos pelo MIT (*Massachusetts Institute of Technology*) sobre detecção de anomalias na rede, sendo que o MIT até organiza campeonatos para o descobrimento de anomalias, campeonato que leva o nome de KDD-Cup. Fries ainda faz menção sobre alguns tipos de anomalias já reconhecidos através do uso deste *dataset*, mais especificamente: ***Denial of Service* (DOS)**, anomalia facilitada por ocorrência de falhas ou segmentação de *bugs* no sistema e caracterizada por tornar o computador *host* ou a rede indisponível para os usuário, sobrecarregando-o com o recebimento de pacotes TCP e, por consequência, sobrecarregando o servidor; ***User to Root* (U2R)**, anomalia caracterizada por um invasor obtendo acesso a uma máquina através do uso de uma conta normal de usuário, mas com privilégios de usuário *root* que causam danos ao usuário invadido; ***Remote to User* (R2L)**, que é caracterizada por um invasor enviar pacotes de uma máquina de um usuário legítimo para o *host*, conseguindo ganhar privilégios de super usuário com a liberdade de explorar vulnerabilidades no servidor, políticas de segurança, dentre outros riscos; e ***Probing* (PROBE)**, que é caracterizado pelo atacante realizar um sondagem no sistema através de uma varredura na rede, coletando informações de segurança e explorando as vulnerabilidades do sistema como um todo. Para

[1] https://goo.gl/V6ZKa4, Acessado em 30, Jan. 2018.

uma leitura mais detalhada sobre os tipos de anomalias sugere-se a leitura de Fries (2017).

Diante do exposto, o presente trabalho propõe uma abordagem para identificação de anomalias que utiliza técnicas estatísticas e de aprendizado de máquina. O restante do texto é dado como segue. Ainda no Capítulo 1 serão apresentados a motivação para o trabalho, a justificativa para executá-lo, a solução proposta e os objetivos propostos. No Capítulo 2 serão apresentados os fundamentos teóricos necessários para acompanhar o entendimento sobre o trabalho, com ênfase em uma desigualdade conhecida da Estatística denominada de Desigualdade de Chebyshev, e em técnicas de Inteligência Computacional, em especial aquelas relacionadas à aprendizado de máquina e representação de conhecimento, apenas com o intuito de apresentar ao leitor leigo a gama de técnicas disponíveis e sem a pretensão de exaurir o conhecimento sobre o tema. No Capítulo 3 são apresentados os detalhes sobre a metodologia de desenvolvimento do trabalho, com especial destaque para os materiais de *hardware* e *software* empregados, e as principais fases do estudo conduzido neste Trabalho de Conclusão de Curso. No Capítulo 4 serão apresentados alguns detalhes sobre a arquitetura da solução proposta, conjuntos de dados usados, e planejamento de experimentos realizados para validar a metodologia proposta nos diferentes cenários de análise. No Capítulo 5 os resultados obtidos são apresentados, e brevemente discutidos. Por fim, no Capítulo 6 algumas considerações finais são feitas à luz dos resultados encontrados e discutidos, assim como são deixadas para futuros trabalhos algumas idéias de continuidade.

1.1 Motivação e Justificativa

Como mencionado na Introdução deste trabalho, estima-se que cerca de três bilhões de pessoas tem acesso a *Internet*. Como consequência o número de empresas que desempenham trabalhos relacionados a área de TI tem ganhado mais espaço no mercado atual. Segundo o jornal o Globo (2016) [2] o Brasil emprega 1,3 milhão de pessoas nesse mercado, que espera um profissional qualificado para cada um dos cerca de 50 mil postos de trabalho em aberto.

Com o crescimento da Internet e da popularização dos computadores, as empresas passaram a depender de alta disponibilidade de seus serviços de TI. Segundo o site Opservices (2014) [3], grandes empresas aplicam Governança de TI para melhorar seus processos internos com uso de *softwares*, assim como práticas que garantem a segurança da informação nos processos que ela realiza. Dentre as práticas mais empregadas estão aquelas relacionadas com o Gerenciamento de Incidentes do ITIL (*Information Technology Infrastructure Library*), que podem se beneficiar de ferramentas que sejam capazes de identificar anomalias no momento em que ocorrem e determinar suas origens (como ataques, invasão e outros) em tempo de reagir com impacto mínimo para o negócio.

[2] https://goo.gl/B6ufqa; Acessado em 21, out. 2018
[3] https://www.opservices.com.br/governanca-de-ti/; Acessado em 21, out. 2018.

Isto posto, é de suma importância que os impactos e prejuízos causados por anomalias e intrusões na rede sejam minimizados, e para isso é preciso que as anomalias sejam descobertas com precisão, diminuindo os índices de falsos positivos e negativos. Para que isso aconteça, novas técnicas e métodos de identificação devem ser estudados. O intuito deste trabalho é justamente propor uma nova abordagem que, se promissora, pode tornar-se futuramente um "motor" para *softwares* de detecção de anomalias na rede.

1.2 Solução Proposta

Mediante o que foi apresentado nas subseções anteriores, este trabalho tem como proposta o uso de técnicas de análise estatística e de inteligência computacional para detecção de anomalias. Ela consiste em uma abordagem em duas fases, sendo a primeira uma pré-análise estatística chamada análise pela desigualdade de Chebyshev, onde por meio de votação sobre o número de discrepâncias encontradas nas variáveis que descrevem os dados da rede ao longo do tempo confirmam-se suspeitas de anomalias em um conjunto restrito de dados, seguido pela segunda fase que realiza uma investigação e classificação de situações anômalas por meio do uso da técnica de aprendizado de máquina denominada *Random Forest* sobre observações ainda não conhecidas. A proposta desenvolvida usa a combinação de ambas as técnicas com decisão baseada em votação para indicação de situações anômalas. É preciso esclarecer que o trabalho busca gerar apenas um 'prova de conceito' para o problema, sem a pretensão de gerar uma aplicação ou produto final pronto para uso no contexto de detecção de anomalias carecendo, portanto, mais investigação para validar o que se propôs.

1.3 Objetivos

Nesta seção serão apresentados os objetivos geral e específicos que, se cumpridos, permitem considerar que a execução da proposta do projeto foi bem sucedida.

1.3.1 Objetivo Geral

- Apresentar, como prova de conceito, uma metodologia que envolve detecção de anomalias de maneira indireta, mediante aprendizado de máquina sobre padrões de assinatura de sinais que foram considerados anômalos em *datasets* contendo informações sobre uso de CPU, uso de memória, volume de dados transmitidos e recebidos, e carga da máquina em número de processos.

1.3.2 Objetivos Específicos

Para que ao fim do projeto se pudesse considerar que o objetivo geral foi cumprido e que a prova de conceito foi válida, é necessário cumprir alguns objetivos menores, denominados objetivos específicos. Quando cumpridos, em conjunto concretizam o objetivo geral. Para este projeto os objetivos específicos propostos foram:

- Implementar identificação de tuplas anômalas mediante uso de desigualdade de Chebyshev;
- Calibrar a implementação inspirada em Chebyshev para reduzir a quantidade de falsos positivos, escolhendo valores específicos para os parâmetros de sensibilidade associados com cada variável do conjunto de dados analisado;
- Treinar um conjunto de florestas aleatórias (*random forests*) para reconhecer a 'assinatura' de uma anomalia utilizando aprendizado sobre tuplas classificadas como anômalas pela desigualdade de Chebyshev;
- Reportar resultados da análise da proposta de identificação sobre um conjunto de dados específico na forma de gráficos e tabelas para discussão dos resultados, visualização dos resultados alcançados e possível validação da técnicas ou identificação de melhorias que possam agregar para o refinamento da proposta.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados toda fundamentação teórica para o desenvolvimento deste projeto, bem como todos os trabalhos relacionados ou aqueles que seguem a mesma linha de pensamento e desenvolvimento que auxiliaram e nortearam a implementação da proposta alvo do presente documento.

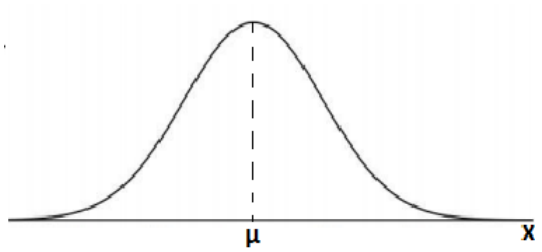
2.1 Desigualdade de Chebyshev

Segundo Walpole et. al. (2010), em seu livro "Probabilidade e Estatística para Engenharia e Ciências", é possível afirmar que a partir da variância de uma variável aleatória X algumas características dos dados podem ser inferidas, como por exemplo, a variabilidade das observações em torno da média. O autor afirma ainda que se uma variável X possui pequenos valores de variância ou pequenos valores de desvio-padrão, é esperado que grande parte dos valores fiquem em torno da média, de forma que a probabilidade de que uma variável aleatória admita um valor em um certo intervalo ao redor da média é maior do que para uma variável aleatória com um grande valor de desvio-padrão. Pensando graficamente, uma distribuição de probabilidade com maior variância possui um maior espalhamento dos valores ao redor da média enquanto que uma distribuição com menor variância tende a concentrar seus valores distribuídos com maior frequência ao redor da média, situações estas ilustradas a seguir. A Figura 1a apresenta uma distribuição platicúrtica onde os dados se encontram mais espalhados e distantes da média (caso com variância e desvio grandes), enquanto que na Figura 1b temos uma distribuição leptocúrtica, com baixa variabilidade e maior concentração de valores ao redor da média. O mesmo argumento também vale para distribuições discretas como observa-se na Figura 2a e 2b.

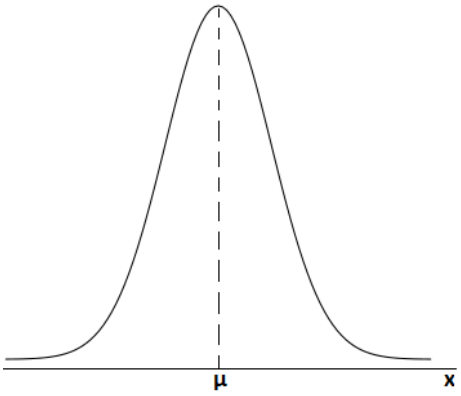
Walpole et al (2010) comentam em seu livro que em meados no século 19 o matemático P. L. Chebyshev propôs que: "(...) a fração da área entre quaisquer dois valores simétricos sobre a média está relacionada ao desvio-padrão (...)". Como a área abaixo da curva do histograma, ou distribuição de probabilidade, resulta em uma soma igual a um, então a área entre dois números quaisquer é a probabilidade da variável aleatória X assumir um valor entre estes dois números. A idéia de Chebyshev foi prover uma "estimativa conservadora" para a probabilidade de variável aleatória X assumir valor a k desvios padrão ao redor de sua média para k real não nulo. Como pode ser apresentado no Teorema 1.

Figura 1 – Variabilidade de observações contínuas em torno da média.

(a) Distribuição Normal Platicúrtica



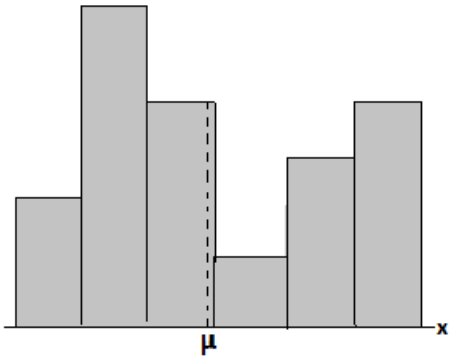
(b) Distribuição Normal Leptocúrtica



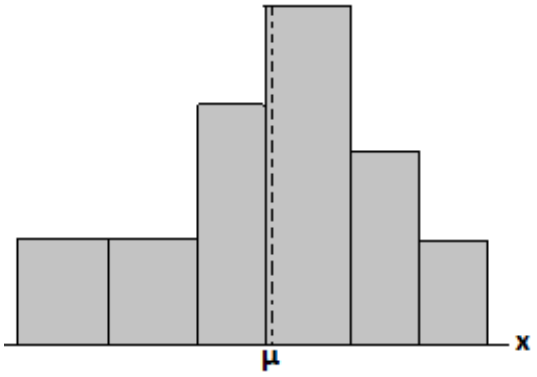
Fonte : Baseado no Livro Probabilidade & Estatística para Engenharia e Ciências 8ª Edição - p. 84

Figura 2 – Variabilidade de observações discretas em torno da média.

(a) Distribuição Discreta Menos Espalhada



(b) Distribuição Discreta Mais Espalhada



Fonte : Baseado no Livro Probabilidade & Estatística para Engenharia e Ciências 8ª Edição - p. 84

Teorema 1 (Toerema de Chebyshev) *A probabilidade de que uma variável aleatória X assuma um valor a k desvios-padrão da média é pelo menos, $1 - 1/k^2$, ou seja,*

$$P(\mu - k\sigma < X < \mu + k\sigma) \geq 1 - 1/k^2$$

A prova para a desigualdade apresentada pelo teorema é dada por Walpole et al (2010), e não será apresentada neste texto, podendo ser consultada diretamente da fonte. Uma questão de destaque sobre o Teorema de Chebyshev é que ele é válido para qualquer distribuição de probabilidades, mas deve ser entendido como um **limite inferior** para

a probabilidade de uma variável aleatória X situar-se à uma quantidade k de desvios ao redor de sua média. A real probabilidade só pode ser determinada conhecendo-se a distribuição de probabilidades em questão. Por ser um resultado **livre de distribuição**, é útil em aplicações em que a forma da distribuição não é conhecida. Villamarín-Salomón e Brustoloni (2008) complementam que métodos de detecção de anomalias baseados na desigualdade de Chebyshev utilizam um resultado direto do Teorema, que é dado por $P\Big(\|X - E(x)\| \geq k \cdot \sigma\Big) \leq 1/k^2$, e que são usados quando (i) a distribuição dos dados disponíveis é desconhecida ou o experimentador não quer fazer considerações acerca de sua distribuição, e (ii) é esperado que as observações sejam independentes umas das outras. Nesta variação, a fórmula estabelece um **limite superior** para a porcentagem de dados cujo valor está k vezes o desvio padrão acima e abaixo da média populacional. Neste caso, valores de k podem ser usados como critério de corte para significância estatística onde dados que estejam além de k desvios padrão ao redor da média são considerados anômalos. Para exemplificar, se k é definido como 4.47, logo o limite superior é 0.05, ou seja, 5% dos valores observados serão considerados anômalos, e estarão localizados à $\pm$ 4.77 desvios padrão ao redor da média da variável X em questão.

2.2 Técnicas de Inteligência Computacional

Nesta subseção serão apresentadas todas as técnicas de inteligência computacional e machine learning que, segundo a literatura especializada, já foram investigadas à luz do tema de detecção de anomalias. Ela é dividida em subseções de 2.2.1 à 2.2.5 que descrevem as seguintes técnicas de Inteligência Computacional (IC): *Random Forests* (Florestas Aleatórias), Árvores de Decisão, *Support Vector Machines*, Sistema Especialista Fuzzy e Redes Neurais. Nelas serão apresentados detalhes necessários para a sua compreensão. Por fim, a subseção 2.2.6 encerra apresentando alguns trabalhos correlatos que empregam as referidas técnicas de IC.

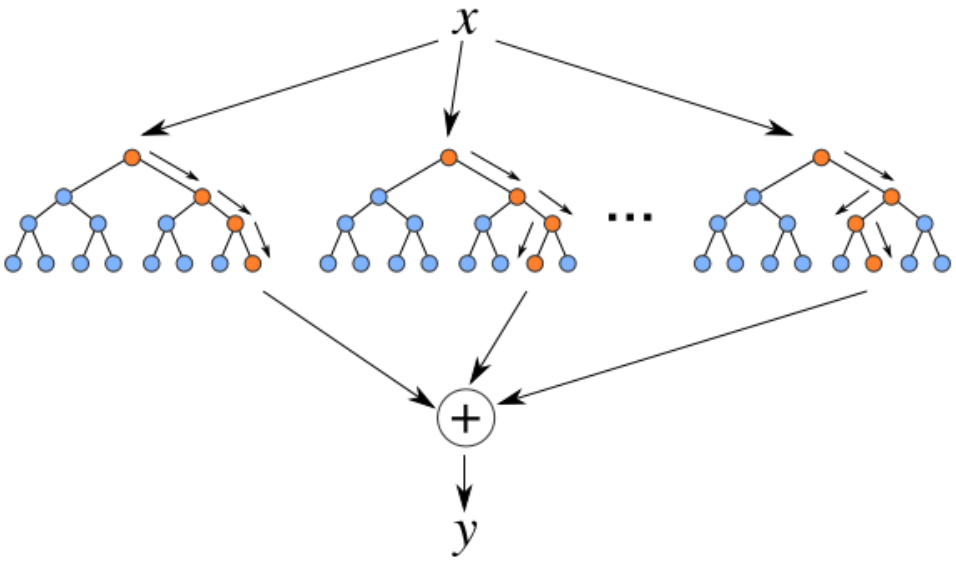
2.2.1 Random Forest

Random Forest ou Florestas Aleatórias, é um método de aprendizado de máquina muito empregada em problemas de classificação, regressão e outras tarefas. É uma técnica derivada do método chamado 'árvores de decisão'. Como afirma Odewald (2018), as árvores randômicas aleatórias são usadas para corrigir um problema encontrado pelo método o qual elas se originaram, chamado de *overfitting*. O termo é muito usado em estatística e descreve a situação onde um modelo se ajusta bem ao conjunto de dados anteriormente observado, mas se mostra ineficaz para prever dados desconhecidos pelo modelo (INVESTOPEDIA, 2018).

Em seus artigo Svetnik (2003) definiu florestas aleatórias como segue: uma *Random*

Forest é um conjunto ***B*** de árvores de decisão (T), ou seja, $T_1(X)$,..., $T_B(X)$, em que X = X_1,..., X_p é um vetor p-dimensional de descritores ou propriedades associadas a um nó da árvore. Cada conjunto de árvores produz uma saída $\hat{Y}_1 = T_1(X)$, ..., $\hat{Y}_B = T_B(X)$, onde $\hat{Y}_b$, b = 1...B é a previsão para a b-ésima árvore. Ao final, as saídas de todas as árvores são agregadas para produzir uma previsão final, $\hat{Y}$, por meio de alguma combinação (*ensemble*) dos resultados de cada árvore do conjunto. Para problemas de classificação, $\hat{Y}$ é a classe prevista pela maioria das árvores, em geral decidido via votação, onde cada saída de uma árvore é um voto. Já para problema de regressão é feito uma média a partir do resultado de todas as saídas das árvores individuais, obtendo assim a resposta para a floresta. A estrutura de decisão de uma floresta aleatória é exemplificada na Figura 3.

Figura 3 – Exemplo de uma *Random Forest*



Fonte: Geoffrey Hodgins(2016)

Bylaiah (2017) Floresta aleatória é um método baseado em conjuntos (*ensemble*), em que os classificadores são construídos a partir de uma combinação de vários outros classificadores base. A independência é teoricamente reforçada pelo treinamento de cada classificador base em um conjunto de treinamento amostrado com a substituição do conjunto de treinamento original. Essa técnica é chamada de *bagging* ou agregação de *bootstrap*. Em algoritmos *Random Forest* a aleatoriedade é introduzida na identificação do melhor recurso de divisão do subconjunto de recursos disponíveis.

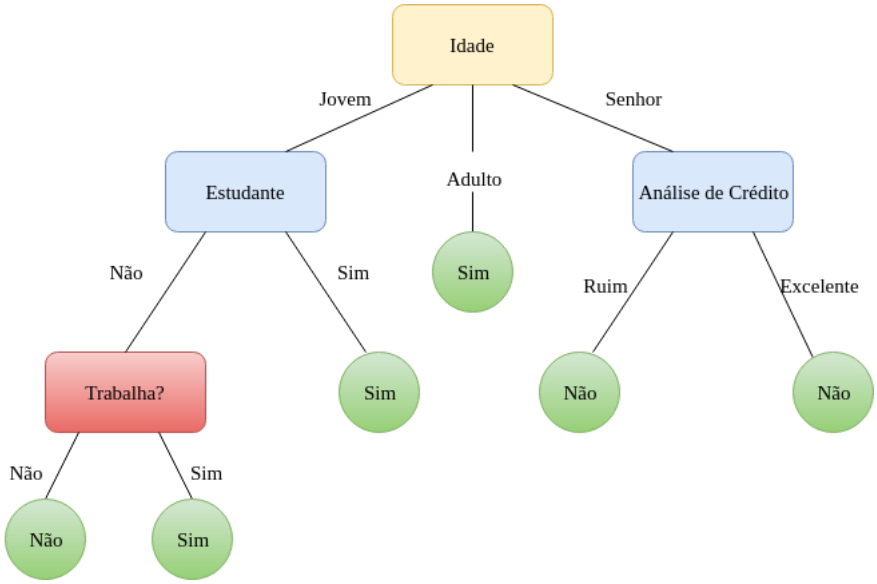
Segundo Prashanth et. al (2008), algoritmos *Random Forest* podem criar padrões com muita eficiência sobre *datasets* balanceados e ter uma ótima performance no que diz respeito a classificação de padrões. Gulenko et. al. (2016) conclui em seu trabalho sobre detecção de anomalias que em testes feitos o algoritmo *Random Forest* obteve uma

acurácia de 99.35% em alguns casos de teste. Logo, trata-se de uma técnica promissora para investigar em problemas de detecção de anomalias, levando em conta novos casos ainda não pesquisados.

2.2.2 Árvores de Decisão

Árvore de decisão ou *Decision Tree* é, segundo Quinlan (1990), uma estrutura recursiva que usa de regras do tipo Se ... Então em cada um de seus nós com objetivo de classificação, onde cada nó analisa um atributo do conjunto de dados. Para classificar um objeto por meio de uma árvore de decisão começa-se analisando os atributos do objeto pela raíz da árvore, analisando as respostas para cada nó intermediário, até que atinja-se um nó folha. Para a classificação, o objeto passa por uma série de etapas, neste caso, nós da árvore, que possuem características mutuamente exclusivas, ou seja, os nós têm espécies de definições de atributos, e os objetos analisados são separados por meio de testes de pertinência feitos em cada um dos nós. A classe ao qual o objeto pertence é definida quando atinge-se, segundo esse procedimento, um nó folha da árvore rotulado com a referida classe. Um exemplo simples de árvore de decisão é apresentado a seguir na Figura 4, que apresenta uma árvore de decisão cujo objetivo é classificar pessoas aptas a obter um benefício de um banco, baseado na idade e ocupação dos interessados.

Figura 4 – Exemplo de uma Árvore de Decisão para análise de clientes para obter benefício de um banco.



Fonte: Elaborado pelo Autor

Segundo Hosokawa (2011), uma árvore de decisão pode ser facilmente transformada num conjunto de regras de classificação do tipo **Se .. Então**, como trechos de código tal

como exibido abaixo:

> **As regras são do tipo: IF L_1 AND L_2 ... AND L_n THEN Classe = Valor, onde L_i são expressões do tipo Atributo = Valor. Para cada caminho, da raiz até uma folha, tem-se uma regra de classificação. Cada par (atributo,valor) neste caminho dá origem a um L_i.**

Para a Árvore de Decisão apresentada no exemplo descrito pela Figura 4, sua transformação em conjunto de regras poderia ser feito da seguinte maneira:

IF Idade = Jovem **AND** Estudante = Sim **THEN** Classe = Sim
IF Idade = Senhor **AND** Analise_Credito = Excelente **THEN** Classe = Sim
IF Idade = Jovem **AND** Estudante = Não **AND** Trabalha = Não **THEN** Classe = Não
IF Idade = Adulto **THEN** Classe = Sim

Segundo Hosokawa (2011), as árvores de decisão são muito usadas na mineração de dados e possuem muitas vantagens quanto a sua simplicidade de interpretação e explicação, além de dispensarem tratamentos nos dados de entrada. Esse tipo de técnica está apta a trabalhar tanto com dados nominais quanto categóricos, além de poder ser validado com uso de testes estatísticos, tornando fácil a verificação da confiabilidade do método. Possui ainda uma enorme eficiência na construção dos modelos, com complexidade média de O (n log(n)).

2.2.3 *Support Vector Machines*

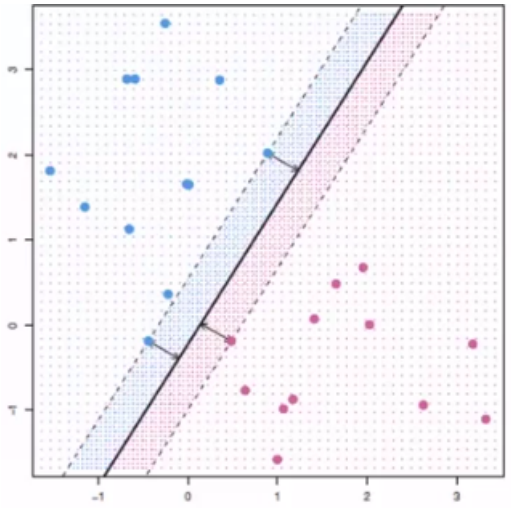
Em uma tradução literal, as Máquinas de Vetores Suporte (*SVM - Support Vector Machines*), consistem em uma técnica de aprendizado de máquina muito utilizada para classificação, seja supervisionada ou não, que segundo Lorena & Carvalho (2003) têm obtido melhores resultados até quando comparado com algoritmos como redes neurais artificiais e algoritmos genéticos.

Informalmente, Tadewald (2018) apresenta no curso intitulado "*Python para Data Science e Machine Learning*"[1] disponível na plataforma *Udemy*[2] um breve funcionamento sobre o método, descrito como segue: dado com um conjunto de dados para treino, sendo marcado cada um deles como uma classe, SVM cria um modelo que engloba novos valores em uma das categorias criadas com auxílio de um classificador não probabilístico. O modelo criado é uma representação de pontos no espaço, que são mapeados e separados de maneira que fiquem com a maior distância possível entre eles. Para definir a maior separação dos dados é necessário analisar a disposição deles no plano.

[1] https://www.udemy.com/python-para-data-science-e-machine-learning/
[2] https://goo.gl/i5NDnt

O algoritmo de SVM busca determinar analiticamente hiperplanos que vão maximizar as margem entre as classes (i.e., as distâncias entre os dados), usando para isso pontos do conjunto de dados que demarcam esses hiperplanos. Tais pontos são determinados de vetores de suporte, ilustrados na Figura 5 como os pontos que estão sobre as margens, destacadas por pontilhados. Quanto maior for a margem, menor é a chance de ocorrer o *overfitting*.

Figura 5 – Margens do Hiperplano Traçado, com destaque para as classes localizadas à esquerda e à direita do hiperplano separador e para os pontos que são usados para formar os vetores de suporte para o conjunto de dados hipotético.

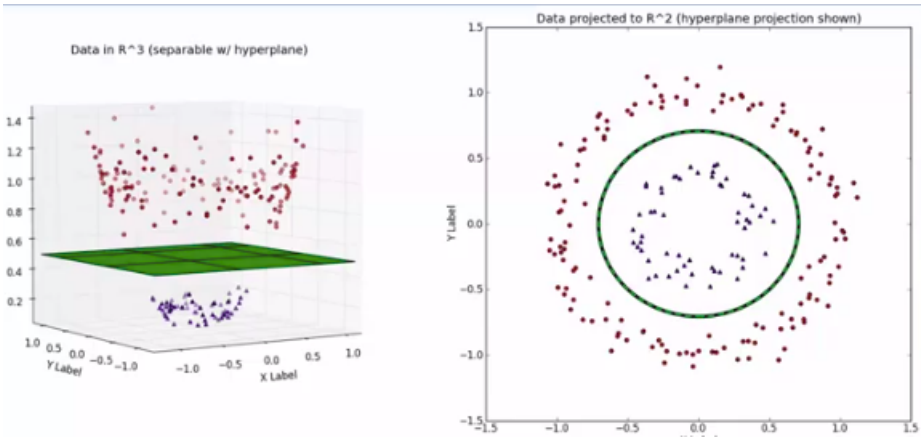


Fonte: https://www.udemy.com/python-para-data-science-e-machine-learning/; Minuto 2:22; Acessado em 03 Jun. 2018

Haijun (2007) afirma que na ocorrência dos dados do conjunto analisado não serem linearmente separáveis (isto é, não puderem ser separados por um único hiperplano separados) o SVM possui um recurso denominado de *Kernel Trick*. Trata-se de uma função que, quando aplicada sobre os dados originais, transforma-os de forma que após sua transformação possam ser separados linearmente. A Figura 6 apresenta um exemplo de conjunto de dados. No exemplo, os dados estão originalmente dispostos em um espaço R^2, como mostra a figura da direita presente na Figura 6. Embora não seja linearmente separável, com a aplicação de um *Kernel Trick* os dados são projetados em R^3, onde um hiperplano separador pode linearmente separá-los.

Deste ponto em diante serão dados alguns *insights* acerca de como o *Support Vector Machine* determina o hiperplano de maior margem de maneira analítica. Segundo Haijun (2007), matematicamente, a superfície de decisão na forma de um hiperplano que realiza

Figura 6 – Hiperplanos em dados não-lineares.



Fonte: https://www.udemy.com/python-para-data-science-e-machine-learning/; Minuto 4:08; Acessado em 03 Jun. 2018

separação dos dados é obtida pela Equação:

$$w^t \cdot x + b = 0 \tag{2.1}$$

Em que x é um vetor de entrada, w é um vetor de peso ajustável e b é um *bias*. Em um problema de classificação se deve estimar é uma função $f : R_n \rightarrow \pm 1$ usando dados de treinamento, por exemplo se pode denotar uma dada classe A com $x \in A, y = 1$, e outra data classe B com $x \in B, y = -1$, $(x_i, y_i) \in R_n \cdot \pm 1$. Se os dados de treinamento são linearmente separáveis, logo existe um par $(w, b) \in R_n \cdot R$ de modo que:

$$w^t \cdot x + b \geq +1, \quad \text{Para todo } x \in A \tag{2.2}$$

$$w^t \cdot x + b \geq -1, \quad \text{Para todo } x \in A \tag{2.3}$$

Com a função de decisão dada por:

$$f_{w,b}(x) = sign(w^t \cdot x + b) \tag{2.4}$$

As desigualdades presentes nas inequações (2.2) e (2.3) podem ser combinadas resultando em:

$$y \cdot (w^t \cdot x + b) \geq 1, \quad \text{Para todo } x \in A \ \cup B \tag{2.5}$$

O classificador com margem máxima otimiza a separação dos dados com o hiperplano de máxima margem. O problema de aprendizagem é então reformulado por $\frac{1}{2}\|w\|^2$ sujeito às restrições de separabilidade (2.5). A otimização é um problema de programação quadrática (QP):

$$\begin{aligned} &\text{Minimize } \Phi(x) = \frac{1}{2}\|w\|^2 \\ &\text{s.t.} \quad y \cdot (w^t \cdot x + b) \geq 1. \end{aligned} \tag{2.6}$$

Isto posto, Lorena e Carvalho (2007), comentam sobre algumas vantagens que justificam o uso desta técnica, são elas: As SVMs tem uma boa capacidade de generalização, ou seja, tem um boa desenvoltura em classificar novos dados que não pertenciam ao conjunto de treinamento, evitando *overfitting* que faz com que o classificador possua baixo desempenho na classificação de novos padrões. As SVMs também possuem robustez quando lida bem com objetos de tamanhos grandes, como por exemplo imagens. Essa técnica ainda possui convexidade da função objetivo, quadrática, que possui apenas um mínimo global, levando vantagem sobre outros algoritmos de aprendizado que possuem mínimos locais em sua função objetivo.

2.2.4 Sistema Especialista *Fuzzy*

Segundo Gomide (1994), os sistemas Especialistas *fuzzy*, ou simplesmente conhecidos como sistemas baseados em lógica *fuzzy* (nebulosa), são aqueles que ao contrário dos convencionais inspirados em lógica, admitem valores verdade segundo uma função de pertinência em detrimento aos valores clássicos verdadeiro (T) e falso (F) sendo, portanto, úteis para modelar incerteza.

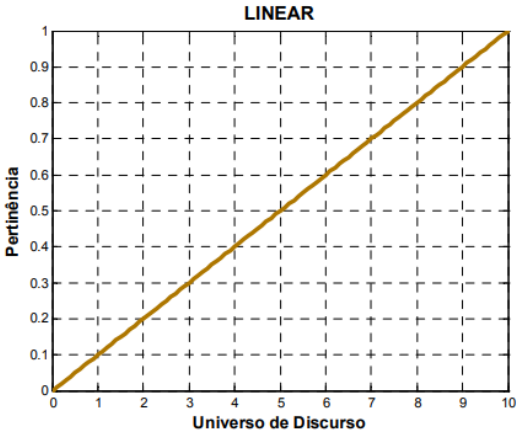
> **Nos sistemas lógicos multi-valores, o valor verdade de uma proposição pode ser ou um elemento de um conjunto finito, num intervalo, ou uma álgebra booleana. Na lógica nebulosa, os valores verdade são expressos linguisticamente, (e.g. : verdade, muito verdade, não verdade, falso, muito falso, ...), onde cada termo linguístico é interpretado como um subconjunto fuzzy do intervalo unitário. (GOMIDE, 1995. p. 1)**

Para melhor entendimento dessa técnica, Coppin (2004), em seu livro chamado "Inteligência Artificial", deu algumas definições tais como: variáveis linguísticas, que tem uso extensivo na lógica *fuzzy*, consistem em um conceito como "altura" que pode ter uma faixa de valores nebulosos como "alto", "baixo", "médio"; conjuntos nebulosos, que contrastam com os conjuntos utilizados na teoria tradicional de conjuntos conhecidos como conjuntos nítidos (crisp), são definidos por uma função chamada de função de pertinência, que tem como objetivo mapear cada elemento do conjunto universo U em um valor entre 0

e 1, comumente chamado fuzzyficação ou nebulização. As principais funções de pertinência usadas são, segundo Neto (2006), as que seguem:

1. Linear, que pode ser tanto crescente quanto decrescente. A Figura 7 exemplifica uma função linear crescente.

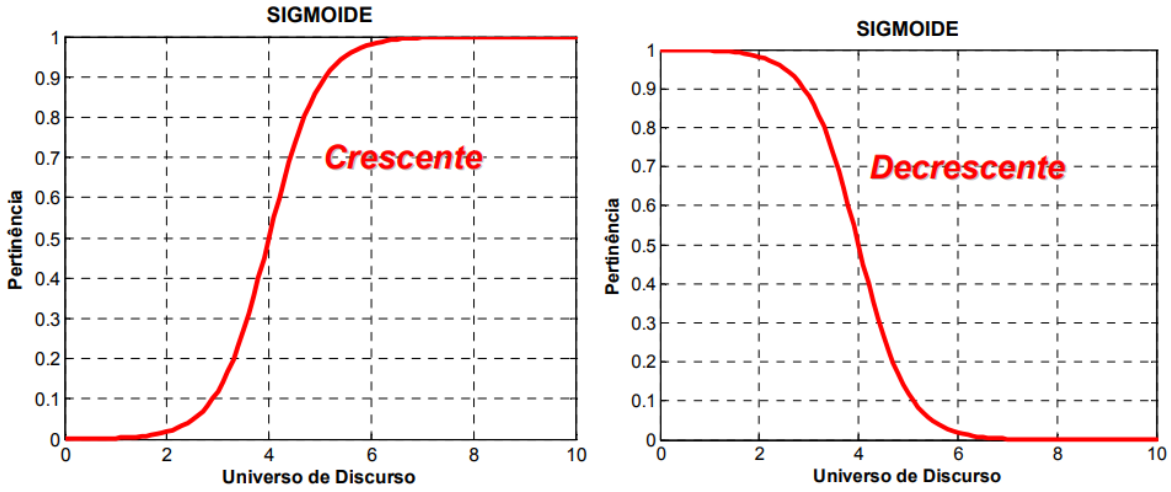
Figura 7 – Função de pertinência linear crescente.



Fonte Neto (2006).

2. Sigmóide, que possui dois tipos de equação: crescente e decrescente. Ambas são apresentadas na Figura 8.

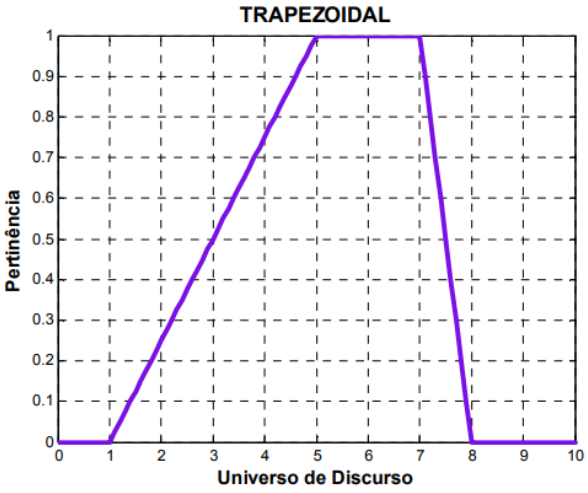
Figura 8 – Funções crescente e decrescente de pertinência Sigmóide.



Fonte Neto (2006).

3. A função Trapezoidal exemplificada graficamente na Figura 9.

Figura 9 – Função de pertinência Trapezoidal.



Fonte Neto (2006).

Segundo Oliveira et al (2005), a estrutura básica de um sistema baseado em regras *fuzzy* é a seguinte: (i) um fuzzificador, que traduz a informação de entrada em Conjuntos *Fuzzy*. A cada variável de entrada são atribuídos termos linguísticos que são os estados da variável, e cada termo linguístico é associado a um Conjunto *Fuzzy* traduzido por uma função de pertinência; (ii) uma base de conhecimento, que contém um conjunto de regras *Fuzzy* do tipo **SE A operador x ENTÃO B = y** (em que os operadores podem ser =, >, <, >=, <, conhecido como base de regras e um conjunto de funções de pertinência conhecido como base de dados; (iii) um método de inferência, que aplica um raciocínio *Fuzzy*, para obter uma saída *Fuzzy*; e (iv) um defuzzificador, que traduz a saída por um valor numérico.

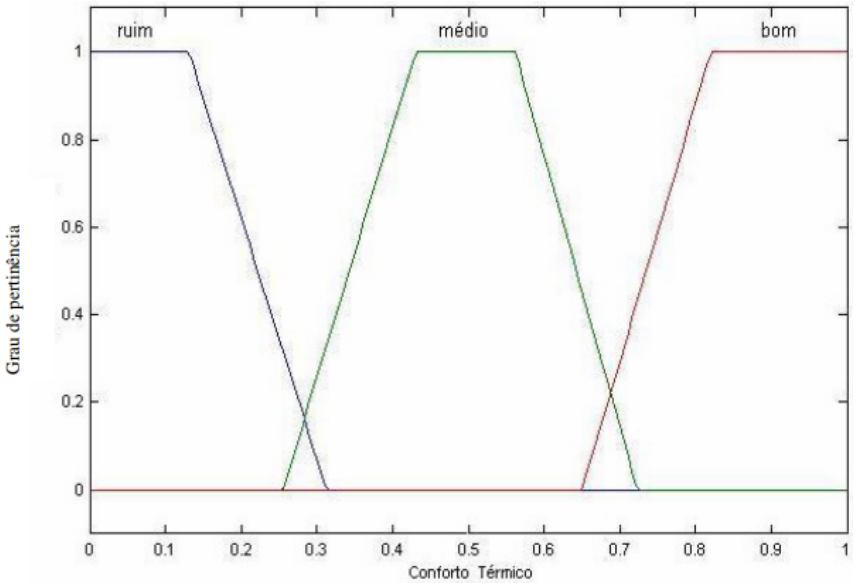
Geralmente em sistemas *fuzzy* o uso de variáveis linguísticas é mais comum e utilizado, pois essas variáveis assumem valores quantitativos em forma de adjetivos. Para exemplificar, se a intenção é modelar uma variável de entrada que representa a temperatura de algo, então para a variável linguística temperatura teríamos os possíveis conjuntos nebulosos representando situações: temperatura alta, temperatura média, e temperatura baixa, dependendo do grau de pertinência de um valor à cada um destes conjuntos. Dito isso se pode afirmar, segundo Salomon (2017), que a função de pertinência varia de 0 a 1 e ela diz respeito a possibilidade de ocorrência de um certo elemento em um conjunto.

Portanto, a partir dessas funções de pertinência é possível definir onde as variáveis de entrada do processo estão localizadas dentre os conjuntos criados. Esses tipos de conjuntos são chamados conjuntos nebulosos. Na clássica teoria (desenvolvida por Aristóteles) os conjuntos são chamados de nítido ou “crisp” e um elemento qualquer dado do universo, pertence ou não a esse conjunto (ABAR, 2004). Abar (2004) afirma que já na teoria dos

conjuntos nebulosos existe um grau de pertinência de cada elemento a um determinado conjunto.

Um exemplo como esses conjuntos são criados para o controle de conforto térmico de temperatura explorado por Oliveira et. al (2005). Nesta aplicação o sistema possui uma variável linguística denominada 'Conforme Térmico', sendo mapeada por três conjuntos nebulosos que representam as situações de conforto térmico ruim, médio e bom, segundo a função de pertinência trapezóide usada. Esse exemplo de uso pode ser visto graficamente na Figura 10.

Figura 10 – Funções de pertinência dos Conjuntos *Fuzzy* assumidos pela variável lingüística conforto térmico, para aves de postura.



Fonte Oliveira et. al (2005).

Para obter um sistema nebuloso capaz de caracterizar as variáveis de entrada em conjuntos, é necessário seguir uma sequência de passos. Segundo Mamdani (1970) são eles: Primeiro determinar o conjunto de regras nebulosas, em segundo lugar "nebulizar" os valores de entrada usando funções de pertinência, após isso combinar as entradas nebulosas de acordo com os antecedentes das regras, depois encontrar o consequente das regras combinando valores, depois combinar os consequentes das regras para obter distribuição de saída e por fim, "desnebulizar" a saída para obter um valor nítido como resultado.

Segundo Souza (2010) a determinação do conjunto de regras *fuzzy* é uma das partes mais importantes do sistema nebuloso. Composto por proposições *fuzzy*, cada uma delas é descrita também em forma linguística (e.g. Se está Chovendo e não tem Sol, Então a temperatura é baixa). Ou de uma maneira mais formal : Se (x_1 é A_1) e (x_2 é A_2) e ... e (x_n é A_n) então (u_1 é B_1) e (u_2 é B_2) e ... e (u_m é B_m). Neste ponto ainda as variáveis linguísticas

são catalogadas e modeladas para conjuntos *fuzzy* ou para funções de pertinência, tais como apelos intuitivos, ajustes de curvas, interpolação e até mesmo rede neurais.

Após todas as etapas intermediárias que são responsáveis por transformar os valores linguísticos em valores matemáticos, é necessário "desnebulizar" a saída para que a mesma fique entendível no mundo real. Há vários métodos para que essa técnica dê certo, tais como: O método da centróide (*Center of gravity* - COG), a média dos máximos (*Mean of Maxima* - MOM), entre tantos outros.

Para a visualização das demais funções, ler o estudo desenvolvido por Neto (2006). Neto ainda define um último conceito relevante para o entendimento da lógica *fuzzy*, sendo ele: **Desfuzzyficação** ou **desnebulização**, essa etapa do processo diz respeito a transformação da forma fuzzificada (nebulosa) para forma determinística ou precisa (numérica), determinando o valor real da saída. O método mais usado nesta etapa é o método da centróide definido por:

> **Supondo-se um universo de discurso discreto, a saída precisa R é dada pelo centro de gravidade do conjunto de conseqüente obtido pela composição das regras. Nesse caso, n indica o número de quantização da saída, d_i representa o valor da variável de saída para o intervalo de quantização i e $\mu A(d_i)$ o grau de pertinência.(Neto 2006, p.28).**

Sendo assim, o ponto de equilíbrio da região *fuzzy* é encontrado através da média ponderada das regiões, tal resultado é tido pelo uso da fórmula:

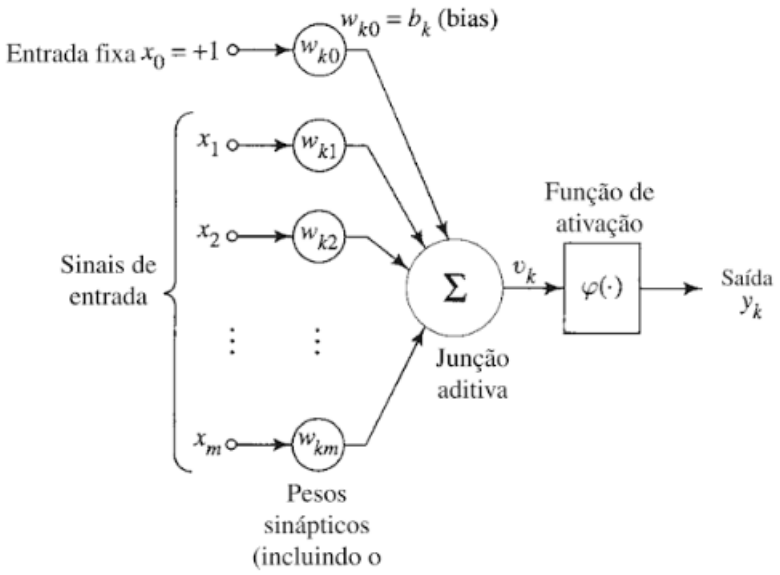
$$R = \frac{\sum_{i=0}^{n} d_i \mu(d_i)}{\sum_{i=0}^{n} \mu(d_i)} \tag{2.7}$$

2.2.5 Rede Neural Artificial

Comumente chamada apenas de Rede Neural (RN), segundo González (1996) é um método computacional inspirado, como muitos outros métodos de inteligência computacional, no funcionamento da natureza, mais precisamente no princípio dos neurônios presentes no cérebro de organismos inteligentes, que adquirem conhecimento através de experiências. Uma RN é uma estrutura conexionista que interliga pequenas unidades de processamento, denominados de neurônios artificiais. De forma análoga às redes neurais naturais, nas redes neurais artificiais cada um de seus nós são neurônios responsáveis pelos cálculos e armazenagem do conhecimento adquirido e que passam os resultados de cada computação adiante para outros neurônios da rede em forma de sinapses. O modelo de um neurônio artificial pode ser visto com mais detalhes na Figura 11 e discutida adiante.

Na Figura 11 estão representados de maneira simplificada como os cálculos e a propagação de resultados de um único neurônio é feita. Primeiramente, as entradas

Figura 11 – Funcionamento do neurônio artificial.



Fonte: Baseado em (HAYKIN, 1999, p. 38).

representadas por "x" ($x_1 ... x_m$), são enviadas para cada neurônio. Neles são aplicados pesos, representados por "w" ($w_{k1} ... w_{km}$), podendo até estar em intervalos, tanto negativos quanto positivos, e podendo ainda ter viés (*Bias* - representado por "b_k") em caso de maior importância do dado de entrada em determinado neurônio. Após a computação de cada nó da estrutura, uma saída é obtida como resultado (no caso representado por "v_k"), os cálculos passam por uma função restritiva, também chamada de função de ativação (representado na figura por "φ"), que restringe o valor de saída a um intervalo finito.

De acordo com Haykin (1999), existem muitos tipos diferentes de funções de ativação. As mais comuns são enumeradas a seguir, em que "φ" representa a função de ativação e v a saída do cálculo feita pela Rede Neural e entrada para função de ativação:

1. Função Limiar, representada matematicamente por:

$$\varphi(v) = \begin{cases} 1 & se \quad v \geq 0 \\ 0 & se \quad v < 0 \end{cases} \tag{2.8}$$

2. Função Linear por partes, representada matematicamente por:

$$\varphi(v) = \begin{cases} 1, & v \geq +1/2 \\ v, & +1/2 > v > -1/2 \\ 0, & v \leq -1/2 \end{cases} \tag{2.9}$$

3. Função Sigmóide, representada matematicamente por:

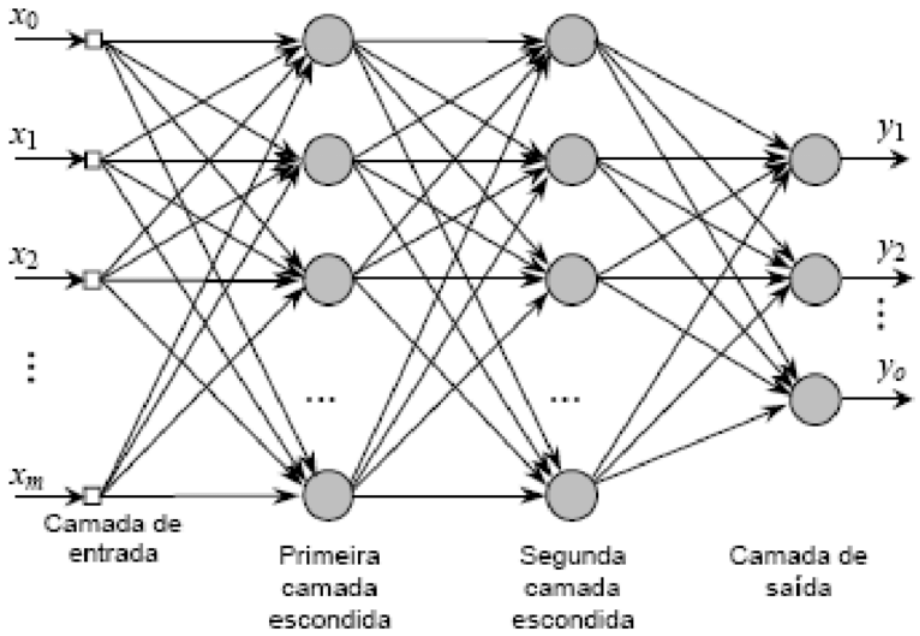
$$\varphi(v) = 1/1 + exp(-av) \tag{2.10}$$

4. Função Gaussiana, representada matematicamente por:

$$\varphi(v) = exp\left(-\frac{\sum_{j=1}^{n}(x_j - w_{ij})^2}{2\sigma^2}\right) \tag{2.11}$$

Existem variações de redes neurais, tais como: recorrente ou realimentada, de estrutura reticulada e *backpropagation*. Segundo Palmiere (2016), a rede neural *backpropagation* uma das arquiteturas mais comuns de redes neurais, que propaga as informações calculadas novamente de volta para os neurônios, usando geralmente mais camadas de neurônios (camadas ocultas), realimentando-os para melhor aprendizagem. É comumente utilizada em reconhecimento de imagens e sons. Um exemplo de estrutura pode ser visto na Figura 12.

Figura 12 – Exemplo de Rede Neural Multicamadas.



Fonte: Oliveira et al (2010).

Segundo Mahanta (2017), redes neurais possuem várias vantagens que tornam essa técnica adequada para alguns tipos de problemas, tais como: (i) capacidade de aprender relações não-lineares e complexas, o que é importante pois dados reais muitas vezes não são lineares e são complexos; (ii) as RNs podem generalizar, ou seja, depois de terem reconhecido e aprendido um padrão, as RNs tem uma alta capacidade de prever novos

objetos não vistos; (iii) as RNs não impõem nenhuma restrição nas variáveis de entrada, como por exemplo a maneira em que elas serão distribuídas. Segundo o autor, muitos estudos mostraram que as RNAs podem modelar melhor a heterocedasticidade, ou seja, dados com alta volatilidade e variação não constante, dada sua capacidade de aprender relacionamentos ocultos nos dados sem impor quaisquer relacionamentos fixos nos dados. Isso é algo muito útil na previsão de séries temporais financeiras (por exemplo, preços de ações) em que a volatilidade dos dados é muito alta.

Porém, segundo Tu (1996), esse tipo de técnica tem algumas desvantagens. Algumas delas são: (i) RNs são uma "caixa preta" e tem uma capacidade limitada de identificar explicitamente possíveis relações causais; (ii) os modelos tem uma complexidade maior de uso dependendo do campo que se deseja implantá-lo; (iii) a modelagem das RNs necessitam na maioria das vezes de recursos computacionais maiores; e (iv) o desenvolvimento de uma RN é empírico e necessita de muitos ajustes metodológicos.

2.3 Trabalhos Relacionados

Stampar e Fertalj (2015) apresentam várias técnicas da literatura de inteligência computacional aplicadas na detecção de anomalias. Uma das técnicas apresentadas é denominada de floresta aleatória, que trabalha com um sistema de "votação" com as árvores de decisão que compõem sua estrutura. Cada árvore analisa os dados e dá seu "voto" sobre como avalia a situação da rede sobre a entrada de dados feita; por fim, a floresta toma a decisão sobre se a rede encontra-se ou não em situação de anomalia a partir do resultado da maioria dos "votos" recebidos.

Dickerson e Dickerson (2011) apresentam em seu trabalho uma aplicação que utiliza lógica *fuzzy* como motor na detecção de anomalias. Os autores propõem uma aplicação que tem sua arquitetura dividida em sub-tarefas tais como um coletor de dados, processador de dados que sumariza e seleciona cuidadosamente as categorias de cada dado lido na etapa anterior a cada passagem do coletor, criando tabelas para os comparativos entre as leituras anteriores e as atuais. Depois, a cada nova leitura de dados os dados são passados para o sistema *fuzzy* que toma ações chamadas por eles de alertas *fuzzy*. As regras de produção *fuzzy* utilizadas são baseadas nos dados coletados, como por exemplo: número total de pacotes em um determinado intervalo de tempo e sua variância com outras observações; o número de conexões TCP/IP (*Transmission Control Protocol - Protocolo de Controle de Transmissão / Internet Protocol - Protocolo de Internet*) realizadas com sucesso em um intervalo de tempo considerado; dados coletados dos cabeçalhos TCP tais como quantidade de dados presentes no pacote, IPs de fonte e destino do pacote; dentre outros. Com os dados e as regras criadas, o sistema caracteriza um novo dado observado em três categorias de ameaça: baixa tendendo a zero, média-baixa e alta.

Segundo Cannady (1998), para o detecção de anomalias na rede podemos aplicar várias técnicas de inteligência computacional (IC). Um exemplo famoso é o uso de redes neurais artificiais, estrutura capaz de analisar os dados vindos da rede de comunicação e reconhecer comportamentos anômalos. Como ataques mudam sua natureza e características, então é necessário que o sistema de detecção seja flexível, capaz de analisar grandes montantes de dados de tráfego de rede e detectar vários tipos de possíveis ataques. Apesar da flexibilidade, a rede neural também possui desvantagens, pois o aprendizado da rede requer grande volume de dados de entrada para seu treinamento, com padrões previamente reconhecidos, podendo tomar vários dias ou meses para se construir um *dataset* para o aprendizado da rede neural.

3 MATERIAIS E MÉTODOS

Nesta seção será apresentado tudo que foi utilizado no desenvolvimento do trabalho, assim como detalhes de configurações no caso de possível replicação, e descrição da metodologia composta por todos os detalhes relevantes para o entendimento do norte seguido pelo projeto.

3.1 Materiais

Esta subseção diz respeito à listagem de todos os *hardwares* e *softwares* utilizados na confecção deste projeto, bem como toda a configuração original dos mesmos, configurações feitas, e outros detalhes necessários para compreender o funcionamento da solução proposta.

3.1.1 Configuração dos *hardwares* utilizados

Todos os materiais utilizados na plataforma de *hardware* usada estão descritos e detalhados nas Tabela 1. Nela constam a configuração dos *hardwares* utilizados para confecção do *software*.

Tabela 1 – Máquina de implementação e testes do *software*.

Item	**Descrição**
Hardware	• Notebook Acer, Modelo ESI-572-36XW • Processador Intel Core i5-4200U CPU@ 1.60Hz 2.30Hz, 3,00 MB de Memória Cache. • Memória Ram: 6,00 GB. • HDD: 500 GB.
Sistema Operacional	Linux Elementary OS 0.4.1 64 bits.
Compilador/Interpretador	Python 3.0.
Ambiente	Sublime Text.

Fonte: Elaborado pelo autor

3.1.2 *Softwares* Utilizados

Para a confecção deste trabalho foram utilizados alguns *softwares* e bibliotecas especializadas externas à linguagem para o problema em questão. A seguir serão enumerados

os principais:

(i) **Python 3.0:** Interpretador da linguagem de programação *python* na versão 3.0, uma das versões mais atualizadas que permite com que o código possa ser usado em outras aplicações e em outras máquinas que possuem o *python* instalado. Neste projeto foi utilizado na implementação do *software*.

(ii) **Pip:** Gerenciador de pacotes que possibilita e facilita a instalação e gestão de pacotes e bibliotecas feitas em *python*, necessárias para a confecção do sistema como um todo. Utilizado neste trabalho para instalação das bibliotecas auxiliares do *software*.

(iii) **Sklearn:** Biblioteca que comporta vários algoritmos de aprendizado de máquina, muito eficiente para mineração e análise de dados, possuindo código aberto, inclui, algoritmo de classificação, regressão e agrupamento, incluindo máquinas de vetor de suporte (SVM), florestas aleatórias (*Random forest*), *gradient boosting*, *k-means* e DBSCAN. Neste projeto foi escolhida como técnica inteligente a ser utilizada o *Random Forest*.

(iv) **Pandas:** Biblioteca escrita na linguagem *python* para manipulação e análise de dados, utilizada neste trabalho para leitura, gravação e manipulação de arquivos do tipo CSV. Essa biblioteca carrega dados para a memória em objeto do tipo *dataframe*, facilitando operações feitas com os dados.

(v) **Matplotlib:** Biblioteca para geração de gráficos para análise, desenvolvida para linguagem *python* e sua extensão matemática Numpy, essa biblioteca oferece suporte com API (*Application Programming Interface*) que agrega objetos gráficos podendo ser ferramentas GUI como Tkinter, wxPython, Qt ou GTK +, utilizada neste trabalho para obtenção de gráficos com os dados advindos dos datasets para investigação externa.

(vi) **Tkinter:** Biblioteca que realiza a ligação entre o *python* e as ferramentas GUI, utilizado nesta aplicação em conjunto com a biblioteca matplotlib para amostragem de gráficos.

(vii) ***Pickle*:** Biblioteca que serve como ferramenta de serialização de objetos, ou seja, persistência dos mesmos na memória não volátil do sistema, para que esses objetos possam ser usados posteriormente. Foi utilizada neste projeto para a permanência de listas e dos objetos treinados pela técnica de IC.

(viii) **Numpy:** Biblioteca com atributos para manipulação matemática mais facilitada, com suporte a vetores e matrizes, das mais diferentes dimensões, utilizada neste trabalho em conjunto com a biblioteca pandas para operações sobre os *datasets* e estatísticas.

(ix) **sys, os:** Bibliotecas desenvolvidas na linguagem C e python para manipulação de dados do sistema, provendo acessos a dados como I/O, dados de arquivos, etc, utilizada também para aprimorar e incentivar a portabilidade de programas desenvolvidos em *python*, usada neste projeto para obtenção de dados advindos do terminal como parâmetros e dados sobre arquivos, como por exemplo a existência dos mesmos.

3.2 Metodologia

A metodologia de desenvolvimento da presente proposta consistiu em realizar o projeto em fases bem definidas, cada qual com um propósito que, se atendido, irá permitir a execução do projeto com êxito. As próximas subseções descrevem tais etapas.

3.2.1 Preparação e Projeto

Para preparação e feitoria de um plano de trabalho simplificado, foi necessário passar por algumas etapas, que englobam desde o estudo em bases de dados de técnicas de IC para entendimento, passando pela a definição de uma técnica a ser utilizada, estudo das bibliotecas externas necessárias, implementação da técnica e testes experimentais feitos, tudo isso será detalhado nas seções posteriores. Mas primeiramente, foi necessário realizar a preparação do ambiente para implementação.

Como mencionado na seção 3.1.2, para confecção do *software* foi preciso fazer a instalação de algumas bibliotecas necessárias para o comprimento dos objeitivos do projeto, todas elas foram feitas através de comandos dados pelo terminal do linux, sistema operacional já citado na tabela 1. Todo o processo pode ser melhor visto e explicado a seguir:

(i) **Pip:** A maioria das bibliotecas precisam da instalação do gerenciador de pacotes da linguagem *python* chamado pip para instalação facilitada das demais bibliotecas. A instalação desse gerenciador é feita desta maneira:

```
$ sudo apt-get install python-pip
```

(ii) **Sklearn:** A instalação biblioteca para o uso do algoritmo inteligente também é feita por intermédio do gerenciador de pacotes pip e é feita da seguinte maneira:

```
$ pip install -U scikit-learn
```

(iii) **Pandas:** Como comentado na seção de *softwares* utilizados para utilização dos *datasets* foi necessária a instalação da biblioteca Pandas, feita da seguinte maneira:

```
$ pip install pandas
```

(iv) **Matplotlib:** Já a instalação da biblioteca que cria os gráficos para exibição dos dados é feita da seguinte maneira:

```
$ pip install matplotlib
```

(v) **Tkinter:** A instalação desta biblioteca, se o sistema linux não possuir já instalada originalmente, para uso de interface gráfica para exibição dos gráficos é feita da seguinte maneira:

```
$  sudo apt-get install python-tk
```

(vi) As demais bibliotecas não necessitam de ser instaladas externamente, basta somente importá-las no projeto, da seguinte maneira:

```
1|import Nome_Biblioteca
2|import Nome_Biblioteca2
```

3.2.2 Estudo das técnicas de IC

A fase de estudo das técnicas de IC, já descritas no referencial teórico, foi desenvolvida com intuito de entender melhor o que o estado da arte diz sobre o uso das técnicas, e como elas estão sendo empregadas na detecção de anomalias da rede. Além disso foram feitos estudos preliminares, não reportados neste documento, sobre as técnicas de IC.

Para os estudos feitos, foram pesquisadas bases de dados tais como: *Scopus* e *Springerlink* através do portal da Capes [1], e o *Google Scholar*[2], onde se é possível encontrar artigos e livros especializados no assunto tema deste trabalho. Para seleção dos materiais a serem estudados, além de pesquisas no *Google Scholar*, foi utilizada uma ferramenta disponível no portal da Capes, que possibilita o *download* de arquivos do tipo ".xlsx" ou ".csv" que contêm todos os dados relevantes individuais de cada trabalho, dados como: nome do material, autor(es), DOI, *link* para acesso, etc. Isso tudo no caso do estado da arte sobre detecção de anomalias com uso de IC, porém foi necessário também o estudo teórico das técnicas de IC e o estudo das bibliotecas existentes na linguagem *python* escolhida para implementação, com isso foi possível definir a técnica que realmente seria utilizada no projeto.

[1] ttp://www.capes.gov.br/
[2] https://scholar.google.com.br/

3.2.3 Definição da técnica IC a ser utilizada

Após o estudo e levantamento de algumas técnicas de IC, uma investigação de cunho qualitativo foi realizada confrontando cada técnica estudada em busca de selecionar aquela que melhor se adaptaria para solucionar o problema proposto.

A escolha foi feita de maneira empírica, onde foram experimentadas as técnicas Redes Neurais Artificiais, *Support Vector Machines* e *Random Forests*. As técnicas foram experimentadas em um conjunto de dados contendo 20 arquivos de *log* com variáveis ligadas à nós de rede, contendo mais de 90 mil registros coletados de 10 em 10 segundos. Nos experimentos preliminares realizados, não reportados neste documento, observou-se que o método que melhor foi capaz de aprender sobre o conjunto de dados realista foi o *Random Forest*. Logo, este método foi adotado para realizar experimentação mais massiva com estratégia de votação em um procedimento que será melhor explorado adiante.

3.2.4 Implementação

A implementação foi feita por meio de uma metodologia de duas frentes movidas por dois aplicativos desenvolvidos na linguagem *python* com auxílio de bibliotecas especializadas. A primeira está relacionada às operações ligadas a análise sob a desigualdade de Chebyshev. Esse aplicativo contém todas as operações estatísticas, além da própria análise embasada no Teorema de Chebyshev, para identificar quais são os padrões ou 'assinaturas' de anomalia nos conjuntos de dados de entrada. Ela inclui a geração de gráficos advindos dos cálculos realizados e um *HTML* com dados para entendimento da investigação feita. A segunda frente é baseada nos aplicativos que realizam todas as funções ligadas ao algoritmo de IC, esse que por sua vez também ao final da execução gera gráficos e um arquivo *HTML* que possui todos os dados analisados e tuplas consideradas anômalas destacadas, além de algumas informações relevantes para o entendimento da predição feita. Mais detalhes sobre a implementação estarão nas seções subsequentes. Toda a implementação está disponível para trabalhos futuros gratuitamente na plataforma *GitHub*[3] a partir do *link*: https://github.com/HenriqueArouca/Uma-Proposta-de-Detecao-de-Anomalias-na-Rede-com-Uso-de-Estatistica-e-Inteligencia-Computacional.git.

3.2.5 Projeto Experimental

Nesta fase, experimentos visando validar a prova de conceito construída foram realizados. A experimentação foi dividida em dois tipos: Experimentos pilotos, visando calibrar parâmetros, e experimentos massivos, visando mostrar o quão robusto a metodologia proposta é para detecção de anomalias. No primeiro tipo de experimento, que diz respeito aos experimentos pilotos, foram feitos testes de acurácia testando arquivos

[3] https://github.com/

advindos da análise feita através da desigualdade de Chebyshev e passando-os para análise no algoritmo de IC, sempre usando os mesmos arquivos para treino e uso, para verificação de acertos e erros cometidos pelo algoritmo inteligente, porém variando-se a quantidade de decisores em uma estratégia que será melhor detalhada adiante inspirada no conceito de *ensemble*, ou seja, de um conjunto de decisores cujas opiniões são combinadas para retornar uma saída única. Já o segundo tipo, que diz respeito aos experimentos massivos, foram feitos teste com tamanhos de *ensembles* diferentes, nos quais era usado uma metodologia de conjuntos para separação, dos arquivos de dados que serão utilizados nas análises de Chebyshev e o treinamento do algoritmo de IC dos arquivos de dados que fariam parte do conjunto de teste e classificação realizados pelo *Random Forest*. Maiores detalhes sobre ambos os tipos de experimentos são dados a seguir, na seção de desenvolvimento.

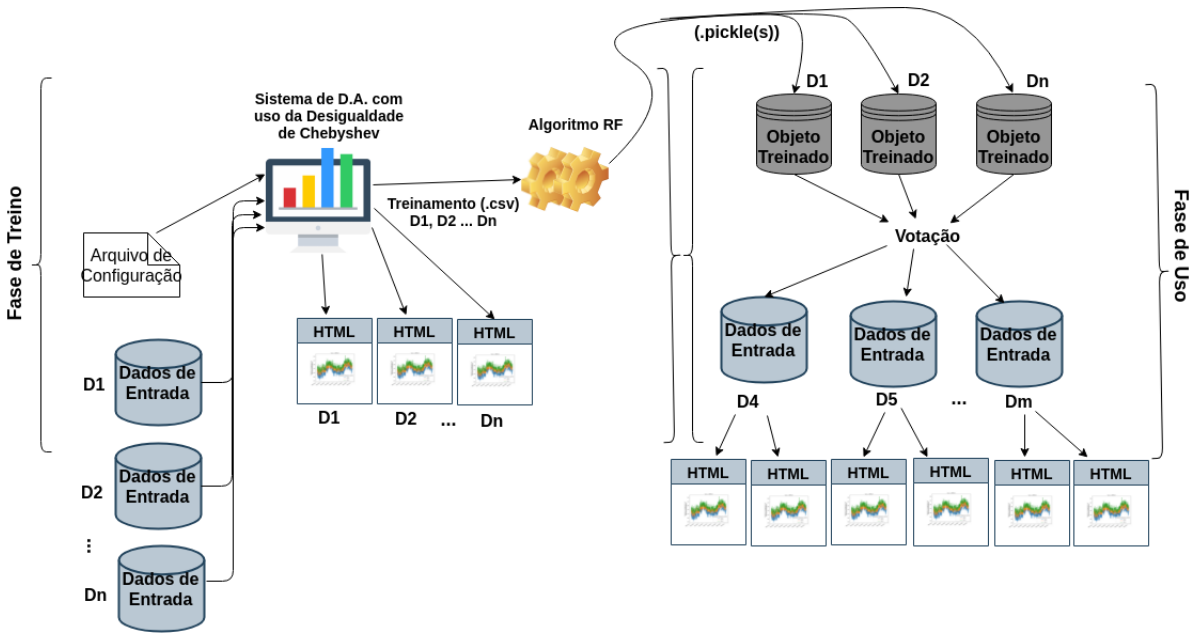
4 DESENVOLVIMENTO

Nesta seção serão detalhados todos os passos seguidos na confecção do projeto e a implementação, o que foi feito e como foi feito, desde a arquitetura do sistema, aquisição do *dataset*, passando pela parametrização dos métodos de IC e por fim os testes realizados no sistema.

4.1 Arquitetura da Solução

A arquitetura do sistema foi dividida em duas frentes, sendo a primeira a fase de treino da técnica de inteligência computacional *Random Forest* e a segunda sendo a fase de uso dos objetos treinados para caracterização de outros *datasets*. O ilustrativo da arquitetura pode ser vista na Figura 13:

Figura 13 – Arquitetura do Sistema.



Fonte: Elaborado pelo autor

Para definição da arquitetura deste projeto, esquematizada na Figura 13, foi escolhida uma metodologia de votação composta por duas fases, sendo elas: (i) fase de treino; e (ii) fase de uso. A primeira das fases consiste em um subsistema que realiza a escolha aleatória de conjuntos para a separação dos *datasets* que seriam usados em cada uma das fases, além de estabelecer o comitê de votação na segunda delas. Na primeira fase ainda, é realizado um pré-processamento dos dados escolhidos para o conjunto de

treino. Esse processamento consiste em uma classificação de *outliers*, neste caso ditos como suspeitas de anomalias, feita mediante uma análise estatística executada com uso da desigualdade de Chebyshev, considerando média e k vezes o desvio padrão dos dados em uma janela de tempo de 10 horas, onde k é o parâmetro de sensibilidade que varia para cada variável do *dataset*. A classificação de suspeita de anomalias é feita por maioria de votos do comitê que contêm todas as variáveis do *dataset*. Posteriormente, esses dados pré-analisados são usados como entrada para o treinamento do algoritmo *Random Forest*, criando assim o segundo comitê de votações, comitê esse que é acionado na fase de uso. Já a segunda das fases consiste basicamente em colocar à prova o classificador gerado, confrontando-o com os *datasets* do conjunto de uso sob a análise do comitê de *Random Forests*, com o critério de maioria de votos para o consenso sobre anomalia. Portanto, utilizam-se comitês em três situações distintas: (i) nas variáveis do *dataset*, analisando tupla a tupla, para indicar suspeita de anomalias; (ii) nas 10 (dez) *Random Forests* geradas a partir de cada *dataset* do comitê de treinamento; e (iii) no comitê de *Random Forests*, para confirmar que todas as florestas acreditam haver anomalias para um dado advindo de um *dataset* de uso.

4.2 Diagrama de Classes

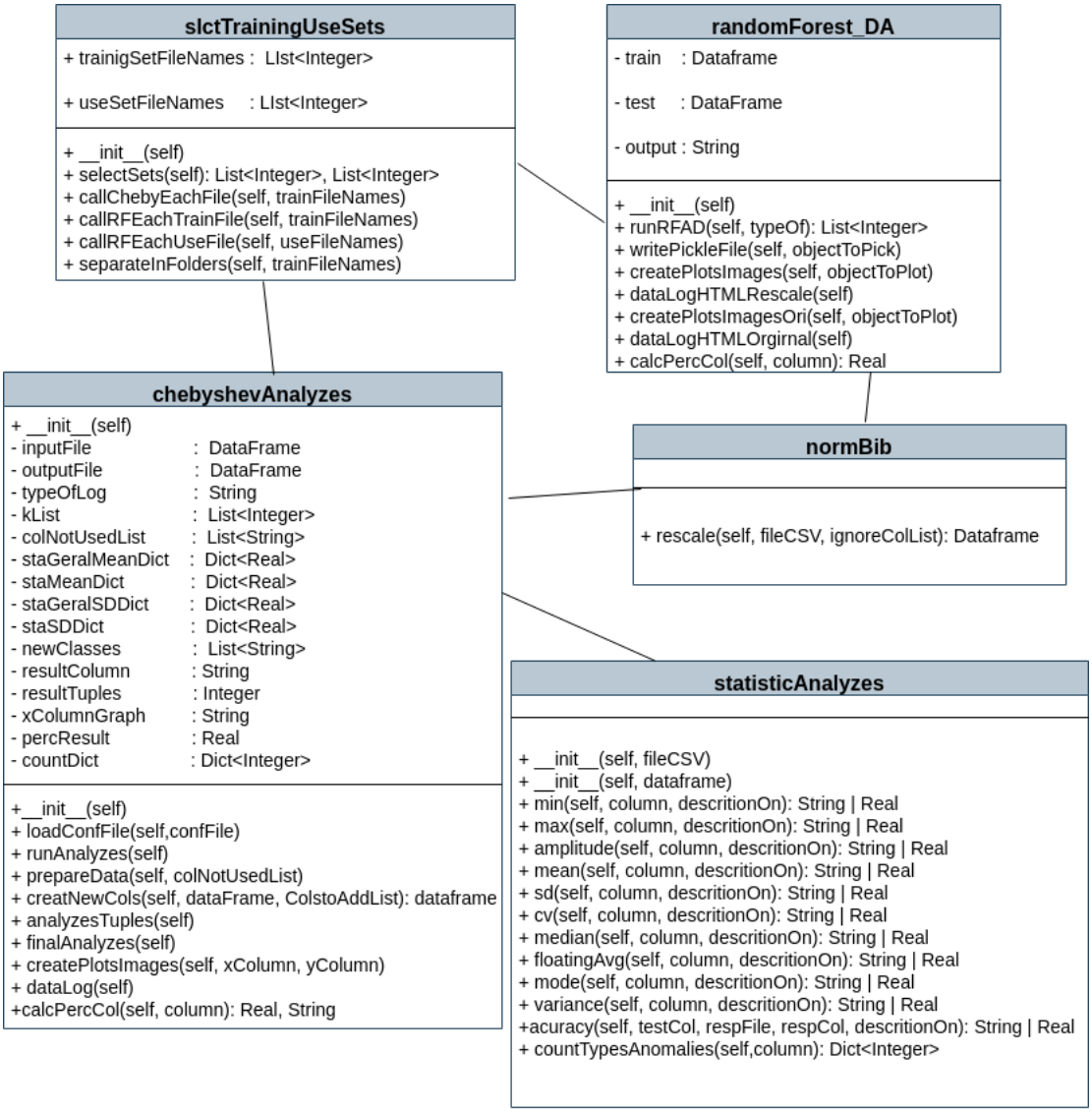
Para o desenvolvimento deste sistema foi necessário o planejamento e execução de um projeto bem estruturado no que diz respeito a separação das classes, sempre com a preocupação para com os objetivos que deveriam ser cumpridos. O diagrama de classes feito pode ser visto na Figura 14. Cada classe será detalhada posteriormente.

Como dito anteriormente as classes foram separadas a fim de facilitar e organizar o projeto, cada classe tem o seu papel específico definido da seguinte maneira:

(i) **slctTrainingUseSets:** Localizada no arquivo “slctTrainingUseSets.py”, essa classe tem como função verificar os arquivos de *datasets* presentes na pasta raiz do projeto. Com essa informação em mãos, as funções da classe selecionam aleatoriamente dentro do conjunto universo de *datasets*, subconjuntos de treino e uso dos arquivos com dados para análise. Após essa seleção, o algoritmo realiza todas as chamadas, para cada conjunto, dos algoritmos de Chebyshev e *Random Forest* tanto para treino quanto para uso e organização da pasta raiz.

(ii) **chebyshevAnalyzes:** Localizada no arquivo “chebyshevAnalyzes.py”, essa classe tem como função realizar todos os cálculos e análises necessárias, após o carregamento e normalização de um arquivo de *dataset*, segundo a desigualdade de Chebyshev, utilizada na investigação de *outliers*, que nesse caso são considerados anomalias na rede. Após a análise, mediante média móvel (com janela de 8640) mais/menos k

Figura 14 – Diagrama de classes.



Fonte: Elaborado pelo autor

multiplicado pelo desvio-padrão, feita pelo algoritmo para cada coluna (variável) do *dataset*, uma última análise é realizada por meio de votação, em que se uma certa porcentagem das variáveis (colhida via arquivo de configurações) de uma determinada tupla for considerada *outlier*, então a linha inteira é dita anômala. Por fim, gráficos com dados anômalos destacados, e um *log*, podendo ser uma página *html* ou texto puro, com todas as informações, tais como média, desvio-padrão, coeficiente de variação, k utilizado, porcentagem e número de tuplas anômalas, porcentagem de anomalia em cada variável, é tida como saída.

(iii) **randomForest_DA:** Localizada no arquivo "randomForest_DA.py", essa classe tem como função executar, após o carregamento e normalização de um arquivo

presente nos conjuntos de treino e uso, às atividades de treinamento do algoritmo *Random Forest* e executar a análise mediante essa técnica. A definição de qual função será cumprida depende do argumento dado na chamada do algoritmo. Na fase de uso é feita uma votação dentre uma porcentagem de *datasets* selecionados, na classe "slctTrainingUseSets", escolhidos aleatoriamente para treino, e se a maioria simples (50% + 1) do votos, para cada tupla, chegarem a conclusão de anomalia, a linha é considerada anômala. Ao fim da execução, duas páginas *html* são tidas como saída, nessas páginas irão conter gráficos com os dados e situações consideradas anômalas destacadas além de a porcentagem e número de tuplas anômalas. O primeiro *html* corresponde aos dados normalizados e o segundo aos dados crus.

(iv) **normBib:** Localizada no arquivo "normBib.py" essa classe tem como função realizar cálculos para a reescala dos dados em um *dataset* carregado, ou seja, para cada dado do arquivo é feito o seguinte procedimento: NovoDado = dadoAtual / max(coluna) - min(coluna). Sendo assim, o dado atual assumirá seu próprio valor dividido pela amplitude da coluna que ele corresponde.

(v) **statisticalAnalyzes:** Localizada no arquivo "statisticalAnalyzes.py", essa classe tem como função realizar todos os cálculos ligados a análises estatísticas, em que muitos deles serão usados na classe "chebyshevAnalyzes". Exemplos de cálculos que podem ser feitos mediante um determinado *dataset* e coluna, passados como parâmetro para cada função, são: média, desvio-padrão, coeficiente de variação, valores mínimo e máximo, amplitude, mediana, moda, média móvel (flutuante), etc. Sendo que, todos os métodos possuem duas maneiras de saída, a primeira sendo a descritiva e a segunda por sua vez sendo a numérica, dependendo do parâmetro que a função receber.

4.3 Formato de Entrada de Dados

O *software* em questão possui dois tipos arquivos que servem como entrada, o primeiro sendo um arquivo de configurações, este somente utilizado no pré-processamento estatístico baseado na desigualdade de Chebyshev, o qual utiliza uma estratégia de variáveis e valores que dentro dos algoritmos possui um papel importante, como por exemplo: Os Ks utilizados nos cálculos de Chebyshev, nome da coluna resultante da análise estatística, coluna do *dataset* que será o "x" no gráfico de saída, coluna que se deseja ignorar nas análises (neste caso a coluna de tempo (*timestamp*)), e por fim, como se trata de uma metodologia baseada em votação para classificação, é necessário que seja informado a porcentagem de votos para que uma tupla do *dataset* seja considerada anômala. Um exemplo do arquivo de configurações pode ser visto na Figura 15:

Figura 15 – Arquivo de configuracoes

```
################################################
#                                              #
#      K multiplier for each column in order   #
#          without the ignored colum           #
#                                              #
################################################

# First Column
K 3.5
# Second Column
K 3.5
# Third Column
K 3.5
# Fourth Column
K 3.5
# Fifth Column
K 3.5
# Sixth Column
K 3.5
# Seventh Column
K 3.5


################################################
#                                              #
#           Column % for analysis,             #
#               example 0.5                    #
#                                              #
################################################

P 0.5
```

Fonte: Elaborado pelo autor

Já o segundo tipo de arquivos de entrada, este que é utilizado tanto na análise de Chebyshev quanto na análise da técnica de IC, são os próprios *datasets* comentados no parágrafo anterior, os quais possuem todos o mesmo padrão e dados relevantes que são utilizados tanto no treinamento quanto nos testes do *software* desenvolvido. Para melhor entendimento da organização do arquivo o dataset será detalhado na seção seguinte.

4.4 *Datasets* Utilizados

Para o desenvolvimento deste projeto foram utilizados arquivos do tipo csv (*Comma-separated values*), em que cada arquivo representa dados coletados durante 14 dias de máquinas espalhadas pelo mundo, com uso do programa Planetlab. Os arquivos que compõem o conjunto de dados são, portanto, reais, e foram utilizados em outro trabalho. Para detalhes de sua aplicação em outro contexto, ver Valadão (2009).

O arquivo segue o padrão de que cada tupla, ou seja, uma linha do *dataset* representa uma leitura feita. Cada leitura é realizada em um intervalo de dez segundos e a primeira das tuplas representa o cabeçalho contendo o nome de cada variável. As colunas do *dataset* por sua vez, representam as grandezas coletadas, tais como: O tempo

em milissegundos em que a coleta foi feita (*timestamp*), porcentagem de uso de CPU de 0% a 100% (cpu%), volume de processamento da máquina (entre 0 e 1.0 a máquina consegue atender à demanda de processamento, acima de 1.0 começa a ocorrer enfileiramento) a cada um, cinco e quinze minutos (load[1, 5, 15]) que representa uma razão entre os processos no estado "pronto" e aquele(s) "em execução" no(s) processador(es), porcentagem de uso de memória de 0% a 100% (mem%) e taxas de *download* (rx) e *upload* (tx) em *bits* por segundo (bps). Ao todo foram utilizados 80 arquivos para o treinamento e o uso e cada arquivo em média possui cerca de 100 mil tuplas, um exemplo do *dataset* pode ser visto na figura 16.

Figura 16 – *Dataset* utilizado.

	-e timestamp	cpu%	load_1	5	15min	mem%	rx	tx(bps)
2	1343942289589	53.8	0.95	0.7	0.62	0.04	0.0	0.0
3	1343942300647	53.8	0.96	0.71	0.62	0.06	0.0	0.0
4	1343942311679	53.8	0.96	0.72	0.63	0.08	0.0	0.0
5	1343942322711	53.8	0.97	0.73	0.63	0.01	0.0	0.0
6	1343942333747	53.8	0.98	0.74	0.64	0.02	0.0	0.0
7	1343942344783	53.8	0.98	0.75	0.64	0.03	0.0	0.0
8	1343942355816	53.8	2.18	1.01	0.72	0.04	0.0	0.0
9	1343942366849	53.8	2.15	1.04	0.74	0.05	0.0	0.0
10	1343942377881	53.8	1.98	1.04	0.74	0.06	0.0	0.0
11	1343942388901	53.8	1.76	1.03	0.74	0.07	0.0	0.0
12	1343942399935	53.8	1.64	1.03	0.74	0.08	0.0	0.0
13	1343942410967	53.8	1.86	1.1	0.77	0.01	0.0	0.0
14	1343942421998	53.8	1.88	1.13	0.78	0.02	0.0	0.0
15	1343942433017	53.8	1.75	1.12	0.78	0.02	0.0	0.0
16	1343942444049	53.8	1.58	1.11	0.78	0.03	0.0	0.0
17	1343942455081	53.8	1.49	1.11	0.79	0.04	0.0	0.0
18	1343942466114	53.8	1.41	1.11	0.79	0.05	0.0	0.0
19	1343942477158	53.8	1.35	1.1	0.79	0.06	0.0	0.0
20	1343942488191	53.8	1.35	1.11	0.8	0.07	0.0	0.0

Fonte: Elaborado pelo autor

4.5 Parâmetros Experimentais

Para a configuração e utilização do sistema de detecção de anomalias desenvolvido foi necessário realizar a escolha de alguns parâmetros relevantes. Os argumentos escolhidos são usados na análise de Chebyshev e no algoritmo *Random Forest* na função que cria o classificador (*RandomForestClassifier*), presente na biblioteca "*sklearn.ensemble.RandomForestClassifier*". Os parâmetros podem ser vistos na Tabela 2.

Tabela 2 – Grandezas e parâmetros utilizados para calibrar implementação e biblioteca *sklearn.ensemble.RandomForestClassifier* do python.

#	Parâmetro	Onde foi Usado	Valor
1	k	Análise de Chebyshev	2.0
2	n_estimators	Random Forest	10
3	warm_start	Random Forest	TRUE
4	criterion	Random Forest	gini

Fonte: Elaborado pelo autor

A escolha dessas grandezas foram feitas mediante testes de acurácia e pesquisas realizadas na literatura especializada, além de análises feitas com base nas saídas gráficas geradas pelo *software*. Esses parâmetros são de suma importância para a replicação do trabalho realizado, pois eles são responsáveis e estão diretamente ligados aos resultados obtidos.

De maneira independente é possível entender essa importância, sabendo o que cada variável representa: O k usado no pré-processamento realizado pela análise com desigualdade de Chebyshev, como explicado no referencial teórico, é responsável por ditar a quantidade de desvios padrões que serão somados a média para separação de *outliers* dos demais dados. Já o parâmetro "*n_estimators*" no algoritmo de IC *Random Forest* possui o papel de representar o número de árvores na floresta (esse argumento diz respeito a uma variável inteira e possui como valor *default* dez). O parâmetro "*warm_start*" corresponde a uma variável booleana e tem como papel, quando setada com o valor verdadeiro, a reutilização da solução da chamada anterior para a ajustar e adicionar mais estimadores ao conjunto. Por fim o parâmetro "*criterion*", cujo possui dois tipos de valores específicos com a função de medir a qualidade de uma divisão de cada árvore. Os critérios suportados são "gini"para a impureza Gini e "entropia"para o ganho de informação.

4.6 Experimentos Realizados

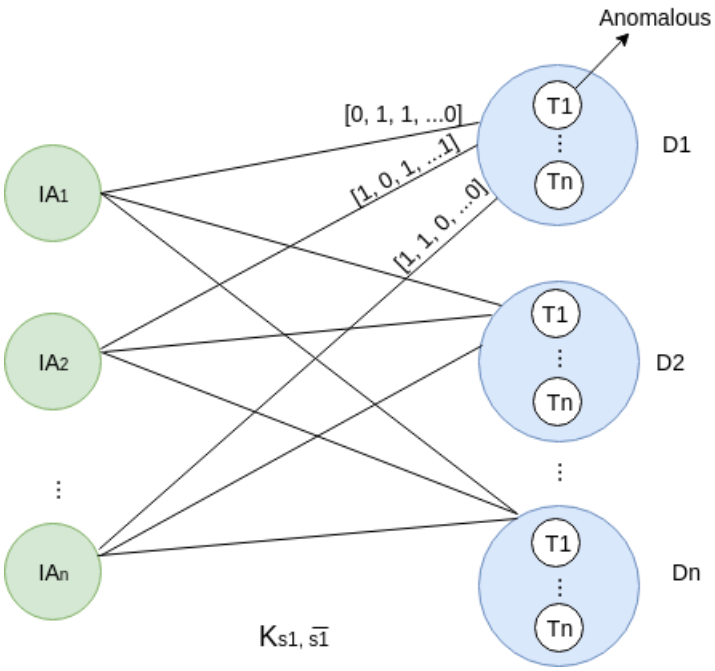
Para validação do projeto alguns experimentos e testes foram necessários. Para tal investigação, foi desenvolvida uma metodologia que usa estratégia de conjuntos, com objetivo de selecionar e separar os dados que serão treinados pela técnica de IC daqueles que serão usados pelo algoritmo *Random Forest*. Todos os cenários possuem um comportamento geral de um grafo bipartido, que modela cada experimento da seguinte maneira: cada IA do conjunto de treino (lado esquerdo do grafo) realiza uma votação para cada *dataset* do conjunto de uso (lado direito do grafo). Essa votação é armazenada em uma estrutura de lista que representa a contagem de votos para cada tupla do *dataset* em análise, ao final da execução esses votos são contabilizados. Sempre que a posição da lista correspondente a uma tupla estiver com maioria simples dos votos, então a tupla é considerada anômala. A estrutura do grafo bipartido pode ser vista na Figura 17.

Para melhor detalhamento, a fase de experimentação foi dividida em três cenários, descritos nas subseções a seguir.

4.6.1 Cenário I

Para o primeiro *ensemble*, cujo o conjunto universo U é de 20 *datasets* de entrada (arquivos de 1 a 20), foi selecionado aleatoriamente um subconjunto S de dez arquivos para treino. Esses arquivos são passados por uma pré-análise feita pela desigualdade de

Figura 17 – Grafo bipartido de comportamento dos cenários



Fonte: Elaborado pelo autor

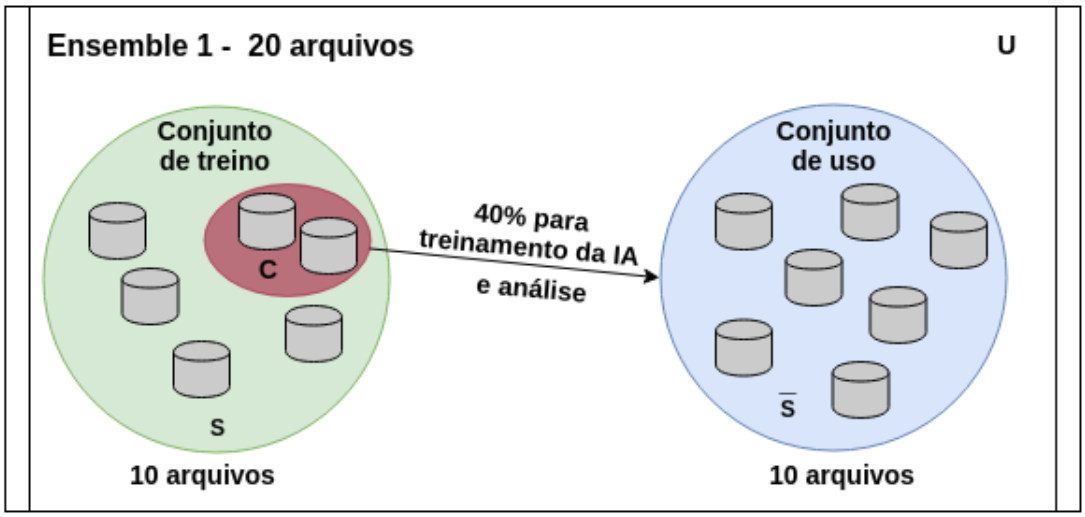
Chebyshev; e posteriormente, um subconjunto C, também selecionado aleatoriamente; de 40% desses arquivos são usados como *inputs* para o treinamento do *Random Forest*. O restante dos outros dez arquivos, que fazem parte do subconjunto de complemento de S, são usados nos testes da técnica de IC. Um esquemático do *ensemble* pode ser visto na Figura 18.

Já as Tabelas 3 e 4 representam os arquivos de *datasets* das máquinas presentes em cada conjunto, treino e uso respectivamente. Elas contêm informações necessárias para o entendimento do experimento. Nas tabelas são descritos: (i) Identificador de contagem de arquivos (#); (ii) máquina alvo da análise; (iii) número de tuplas presentes no *dataset* da máquina correspondente. A Tabela 3 que mostra o conjunto de dados de treinamento estão destacados em negrito e itálico aquelas máquinas que foram selecionadas para serem os *inputs* para o treinamento na *Random Forest*, dentro dos 40% escolhidos do *ensemble*.

4.6.2 Cenário II

Para o segundo *ensemble*, cujo o conjunto universo U é de 20 *datasets* de entrada (arquivos de 1 a 20), foi selecionado aleatoriamente um subconjunto S de 10 arquivos para treino. Esses arquivos são passados por uma pré-análise feita pela desigualdade de Chebyshev; e posteriormente, um subconjunto C também selecionado aleatoriamente, de 30% desses arquivos, são usados como *inputs* para o treinamento do *Random Forest*. O

Figura 18 – Cenário I.



Fonte: Elaborado pelo autor

Tabela 3 – Conjunto de treino - *Ensemble* 1.

#	Máquina	Número de Tuplas
1	***3***	***101004***
2	5	91004
3	6	101009
4	***8***	***90584***
5	10	101128
6	***11***	***90908***
7	12	91051
8	13	90641
9	***15***	***101101***
10	18	91054

Fonte: Elaborado pelo autor

restante dos outros 10 arquivos, que fazem parte do subconjunto de complemento de S, são usados nos testes da técnica de IC. Um esquemático do *ensemble* pode ser visto na Figura 19.

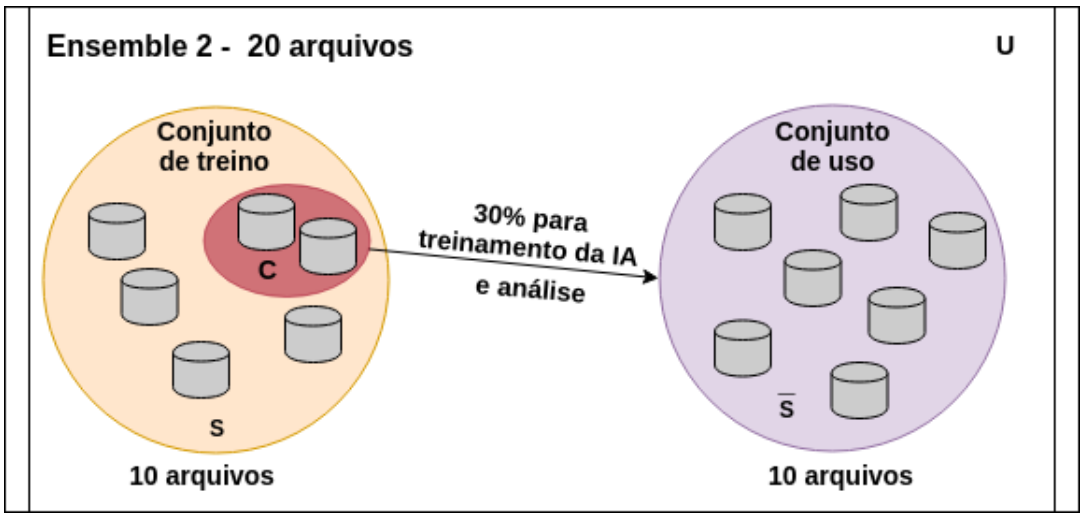
Assim como no *ensemble* 1, este ***ensemble* 2** realiza todas as análises e classificações nos *datasets* presentes nos mesmos conjuntos de treino e uso listados nas Tabelas 3 e 4, porém com a diferença da quantidade de máquinas votantes que neste caso é igual a 30% (3 Máquinas) do conjunto de treino. Para o experimento deste cenário escolheu-se os *datasets* referentes às Máquinas 8, 12 e 18.

Tabela 4 – Conjunto de uso - *Ensemble* 1.

#	Máquina	Número de Tuplas
1	1	91034
2	2	101130
3	4	101078
4	7	90920
5	9	101136
6	14	91335
7	16	91050
8	17	101025
9	19	101062
10	20	91004

Fonte: Elaborado pelo autor

Figura 19 – Cenário II.



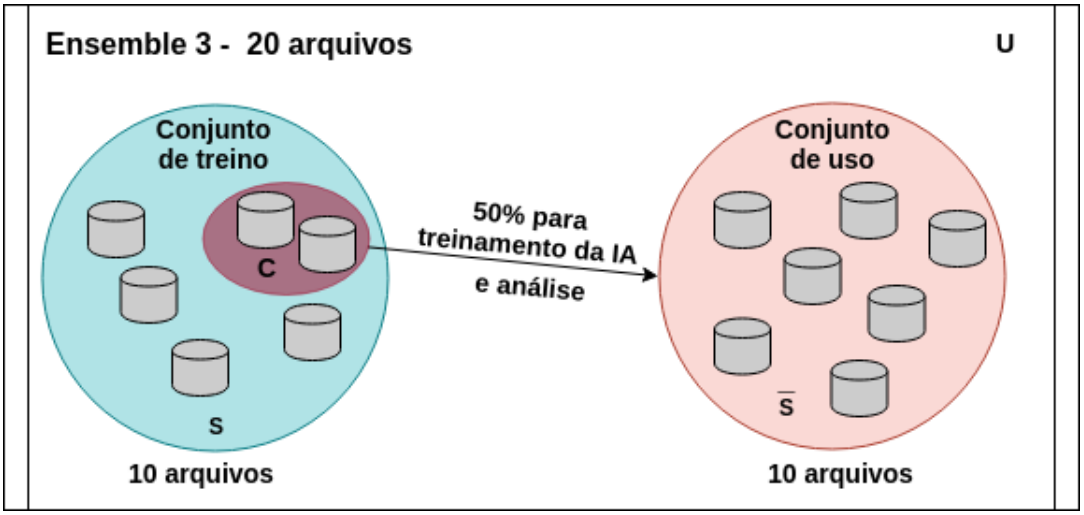
Fonte: Elaborado pelo autor

4.6.3 Cenário III

Para o terceiro e último *ensemble*, cujo o conjunto universo U é também de 20 *datasets* de entrada (arquivos de 1 a 20), foi selecionado aleatoriamente um subconjunto S de 10 arquivos para treino. Esses arquivos são passados por uma pré-análise feita pela desigualdade de Chebyshev; e, posteriormente, um subconjunto C, também selecionado aleatoriamente, de 50% desses arquivos, são usados como *inputs* para o treinamento do *Random Forest*. O restante dos outros 10 arquivos, que fazem parte do subconjunto de complemento de S, são usados nos testes da técnica de IC. Um esquemático do *ensemble* pode ser visto na Figura 20.

Assim como nos dois *ensembles* anteriores, o ***ensemble* 3** possui os mesmos

Figura 20 – Cenário III.



Fonte: Elaborado pelo autor

conjuntos de treino e uso listados nas Tabelas 3 e 4, porém possui o número de votantes no comitê de *Random Forests* igual a 50% (5 Máquinas) do conjunto de treino. Neste cenário, os *datasets* escolhidos foram os arquivos correspondentes às medições feitas para as Máquinas 3, 5, 8, 10 e 18.

4.6.4 Outros Experimentos

Diversos outros experimentos foram realizados a fim de realizar testes de prova do algoritmo e até de outras técnicas de IC, pelo motivo de escolher a técnica que mais se encaixa e que tem melhores resultados mediante a classificação de dados anômalos e não anômalos, sempre visando a diminuição de falsos positivos e falsos negativos. O principal teste secundário feito foi aquele que diz respeito a técnica de IC denominada SVM. Nesses testes foram experimentados arquivos de diversos tamanhos e variedades, além do teste de prova da técnica, onde coloca-se o arquivo usado no treinamento como entrada no uso. Como os resultados não foram promissores, seja pela dificuldade de ajustar seus parâmetros, seja pela dificuldade do próprio método em operar sobre um *dataset* grande e real, o autor deste trabalho optou por não registrar os testes realizados, ficando como trabalhos futuros investigar melhor o método SVM à luz de sua aplicação em detecção de anomalias usando o *dataset* escolhido para este trabalho.

5 RESULTADOS E ANÁLISE

Nesta seção serão apresentados todos os resultados obtidos por todos os cenários propostos na seção anterior, bem como análises detalhadas qualitativas e quantitativa sobre cada um deles.

5.1 Cenário I

Conforme mencionado, o **Cenário I** possui um conjunto de 20 arquivos sendo 10 usados na análise primária de Chebyshev, dos quais 4 deles (40%) foram tomados e usados como entrada para o treinamento do *Random Forest*. A Tabela 5 apresenta os resultados obtidos com a aplicação da análise de Chebyshev sobre o conjunto de dados correspondente à Máquina 3, que faz parte do conjunto C escolhido. Nela são apresentados (i) o identificador da variável do *dataset*, (ii) o nome da coluna no *dataset*, correspondente à variável; (iii) o valor da média geral obtida a partir das observações da referida variável; (iv) o desvio padrão amostral das observações da referida variável; (v) o coeficiente de variação que mede a variabilidade relativa dos valores observados; (vi) o parâmetro de sensibilidade K usado na desigualdade de Chebyshev para indicar, variável por variável, quais observações estão fora dos limiares aceitáveis, quando considerando o valor observado acima ou abaixo de $\overline{x} \pm K \cdot s$ quando considerando-a em relação aos valores observados da mesma janela móvel (i.e., em relação à 10 horas de amostragem); (vii) a quantidade de tuplas consideradas anômalas por estarem fora dos limiares calculados; (vii) o percentual de tuplas anômalas, em relação ao total de tuplas da mesma variável.

Tabela 5 – Tabela Resultante da análise de chebyshev.

#	***Column***	***Mean*(x̄)**	**SD**	**CV**	***K***	**# *Anomalous***	***Anomalous*(Per/Var)**
0	Col.cpu%	0.84946	0.08295	0.09765	2.0	6319	6.25371%
1	Col.load_1	0.31899	0.17592	0.5515	2.0	8154	8.06975%
2	Col.load_5	0.42981	0.18466	0.42964	2.0	8715	8.62496%
3	Col.load_15	0.512	0.19684	0.38446	2.0	9521	9.42263%
4	Col.mem%	0.71312	0.18443	0.25863	2.0	10589	10.47959%
5	Col.rx	0.22597	0.26242	1.16131	2.0	7018	6.94549%
6	Col.tx	0.24427	0.27417	1.12238	2.0	7903	7.82135%

Fonte: Elaborado pelo autor

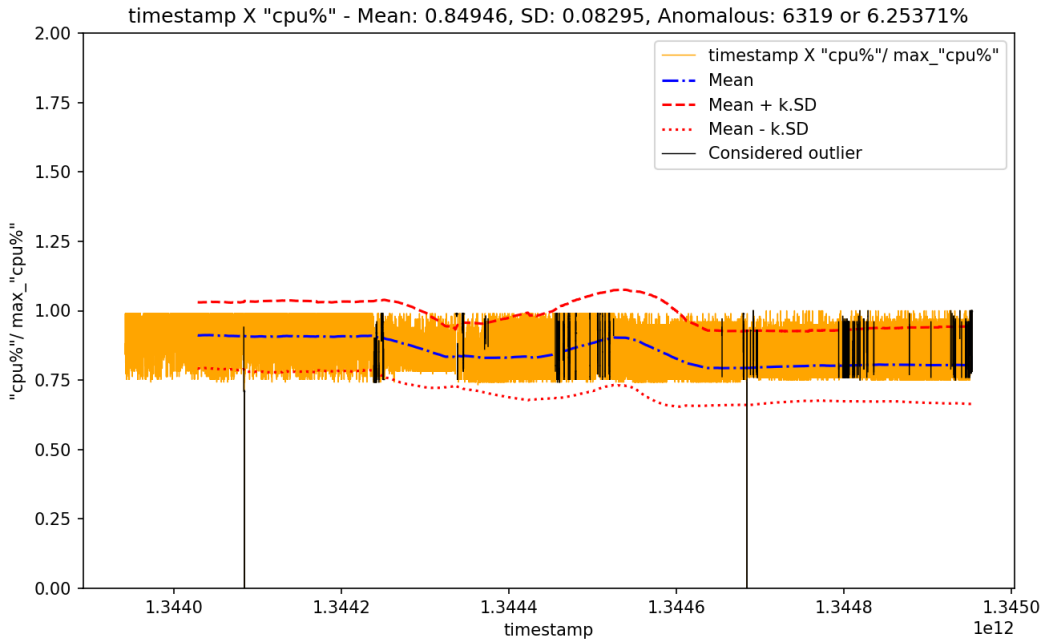
Antes de analisar os dados contidos na Tabela 5, cabe uma consideração. As colunas "*Anomalous*" e "*Anomalous (Per/Var)*" referem-se aos totais absolutos e relativos de tuplas que foram indicadas pela análise de Chebyshev como anômalas quando consideradas apenas dentro das janelas móveis de 10 horas, isto é, considerando a média móvel e desvio padrão

amostral dos dados dentro da janela móvel. Desta forma os percentuais observados devem ser considerados à luz desta condição, sendo incomparáveis com os valores esperados da mesma amostra com média global e desvio padrão global. Isto posto, os percentuais observados estiveram entre 6.2% e 10.4%, aproximadamente. É preciso que fique claro que estes valores foram computados considerando-se apenas cada variável de maneira independente.

Ao considerarmos todas as variáveis em conjunto, os resultados se alteram. Seja o caso onde a tupla é considerada anômala apenas quando a maioria de suas variáveis possuem valores observados com suspeita de anomalia. Desta forma, convencionou-se uma tupla é anômala se $50\% + 1$ de suas variáveis estiver indicando suspeita de anomalia. Ao computar o percentual de tuplas consideradas anômalas após a votação, para a Máquina 3 em questão observou-se que este percentual cai para 2.65033%, ou 2678 tuplas do total de 101044. Este resultado é de maior interesse pois mostra que usando a metodologia de votação sobre as tuplas com variáveis suspeitas reduz em muito a quantidade de falsos positivos que poderiam estar sendo gerados se a análise da tupla considerasse apenas cada variável em separado.

Para ter uma melhor compreensão sobre o processo, sejam as Figuras 21, 22, 23, 24, 25, 26, 27. Cada uma delas apresenta o comportamento e tendências das medidas "cpu%", "*load1*", "*load5*", "*load15*", "mem%", "rx", "tx", que correspondem, respectivamente, às variáveis percentual de uso da CPU, carga da CPU no último minuto, carga da CPU nos últimos 5 minutos, carga da CPU nos últimos 15 minutos, percentual de uso da memória, taxa em *bits*/segundos de recepção dos dados, e taxa em *bits*/segundos da transmissão de dados. Em cada caso, o gráfico contém os valores observados de cada uma destas grandezas ao longo do tempo. No eixo X estão caracterizados os tempos de amostragem, espaçados de 10 em 10 segundos, ao longo de 14 dias, aproximadamente. No eixo Y estão caracterizados os valores de cada grandeza, oscilando entre 0 e 1 devido à normalização (reescala linear) aplicada sobre os dados para que cada grandeza tenha seus valores observados entre 0 (para o menor valor do *dataset* original) e 1 (para o maior valor do *dataset* original). Em laranja estão os valores observados. A linha em azul ilustra a variação da média amostral dentro da janela móvel, isto é, corresponde ao valor da média móvel para janelas de 10 horas cada. Em vermelho, são apresentados os limites para considerar uma variável suspeita, computados por meio do parâmetro K de sensibilidade, do valor da média amostral móvel, e do valor do desvio padrão amostral da janela móvel. Qualquer medição realizada acima da linha vermelha indica tupla e variável com suspeita de anomalia. Desta forma, a linha vermelha atua como um *upper bound* e *lower bound* sobre os limites aceitáveis sem suspeita de anomalia. Ainda nestas figuras existe uma marcação especial, em cor preta, para as tuplas que foram consideradas anômalas de *facto* após a votação. Foram caracterizadas em preto observações onde, no mesmo instante de tempo, observou-se suspeita de anomalia em mais da metade das variáveis da tupla.

Figura 21 – Gráfico de CPU% resultante da análise de chebyshev.



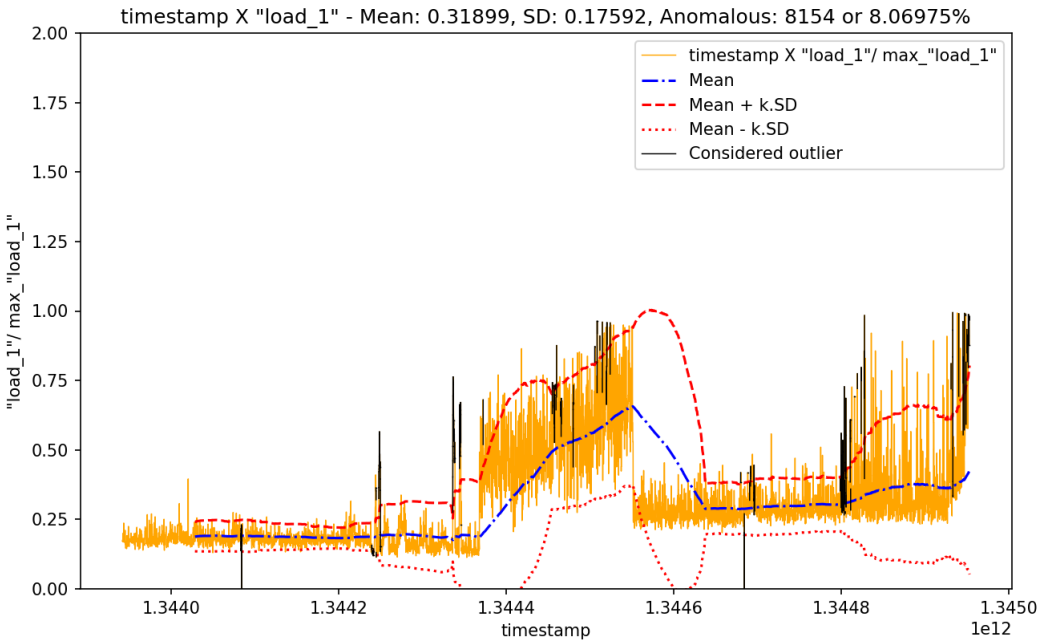
Fonte: Elaborado pelo autor

Para a Máquina 3, objeto desta discussão, nota-se que a variável percentual de uso da CPU exposta na Figura 21 não teve grandes variações observadas. Entretanto, em alguns instantes de tempo (próximos das UTC (*Universal Time Coordinated*) de valores 1.3443, 1.3445 e 1.3448 e 1.3449) houve marcação de tuplas anômalas, marcadas em cor preta. Embora em alguns casos já havia suspeita de anomalia considerando apenas o percentual de uso da CPU, a confirmação sobre a anomalia ocorreu apenas nas proximidades destes intervalos devido à votação considerando todas as variáveis. Somente pela CPU seria difícil detectar prováveis anomalias. Logo, a explicação ficará mais clara conforme novas variáveis sejam apresentadas ao leitor.

Na Figura 22, que mostra a evolução da carga da CPU no último minuto ao longo dos 14 dias, nos mesmos valores de UTC apresentados no caso anterior existe confirmação de anomalia. Nota-se, entretanto, que no caso anterior as variações ocorreram, mas sozinhas não foram suficientes para confirmar a anomalia; já neste caso as confirmações ocorreram na maioria dos casos onde houve uma variação abrupta nos valores observados da carga em relação às observações anteriores. Isso nos permite conjecturar que a variável carga da CPU no último minuto é, provavelmente, uma das variáveis mais impactantes para a confirmação da anomalia visto que a "assinatura" da anomalia identificada confere com as variações de suas leituras.

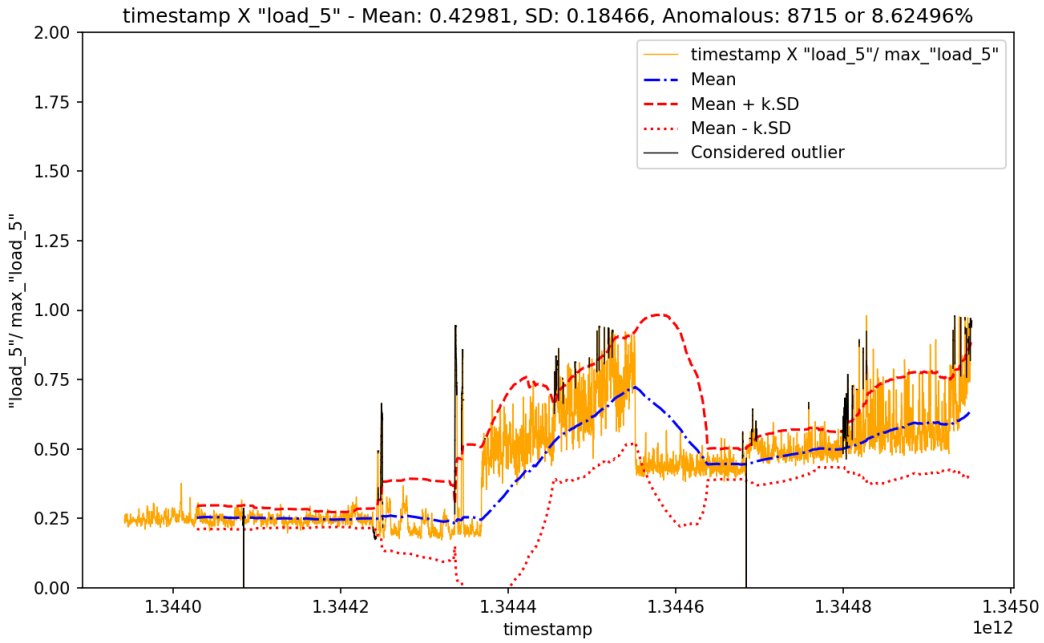
Já na Figura 23, pode-se observar a carga nos últimos 5 minutos de CPU da Máquina 3 durante o intervalo de coleta de 14 dias. Segundo o resultado disponível

Figura 22 – Gráfico de *Load*(1) resultante da análise de chebyshev.



Fonte: Elaborado pelo autor

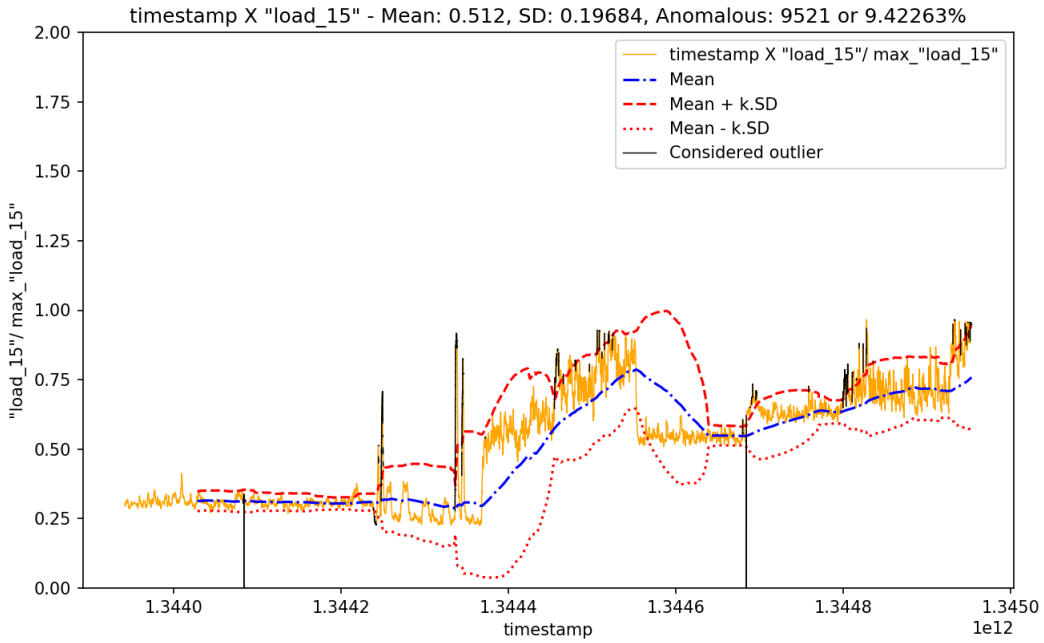
Figura 23 – Gráfico de *Load*(5) resultante da análise de chebyshev.



Fonte: Elaborado pelo autor

em visão gráfica, algumas tuplas foram consideradas anômalas nos mesmos intervalos apresentados na Figura 22, definido por maioria de votos sobre as suspeitas de anomalia de cada variável das referidas tuplas. Por ser uma medida análoga à variável anterior, diferenciando-se apenas o período usado para consolidar o uso (antes de 1 minutos, agora de 5 minutos), observa-se que as anomalias foram novamente registradas em instantes de tempo onde houveram mudanças bruscas nos valores observados, normalmente nos picos.

Figura 24 – Gráfico de *Load*(15) resultante da análise de chebyshev.

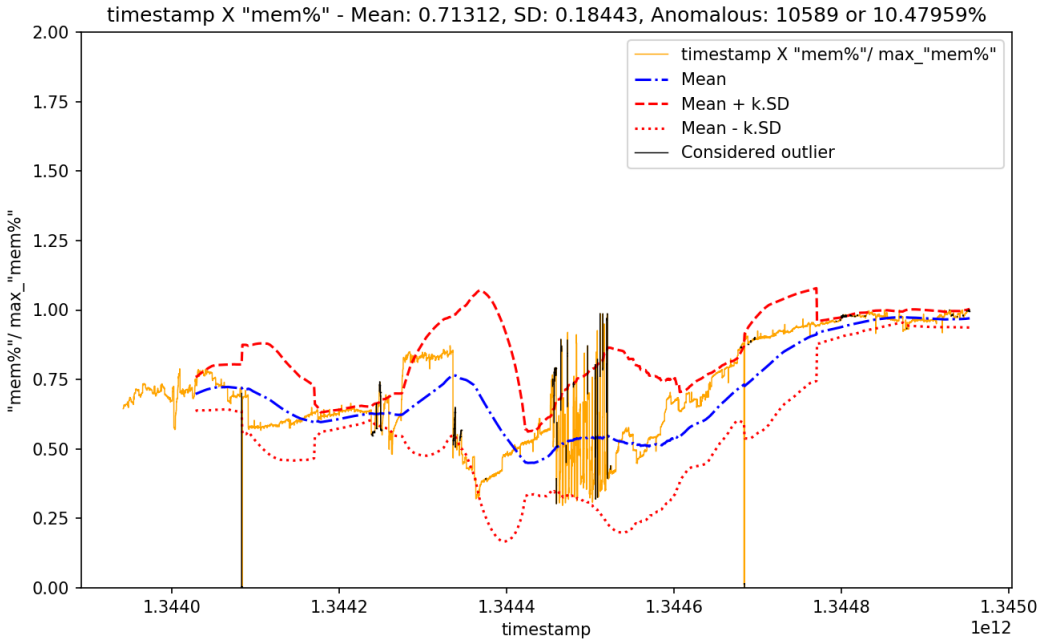


Fonte: Elaborado pelo autor

Assim como às Figuras 21, 22 e 23, a figura 24, que representa os dados referentes a carga de CPU no intervalo dos últimos 15 minutos, também é de fácil percepção que anomalias foram apontadas nos mesmos intervalos de tempo UTC, comprovando que para os dados analisados e os padrões reconhecidos pela desigualdade de Chebyshev as tuplas as quais esses intervalos de tempo correspondem são anômalas.

A Figura 25 representa o gráfico referente aos dados de porcentagem de memória no período de coleta de 14 dias, também possui os mesmos valores de *timestamp* UTC assinalados como anomalias na cor preta pelas variáveis. Como a confirmação de anomalia ocorre devido à votação, é esperado que cada tupla anômala apresente indicação de anomalia sempre no mesmo instante de tempo para as diversas variáveis analisadas. Neste caso em especial observa-se que, um dos picos de memória, ocorrido ao redor do instante de tempo 1.3443, não foi capturado como assinatura de anomalia. Isso ocorre porque apesar do pico ter sido observado para essa variável, a variância dos dados na janela de tempo correspondente determinou um limiar de suspeita, segundo a desigualdade de

Figura 25 – Gráfico de percentual de memória (mem%) resultante da análise de chebyshev.



Fonte: Elaborado pelo autor

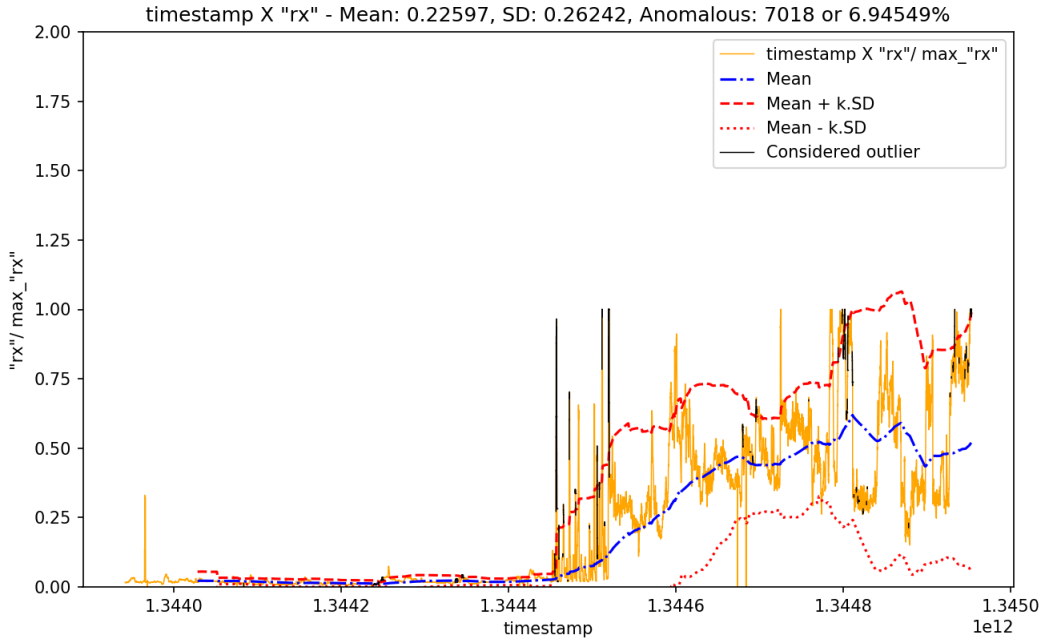
Chebyshev, mais alto que a própria leitura feita da memória neste instante (ver linha vermelha tracejada). Desta forma a variável não foi indicada como suspeita, e não deu parecer favorável na votação correspondente.

Assim como os demais gráficos já discutidos, o gráfico presente na Figura 26 possui um padrão de assinatura que possui algumas variações referentes aos dados de *download* em *bits*/segundo da Máquina 3, aparentemente ocorridas na transição de uma leitura de valor mais baixo para uma leitura consecutiva de valor mais alto. As variações ocorridas nesta variável, que também foram observadas em outras variáveis já discutidas e que excedem o limiar calculado pela desigualdade de Chebyshev, foram confirmadas via votação, e encontram-se destacadas no gráfico. As variações que excedem o limiar mas não foram destacadas não foram confirmadas por tratar-se de uma variação local, apenas nesta variável, ou em apenas algumas das demais variáveis em quantidade menor do que 50% + 1. É o caso, por exemplo, das tuplas que encontram-se na localidade da UTC 1.3446.

Semelhante às Figuras 21, 22 e 23, 24, 25, 26 a Figura 27 representa dados referentes de *upstream*, ou seja, dados de transmissão da Máquina 3 coletados durante 14 dias. Nos mesmos espaços de tempo UTC também foram consideradas e ditas por pelo menos a maioria simples das variáveis da determinada tupla em análise feita pela desigualdade de Chebyshev.

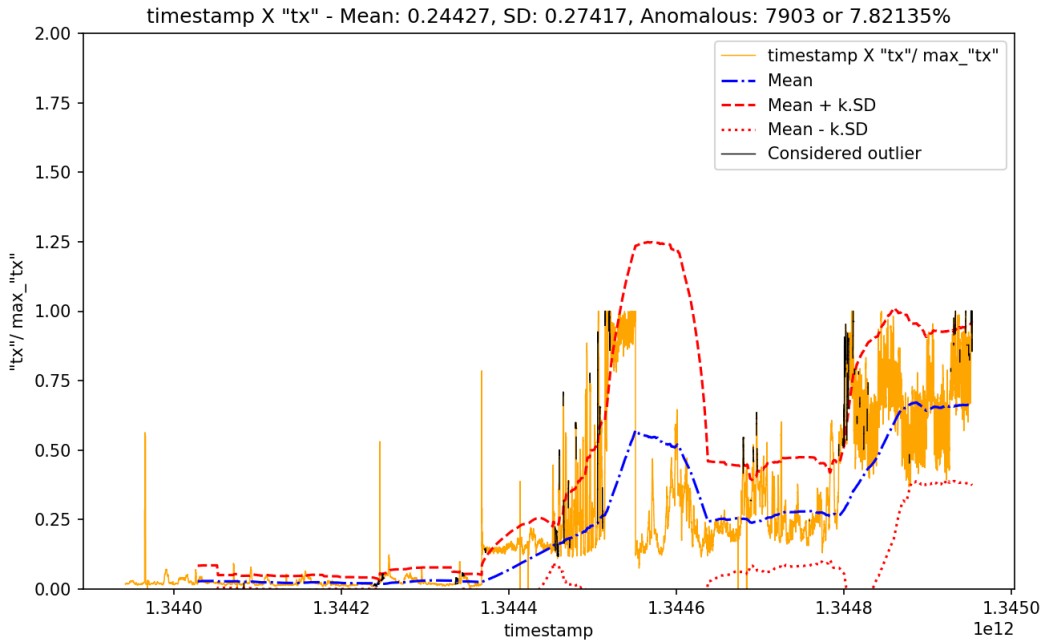
Os demais resultados dos datasets usados nos treinamentos estão presentes na

Figura 26 – Gráfico da taxa de recepção de dados em *bits*/segundo (rx) resultante da análise de Chebyshev.



Fonte: Elaborado pelo autor

Figura 27 – Gráfico de tx resultante da análise de chebyshev.



Fonte: Elaborado pelo autor

Tabela 6, em que se encontram dados relevantes para análise dos demais resultados obtidos da investigação e classificação feita pela análise de Chebyshev. Nesta tabela encontram-se dados como: (i) Identificador da máquina em análise. (ii) Número de tuplas correspondente ao *dataset* na máquina alvo. (iii) Número de tuplas anômalas classificadas pela desigualdade de Chebyshev. (iv) Porcentagem de anomalias encontradas referente ao total das tuplas analisadas.

Tabela 6 – Demais resultados para arquivos de treino.

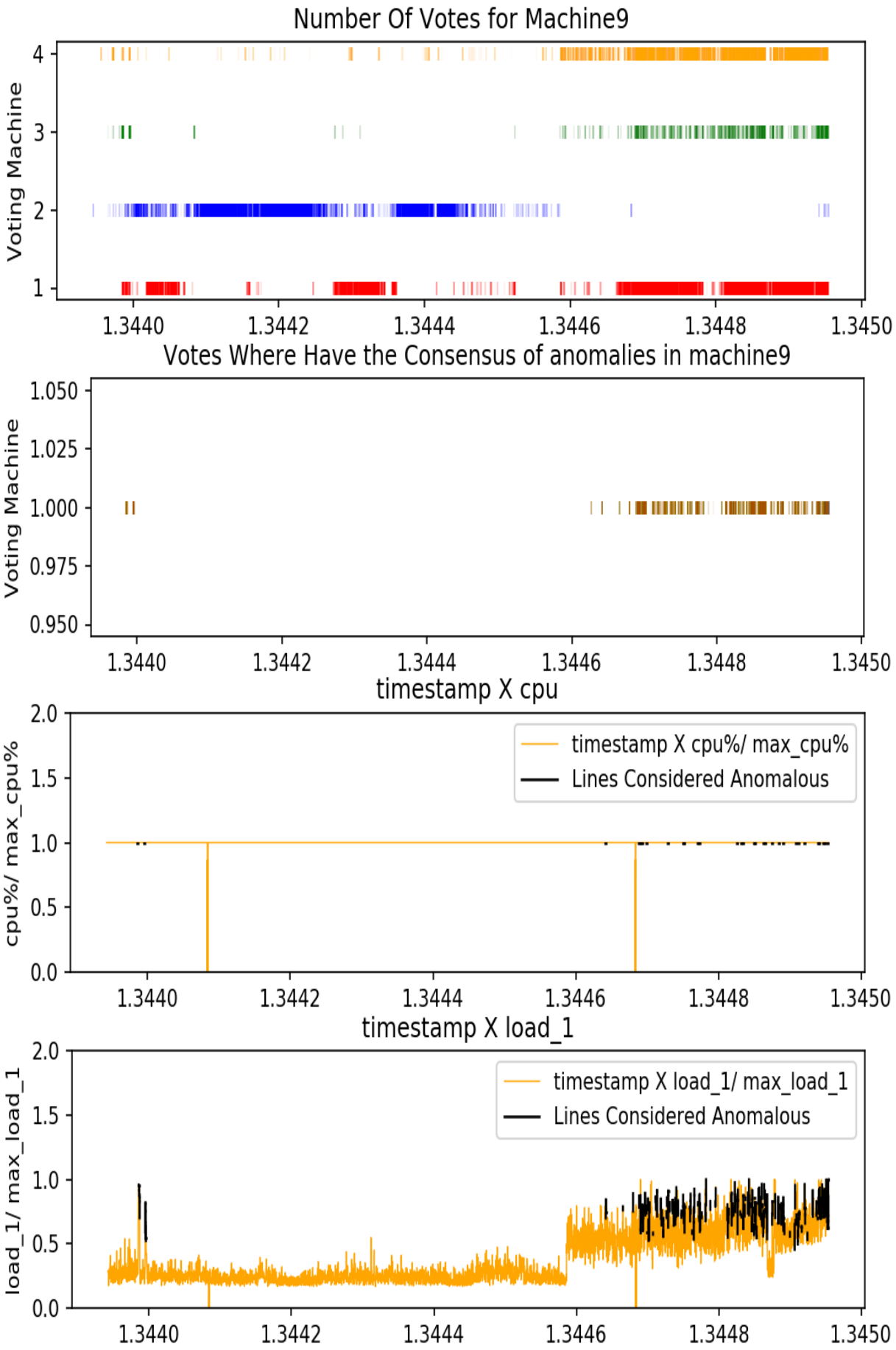
# Máquina	# Tuplas	# Tuplas Anômalas	% Anomalia
8	90584	302	0.33339%
11	90908	256	0.2816%
15	101101	1994	1.97229%

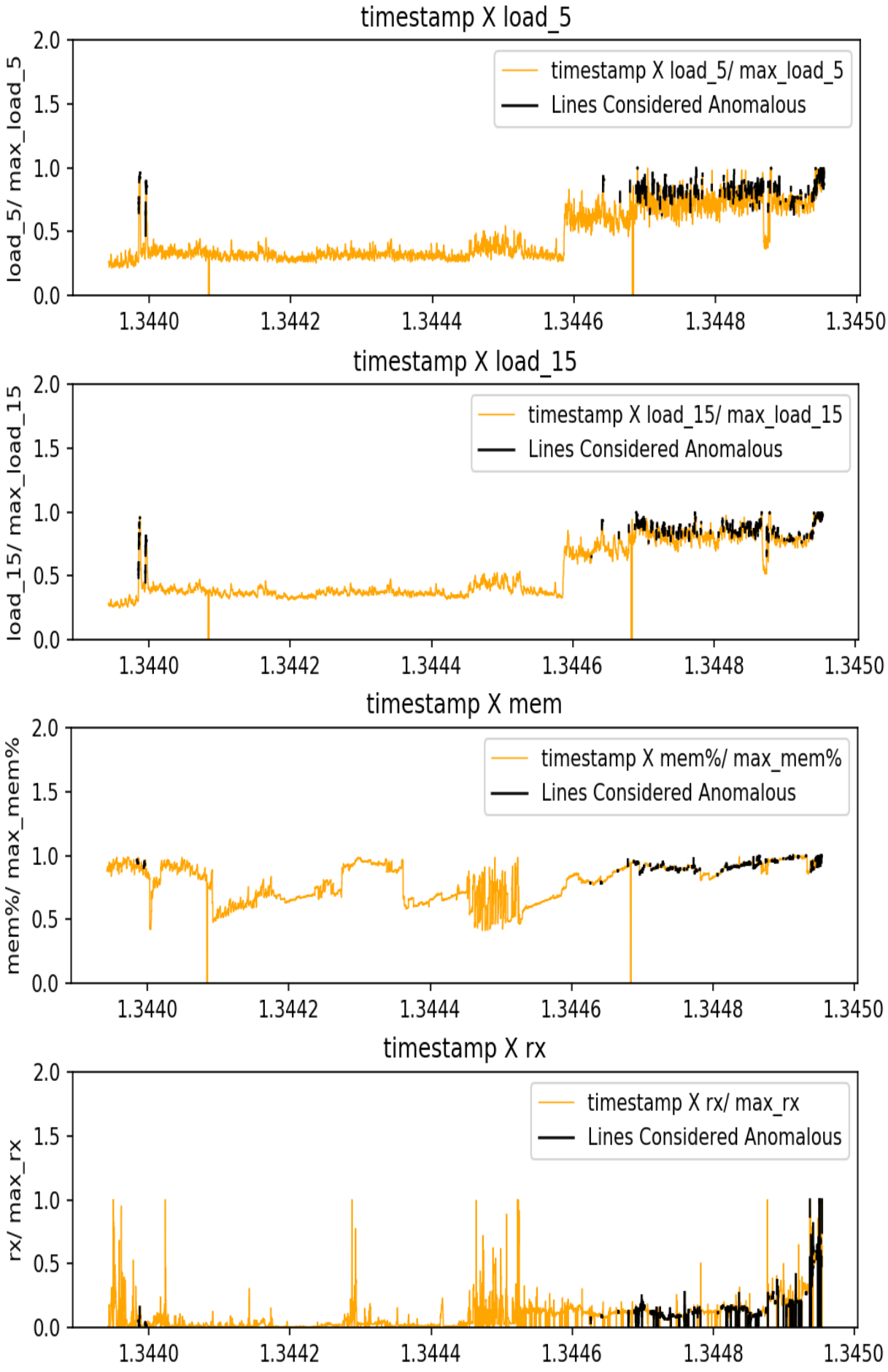
Fonte: Elaborado pelo autor

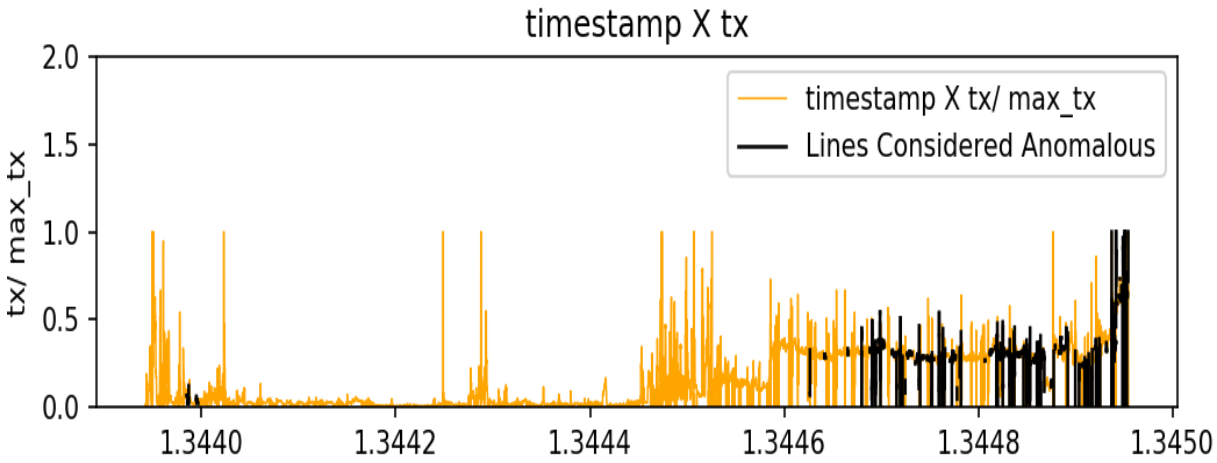
Na Tabela 6 descrita, onde se encontram as máquinas usadas como *input* para os treinamentos das *Random Forests*, percebe-se que na Máquina 8, dentre as 90584 tuplas 302 (0.33339%) delas foram caracterizadas com comportamentos fora do normal. Já na Máquina 11 do total de 90908 tuplas 256 (0.2816%) delas foram ditas anômalas. Por fim na Máquina 15 que ao todo possui 101101 tuplas, 1994 (1.97229%) delas foram classificadas como anômalas segundo a desigualdade de Chebyshev.

Após cada procedimento semelhante a esse descrito, um arquivo ".csv" com as análises e votações é obtido. Este mesmo arquivo é usado como entrada para o treinamento do algoritmo *Random Forest*, que por sua vez gera objetos frutos dos treinamentos que são usados na avaliação dos demais *datasets* das máquinas remanescentes que fazem do conjunto de uso. Um dos arquivos examinados foi o referênte ao *dataset* da Máquina 9, cujos resultados estão expostos na 28. Ela é dividida em nove (9) gráficos com as seguintes informações: os gráficos "*Number of Votes for Machine9*" e "*Votes Where Have Consensus of anomlaies in Machine9*" possuem os votos de cada máquina participante do *ensemble*, totalizando em quatro votos para o cenário discutido. O primeiro destes dois gráficos representa o voto concedido por cada membro do comitê que forma o *ensemble*, e o segundo representa a consolidação dos votos quando houve consenso na maioria das máquinas do comitê, indicando uma possível anormalidade dos dados. Já os demais sete (7) gráficos representam os dados de cada variável, já comentadas anteriormente, advindas do *dataset* referente a Máquina 9 na cor laranja, com destaque de marcações na cor preta na sequência dos dados nos instantes de tempo em que foram indicadas anomalias mediante a votação do *ensemble*.

Figura 28 – Votos e gráficos resultantes das análises da *Random Forest*.




timestamp X load_5
load_5/ max_load_5
timestamp X load_5/ max_load_5
Lines Considered Anomalous
timestamp X load_15
load_15/ max_load_15
timestamp X load_15/ max_load_15
Lines Considered Anomalous
timestamp X mem
mem%/ max_mem%
timestamp X mem%/ max_mem%
Lines Considered Anomalous
timestamp X rx
rx/ max_rx
timestamp X rx/ max_rx
Lines Considered Anomalous
0.0
0.5
1.0
1.5
2.0
1.3440
1.3442
1.3444
1.3446
1.3448
1.3450



Fonte: Elaborado pelo autor.

Com a análise de todos os nove (9) gráficos percebe-se que uma certa porcentagem de anomalias foi diagnosticada com base no comitê formado pelas florestas aleatórias mediante classificação dos dados usando os objetos treinados de cada floresta. No primeiro dos gráficos ("*Number of Votes for Machine9*"), que representa os votos dados por cada Máquina presente no *ensemble*, é possível analisar que nos momentos um pouco antes do tempo UTC 1.3440 e no intervalo UTC de 1.3446 à 1.3450 aproximadamente, é tido um consenso entre as florestas confirmando assim possíveis anomalias. Isso pode ser melhor analisado no segundo dos gráficos ("*Votes Where Have the Consensus of Anomalies in Machine9*"), que representa os momentos onde as Máquinas do *ensemble* concordaram sobre a votação, nos mesmos instantes de tempo.

No gráfico **timestamp x CPU**, que representa os dados de porcentagem de CPU da Máquina 9 colhidos durante 14 dias, a classificação de anormalidade dos dados se torna mais fácil de perceber, pois nos mesmos espaços de tempo UTC assinalados pelos gráficos de votação, foram marcados com linhas pretas no gráfico de CPU indicando que naqueles espaços de tempo as Máquinas votantes indicaram anomalias. Em alguns outros momentos, como por exemplo no tempo UTC 1.3447 é perceptível que houve uma mudança abrupta nos padrões de dados, porém essa mudança não foi assinalada pelas *Random Forests*, pois nesses momentos não houve consenso entre as Máquinas presentes no ensemble sobre anomalia.

Os quarto, quinto e sexto gráficos representam dados colhidos sobre a carga de processos da Máquina 9 (***timestamp* x *load*_[1,5,15]**), nos últimos 1, 5 e 15 minutos. Nesses gráficos é possível perceber que nos mesmos momentos de UTC apresentados pelos gráficos de votação, 1.3440 e no intervalo 1.3446 à 1.3450, houve a confirmação e a marcação com linhas pretas nos gráficos referentes a essas variáveis. Isso ocorre porque, na prática, essas três variáveis estão correlacionadas, embora meçam o efeito da carga da CPU em janelas de tempo de diferentes tamanhos. Isto posto, o efeito sendo mais

forte na janela de 1 minutos, e mais fraco na janela de 15 minutos, mas se o padrão é perceptível, ele o será em todos os casos, com maior ou menor 'força'. E da mesma forma que na variável porcentagem de CPU, as variáveis de *load* possuem algumas variações não assinaladas, mas isso é devido a falta de consenso entre as Máquinas votantes.

Por fim, nos últimos três gráficos referente aos dados de porcentagem de memória (***timestamp* x Mem**), taxa de *download* (***timestamp* x rx**) e *upload* (***timestamp* x Mem**) em *bits*/segundo, coletados por 14 dias, tudo que foi dito nos parágrafos anteriores também se aplica, pois nos tempos de UTC 1.3440 e no intervalo 1.3446 à 1.3450, é possível ver que foram assinalados anomalias confirmadas por maioria simples das Máquinas presentes no comitê, e da mesma forma algumas áreas de tempo não foram marcadas pois o consenso não ocorreu.

Os demais resultados dos *datasets* restantes, que formam junto com a máquina 9 analisada o conjunto de dados de uso/teste, podem ser encontrados na Tabela 7. Nela estão contidos os seguintes dados, coluna a coluna: (i) Identificador da Máquina cujo *dataset* foi analisado; (ii) número de tuplas presentes no arquivo de *dataset*; (iii) número de tuplas consideradas anômalas pela análise feita pelo algoritmo *Random Forest* com consenso do comitê *ensemble*; e (iv) porcentagem de anomalias computadas a partir da razão entre o total de tuplas anômalas e o total tuplas analisadas.

Tabela 7 – Demais resultados das nove máquinas restantes no conjunto de uso, para o *ensemble* 1, após a análise das *Random Forests*.

# Máquina	# Tuplas	# Tuplas Anômalas	% Anomalia
1	91034	2	0.0022%
2	101130	196	0.19381%
4	101078	281	0.278%
7	90920	999	1.09877%
14	91335	12	0.01314%
16	91050	2832	3.11038%
17	101025	747	0.73942%
19	101062	507	0.50167%
20	91004	49	0.05384%

Fonte: Elaborado pelo autor

5.2 Cenário II

Semelhante ao Cenário I, o **Cenário II** realiza as mesmas operações, seguindo os mesmos padrões explanados na subseção anterior, contando com a mesma seleção de *datasets* para os conjuntos de treino e uso. O que varia no caso do Cenário II em relação ao Cenário I é o tamanho do comitê de *Random Forests*. Como dito na seção 4.6.2 foram selecionados aleatoriamente 30% (3 *datasets*) para o treinamento da *Random Forest*. Neste caso foram escolhidos os arquivos referentes às Máquinas 8, 12 e 18 . Seguindo a mesma

linha de raciocínio, os dados dos arquivos presentes no conjunto de treino são passados por uma análise de Chebyshev, é gerado um comitê de *Random Forests*, para que os arquivos do conjunto de uso sejam posteriormente colocados à prova mediante a análise e classificação do referido comitê.

Após a conclusão do processo, os resultados obtidos foram reportados na Tabela 8, que contém todos os dados necessários para o entendimento. Os dados presentes nela são: (i) Identificador correspondente ao número da máquina em análise; (ii) número de tuplas presente no *dataset* em questão; (iii) número de tuplas anômalas apontadas pela votação dos algoritmos *Random Forest*; e (iv) porcentagem de anomalias assinaladas pela votação mediante ao total de tuplas que o *dataset* possui.

Tabela 8 – Resultados das 10 máquinas presentes no conjunto de uso, para o *ensemble* 2, após a análise das *Random Forests*.

# Máquina	**# Tuplas**	**# Tuplas Anômalas**	**% Anomalia**
1	91034	0	0.0%
2	101130	14610	14.44675%
4	101078	7989	7.9038%
7	90920	242	0.26617%
9	101136	2667	2.63704%
14	91335	0	0.0%
16	91050	661	0.72597%
17	101025	7123	7.05073%
19	101062	640	0.63327%
20	91004	1	0.0011%

Fonte: Elaborado pelo autor

Através da análise dos resultados presentes na Tabela 8 é possível entender que a metodologia desenvolvida baseada em comitês de votação varia seus resultados dependendo de quem e de quantos são os votantes. No caso deste *ensemble* de número dois, os resultados obtidos presentes na coluna "# Tuplas Anômalas" variam desde 0 (zero) tuplas consideradas anômalas até 14610 tuplas ditas anormais, sendo que a média dos resultados se encontra em torno de 3393 ocorrências de anomalia. Um número nulo de anomalias consideradas pode ocorrer por dois motivos aparentes. O primeiro deles é que pode não ter havido consenso entre o comitê de votos composto por três máquinas indicando anomalias nas linhas dos *datasets*. Já o segundo dos motivos diz respeito ao padrão para o qual o comitê de *Random Forest* foi treinado, ou seja, as máquinas escolhidas aleatoriamente para o treinamento reconhecem um tipo de padrão de normalidade e um outro tipo de padrão de anormalidade, logo se a entrada presente no conjunto de uso não contiver algo parecido há uma certa chance das máquinas no comitê não indicarem anomalias. De maneira semelhante é possível analisar a coluna de nome " % Anomalia" que diz respeito a porcentagem de anomalias comparadas ao total de tuplas presentes no *dataset* luz da análise. Os resultados nesta coluna variam entre 0.0% e 14.44675%, em que a média do

resultados é igual a 3.36600% de tuplas anômalas.

5.3 Cenário III

Assim como nos Cenários I e II, este **Cenário III** segue a mesma metodologia de uso. A seleção dos arquivos para cada conjunto (teste/uso) é feita conforme explicado em seções anteriores. Para o Cenário III foram escolhidos aleatoriamente 50% do conjunto separado para treinamento S (i.e., 5 *datasets*) , mais especificamente, os arquivos correspondentes às Máquinas 3, 5, 8, 10 e 18. Após a seleção dos conjuntos e subconjuntos, todas as demais operações foram repetidas, desde a passagem do conjunto de treino pela análise pela desigualdade de Chebyshev, passando pelo treinamento dos *datasets* selecionados para as *Random Forests*, e, por fim, a classificação feita mediante o conjunto de uso semelhante em todos os cenários. Seguindo como referência as colunas das Tabelas 7 e 8 apresentadas anteriormente nos demais cenários, a Tabela 9 possui os mesmos conjuntos de dados porém agora referentes ao *ensemble* 3.

Tabela 9 – Resultados das 10 máquinas presentes no conjunto de uso, para o *ensemble* 3, após a análise das *Random Forests*.

# Máquina	# Tuplas	# Tuplas Anômalas	% Anomalia
1	91034	5793	6.36356%
2	101130	2403	2.37615%
4	101078	20482	20.26356%
7	90920	261	0.28707%
9	101136	3789	3.74644%
14	91335	26	0.02847%
16	91050	1284	1.41021%
17	101025	12672	12.54343%
19	101062	977	0.96673%
20	91004	63	0.06923%

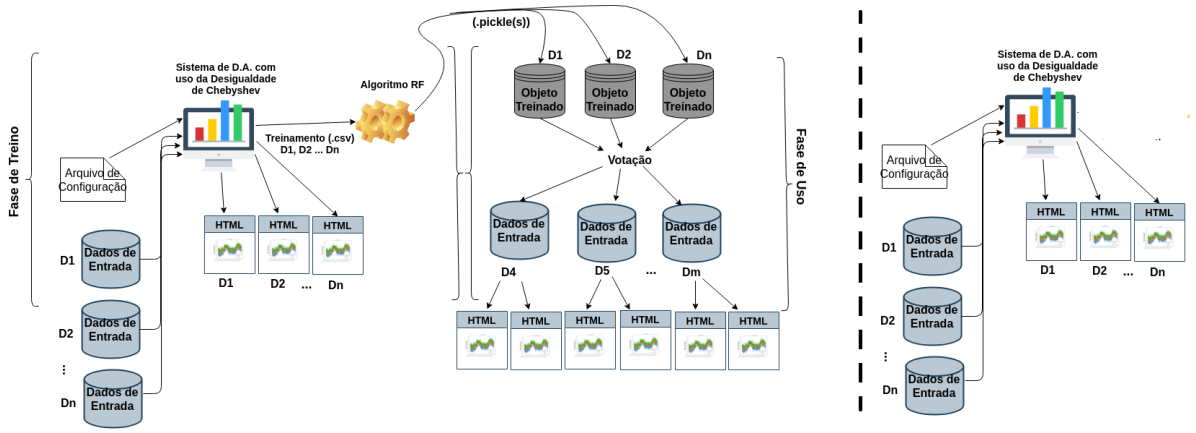
Fonte: Elaborado pelo autor

Mediante aos resultados obtidos após a conclusão do *ensemble* 3, é possível perceber que os valores referentes a coluna "# Tuplas Anômalas" (número de tuplas anômalas) tem seu valor mínimo igual a 26 (para o arquivo 14), e seu valor máximo igual a 20482 (para o arquivo 4), variando seus valores nesta amplitude para os demais arquivos do conjunto de uso, possuindo média de tuplas consideradas anômalas igual a 4775. De maneira similar, a coluna "% Anomalia" (porcentagem de anomalias) segue a variação de valores entre 0.02847% e 20.26356% e possui média de porcentagem de tuplas anômalas mediante ao total de tuplas dos *datasets* igual a 4.80500%.

5.4 Resumo

Para efeito de comparação e análise de qual dos *ensembles* testados foi aquele que obteve melhores resultados, estabeleceu-se neste trabalho que o entendimento sobre o que é considerado melhores resultados está relacionado com a indicação sobre qual tamanho de *ensemble* identifica o menor número de falsos positivos e negativos no que diz respeito a detecção de anomalias na rede. Para tal, foram realizados testes secundários para cada *ensemble* utilizado e reportado acima. Os testes secundários baseiam-se na comparação dos resultados obtidos pelo procedimento Determinação de Padrão usando Chebyshev + Comitê *Random Forests* + Aplicação no Conjunto de Uso, já reportados nos resultados dos Cenários I, II e III, com uma análise simples feita utilizando apenas a desigualdade de Chebyshev aplicada sobre o Conjunto de Uso.

Figura 29 – Arquitetura de comparações entre as analises de Chebyshev e a *Random Forest*.



Fonte: Elaborado pelo autor

Neste experimento confrontou-se, portanto, as anomalias detectadas usando o procedimento que considera o comitê de *Random Forests* com as anomalias detectadas usando apenas a desigualdade de Chebyshev. Para discutir a comparação e ter uma melhor visualização da diferença entre as duas análises, computou-se a variação do número de anomalias de ambos procedimentos via diferença dos resultados. A Tabela 10 possui os dados necessários para o entendimento dessa comparação. Ela possui as seguintes colunas: (i) Identificador correspondente a máquina analisada; (ii) porcentagem de anomalias apontados na análise feita pela desigualdade de Chebyshev para o referido arquivo presentes no conjunto de uso; (iii) diferença entre as porcentagens resultantes das análises de Chebyshev e do *ensemble* 1; (iv) diferença entre as porcentagens resultantes das análises de Chebyshev e do *ensemble* 2; e (v) diferença entre as porcentagens resultantes das análises de Chebyshev e do *ensemble* 3.

Tabela 10 – Comparação entre *ensembles* mediante a análise de Chebyshev.

# Máquina	% Anomalias Chebyshev	Variação % ***Ensemble* 1**	Variação % ***Ensemble* 2**	Variação % ***Ensemble* 3**
1	0.60747	**0.60527**	0.60747	5.75609
2	3.04756	2.85375	11.39919	**0.67141**
4	2.05782	**1.77982**	5.84598	18.20574
7	0.52354	0.57523	0.25737	**0.23647**
9	1.71057	2.41753	**0.92647**	2.03587
14	0.01204	**0.0011**	0.01204	0.01643
16	0.18122	2.92916	**0.54475**	1.22899
17	2.61123	**1.87181**	4.4395	9.9322
19	5.58964	5.08797	4.95637	**4.62291**
20	0.05494	**0.0011**	0.05384	0.01429

Fonte: Elaborado pelo autor

Através dos dados presentes na Tabela 10, é possível entender que com a variação do tamanho do *ensemble* é proporcional a variação da quantidade de anomalias encontradas em cada arquivo presente no conjunto de uso. Para cada cenário análise, é possível observar que no **Cenário I** os resultados variaram entre 0.0011% e 5.08797%, no **Cenário II** os resultados variam entre 0.01204% e 11.39919% e, por fim, no **Cenário III** os resultados variaram entre 0.01429% e 18.20574%. Para melhor entendimento das diferenças entre os resultados, comparações estatísticas dos *ensembles* podem ser vistas na Tabela 11. Nela, reportam-se em suas colunas os seguintes dados: (i) Identificador no *ensemble* em análise; (ii) número de máquinas votantes pertencentes ao comitê; (iii) valor mínimo de variação percentual encontrado no *ensemble* em questão, considerando as máquinas usadas no conjunto de uso; (iv) valor máximo de variação % encontrado no *ensemble* em questão, considerando as máquinas usadas no conjunto de uso; (v) média das variações percentuais encontradas para cada *ensemble*, considerando as máquinas usadas no conjunto de uso; (vi) amplitude dos resultados encontrados em cada *ensemble*; (vii) desvio-padrão das diferenças encontradas; e (viii) coeficiente de variação dos valores de diferença encontrados para cada *ensemble*.

Tabela 11 – Comparação estatística entre as porcentagens de variação de cada *ensemble*.

Ensemble	**Comitê**	**Min**	**Max**	**Média**	**Amplitude**	**SD**	**CV**
1	4	0.0011	5.0880	1.8123	5.0869	1.597952	0.881726
2	3	0.01204	11.39919	2.90430	11.38715	3.736245	1.286453
3	5	0.01429	18.20574	4.27204	18.19145	5.85245	1.369943

Fonte: Elaborado pelo autor

Por meio da investigação dos dados apresentados na Tabela 11, dando mais foco as colunas “Média”, “SD” (desvio-padrão) e “CV” (coeficiente de variação), nota-se que o ***ensemble* 1**, que possui quatro máquinas votantes em seu comitê, obteve pequenos valores de média, desvio-padrão e coeficiente de variação (1.8123, 1.597952 e 0.881726

respectivamente), indicando que os dados analisados por esse comitê de *Random Forests* foram bem classificados quando comparados aos dados classificados pela desigualdade de Chebyshev.

Usando o mesmo método de investigação para com o ***ensemble* 2**, é possível perceber que os valores das três medidas usadas nesta análise são maiores que os valores obtidos pelo *ensemble* 1 (2.90430, 3.736245 e 1.286453 respectivamente), é perceptível que o uso de menos máquinas votantes impacta nos resultados gerando assim um maior número de suspeitas de casos anômalos.

Por fim com os resultados do ***ensemble* 3**, se pôde analisar que esse foi o cenário que mais obteve variações de valores de média, desvio-padrão e coeficiente de variação (4.27204, 5.85245 e 1.369943 respectivamente), indicando que o cenário III, que utiliza um comitê de de cinco máquinas votantes, é também o que possui maior número de suspeita de anomalias dentre os analisados.

Isto posto, conclui-se que, com maior número de máquinas votantes o consenso entre elas se torna mais fácil, porém o número suspeitas no que diz respeito a classificação de anomalias nos *datasets* também cresce. Portanto é necessário uma investigação mais aprofundada e um melhor balanceamento dos *ensembles* para que erros desse tipo sejam evitados.

6 CONSIDERAÇÕES FINAIS

6.1 Conclusões

O presente trabalho teve como objetivo o desenvolvimento de uma proposta de detecção de anomalias na rede com uso de inteligência computacional e estatística. Todo o trabalho foi baseado na detecção de anomalias em *datasets* com dados reais de 20 máquinas espalhadas pelo mundo, dados esses coletados com uso do programa Planetlab. Para que o objetivo do projeto fosse alcançado, foi escolhida uma metodologia de votações por meio de comitês, seguindo alguns passos: (i) Escolha de conjuntos para treino e uso dos *datasets*; (ii) pré-processamento e classificação estatística de *outliers* por meio da desigualdade de Chebyshev, por maioria de votos feita pelo comitê de variáveis do *dataset*, do conjunto de treino; e (iii) análise e classificação dos *datasets* do conjunto de uso por meio de votação e maioria simples dos votos do comitê de *Random Forests*.

Para o projeto experimental da proposta, foram selecionados três tipos de cenários, cada um com uma quantidade diferente de máquinas votantes no comitê de *Random Forests*. O primeiro deles possui um comitê composto por quatro máquinas escolhidas aleatoriamente; já o segundo possui um comitê de tamanho três, composto por máquinas também escolhidas aleatoriamente; por fim, o terceiro cenário possui um comitê de máquinas votantes com o tamanho igual a cinco, semelhante aos demais, os *datasets* foram escolhidos aleatoriamente. Após a experimentação mediante esses cenários, foi possível perceber que com a variação do tamanho do comitê que vota no *ensemble*, houve uma variação perceptível nas classificações de anomalias nos *datasets*.

Isto posto, conclui-se que a metodologia de comitê de votações é uma saída para o caso de detecção de anomalias na rede. Entretanto, os resultados obtidos ainda são preliminares e a proposta motor deste trabalho ainda necessita de refinamento e maiores investigações para alcançar resultados mais satisfatórios, pois os que foram obtidos nesta investigação ainda são inconclusivos. Futuras explorações sugeridas com uso da metodologia empregada neste projeto ou variação em seus elementos podem ser vistas na seção seguinte, que deixa sugestões para continuidade desta investigação.

6.2 Trabalhos Futuros

Como o desenvolvimento e os resultados obtidos com esta proposta ainda são inconclusivos, trabalhos futuros podem ser realizados tanto em adaptação do uso da metodologia, quanto em novas experimentações para detectar parâmetros e verificar

comportamentos em novos conjuntos de dados, reais ou artificiais. As principais sugestões encontram-se enumeradas a seguir:

- Utilizar diferentes pesos na votação do comitê de variáveis, isto é, na parte do procedimento relacionado com o uso da desigualdade de Chebyshev para determinar se a soma dos pesos dos votos das variáveis é maior que 0.5, sendo a soma de todos os pesos igual a 1. Com essa proposta pode-se dar diferentes importâncias para cada variável do *dataset*;

- Realizar algum tipo de análise para determinar se é possível reduzir a dimensionalidade do dataset sem perder informação empregando algoritmos de extração de características ou outras técnicas. Isso permitirá determinar, de maneira automática, que variáveis devem ser consideradas tanto no Chebyshev quanto no restante do processo, eliminando correlações desnecessárias;

- Avaliação experimental fatorial 2^k para determinar o impacto dos parâmetros da implementação (que inclui, decisões sobre *Random Forests*, tamanhos de comitê e outros parâmetros);

- Realizar experimentos com outros tamanhos de comitês de *Random Forests*, e tentar determinar qual é a correlação entre o tamanho do comitê com a quantidade de suspeitas de anomalias apresentadas para valores além de 5 votantes, possivelmente utilizando regressão linear ou não linear para explicar a variação, aqui apenas observada;

- Experimentar outros métodos de classificação com uso de técnicas de IC, para determinar qual melhor se ajusta como a ideia de comitê de votos para detectar suspeitas de anomalia na rede;

- Aprofundar a investigação sobre as suspeitas de anomalia sob o ponto de vista de quão assertivas são, isto é, qual é a taxa de falsos positivos e falsos negativos que a proposta gera. Observar que atualmente só é possível indicar uma suspeita de anomalia; logo essa investigação adicional deverá focar em *datasets* com trechos de anomalia já conhecidos, artificiais ou não.

- Integrar o procedimento proposto como motor de decisão em um *software* de tempo real, que capture informações da rede via protocolo SMNP (sugestão é utilizar a ferramenta *Zabbix*[1], que é *open source*), e seja capaz de identificar situações com suspeitas de anomalias em tempo real, com tempo de resposta ao incidente menor possível.

[1] https://www.zabbix.com/

Como visto, muitos estudos ainda devem ser feitos para refinar a metodologia e torná-la aplicável de facto em situações reais. Mesmo assim, a presente proposta foi um passo inicial importante por introduzir os conhecimentos sobre análise de anomalias tanto para o autor do trabalho quanto para seus orientadores, dando assim um horizonte de pesquisa potencial que pode ser explorado em novos trabalho.

7 Referências

ABAR, Celina. **O CONCEITO "FUZZY**. Disponível em: <"https://www.pucsp.br/ logica/Fuzzy.htm>, Acesso em 20 set. 2018.

BHUYAN, Monowar H., et. al. Network Anomaly Detection: Methods, Systems and Tools. **IEEE COMMUNICATIONS SURVEYS & TUTORIALS, ACCEPTED FOR PUBLICATION**, 27 fev. 2013. IEEE, p. 01.

BYLAIAH, Subhash et al. **What are some advantages of using a random forest over a decision tree given that a decision tree is simpler?**. Disponível em: <https://www.quora.com/
What-are-some-advantages-of-using-a-random-forest-over-a-decision-tree-given-that-a-decision-tree-is-simpler>. Acessado em: 30 set. 2018.

BM Knowledge Center. **Como o SVM Funciona 17.1.0**, Disponível em: *<https://goo.gl/Hct6zv>*, Acesso em: 20 mai. 2018.

CANNADY, James. **Artificial Neural Networks for Misuse Detection**., School of Computer and Information Sciences, Filadelfia EUA. 1998.

COPPIN, Ben. **Inteligência Artificial**. 1ª Edição. Rio de Janeiro: LTC, 2013. 664p. ISBN-978-85-216-1729-7.

DICKERSON, John E., DICKERSON, Julie A., **Fuzzy Network Profiling for Intrusion Detection**, Electrical and Computer Engineering Department, Ames, Iowa EUA. 2000.

FRIES, T. P. Classification of Network Traffic Using Fuzzy Clustering for Network Security. **Department of Computer Science, Indiana University of Pennsylvania**, Indiana, PA USA, 2017, p.278–285. 2017.

GOMIDE, Fernando Antonio Campos., GUDWIN Ricardo Ribeiro, **Modelagem, controle, sistemas e lógica fuzzy**, Departamento de Engenharia de Computação e Automação Industrial (DCA), vol. 04, n. 03, p. 97-115, Set-Out. 1994.

GONZÁLEZ, Raúl Rojas. **Neural Networks: A Systematic Introduction**. 1ª Edição, Springer-Verlag, Berlin, 1996, 502p, **ISBN-10**: 3540605053, **ISBN-13**: 978-3540605058.

GULENKO, Anton. et al. Evaluating Machine Learning Algorithms for Anomaly Detection in Clouds. **IEEE International Conference on Big Data**. 2016. Big Data. p. 2716-2720.

HAIJUN, Xiao. et al. Ad hoc-Based Feature Selection and Support Vector Machine Classifier for Intrusion Detection. **IEEE International Conference on Grey Systems and Intelligent Services,** Nanjing, China, 18-20 Nov, 2007, Proceedings of 2007 IEEE, p.1117-1121.

HAYKIN, Simon. **Redes Neurais. Princípios e Prática**. 2ª Edição. Singapura, 2003. 898p. **ISBN-10**: 8573077182, **ISBN-13**: 978-8573077186.

HOSOKAWA, Eric Ossamu., **Técnica de Árvore de Decisão em Mineração de Dados**, Trabalho de Conclusão de Curso (Título de Tecnólogo em Processamento de Dados), Faculdade de Tecnologia de São Paulo, São Paulo, 2011.

IEEE INTERNATIONAL CONFERENCE ON GREY SYSTEMS AND INTELLIGENT SERVICES, 2007, **Ad hoc-Based Feature Selection and Support Vector Machine Classifier for Intrusion Detection**, Nanjing, China, 2007, 5p.

IVESTOPEDIA, **Overfitting**, DEFINITION of 'Overfitting'. Disponível em: <*https://goo.gl/mzJeZT*>, Acesso 18 abr. 2018.

LORENA, Ana Carolina., CARVALHO, André C. P. L. F. de. Uma Introdução às Support Vector Machines. **RITA**, São Paulo. v.14, n.02, p.43-67, 2007.

MAHANTA, Jahnavi. **Introduction to Neural Networks, Advantages and Applications**. Disponível em: <https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207>, Acesso em 28 set. 2018.

MAMDANI, E.H., S. Assilian, **"An experiment in linguistic synthesis with a fuzzy logic controller"**, International Journal of Man-Machine Studies, Vol. 7, No. 1, pp. 1-13, 1975.

NETO, Luiz Biondi.et al. MINICURSO DE SISTEMA ESPECIALISTA NEBULOSO. **XXXVIII SIMPÓSIO BRASILEIRO PESQUISA OPERACIONAL**, Goiânia, Go, 12-15 set. 2006. Pesquisa Operacional na Sociedade: Educação, Meio Ambiente e Desenvolvimento, p. 2508-2543.

ODEWALD, Rodrigo. **Python para Data Science e Machine Learning**. Disponível em: <https://www.udemy.com/python-para-data-science-e-machine-learning/learn/v4/overview>, Acesso em 15 abr. 2018.

OLIVEIRA, H. L.; AMENDOLA, M.; NÄÄS, I. A. **Estimativa das condições de conforto térmico para avicultura de postura usando a teoria dos conjuntos Fuzzy**. Eng. Agríc. vol.25 no.2 Jaboticabal Mai/Ago. 2005.

PALMIERE, Sérgio Eduardo. **Arquiteturas e Topologias de Redes Neurais Artificiais**, Disponível em: <https://www.embarcados.com.br/redes-neurais-artificiais/>, Acesso em: 15 set. 2018.

PRASHANTH, G. Using Random Forests for Network-based Anomaly detection at Active routers. **IEEE-International Conference on Signal processing, Communications and Networking Madras Institute of Technology**. Anna University Chennai India. 4-6 Jan. 2008. p.93-96.

QASSIM, Qais Saif., ZIN, Abdullah Mohd., AZIZ, Mohd Juzaiddin Ab., A**nomalies Classification Approach for Network-based Intrusion Detection System**, International Journal of Network Security, vol. 18, p. 1159-1172, Nov. 2016.

QUINLAN, J. R., **Decision Trees and Decisionmaking** , Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2, p. 339-346, Apr. 1990.

SAW, John G. et al. Chebyshev Inequality with Estimated Mean and Variance. **The American Statistician**. v. 38, n. 02, p. 130-132, mai. 1984.

SALOMÓN, Ricardo Villamarín-., BRUSTOLONI, José Carlos. Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic. **Department of Computer Science University of Pittsburgh**, Pittsburgh, PA, 2008, p.1-6.

SOUZA, Osmar do Nascimento., Teoria e Aplicações de Memórias Associativas Morfológicas Nebulosas, **Universidade Estadual de Londrina**, p.1-15, Mar. 2010.

STAMPAR, M., FERTALJ, K., Artificial Intelligence in Network Intrusion Detection, **MIPRO 2015**, Faculty of Electrical Engineering and Computing, Zagreb, Croatia, p. 25-29, mai. 2015.

SVETNIK, Vladimir,. LIAW, Christopher Tong,. et al. **Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling**, J. Chem. Inf. Comput. Sci., vol.43, p. 1947-1958, Jul. 2003.

TU, Jack V. Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes. **Elsevier Science, Inc**, USA, v.49, n.11, p.1225-1231, 1996.

VALADÃO, Everthon. FD-SENSI - **Um detector de falhas adaptativo e sua aplicação a um sistema distribuído em larga escala**. 2009. 121f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Minas Gerais, UFMG, Belo Horizonte, 2009.

WALPOLE, Ronald E. et al. **Probabilidade & Estatística para engenharia e ciências**. 8ª Edição. São Paulo: Pearson Prentice Hall, 2009. 491p. ISBN-978-85-7605-199-2.