An Intrusion Detection System for Web-Based Attacks Using IBM Watson

Ricardo C. C. da Silva, Marcos P. de O. Camargo, Matheus S. Quessada,
Anderson C. Lopes, Jacinto D. M. Ernesto, and Kelton A. P. da Costa

***Abstract*—The internet and web applications have been growing steadily and together with the increasing number of cyber attacks. These attacks are carried out through requests that are considered normal or abnormal (attack requests). Therefore, an intrusion attack can be considered as a classification problem. Machine learning algorithms are used as a way to train models to classify these requests in order to increase the security of web systems. The data used to carry out the training and tests in this work come from the CSIC 2010 dataset. The J48, Naive Bayes, OneR, Random Forest and IBM Watson LGBM algorithms were tested. The metrics used were t-rate, precision, recall and f-measure. The results showed that the algorithm used by the Watson tool (LGBM) was the one that did the best in all metrics when compared to the other algorithms in the literature.**

***Index Terms*—Web applications, IBM Watson, cyber attacks, intrusion attack, machine learning, classification.**

I. Introdução

Devido a crescente evolução da internet e ao crescimento das aplicações web, casos relacionados a crimes cibernéticos, como ataques a dados de empresas, roubo de informações privadas, ataques de negação de serviços e ataques de injeção de códigos SQL, vem se tornando mais frequentes nesse meio [1], [2]. Dessa forma, ferramentas e algoritmos, a fim de detectar, combater e detectar esses ataques se fazem necessárias [3].

Os ataques mais comuns quando falamos do cenário de aplicações web, são os de DoS ou DDoS (do inglês, *Denial of Service*), *buffer overflow* e de força bruta, que no geral podem ser considerados como ataques de intrusão. Esses ataques, na sua essência, tem foco nas requisições realizadas pelos clientes aos servidores que hospedam essas aplicações web, sendo suscetíveis a invasões e ataques [2], [4].

Ataques de intrusão são executados dia a dia em todos os tipos de sistemas e redes. A detecção de intrusão nada mais é do que detectar quando um sistema ou rede está sobre ataque. Esse processo é realizado através da análise de dados coletados anteriormente de outros ataques, para que com base nesses dados, a reincidência possa ser detectada e a intrusão combatida [5].

Um ataque de intrusão pode ser considerado como um problema genérico de classificação (uma requisição maliciosa (ataque) ou uma requisição normal) e assim resolvido como tal [6]. Dessa forma, a detecção de intrusão na web é muito utilizada em conjunto com ferramentas e algoritmos de aprendizado de máquina para classificação [6], [7].

O aprendizado de máquina é um ramo da computação onde é possível que o computador tenha a habilidade de aprender a fazer predições e detecções sem necessariamente ter sido programado para isso [7]. Alguns algoritmos conhecidos na literatura para lidar com o problema de classificação são utilizados em conjunto com os Sistemas de Detecção de Intrusão (SDIs) nesse processo, como J48, NaiveBayes dentre outros [7], [8].

O conjunto de ferramentas fornecidas pela plataforma IBM Cloud [9] são frequentemente usadas em conceitos de aprendizado de máquina e outras ferramentas de classificação, seja na parte de sistemas de recomendação, *chat tools* e outros [10], [11], [12]. O IBM Watson é uma das ferramentas desse conjunto disponíveis na IBM Cloud. No sistema proposto, ele é utilizado como ferramenta de classificação para detecção de intrusão em conjunto com o dataset.

O uso do *machine learning* como ferramenta de classificação de ataques é altamente difundido nesse meio. Em conjunto com alguns *datasets* conhecidos como o HTTP CSIC 2010 [13] e o NSL-KDD [14], esses modelos de classificação são treinados e testados para que possam ser capazes de detectar e prever ataques de intrusão.

Esse artigo tem como objetivo utilizar o dataset HTTP CSIC 2010 como base para um modelo de classificação de detecção de intrusão juntamente com a plataforma IBM Watson, mais precisamente com a ferramenta *Watson Knowledge Studio* [15], para verificar quais requisições dispostas no dataset são consideradas anômalos ou normais e realizar uma comparação com outros algoritmos conhecidos na literatura. Sendo assim, as seguintes contribuições do trabalho são destacadas:

- Desenvolver um sistema de detecção de intrusão para ataques web;
- Aumentar a segurança de aplicações web através de uma classificação mais eficiente de ataques;
- Utilizar o dataset HTTP CSIC 2010 como base de testes para treinamento e validação do trabalho;
- Utilizar o IBM Watson e o algoritmo LGBM como forma de classificação dos ataques;

Ricardo C. C. da Silva, Universidade Estadual Paulista, São José do Rio Preto, São Paulo, Brasil (e-mail: ricardo.conde@unesp.br)

Marcos P. de O. Camargo, Universidade Estadual Paulista, Rio Claro, São Paulo, Brasil (e-mail: mp.camargo@unesp.br)

Matheus S. Quessada, Universidade Estadual Paulista, São José do Rio Preto, São Paulo, Brasil (e-mail: matheus.quessada@unesp.br)

Anderson C. Lopes, Universidade Estadual Paulista, São José do Rio Preto, São Paulo, Brasil (e-mail: anderson.claiton@unesp.br)

Jacinto D. M. Ernesto, Universidade Estadual Paulista, Bauru, São Paulo, Brasil (e-mail: jacinto.diassala@unesp.br)

Kelton A. P. da Costa, Universidade Estadual Paulista, Bauru, São Paulo, Brasil (e-mail: kelton.costa@unesp.br)

- Analisar e comparar os principais algoritmos da literatura em comparação ao algoritmo LGBM;

O artigo está organizado da seguinte forma: na Seção II são abordados os trabalhos relacionados a detecção de intrusão, tipos de ataques relacionados a web e uso do IBM Watson. Na Seção III será discutido como foi feita a separação do dataset e quais os critérios e algoritmos utilizados para realizar a classificação. Na Seção IV, serão discutidos os resultados obtidos através dos algoritmos. Por último na Seção V serão apresentadas as conclusões e diretrizes para trabalhos futuros.

II. Trabalhos Relacionados

Nesta seção serão discutidos alguns trabalhos relacionados com o conteúdo de sistemas de detecção de intrusão [2], [4], [16] e o uso do IBM Watson [11], [10] com o uso aplicado a aprendizado de máquina.

Em [2] os autores utilizaram o dataset SNMP-MIB como base para testes de classificação e treinamento para um modelo de detecção de intrusão. Foi feita uma análise acerca dos tipos de ataques de negação de serviços (DoS, do inglês *Denial of Service*), analisando quais ataques eram mais fáceis de serem previstos e quais eram mais difíceis utilizando como base dois algoritmos de aprendizado de máquina, o Random Forest e o Logistic.

Uma outra abordagem para detecção e mitigação de negação de serviços de forma distribuída (DDoS, do inglês *Distributed Denial of Service*) em computação em nuvem é apresentada em [4]. Os autores realizaram um levantamento dos tipos de DDoS listando suas características e particularidades. Foram utilizados dois algoritmos para classificação dos casos (ataques e normais), Naive Bayes e Random Forest.

Já em [16], os autores apresentam uma visão geral dos ataques de injeção de código SQL (do inglês, *Standard Query Language*). Foram utilizados os métodos Taxa de Custo Total e Informação Mútua para extrair características das requisições HTTP para utilização dos classificadores. Utilizaram três datasets para realização dos experimentos, CSIC-2010, DRUPAL e PKDD2007. Os algoritmos utilizados para classificação foram Bayesian Naif e Bayesian Multinomial, classificando quais eram requisições normais e quais eram maliciosas.

Uma análise utilizando a ferramenta *IBM Watson Analytics* foi feita em [11]. Os autores propuseram um modelo para um sistema de recomendação para os usuários utilizando o algoritmo de Fatoração de Matriz. O IBM Watson foi utilizado como forma de analisar e prever quais as melhores comparações para aqueles usuários serão as mais adequadas de acordo com o algoritmo proposto, baseado nas prévias experiências de recomendação feitas para os usuários.

Em [10] foi proposta uma solução de processamento de linguagem natural para melhoria de ferramentas de chat. Utilizaram o *IBM Watson Knowledge Studio* para criar um modelo de linguagem onde analisaram históricos de mensagem dos chats para avaliar a performance da solução proposta por eles.

Como pode-se observar, há alguns trabalhos que tratam sobre sistemas de detecção de intrusão com aprendizado de máquina [2], [4], [16] e outros que tratam do uso do IBM Watson [11], [10]. Porém, nenhum estudo foi encontrado utilizando o IBM Watson com aprendizado de máquina para detecção de intrusão em ataques web especificamente. Dessa forma, o trabalho proposto se faz interessante para analisar e comparar qual o desempenho do sistema proposto utilizando o IBM Watson em relação a outros algoritmos conhecidos.

III. Desenvolvimento

Nesta seção é abordado características sobre o dataset utilizado, tratamento e limpeza, os algoritmos de classificação que foram utilizados e o uso da ferramenta Watson como forma escolhida. Para um melhor entendimento, a seção foi dividida em 3 partes:

1) Dataset e preparação: são abordados pontos como a origem dos dados utilizados, qual o processo de limpeza realizado e a geração dos arquivos necessários para o treinamento;
2) Classificação e treinamento: são abordados os tipos de classificação dos ataques e como o treinamento foi feito;
3) Algoritmos: são abordados os algoritmos aprendizado de máquina utilizados para classificação e a ferramenta escolhida (IBM Watson).

A. Dataset e Preparação

Para realização desse trabalho, foi utilizado o dataset HTTP CSIC 2010 [13]. O dataset contém requisições web (HTTP) geradas automaticamente. Foi desenvolvido pelo Instituto de Segurança da Informação do Conselho Nacional de Pesquisa Espanhol. Remete a testes de proteção de sistemas contra ataques web. No seu total, o dataset é dividido em três partes, sendo elas:

1) Tráfego Normal (treinamento): contém as requisições normais que devem ser utilizadas para realização do treinamento dos modelos. Contém 36000 instâncias;
2) Tráfego Normal (teste): contém as requisições normais que devem ser utilizadas para teste dos modelos. Contém 36000 instâncias;
3) Tráfego Anormal (teste): contém as requisições anormais que devem ser utilizadas para teste dos modelos. Contém 25000 instâncias.

Nesse sentido, foram selecionados 16 atributos empiricamente, sendo 9 atributos padrões das requisições HTTP e 6 atributos sendo recuperados através de um processamento de texto mais profundo. Os atributos selecionados são apresentados a seguir: (1) $Content\text{-}Length$, (2) $Content\text{-}Type$, (3) $Label(norm|anom)$, (4) $Keywords$, (5) $Length\_HTTP$, (6) $Pragma$, (7) $Host$, (8) $Connection$, (9) $Cache\text{-}Control$, (10) $Accept\text{-}Language$, (11) $Accept\text{-}Encoding$, (12) $Email$, (13) $User\text{-}Agent$, (14) $nombre$, (15) $Password\_Count$, (16) $Login$. Nesse sentido, os atributos (4) $Keywords$, (5) $Length\_HTTP$, (15) $Email$, (17) $nombre$, (18) $Password\_Count$, (20) $Login$ representam, respectivamente, as palavras-chaves de uma operação na linguagem SQL, a quantidade total de caracteres contidos em toda a requisição, *e-mail* do usuário, nome do usuário, quantidade de caracteres da senha de acesso ao sistema, nome de acesso ao sistema.

Portanto, as requisições web contidas no dataset podem ser divididos em três categorias: GET, POST e PUT. Dessa forma, na Tabela I é possível ver os atributos padrões para cada tipo de requisição.

TABELA I
ESTRUTURA DAS REQUISIÇÕES DO DATASET UTILIZADO

GET	POST−PUT
User-Agent	User-Agent
Pragma	Pragma
Cache-Control	Cache-Control
Accept	Accept
Accept-Encoding	Accept-Encoding
Accept-Charset	Accept-Charset
Accept-Language	Accept-Language
Host	Host
Cookie	Cookie
Connection	Content-Type
	Connection
	Content-Length
	Campo de Dados

Para a atividade de processamento de informações, como seleção de características relevantes para a classificação das requisições, foi criado e utilizado um algoritmo desenvolvido em linguagem C. Sua escolha foi feita devido a experiência em processamento de texto e debugging presentes nessa linguagem. A função principal do programa é extrair de cada requisição HTTP os atributos selecionados e organizá-los no formato *.csv*. O formato *.csv* é utilizado nas ferramentas e nos mecanismos de aprendizagem, assim como o formato *.arff* (facilmente convertido do formato *.csv*), tanto na ferramenta *Weka* (do inglês, *Waikato Environment for Knowledge Analysis*) utilizada para teste dos algoritmos mais conhecidos, como na ferramenta IBM Watson.

B. Classificação e Treinamento

No ambiente da internet, tudo é realizado a partir de requisições. Uma intrusão ou um ataque a um sistema web pode ser considerada uma atividade anormal quando comparada a outras requisições consideradas normais ao sistema. Sendo assim, é possível categorizar o problema de intrusão de sistemas web ou esse tipo de ataques como um problema de classificação. Dessa forma, as requisições de ataques são consideradas anormais e as requisições comuns, de usuários que realmente estão utilizando e acessando o sistema, como requisições normais.

Tendo isso como base, é possível treinar um modelo para que o mesmo seja capaz de diferenciar quais requisições são ataques (anormais) e quais são requisições comuns (normais). Esses modelos realizam a classificação através de algoritmos de aprendizado de máquina que tem como objetivo diferenciar quais casos são de um tipo ou de outro, de acordo com o treinamento que esses modelos são submetidos.

Dado o dataset utilizado como origem dos dados (CSIC 2010) e sua estrutura (tráfego normal treino, tráfego normal teste e tráfego anormal teste), foi realizada a junção de duas dessas partes para obtenção de um dataset mais robusto e com maior diversidade para conseguir um resultado mais positivo. Dessa forma, o dataset utilizado para treinamento e teste do modelo é a junção das partes do dataset tráfego normal (treinamento) com o tráfego anormal (teste).

O treinamento do modelo foi realizado em duas plataformas diferentes, na plataforma *Weka* e na *IBM Watson Knowledge Studio*.

O *Weka* é uma coleção de ferramentas e algoritmos de aprendizado de máquina para tarefas de mineração de dados e tarefas da área de inteligência artificial (IA) dedicada ao estudo de aprendizagem de máquina. Nela, é possível utilizar esses algoritmos e realizar classificações, análise de padrões e treinamento de modelos [17].

O IBM Watson é um supercomputador arquitetado para aplicações de IA que resolve problemas cognitivos e atualmente é considerado o mais renomado na área. É considerado um computador e um software que simula o processo cognitivo de aprendizado humano para uma máquina, permite assim o treinamento de modelos de aprendizado de máquina de forma customizada para extração de dados. O acesso a ferramenta, é realizado através da plataforma da *IBM Cloud*, conhecida como *IBM Watson Knoweledge Studio*.

No *Weka*, foram realizados os treinamento utilizando os algoritmos J48, NaiveBayes, OneR e Random Forest. No *IBM Watson Knowledge Studio*, foram realizados os treinamentos utilizando o algoritmo LGBM [18] definido automaticamente pelo Watson através de auto IA. Este método realiza a leitura dos datasets apenas no formato *.csv*. Após a realização de *upload* do arquivo *.csv* já é possível dar início ao processo de auto IA. Numa primeira etapa ocorrem a leitura dos dados, divisão dos dados de validação, leitura dos dados de treinamento, pré-processamento e enfim a seleção de dois modelos de algoritmos mais adequados para a problemática em questão. Na sequência o Watson realiza um primeiro teste com um dos modelos, selecionados de forma automática, momento em que vai realizando otimização de hiperparâmetro e engenharia de recurso até obter quatro testes completos de saída. Por fim, o Watson repete os testes com o segundo algoritmo. No contexto de nossa pesquisa o Watson utilizou um algoritmo de árvore de decisão e um algoritmo LGBM. O classificador LGBM é uma estrutura de *gradient boosting* que usa o algoritmo de aprendizado baseado em árvore de folha a folha (horizontal). As etapas mencionadas acima e o resultado final apresentados pelo IBM Watson podem ser observados na Fig. 1.

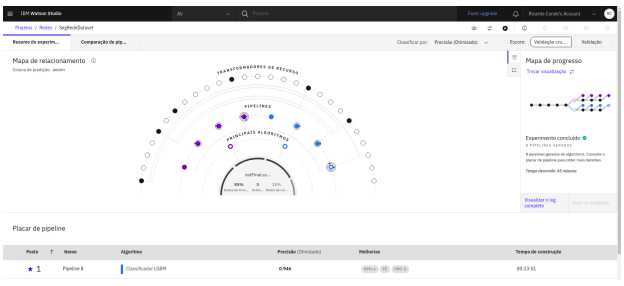

Fig. 1. Auto IA do Watson definindo o algoritmo LightGBM como mais adequado para a problemática.

O modelo foi treinado em ambas as plataformas com base em uma técnica chamada *split*. Essa técnica consiste em dividir (*split*) o dataset em duas diferentes porções, uma parte para

o treinamento do modelo e outra para teste. Dessa forma, a divisão utilizada foi de 85% do dataset para treinamento e os outros 15% para testes e validação do modelo proposto.

C. Algoritmos

As classificações foram feitas utilizando alguns algoritmos de classificação supervisionada conhecidos na literatura como: J48 [19], NaiveBayes [20], OneR [21] e Random Forest [22], e o algoritmo LGBM [18] utilizado pela ferramenta IBM Watson.

Os classificadores Supervisionados classificam os objetos com base em conhecimentos prévios obtidos através de uma base de treinamento contendo objetos rotulados, tal processo é denominado fase de treinamento ou aprendizado. O reconhecimento dos objetos ocorre através de similaridade e o que separa uma classe da outra é a dissimilaridade dos objetos. Nesse sentido, quanto mais dissimilares forem os objetos de uma referida classe em relação a outra, mais facilmente serão classificados pelo algoritmo.

O J48 é um algoritmo de classificação baseado em árvores de decisão que reimplementa na suíte Weka o algoritmo C4.5 desenvolvido por Quinlan. “A árvore de decisão contém um teste, cujo resultado é usado para decidir qual ramificação seguir daquele nó” [19]. O J48 também possui uma fase de pós-poda da árvore após a expansão, na qual são convertidas para folhas as sub-árvores que não representam ganhos de informação acima de um limiar especificado. Segundo [23], “Classificadores de AD são bastante robustos quanto a presença de ruídos e não requerem quaisquer suposições a priori quanto ao tipo de distribuição de probabilidades satisfeitas pela classe e outros atributos”.

O algoritmo Naive Bayes [20] é um algoritmo classificador probabilístico simples baseado na aplicação do teorema de Bayes com fortes suposições de independência. O algoritmo assume que a presença (ou ausência) de uma característica particular de uma classe é não relacionado à presença (ou ausência) de qualquer outro recurso. Mesmo que essas características dependem umas das outras ou da existência uma da outra características, considera todas essas propriedades para contribuir de forma independente para a probabilidade. A Equação a seguir representa a fórmula do teorema de Bayes.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

Sendo:

$P(A|B)$ é a probabilidade do evento A ocorrer dado que B ocorreu.
$P(B|A)$ é a probabilidade do evento B ocorrer dado que A ocorreu.
$P(A)$ é a probabilidade do evento A ocorrer.
$P(B)$ é a probabilidade do evento B ocorrer.

OneR [21] ou One Rule é um algoritmo de classificação simples, mas preciso, que gera uma regra para cada preditor nos dados e, em seguida, seleciona a regra com o menor erro total como “uma regra”. O OneR produz regras apenas um pouco menos precisas do que os algoritmos de classificação de última geração, porém produz regras que são simples de interpretar por seres humanos. Essas regras de classificação impulsionadas por sistemas de aprendizado de máquina são aferidas por dois critérios: precisão de classificação em um conjunto de dados teste, e sua complexidade. Segundo [24], muitos conjuntos de dados do mundo real têm ”poucos picos (muitas vezes apenas um)” e, portanto, são ”fáceis de aprender”.

O algoritmo Random Forest [22] (do português, floresta aleatória) é uma implementação avançada de um algoritmo de *bagging* com um modelo de árvore como modelo básico. Cada árvore no conjunto é construída a partir de uma amostra retirada com substituição do conjunto de treinamento. Ao dividir um nó durante a construção da árvore, a divisão escolhida não é mais a melhor divisão entre todos os recursos. Em vez disso, a divisão escolhida é a melhor divisão entre um subconjunto aleatório de recursos. Por causa dessa aleatoriedade, o viés da floresta geralmente aumenta ligeiramente (em relação ao viés de uma única árvore não aleatória), mas, devido à média, sua variância também diminui, geralmente mais do que compensando o aumento do viés, portanto, rendendo um modelo geral melhor. São dois os parâmetros fundamentais para ajustar os modelos de floresta aleatória: mtry - que define aleatoriamente o número de recursos a serem utilizados em cada divisão; e ntree - a quantidade de árvores no modelo. Segundo [25], valores grandes aumentam a correlação entre as árvores, mas melhoram a força (precisão) de cada árvore. A partir do conjunto de dados de treinamento um subconjunto é carregado para treinar cada árvore na ”floresta”. Em média, cada árvore utiliza próximo a dois terços do conjunto de dados de treinamento. Os elementos não utilizados são chamados de amostras Out Of Bag (OOB), que podem ser usados para validação [26].

O algoritmo utilizado pelo IBM Watson é o LGBM [18]. O LGBM ou LightGBM, é um algoritmo baseado no algoritmo de Árvore de Decisão do Gradient Boosting (do inglês, *Gradient Boosting Decision Tree (GBDT)*). O classificador LGBM é uma estrutura de *gradient boosting* que usa o algoritmo de aprendizado baseado em árvore de folha a folha (horizontal). A principal diferença entre o LightGBM e outro algoritmo baseado em árvore é que sua árvore cresce verticalmente. Na Fig. 2 é possível compreender melhor a forma de crescimento de uma árvore pelo algoritmo Light GBM em relação a outros algoritmos de árvore de decisão.

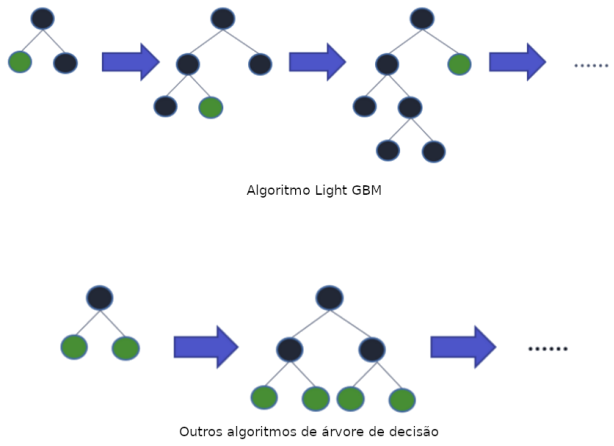

Fig. 2. LightGBM vs outros algoritmos de árvore de decisão.

Trata-se de um algoritmo que vem ganhando notoriedade, pois é considerado rápido, com baixo custo de memória e com suporte ao processamento em GPUs. Tal algoritmo contempla mais de 100 parâmetros que podem ser ajustados para o processamento. No entanto, em nosso contexto ao utilizar o IBM Watson, os parâmetros foram ajustados e adequados para as predições de nossos testes de forma automática.

IV. Avaliação e Resultados

Nesta seção são abordadas as métricas utilizadas para a avaliar os algoritmos e o modelo proposto, bem como os resultados obtidos por essas avaliações.

Para avaliar a eficiência dos algoritmos, foram selecionadas quatro métricas, a saber : (i) TP-Rate (positivos verdadeiros), (ii) Precision (precisão), (iii) Recall e (iv) F-Measure (média precisão + revocação). Abaixo, são apresentadas as equações relacionadas a cada métrica.

$$TP\text{-}Rate\ \ (TP) = Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Precision\ \ (P) = \frac{TP}{TP + FP} \tag{3}$$

$$F\text{-}Measure = 2 * \frac{P * TP}{P + TP} \tag{4}$$

Sendo:

TP é o verdadeiro positivo.
FN é o falso negativo.
FP é o falso positivo.
P é a métrica Precisão.

A métrica TP-Rate (positivos verdadeiros) indica o grau de concordância que há entre o resultado da medição e o dito valor verdadeiro. Em nosso contexto, a capacidade de identificação correta dos registros anômalos. Para isso considera-se a predição realizada pelo classificador, na fase de testes, quanto aos registros anômalos presentes no dataset.

A métrica Precision é a uma medida da concordância entre determinações repetidas de uma mesma grandeza. A precisão é usualmente quantificada como o desvio padrão de uma série de medidas. Em nosso contexto, a precisão é a aferição da identificação realizada pelo classificador descontados os falsos positivos dividido pela quantidade real de registros presentes no dataset na fase de testes.

A métrica Recall pode ser usada em uma situação em que os Falsos Negativos são considerados mais prejudiciais que os Falsos Positivos. Por exemplo, o modelo deve de qualquer maneira encontrar todos os pacientes doentes, mesmo que classifique alguns saudáveis como doentes (situação de Falso Positivo) no processo. Ou seja, o modelo deve ter alto *recall*, pois classificar pacientes doentes como saudáveis pode ser uma tragédia.

A métrica F-Measure é uma média harmônica das duas medidas (Precision e Recall). A média harmônica entre dois números a e b tende a ser próxima de min(a,b). Portanto, F-measure será alto somente se precision e recall forem razoavelmente altos.

Dessa forma, os testes foram realizados através das plataformas *Weka* e *IBM Watson Knowledge Studio* como já descrito na Seção III. Conforme sinalizado na Seção III, o método de validação utilizado foi o *split*. Nesse método foi definido 85% do dataset como treinamento e 15% para os testes efetivos dos algoritmos classificadores. Os resultados podem ser vistos de forma geral na Tabela II.

TABELA II
RESULTADOS OBTIDOS COM OS TESTES NOS ALGORITMOS - MÉTODO SPLIT (85% TREINAMENTO E 15% VALIDAÇÃO)

Algoritmo	TP-Rate	Precision	Recall	F-Measure
J48	87,5%	92,7%	87,5%	90,0%
Naive Bayes	30,8%	64,8%	30,8%	41,8%
OneR	81,3%	80,5%	81,3%	80,9%
Random Forest	88,7%	92,1%	92,1%	92,1%
IBM Watson (LGBM)	94,6%	93,9%	92,9%	93,4%

Na Fig. 3 em relação a métrica de aferição TP-Rate, o Watson se sobressaiu em relação aos outros algoritmos com o valor de 94,6%, sendo o segundo melhor o algoritmo Random Forest 88,7% e o pior o algoritmo Naive Bayes com 30,8%.

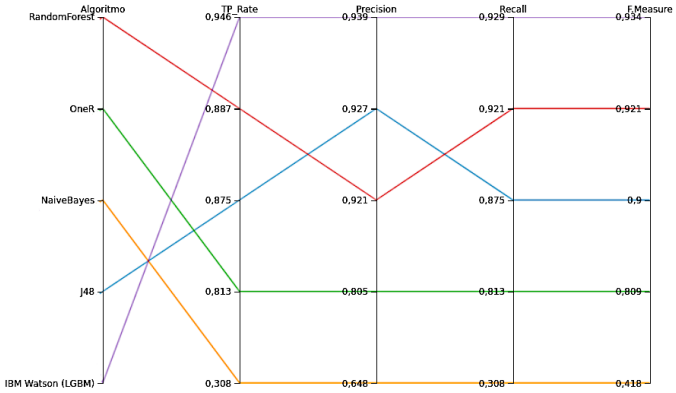

Fig. 3. Gráfico de métricas.

Também é possível observar que o algoritmo usado pelo Watson (LGBM) foi o que obteve um valor maior na métrica Precisão com 93,9%. O algoritmo J48 também se saiu bem, porém ficou abaixo do Watson com 92,7%. O Naive Bayes, foi considerado o pior nessa métrica devido a ser o menor valor com 64,8%.

Na métrica Recall, o algoritmo Naive Bayes obteve o pior resultado com um valor 30,8%. Os algoritmos que obtiveram o melhor resultado foram o Random Forest e o Watson, com 92,1% e 92,9% respectivamente. Sendo assim, o algoritmo utilizado pelo Watson se saiu melhor que os outros testados. Na métrica F-Measure, o Watson obteve o melhor valor com 93,4%. Muito próximos ao valor do Watson, encontram-se os algoritmos J48 (90,0%) e Random Forest (92,1%).

De forma geral, é possível observar quem em todas as métricas utilizadas o algoritmo do Watson se saiu melhor do que os outros. O algoritmo Naive Bayes obteve o pior resultado em todas as métricas em relação aos outros algoritmos. Os algoritmos que mais se aproximaram aos valores alcançados pelo Watson foram o J48 e o Random Forest. Por fim, o algoritmo OneR se mostrou mediano em todas as comparações.

Esse resultado se deve, em parte, pela auto IA do IBM Watson realizar ajustes de Hiperparâmetros de forma automática para a obtenção de melhoria no poder preditivo do modelo e pelo fato do algoritmo LGMB utilizar duas técnicas principais distintas dos demais algoritmos: Gradient-based One-Side Sampling (GOSS). Exclusive Feature Bundling (EFB). O foco do Gradient-based One-Side Sampling (GOSS) é preparar uma amostragem mantendo as instâncias com altos gradientes, de maneira a garantir uma maior acurácia com o ganho de informações. O Exclusive Feature Bundling objetiva auxiliar na redução do número de features, aplicando lógica de agrupamento das features que sejam exclusivas.

V. Conclusão

Esse trabalho apresenta um sistema de detecção de intrusão que utiliza o IBM Watson como forma de treinamento e avaliação para um modelo de aprendizado de máquina. Esse modelo foi desenvolvido com base no dataset HTTP CSIC 2010 e foi avaliado em comparação a outros algoritmos conhecidos na literatura (J48, Naive Bayes, OneR e Random Forest). Desta forma, o algoritmo utilizado pelo IBM Watson (LGBM), demonstrou-se melhor do que os outros com que foi comparado. Na métrica TP-Rate (positivo verdadeiro), o Watson superou os outros algoritmos atingindo a marca de 94,6%. Na métrica Precisão, o Watson conseguiu 93,3% enquanto o menor (Naive Bayes) atingiu 64,8%. Na métrica Recall, o Watson atingiu 92,9%, seguido do Random Forest (92,1) e J48 (87,5%). Na métrica F-Measure, da mesma forma que na Recall, o Watson atingiu um melhor valor em primeiro (93,4%) seguido do Random Forest (92,1%) e J48 (90,0%).

Como diretrizes para trabalhos futuros, pretende-se realizar outras comparações utilizando outros algoritmos e realizar os testes em outros datasets para avaliar a eficiência do modelo proposto.

REFERÊNCIAS

[1] M. Husák, J. Komárková, E. Bou-Harb, and P. Čeleda, "Survey of attack projection, prediction, and forecasting in cyber security," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 640–660, 2019.

[2] M. Abushwereb, M. Mustafa, M. Al-kasassbeh, and M. Qasaimeh, *Attack based DoS attack detection using multiple classifier*, 01 2020.

[3] D. Balzarotti, M. Cova, V. V. Felmetsger, and G. Vigna, "Multi-module vulnerability analysis of web-based applications," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 25–35. [Online]. Available: https://doi.org/10.1145/1315245.1315250

[4] A. Amjad, T. Alyas, U. Farooq, and M. Tariq, "Detection and mitigation of ddos attack in cloud computing using machine learning algorithm," *ICST Transactions on Scalable Information Systems*, vol. 6, p. 159834, 07 2018.

[5] A. Kott, *Towards Fundamental Science of Cyber Security*, 06 2014, vol. 55, pp. 1–13.

[6] Y. Ding and Y. Zhai, "Intrusion detection system for nsl-kdd dataset using convolutional neural networks," ser. CSAI '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 81–85. [Online]. Available: https://doi.org/10.1145/3297156.3297230

[7] Y. Hamid, M. Sugumaran, and L. Journaux, "Machine learning techniques for intrusion detection: A comparative analysis," in *Proceedings of the International Conference on Informatics and Analytics*, ser. ICIA-16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2980258.2980378

[8] P. Tungjaturasopon and K. Piromsopa, "Performance analysis of machine learning techniques in intrusion detection," in *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*, ser. ICNCC 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 6–10. [Online]. Available: https://doi.org/10.1145/3301326.3301335

[9] I. B. M. Corporation. (2020) Introdução ao ibm cloud. Acessado em: 29/10/2020. [Online]. Available: https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform

[10] S. Packowski and A. Lakhana, "Using ibm watson cloud services to build natural language processing solutions to leverage chat tools," in *Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '17. USA: IBM Corp., 2017, p. 211–218.

[11] P. Lak, C. Kavaklioglu, M. Sadat, M. Petitclerc, G. Wills, A. Miranskyy, and A. B. Bener, "A probabilistic approach for modelling user preferences in recommender systems: A case study on ibm watson analytics," in *Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '17. USA: IBM Corp., 2017, p. 38–47.

[12] P. Lak, M. Sadat, C. J. Barrelet, M. Petitclerc, A. Miranskyy, C. Statchuk, and A. B. Bener, "Preliminary investigation on user interaction with ibm watson analytics," in *Proceedings of the 26th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '16. USA: IBM Corp., 2016, p. 218–225.

[13] G. l. M. Carmen Torrano Giménez, Alejandro Pérez Villegas. (2010) Http dataset csic 2010. Acessado em: 20/10/2020. [Online]. Available: https://www.isi.csic.es/dataset/

[14] U. of New Brunswick. Nsl-kdd dataset. Acessado em: 20/10/2020. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[15] I. B. M. Corporation. (2020) Watson knowledge studio. Acessado em: 29/10/2020. [Online]. Available: https://www.ibm.com/br-pt/cloud/watson-knowledge-studio

[16] I. Jemal, O. Cheikhrouhou, H. Hamam, and A. Mahfoudhi, "Sql injection attack detection and prevention techniques using machine learning," *International Journal of Applied Engineering Research*, pp. 569–580, 01 2020.

[17] U. of Waikato. Weka: The workbench for machine learning. Acessado em: 18/11/2020. [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/

[18] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 3149–3157.

[19] J. R. Quinlan, *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[20] "An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, a. m. f. r. s," *Philosophical Transactions of the Royal Society of London*, vol. 53, pp. 370–418, Dec. 1763. [Online]. Available: https://doi.org/10.1098/rstl.1763.0053

[21] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," pp. 63–90, 1993. [Online]. Available: https://webdocs.cs.ualberta.ca/ holte/Publications/simple_rules.pdf

[22] T. K. Ho, "Random decision forests," ser. ICDAR '95. USA: IEEE Computer Society, 1995, p. 278.

[23] CTAN, "The comprehensive tex archive network," 2009. [Online]. Available: http://www.ctan. org

[24] L. Rendell and R. Seshu, "Learning hard concepts through constructive induction: Framework and rationale," *Computational Intelligence*, vol. 6, no. 4, pp. 247–270, 1990.

[25] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 25, pp. 5–32, 2001.

[26] P. Angelo and A. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Computing Surveys*, vol. 51, no. 26, 05 2018.

Ricardo Conde Camillo da Silva Mestre em Informática (Inteligência Computacional) pela UTFPR - Universidade Tecnológica Federal do Paraná (2018), Graduado em Análise e Desenvolvimento de Sistemas pelo Centro Universitário Claretiano (2015), Especialista em Planejamento, Implementação e Gestão da Educação a Distância pela UFF - Universidade Federal Fluminense (2013), Especialista em Gestão da Segurança da Informação pelo Centro Universitário Araraquara (2012), Graduado em Matemática Licenciatura pelo Centro Universitário de Rio Preto (2009), Técnico em Telecomunicações pela escola Philadelpho Gouvêa Netto (1993), Com experiência há mais de 20 anos na área de eletrônica e tecnologia da informação. Atualmente é aluno especial do Programa de Pós-Graduação em Ciência da Computação pela Universidade Estadual Paulista Júlio de Mesquita Filho (UNESP) e do Programa de Pós-Graduação em Ciência de Computação e Matemática Computacional pela Universidade de São Paulo (USP), desenvolve projetos de pesquisa para sistemas inteligentes, visão computacional e robótica, docente no Instituto Federal de São Paulo- IFSP.

Marcos P. de O. Camargo Bacharelado em Ciências da Computação pela Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP) em 2019. Atualmente é aluno do Programa de Pós-Graduação em Ciência da Computação pela UNESP. Sua linha de pesquisa abrange a integração da autoadaptação em sistemas ciber-físicos.

Matheus Sanches Quessada Tecnólogo em Análise e Desenvolvimento de Sistemas pelo Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP) em 2019. Atualmente é aluno do Programa de Pós-Graduação em Ciência da Computação pela Universidade Estadual Paulista Júlio de Mesquita Filho (UNESP). Possui experiência em desenvolvimento de software, aplicações móveis, aplicações web. Sua linha de pesquisa é em redes veiculares, gerenciamento de recursos e sistemas de transporte inteligentes.

Anderson C. Lopes Graduado em Ciência da Computação pela UNORP - Centro Universitário do Norte Paulista em 2004, com especialização em Gestão Empresarial pela mesma universidade. Atualmente é aluno do Programa de Pós-Graduação em Ciência da Computação pela Universidade Estadual Paulista Júlio de Mesquita Filho (UNESP). Possui experiência como docente em engenharia de software, banco de dados e administração de empresas com foco em TI.

Jacinto D. M. Ernesto Carreira desenvolvida na área de Tecnologia da Informação, com ampla experiência em infraestrutura, coordenação de equipes, suporte, administração de Redes, e-mail, chats, backups e segurança. Atualmente frequenta o mestrado em Ciência da Computação pela Universidade Estadual Paulista Júlio de Mesquita Filho (UNESP).

Kelton Augusto Pontara da Costa Possui graduação em Análise de Sistemas pela Universidade Sagrado Coração - USC (2000), mestrado em Ciência da Computação pelo Centro Universitário Eurípides de Marília - UNIVEM (2004), doutorado pela Universidade de São Paulo - USP (2009), pós-doutorado em Redes de Computadores pelo Instituto de Computação da Universidade Estadual de Campinas - UNICAMP e pós-doutorado pelo Departamento de Computação da Universidade Estadual Paulista Júlio de Mesquita Filho - UNESP. Atualmente é professor da Faculdade de Tecnologia (FATEC-campus Bauru) dos cursos de Tecnologia em Banco de Dados e Tecnologia em Redes de Computadores e também professor do curso de Ciência da Computação e Sistemas de Informação da Universidade Estadual Paulista Julio de Mesquita Filho (UNESP-campus Bauru). É avaliador de cursos de graduação do INEP/MEC, Docente do Programa de Mestrado em Ciência da Computação da UNESP (São José do Rio Preto/Bauru) e possui experiência na área de Ciência da Computação, com ênfase em Arquitetura de Sistemas de Computação e Sistemas Distribuídos, atuando principalmente nos seguintes temas: Gerência em Redes de Computadores, Segurança em Computadores, Sistemas de Detecção de Anomalias e Assinaturas em Redes de Computadores e Análise de Fluxo de Dados em Redes de Computadores.