J48 Algorithm in Cyber-Attacks prevention

Enmanuel Aquino, ThanhVi Dang, Jeremy Martínez, Ujjwal Samanta

I. Problem Background

Keeping **Cybersecurity** among our priorities in the tech industry has proved to be a wise decision at every possible level, from the consumer-oriented giant companies like Amazon, Apple, or Meta, to the narrowed use of technological solutions that we implement on a daily basis like using a computer at home or paying with our phones at convenience stores.

Whether it is implemented at a technical level or at a social setting, cybersecurity is the main thing standing in the way of virtual catastrophes, the inherent risks of living in a highly digitalized society, and ourselves. Focusing on the safekeeping of integrity, confidentiality, and availability of hardware and software, it is cybersecurity's role to ensure that our platforms, infrastructure, and information systems are kept from unauthorized access or induced unwanted behavior.

The main reason why there has been a dramatic increase in the number of cyber-attacks that have been executed in the last decades, is how directly proportional to the advancement and globalization of technology they are, the more exposed we are to technology, the more likely we are to rely on it to a point in which people might want to find benefit in exploiting that relationship. This 'dramatic increase' we refer to, is not named so lightly, the 2022 - 2023 Fiscal Year of the United States of America [7] contemplates the herculean amount of 2.6 billion dollars under the Department of Homeland Security (DHS) to allocate exclusively towards cybersecurity solutions to safeguard the country from the consequences of constant cyber attacks, this same budget was of 1.8 billion back in 2018, showing a 36% increase over 4 years and forecasted to be 10.46 billion in a combined amount for all departments and agencies of the government in 2023 [11].

The level of commitment the US government puts into cybersecurity is not exclusive to the country in any sense, *Cybersecurity Ventures* forecasts that **Cybercrime** could cost the world 10.5 trillion dollars each year by 2025 [9] and it is only safe to say that if we are willing to invest this amount of resources into preventing these attacks, it is because the inherent costs of falling to them are way higher than that.

Luckily, the sustained growth of knowledge in **Data Science** and data-oriented solutions to the day-to-day problems we face as society has allowed us to create resistance in the form of models used to predict possible cyberattacks.

This paper explores the idea of using a classification Algorithm named ***J48,*** in cyberattack prevention, and compares it to other categorical solutions that have surfaced through the years like the usage of **Neural Networks** or **Gradient Boosting**.

The J48 **Classification Algorithm** is based on the concept of Decision Trees, a structure similar to a flowchart that illustrates in an easy-to-interpret way a set of possible outcomes from a series of decisions, and it has been used in areas like marketing, research, and, of course, cybersecurity. The reason behind its great use across multiple disciplines is how good it performs at classifying categorical data with high levels of accuracy. Before going in-depth about J48, we should allude that it is considered a **Data Mining** technique, meaning that it *drills* databases with the intention of finding meaningful patterns and converting the extracted data into information that harnesses educated actions, in this particular case, information that helps to avoid a cyberattack from analyzing network data.

A J48 decision three includes a **root node**, which represents the start point and normally holds the attribute that better divides de data, and **branches**, that connect nodes to each other and indicate a decision like "yes", "no" or "sunny", **internal nodes** that further evaluate the variables and finally the **leaf nodes,** which represent the prediction made based on all the conditions and rules.

For the use case of detecting virtual attacks using network data, the framework of a J48-based solution would look like this:

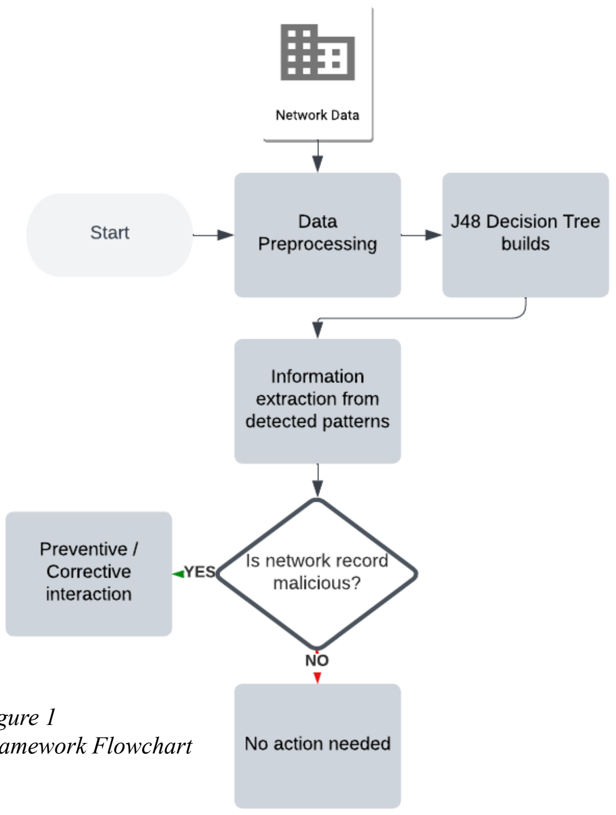

Network Data
Start
Data Preprocessing
J48 Decision Tree builds
Information extraction from detected patterns
Is network record malicious?
YES
Preventive / Corrective interaction
NO
No action needed

Figure 1
Framework Flowchart

II. Comparisons

Many data mining techniques have been used in this field for intrusion detection and cyber attacks. The fast-growing surge of cyberspace in this interconnected world also correlates to the high rates of network instructions or attacks. Therefore, intrusion detection systems (IDS) were developed as shield protection for information systems. Ultimately, their goal is to detect cyberattacks against any external or internal intruders by analyzing suspicious behaviors through either anomaly or signature detection [2]. Due to the rapid growth of digital services that are exposed to cyber-insecurity, this topic has received a pivotal spotlight in research, along with the application of machine learning in this field.

Machine learning-based methods for IDS have proven to be an effective approach for either anomaly or multiple classes of intrusion detection (as we will discuss in this section how that might be). Specifically, the J48 Decision Tree algorithm has been increasingly used to better detect network intrusion. Many previous studies have shown a high (>95%) detection rate when using the J48 algorithm, paralleling the research results conducted by Dr. Rahman and his colleagues as they used this popular machine learning algorithm to implement it into cybersecurity [6]. Others even compare their J48 performance evaluation against other models such as Naive Bayes (NB) and the Random Tree algorithm. The proposed methods of the J48 Decision Tree algorithm used in many studies have outperformed the expectations of cyber attacks and intrusion detection.

The research work of [1] presented a 97.23% detection accuracy using the J48 decision tree for network intrusion detection through a Kyoto 2006+ dataset. They wanted to conduct the J48 model on this dataset to detect the current network situation and the latest attack trends since previous studies mainly focused on using the KDD Cup 99 dataset. As the true positive (TP) shows the number of correct predictions that instances belong to the same class, this classifying technique reports the TP rate to be 0.997 for normal attacks, attack packets, and unknown attacks, which corresponded to the success of this algorithm. This study shows how versatile the J48 decision tree algorithm can be for various types of cyber attacks in different datasets but still remain a very high accuracy detection rate.

On the other hand, in the recent work by [2], Bagui et al. sought to use their hybrid feature selection process and two classification algorithms (NB and J48) to compare their detection accuracy. NB is a probabilistic classifier that assumes the independence of class condition, meaning the impact of an attribute value is independent of other attribute values [2]. Using the UNSW-NB15 dataset containing 9 families of attacks and 49 features, the results were in favor of the NB classifier as it yielded in higher classification accuracy rate with their feature selection process. For rare attacks such as Analysis, BackDoor, Shellcode, and Worms, the NB classifier also did significantly better when feature selection was involved. The shocking conclusion of this study was the only one of many research conducted using the J48 algorithm for cyber attacks that did not correlate to a high performance of this classifier. However, in all fairness, the J48 classifier in this study correctly classified the 9 families of attacks, with >97% accuracy, without the feature selection that this study performed.

In contrast to the study above, this study [4] also conducted a performance analysis of both J48 and NB algorithms to differentiate regular traffic and network intrusions. However, the results from this research suggested that the J48 outperformed NB, as its detection rate was 99.9% for both the training and testing set. The NB algorithm, however, only correctly classified 90% of instances and a high 10.2% false positive (FP) rate. Overall, both of the machine learning-based algorithms successfully executed intrusion detection, but as per the statistics of this paper, the J48 decision tree algorithm outdid the NB classifier.

In a modern-day application, the J48 algorithm was studied regarding attack detection in 5G Networks security [5]. Interestingly, this paper examines two different decision tree algorithms: Random Tree and J48. Although the J48 algorithm achieved and fitted better with the UNSW dataset, it was interesting to see the workings of another decision tree algorithm being analyzed. Similarly to [2], they also used the UNSW-NB15 dataset, but in the simulation of device-to-device connections that 5G cellular network support. As the 5G networks are an up-and-coming field and are gaining attention in research, it is fascinating to see an example of what J48 can do with an imitation 5G dataset, let alone when 5G becomes more prevalent with real datasets produced.

Overall, these studies showed many disciplines of cyber detection that the J48 algorithm has been tested. For the most part, J48 has shown great strengths rather than weaknesses, which gives researchers the opportunity to continue to study this field and how data mining techniques can be successfully applied.

III. Potential Solution/Improvement

When thinking of an improvement over the current framework, we set ourselves to explore the usage of *Gradient Boosting* and *Neural Networks* as two possible alternatives to solve the same problem. The current understanding is that the studied J48 framework positions itself as the superior model, even after doing deep research into other papers and experiments, we found a 95% to 97% accuracy rate at most, which falls under the claimed 99% by Rahman A., Al-Saggaf Y. and Zia T.

This section focus is exploring the concept of gradient boosting for the bulk of the problem-solving and using SHAP values to provide insights about the variables and their impact on the final results.

A. Looking into Gradient Boosting.

Gradient boosting is a supervised machine learning model [14] for classification and regression problems, which produces a model in a decision tree form. It uses the basic concepts of a decision tree, such as root nodes, internal nodes, and leaf nodes. The core concept of this algorithm once the rules are made from the root node to the leaf node, it tries to optimize the result, by fixing the error the previous tree made using the loss function. It continues to build trees until adding additional trees does not improve the results (i.e., reduce the residuals) or it reaches the maximum number of trees the researcher stated in the parameters.

Furthermore, the loss function calculates the residual at each leaf node, which is the difference between the observed value and the predicted value. It evaluates how well can the target variable be predicted, and its relatable to the squared residuals or mean squared error. Therefore, this technique follows a couple of steps until its result.

Step 1 minimizes the sum of squared residuals, ending up with the derivative of each term with respect to the predicted value, equal to zero. The goal is to predict the initial leaf, which would be the average of all target variables observed. Afterward, the algorithm embarks on an iterative process from step 2 to step 6, in respect to the number of trees the researcher will input to the desired outcome.

Step 2 sets the number of trees from 1 to numerous possibilities of trees.

Step 3 calculates the pseudo residuals for each observation, errors made from the observed value, and the initial leaf found in step 1.

Step 4 fits a classification tree with the objective of using independent variables to predict the residuals, instead of the class variable. Where the residual of each observation will end up depending on the conditions the decision trees applied.

Step 5 uses the same approach as step 1, but with the prediction found in step 4. A small variation of the usage of step 1, will be that the output values in every leaf node are always the average of the residuals that end up in the same leaf node.

Step 6 makes new predictions for each observation, based on the initial leaf value in step 1, the output values from Step 5, and an adjusting term called the learning rate [12].

As well as in decision trees, gradient boosting uses a scale contribution called learning rate, to solve any issue in terms of generalizing the training dataset too well, causing overfitting. Finally, once it has looped for the established trees and saved the leaf node values in every iteration, the validation data can be used to predict the desired value. [13]

Beyond the effectiveness of the Gradient Boosting model, in comparison to the J48 algorithm, there is a specific distinction in terms of the interpretability of each technique. Gradient Boosting is known as a black box model, in which the researcher cannot visualize what is happening behind scenes, once the results are available. It means that trees cannot be visualized, nor understand or interpret the results. But there are a couple of approaches to help solve this problem, either using SHAP values or surrogate models.

SHAP (Shapley Additive exPlanation) Values, is a technique used to solve the interpretability issue black box models produced [15]. Instead of providing logic rules, as the decision tree uses, it yields how each independent variable affects positively or negatively to the target variable. The resulting graph shows how each observation magnitude affects the outcome, via the training dataset. Another technique would be surrogate models. Which is a model used to approximate the predictions black box type of model but be interpretable at the same time [16].

It would use the prediction function of the black model in hand but use an interpretable model to provide patterns of the results, such as a decision tree like J48. Importantly, there is a measurement called R squared, which is useful to measure how well the surrogate model replicated the black model. Depending on the results, it will then say if the model passes or fails to explain the black box model.

```
Input: Training set $\{(x_i, y_i)\}_{i=1}^{n}$ a differentiable loss function $L(y, F(x))$, number of iterations M.

Step 1: Initialize model with a constant value:
    $F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: For m=1 to M.

Step 3: Compute pseudo residuals:
    $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n$

Step 4: Fit a regression tree to the rim values and create terminal regions $R_{jm}$, for j=1…$J_m$

Step 5: For j=1…$J_m$ compute:
    $\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$

Step 6: Update the model:
    $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$

Output: $F_m(x)$
```

Figure 2: Optimization process pseudocode.

B. Looking into Neural Networks.

The concept of a Neural Network imitates the way a human brain works, several neurons receive inputs and get activated producing an output which then is “passed” through N numbers of layers to achieve a final result.

Neural networks are capable of detecting patterns and relations within entries of data that are not easily comparable to each other, if we take a minute to go back to the idea of using *network data* as the input for the models presented here, we have to take into account that the *WorkingHours.pcap_ISCX* dataset provided by the Canadian Institute of Cybersecurity (and any other network interaction dataset), contains attributes like; "Fwd Packet Length Max", "PSH Flag Count", "Init_Win_bytes_forward" or "Bwd Packets" which are attributes that at first glance provide little to no insight into the question of whether that entity conforms a cyberattack or doesn't, but being the entire goal of a Neuronal Network to detect hidden patterns and correlations, they seem to be a great fit to solve the problem.

Why could NNs be a good model to predict cyberattacks?

- They shine when detecting underlying patterns that are not easily perceived by other patterns or human supervision.

- The evolving nature of NNs creates an open way to future improvement and the betterment of the model through the years.

- An NN approach would be less vulnerable to changes in the data structure and have way better adaptability options.

- A model based on the Neural Network approach could not only classify a network interaction as benign or intrusive, it could also be expanded or paired with another model that could perform clustering operations to segment the intrusions and build action plans based on the nature of each attack.

- The existence of hyperparameters allows for further tuning of the model and once again higher levels of adaptation to the environment.

Naturally, any possible method to be used for prediction has some sort of negative or flaw, there is no such thing as a 100% accuracy or 100% coverage model, here are some of the difficulties an NN model could face:

- They are very hardware dependent and based on how complex the problem is, this dependency grows higher.
- Connected to the previously mentioned point, they are not only dependent on the hardware, but they are also expensive in this manner, hosting a live-running solution for cyberattack prevention would mean that a lot of resources would need to be constantly assigned to the model. Let's explore this case for a minute:

Using Amazon's (AWS) ML model for batch prediction pricing calculator, we found that they currently charge $0.10 per 1000 predictions [8], without considering any S3 (storage solution) or any other type of extra expense like Real-Time Prediction or EC2 pods. If we convert this data to a real-life scenario:

Amazon itself gets close to 2 billion user connections a year [10] which equals 5,4 million users a day, for the sake of the example, we can consider those users as single network connections to be handled by Amazon servers, the ratio of 5,4 million users a day to 1,000 requests comes to 5,400 charges of $0.10 each day, in the absolute best-case scenario of every user making only one connection to the site which we know is not the real case.

Our calculations translate to $540 a day, Amazon has existed for 28 years or 10,220 days, bringing the cost of maintaining this (once again) absolute best-case scenario Neural Network to 5,5 million dollars, without taking into consideration the costly training process and all the extra needed infrastructure.

Knowing the pros and cons of this model, and the thought process behind why we consider that it could work for predictions in the cybersecurity industry, here's how the model would look on simplified backpropagation pseudocode:

```
Train(Network_Data, Epochs, LR)
  Input <- Network_Data
  Y <- Labels for records Input
  Ws <- Randomly initialize weights
  L <- Number of layers
  run <- True
  Repeat:
     for n in Epochs:
          ForwardProp(): Runs the initial random weights through the network and generates Y values based on them. Neurons get activated in a Relu function.
          BackProp(): Checks the general error and uses a gradient descent algorithm to lower it.
     end for
  until !run

  Train(Network_Data): Trains the model based on the previously shown frame, lowering the error as much as possible.
```

IV. Neuronal Network implementation accuracy test

Based on the previously done research, the initial J48 implementation seems to be the better alternative to run a classification model over the cyberattack detention system. This paper does a great job of explaining why this is the case in a theoretical way, but, we set up to demonstrate an example of a Neuronal Network-based model to see how it performs when compared to the 95% - 99% accuracy J48 Decision Tree models.

To run this experiment we used a hosted notebook with a Python 3.9 instance paired with Pandas, Numpy, Matplotlib, Seaborn, and Sklearn for the preprocessing. For the construction of the model itself, we used Keras.

It was key for us to use the same dataset that was used by Rahman A., Al-Saggaf Y., and Zia T. in their research paper, having the same data source is key to being able to make a reasonable comparison between the models, they are completely different and adding different forms of data on top of that defeats the purpose of this experiment..

Two different CSVs were mounted on the notebook with the following shapes:

- I2-benign.csv: which contains 19,808 entries of benign network communication entities.
- I2-malicious.csv: which contains 249,837 entries of malicious network communication entities.

The following steps were taken in the experimentation process:

Step 1: Both CSVs were merged into a single Pandas data frame with 35 total columns, 34 attributes, and 1 class column which is Benign or Malicious.

Step 2: The resulting DF was then shuffled and the index reset to reduce bias. We can now see a 269,643 data frame with mixed Malicious and Benign entries.

```
0         Malicious
1         Malicious
2         Malicious
3         Malicious
4         Malicious
             ...
269638    Malicious
269639       Benign
269640    Malicious
269641    Malicious
269642    Malicious
Name: Label, Length: 269643, dtype: object
```

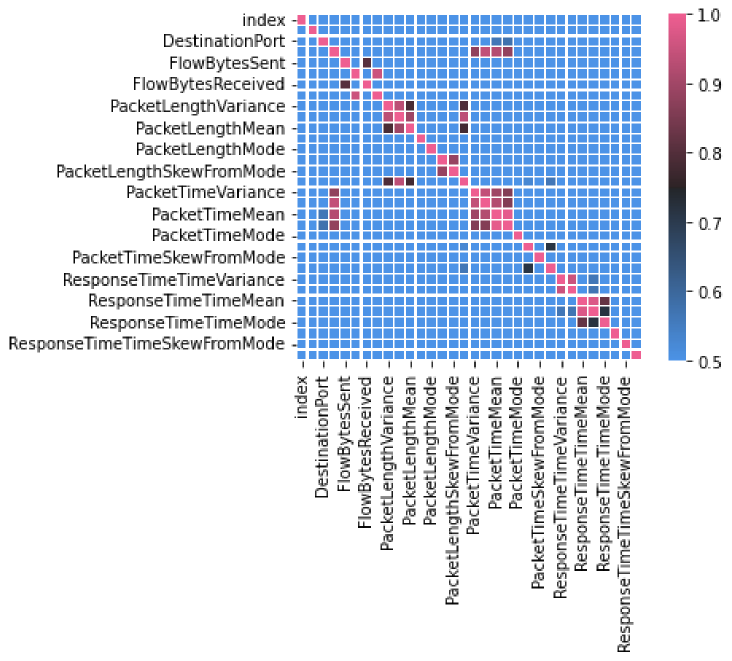
Step 3: At this point, we are aware that Neuronal Network models behave better with categorical data, we decided to convert the Label class to 1 and 0 rather than Malicious and Benign.

```
0         1
1         1
2         1
3         1
4         1
         ..
269638    1
269639    0
269640    1
269641    1
269642    1
Name: Label, Length: 269643, dtype: int64
```

Step 4: Feature selection is extremely important for this type of model, not all features are useful in a learning process. For instance, this DF had a timestamp column that recorded the time at which the network interaction was perceived, this is not useful for the learning process.

Aside from that, among the useful attributes, there are some with a "Strong Positive Correlation", meaning that having them together in the training process would alter the results in a non-realistic way, we want to drop these values as well.

Running a Seaborn Heatmap we get the following result:



Based on this graph, we make an educated feature selection of the following variables:

```
'DestinationPort','FlowBytesSent',
'FlowBytesReceived',
'PacketLengthVariance',
'PacketLengthSkewFromMode',
'PacketTimeMode',
'PacketTimeSkewFromMode',
'ResponseTimeTimeVariance','ResponseTim
eTimeMean',     'ResponseTimeTimeMode',
'ResponseTimeTimeSkewFromMode', 'Label'
```

Step 5: The data was separated into Training and Testing, we went with a 40% test_size since we have a significant amount of entries in the dataset.

Step 6: Building the Neuronal Network

- Dense Layer: 10 Neurons, Relu
- Dense Layer: 10 Neurons, Relu
- Dense Layer: 1 Neuron

Step 7: Compiling the model with an optimizer and a loss function, we used Adam and MSE respectively.

Step 8: Since we decided to run this experiment on a Google Collab for shareability purposes, we can't guarantee that a Cuda device will be always able to run the model and the computing resources might be somewhat limited. Based on that we decided to implement an Early Stopping Mechanism to stop the model from fitting if no more improvements were made at some point.

Step 9: We fitted the model with 150 Epochs and it Autostopped in the 16th iteration with a maximum accuracy of 92%

```
Epoch 9/150
5056/5056 [==============================] -
Epoch 10/150
5056/5056 [==============================] -
Epoch 11/150
5056/5056 [==============================] -
Epoch 12/150
5056/5056 [==============================] -
Epoch 13/150
5056/5056 [==============================] -
Epoch 14/150
5056/5056 [==============================] -
Epoch 15/150
5056/5056 [==============================] -
Epoch 16/150
5056/5056 [==============================] -
<keras.callbacks.History at 0x7f50bc7c39d0>
```

Step 10: running an evaluation of the model

```
5056/5056 [==============================]
[0.06869157403707504, 0.9259016513824463]
```

We can see 0.6% of loss and 92% of accuracy was reached with minimal HyperParams tweaking.

Conclusion: a model with 92% accuracy is very competitive and we're confident that with further improvements on parameters like the number of Epochs, Learning Rate adjustments, and function optimization it could return even better results.

That said, this experiment shows that in some sense, it is easier to achieve better results with a Decision Tree model like the J48 Algorithm and now we feel more confident to back up its level of reliability and efficiency.

For an extended review of the Neuronal Network model, feel free to visit the Google Notebook:

https://colab.research.google.com/drive/1PdULoCKekppTEqxpf1f1bIJfEFdklrNA?usp=sharing

References

[1] Sahu, S., & Mehtre, B.M. (2015). Network intrusion detection system using J48 Decision Tree. *2015 International Conference on Advances in Computing, Communications, and Informatics (ICACCI)*, 2023-2026.

[2] Bagui, S., Kalaimannan, E., Bagui, S.C., Nandi, D., & Pinto, A. (2019). Using machine learning techniques to identify rare cyber‐attacks on the UNSW‐NB15 dataset. *Security and Privacy, 2*.

[3] Alsariera, Y.A. (2021). Detecting Generic Network Intrusion Attacks using Tree-based Machine Learning Methods. *International Journal of Advanced Computer Science and Applications, 12*.

[4] Bashir, U., & Chachoo, M.A. (2017). Performance Evaluation of J48 and Bayes Algorithms for Intrusion Detection System. *International Journal of Network Security & Its Applications, 9*, 01-11.

[5] Steele, B., & Kholidy, H. A. (2020). 5G Networks Security: Attack Detection Using the J48 and the Random Forest Tree Classifiers: A Capstone Report. Department of Network and Computer Security, College of Engineering, SUNY Polytechnic Institute.

[6] Rahman, M.A., Al‑Saggaf, Y., & Zia, T.A. (2020). A Data Mining Framework to Predict Cyber Attack for Cyber Security. *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 207-212.24.

[7] Cybersecurity funding - plum book. (n.d.). Retrieved October 16, 2022, from https://www.govinfo.gov/content/pkg/BUDGET-2020-PER/pdf/BUDGET-2020-PER-5-8.pdf

[8] Alpaydin, E. (2021). Machine learning. Amazon. Retrieved October 15, 2022, from https://docs.aws.amazon.com/machine-learning/latest/dg/pricing.html#w4aab7c20c14

[9] Cybercrimemag. (2021, April 27). Cybercrime to cost the world $10.5 trillion annually by 2025. Cybercrime Magazine. Retrieved October 15, 2022, from https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/

[10] Published by Daniela Coppola, &amp; 14, J. (2022, July 14). Web visitor traffic to Amazon.com 2022. Statista. Retrieved October 15, 2022, from https://www.statista.com/statistics/623566/web-visits-to-amazoncom/

[11] Published by Statista Research Department, &amp; 13, S. (2022, September 13). U.S. government cybersecurity spending FY 2023. Statista. Retrieved October 15, 2022, from https://www.statista.com/statistics/737504/us-fed-gov-it-cyber-security-fy-budget/

[12] Natekin, Alexey.(2013). Gradient Boosting machines, a tutorial. https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021/full

[13] Geron, Aurelien. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd Edition.

[14] Angone, Claudio. (2022). Using machine learning as a surrogate model for agent-based simulations. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0263150

[15] Lundeberg, Scott. (2017). A Unified Approach to Interpreting Model Predictions. https://papers.nips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

[16] Molnar, Christoph. (2022). Interpretable Machine Learning: A Guide For Making Black Box Models Explainable. 2nd Edition.