SURVEYING INTERNET TRAFFIC ACROSS FREQUENTED WEBSITES

Alex Christensen
William & Mary
Williamsburg, VA
aachristensen@wm.edu

Arya Ray-Shryock
William & Mary
Williamsburg, VA
ijrayshryock@wm.edu

Miranda Manning
William & Mary
Williamsburg, VA
elmanning@wm.edu

Yibarek Tadesse
William & Mary
Williamsburg, VA
yetadesse@wm.edu

ABSTRACT

Our goal for this project was to monitor and classify network traffic based on its originating website to understand the different behaviors of web communications. Using Wireshark—a popular network protocol analyzer—we collected traffic data from several commonly visited websites, such as Wikipedia and Best Buy. The captured data was filtered to extract relevant features such as packet counts, which were then used to train our classification models: K-Nearest Neighbors (KNN), Ridge Classifier, and Gradient Boosting. With machine learning, we attempted to predict the source website solely off the input features extracted from our gathered data. Evaluation results revealed that the Gradient Boosting model achieved the highest performance, with a near 0.98 accuracy and F1 score, indicating its strong ability to distinguish between websites in our case. In our group collective, we demonstrated the effectiveness of applying machine learning to network traffic classification while getting a sneak peek at the future of automation.

KEYWORDS

Network Traffic Classification, Machine Learning, Reddit, YouTube, Best Buy, Honda, Wikipedia, Gradient Boosting Classifier

1 INTRODUCTION

Classifying network traffic accurately is now more important than ever to ensure effective network management, cybersecurity, and Quality of Service (QoS) [5, 9]. As internet usage and online services expand and diversify, automatically categorizing traffic patterns has become increasingly imperative [5, 7]. Conventional methods like port-based identification or payload-based Deep Packet Inspection (DPI) are significantly restricted by the prevalence of dynamic port allocation and end-to-end encryption concealing packet contents[5, 7, 9].

Machine Learning (ML) techniques present a powerful alternative for network traffic analysis that circumvents these challenges [5, 8, 9]. ML approaches typically entail the analysis of statistical features from traffic flows (such as packet sizes, timing, counts, etc.) rather than inspecting the payload, which is often encrypted. This makes ML techniques particularly suitable for the modern era. Abundant research has demonstrated the efficacy of ML, including algorithms like logistic regression, support vector machines, decision trees, and ensemble methods, for the classification of various types of network traffic and applications.

This project specifically investigates the feasibility of classifying network traffic by originating website only, based on statistics from captured packet data. We have captured network traffic generated while interacting with five different websites: YouTube, Reddit, Wikipedia, Best Buy, and Honda. We hypothesized that machine learning algorithms would be able to differentiate network traffic by its originating website using solely statistical properties derived from packet data. To test this hypothesis, we captured traffic using Wireshark, extracted relevant statistical features, and formatted a dataset suitable for supervised learning. The dataset was split into training, validation, and testing sets at a ratio of 3:1:1. We implemented and compared the performance of three classification algorithms: K-Nearest Neighbors (KNN) and Ridge Classifier, which served as our baselines, with our main model being a Gradient Boosting Classifier. Gradient Boosting is an ensemble learning method where, as described by Natekin and Knoll (2013), “new models [are added] to the ensemble sequentially… [with each] new weak, base-learner model… trained concerning the error of the whole ensemble learnt so far.” This iterative approach allows the model to progressively refine its predictions. The key contribution of this paper is the empirical evidence of traffic classification at the website level using the statistical predictive power of ML algorithms without relying on payload information. Our findings offer insight into the effectiveness of specific machine learning models, especially Gradient Boosting, in

distinguishing website traffic based on statistical packet features alone. This paper documents our data gathering process, model implementation, and comparative evaluation, concluding with a discussion of the results and potential future work.

2 PROPOSED METHOD

This section documents our methodology for monitoring, analyzing, and classifying computer network traffic based on its originating website. For our model building and selection, we used the sklearn Python library, which provides tools for training, testing, and predicting ML models.

2.1 Data Preparation

To monitor, analyze, and classify computer network traffic by website, we utilize the structured data preparation workflow illustrated in Fig. 2.

2.1.1 *Tool Selection.* This project makes use of the network protocol analyzer software Wireshark (as shown in Fig. 1) to intercept and analyze network traffic data. Wireshark is a well-regarded software that provides detailed insight through the inspection of data packets. Important information, such as protocols used, packet size, and data, is exposed by Wireshark. Such fine-grained detail is crucial to our objective of classifying traffic based on specific attributes. Wireshark's support for numerous protocols ensures it performs well with various network environments. Furthermore, its graphical user interface ensures that it is simple to analyze using features like real-time capture, filtering, and simple examination of packet contents.

2.1.2 Data Collection. To gather traffic data from a website, we set Wireshark to use 10-second intervals. This caused data collection to cease 10 seconds after it started, which was important to help limit independent variables. During each run of Wireshark, we would click on a link to have the website transfer data to our computers, and Wireshark would keep track of every packet transferred during the interval. To filter data from other sources out of our results, we used capture or display filters within Wireshark. This involved finding the IP address the website was using to interact with our machines, and setting that IP as the filter to use. Once the 10-second interval finished, we would download the raw data as a CSV file into our repository's "data" folder. We collected traffic data from five websites: Reddit, YouTube, Best Buy, Honda, and Wikipedia. For each website, we collected an average of 23.6 raw data files for a total of 118 (min=21, max=25).

2.1.3 Feature Engineering. To transform the raw data we gathered via Wireshark into usable statistical data, we wrote a Python script to refine the data. This Python script looped through our data directory, digging into each website folder to reach the individual data CSV files. It used the raw data within each file to create a row of calculated data, including nine pieces of statistical information:

- **Total packet count:** This is the total number of packets that were found by Wireshark across the 10-second interval. This is a good measurement of how much traffic there was between the user and the server.
- **Total packet length:** This is the total size of the collected packets, and can help reveal insights into the kind of traffic that is occurring.
- **Average packet interval:** This is the average amount of time between each packet, which can help reveal how steady the flow of traffic is to and from the website in question.
- **Maximum packet interval:** This is the longest interval between two packets found within the 10-second interval. This information can help narrow down connection issues to the website.
- **Minimum packet interval:** This is the shortest interval between two packets found within the 10-second interval, which is useful for gauging how responsive the website is.
- **Average packet length:** This is the mean size of the collected packets, which can help identify what kind of data the website is transferring.
- **Maximum packet length:** This is the size of the largest packet transferred during the interval, which can help narrow down what types of files are being transferred by the website.
- **Minimum packet length:** This is the size of the smallest packet transferred during the interval. This is useful for gathering insight into the minimum requirements for the website's data transfer speeds.
- **Most common packet length:** This is the most common size found in the packets during the 10-second interval. This is useful for making conclusions about what kinds of files the website transfers most frequently.

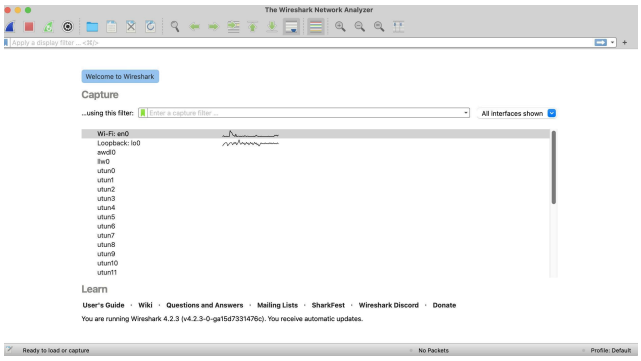


Figure 1: The interface of Wireshark, demonstrating its comprehensive design

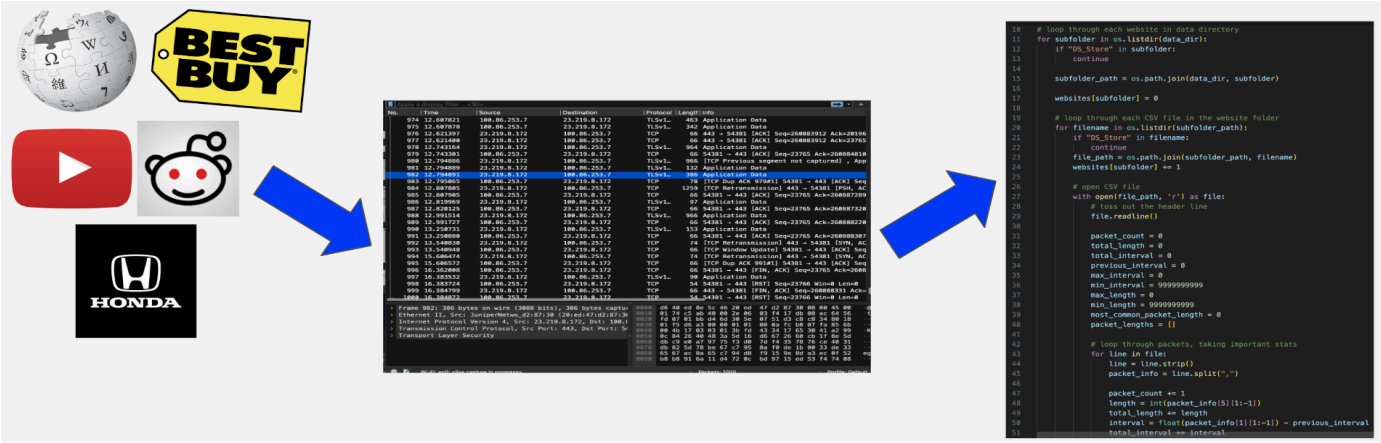


Figure 2: The pipeline of data preparation.

2.2 Employed Method

To classify the originating website based on the collected network traffic data, we used a gradient boosting classifier as our main model. We chose this model for several reasons, key among them being its ability to easily handle multi-class classification problems, as well as its robustness when dealing with smaller datasets.

2.2.1 *Gradient Boosting Classifier.* Gradient boosting is an ensemble learning method primarily used for classification and regression tasks. It operates by constructing many weak decision trees, training each tree to minimize the loss function of the full model based on the residuals given by the previous tree. Once all trees have been trained, the contributions of all are averaged to give a final prediction.

The gradient boosting algorithm (as shown in Fig 3) includes three primary steps:

- **Initialization**: The first tree in the series is trained on the true data and labels, from which residuals are calculated and then used to train the next tree.
- **Loss function minimization**: A loss function is used by the model to calculate the residuals for each tree. There are many different choices, each suitable for different goals. The best fit for our classification problem is the logarithmic loss function. This is the function that the residual trees will be trained to minimize.
- **Final prediction**: The contributions of all trained trees are averaged to come to an overall model decision.

The gradient boosting tends to be less vulnerable to overfitting than basic models. However, several strategies can be taken to further reduce this risk. We employed two of them during the training of our model:

- **Shrinkage**: A consequence of the tree residual training is that data outliers can result in skewed residuals, which can poison the results of the model. To address this, a fixed scalar (in our case, 0.1) is applied to each tree's output, excluding the initialization tree. This lowers the individual contributions of each, effectively smoothing the outliers down, and increases reliability.
- **Early stopping**: Another risk to this algorithm is the possibility that the loss function is minimized before the algorithm stops, leading to overfitting as the trained trees try to optimize a loss function that is already effectively optimized. We fix this by checking the sum of residuals and stopping the algorithm early upon reaching a sum of zero, indicating a fully-minimized loss function.

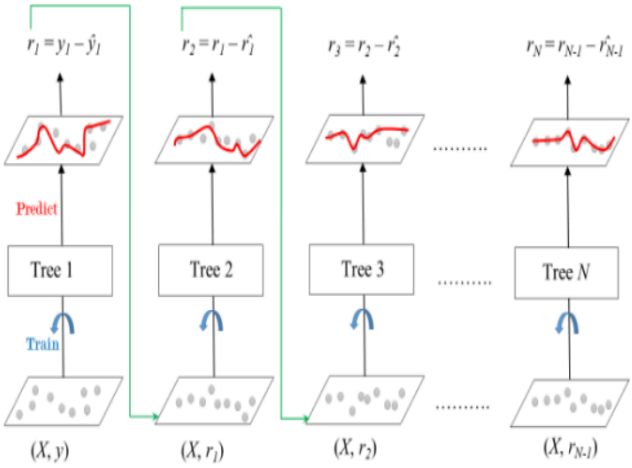


Figure 3: Gradient Boosting Algorithm's Loss Function Minimization Visualization

2.2.2 *Baselines.* We compared two baselines:

- **K-Nearest Neighbors** [10] serves as a baseline model for general predictions due to its simplicity and straightforward implementation. The model algorithm, shown in Fig. 5, looks at the k-closest data points in the training set to the data it is attempting to predict, and makes its decision based on the majority label of those closest data points. This gives a great baseline evaluation for the complexity of this problem, as high performance from a model as simple as this one could indicate that perhaps this problem could be tackled by a much simpler and lighter model than a gradient boosting classifier.

- **Ridge Classifier** [11] is chosen as a baseline model for multiclass categorical classification tasks due to its interpretability and effectiveness in probability-based predictions. It employs a ridge function to convert labels into a numerical category, then uses linear regression to make predictions. This makes it an excellent baseline, capable of providing a performance metric for a basic regression model as opposed to the more complex ensemble method used by the gradient boosting classifier.

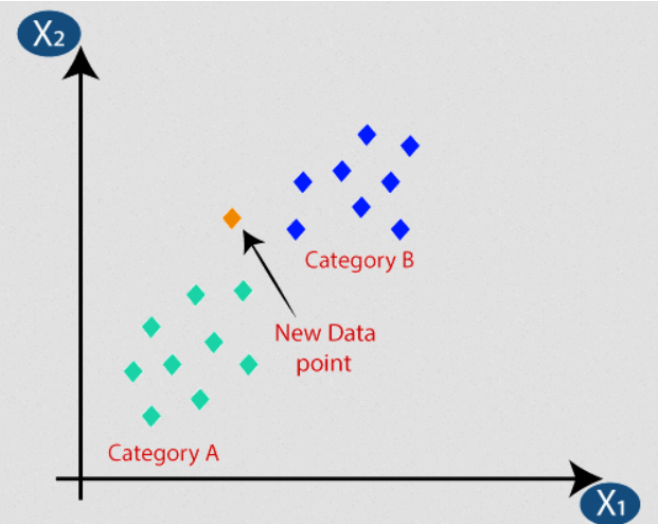


Figure 4: K-Nearest-Neighbors Algorithm Visualization

2.2.3 Model Training:

To train our model, we used a technique known as k-fold cross-validation. First, we split off 20% of our dataset to be used as testing data, and the remaining 80% was used for training and validation.

- **K-fold technique**: to perform cross validation with this technique, first the training dataset is split into k subsets of equal size, with **k=4** in the case of our model. Model fitting is then performed k times, each time using a different subset as the validation set, with the rest being used for fitting. This results in **k** fitted models, each seeing a different set of data for their training phase.
- **K-fold advantages:** We chose this cross-validation method because it partially randomizes the data seen by the model during its training phase, resulting in a robust final model that can handle more diverse data. It also allows us to detect any potential inconsistencies in our dataset, which would be evidenced by large deviations in the performances of individual k-fold models—a problem we fortunately did not encounter.

3 EVALUATION

In this section, we perform an in-depth evaluation of our models and their performance on the dataset.

3.1 Dataset

Table 1 shows the size of the data from each website.

Table 1: Dataset statistics

Label	Size of Refined Data
Wikipedia	25
BestBuy	25
YouTube	24
Reddit	23
Honda	21
Total	118

3.2 Evaluation Metrics

We use two primary metrics to evaluate the models: Accuracy and Macro F1 score. These metrics provide a comprehensive view of performance across all classes.

The **Accuracy** is defined as:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}}$$

The **Precision** is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Precision indicates how many of the *predicted* positive samples are positive.

The **Recall** is defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Recall reflects how many of the *actual* positive samples are *correctly identified* as positive.

The **Macro F1 Score** is calculated by taking the harmonic mean of precision and recall for each class, and then finding the average of these scores across all classes:

$$\text{Macro F1 Score} = \frac{\sum_{i=1}^{n} \text{F1 Score}_i}{n}$$

3.3 Evaluation Results

The results in Table 2 show that the gradient boosting classifier model achieves the highest accuracy (0.9787) and Macro F1 score (0.9782), outperforming both the K-nearest-neighbors and ridge classifier models. This indicates that the gradient boosting classifier model is more effective at capturing complex relationships in the data.

Table 2: The comparison result

Model	Accuracy	F1 Score
KNN	0.5	0.409
Ridge Classifier	0.7994	0.7894
Gradient Boosting	**0.9787**	**0.9782**

3.4 Visualization Results

To evaluate how each model's predictions align with actual values, we visualize the predictions and real values for each model: K-nearest-neighbors (Fig. 5), ridge classifier (Fig. 6), and gradient boosting classifier (Fig. 7).

The results show that the gradient boosting classifier model exhibits the closest alignment between predictions and actual values, reinforcing its superior performance observed in the quantitative metrics.

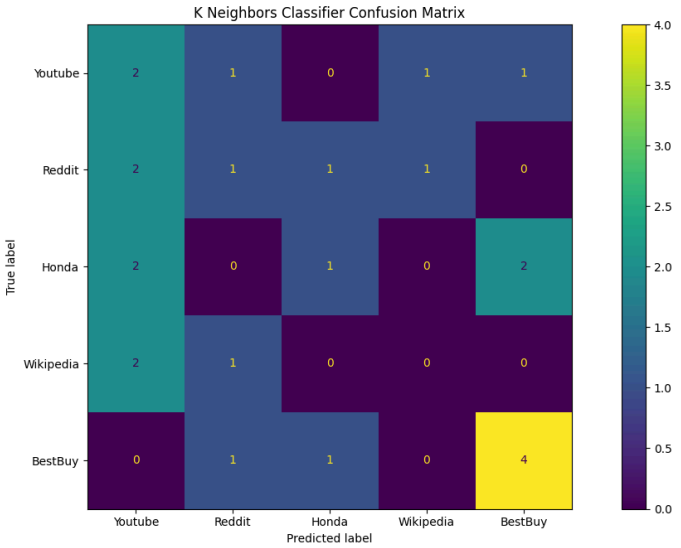
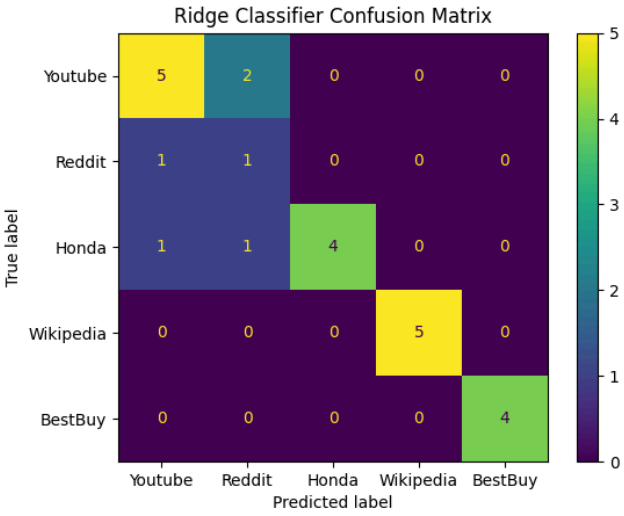


Figure 5: K-Nearest-Neighbors Prediction Visualization



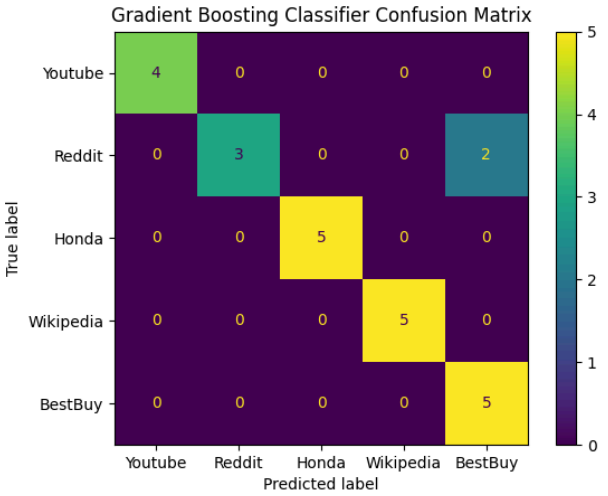


Figure 6: Ridge Classifier Prediction Visualization

Figure 7: Gradient Boosting Classifier Prediction Visualization

4 DISCUSSION & FUTURE WORK

This section will discuss our interpretations of our results and what they mean to us. In addition, we will discuss some ideas we have brainstormed to elevate this project in potential future iterations.

4.1 Interpretation of Results

Three models were tested: KNN, ridge classifier, and gradient boosting classifier. The results indicate that the gradient boosting classifier outperformed both KNN and the ridge classifier for both Accuracy and Macro F1 scores. Specifically, the gradient boosting achieved an accuracy of 0.9787 and a Macro F1 score of 0.9782, far greater than the latter two models shown in Table 2. The discrepancies in our results highlight the gradient boosting classifier's ability to recognize key patterns in the extracted data, making it suitable for our instance of network traffic classification.

With our confusion matrices seen in Figs. 5-7, the figures showcase that the gradient boosting model predictions align closely with the actual values, giving us a good visual indication of the model's performance.

The gradient boosting model did its job in effectively classifying unique traffic patterns, marking its role in upholding network security.

4.2 Future Work

Despite our relative success, we did come across some potential hurdles or ideas we would implement if we were to revisit this for a more robust result:

- **Dataset Size:** Due to the nature of data collection, there is only so much time we can allot to collecting it. For the scope of the task, we believe we aggregated enough data to train our models, yet it would have benefited our models to analyze more data to detect missed patterns. However, that would have been an unrealistic time sink with our current setup.
- **Data Capture:** We tried to ensure that data captures were as consistent as possible (e.g, 10-second capture window), but there very well could have been unwanted inconsistencies with the introduction of human error or variables unbeknownst to us that affected our model training.
- **Model Variety:** Many training models exist for classification purposes, with their own specific characteristics, but we decided to focus only on three. Other models could have proved more useful or even showcased their impracticality for our data sets.

To address the above and enhance the project, we consider some possible options:

- **Automated Collection:** This would need to be explored much further to consider viable options, but some implementation of a methodology that would allow us to automate data collection, or at the very least significantly reduce the amount of human input necessary, would be helpful. This could aid in gathering a large amount of data from a myriad of sites.
- **Strict Parameters:** To improve consistency, setting stricter controls during data capture, like limiting background activity or standardizing system conditions, could have reduced noise. This would help ensure cleaner input for training, making it easier for models to detect patterns accurately and reducing the risk of learning from irrelevant or inconsistent data.
- **Introduce New Models:** While it would introduce much complexity to our work, implementing more models to train would promote a stronger, more confident grasp of what is and is not a suitable choice based on different model frameworks. Because in the end, choosing the most cost-effective and efficient model is crucial in the tech industry as a whole.

5 CONCLUSION

This project successfully demonstrated the application of machine learning techniques for network traffic classification using statistics derived from packet data. We used Wireshark to collect network traffic from distinct websites (Wikipedia, Best Buy, YouTube, Reddit, and Honda) and extracted nine key statistical features. Three classification models- K-Nearest Neighbors (KNN), Ridge Classifier, and gradient boosting- were trained and evaluated on this dataset.

Our findings indicate that the Gradient Boosting classifier significantly surpassed the baseline models with a high accuracy of 98% as well as a Macro F1 score of approximately 0.978. This demonstrates the power of ensemble algorithms like Gradient Boosting in capturing the complex, non-linear patterns present in network traffic at the website level.

Two key takeaways can be derived from this work. First, the Gradient Boosting model is highly effective at distinguishing website traffic patterns based solely on packet statistics, proving robust even with a moderately sized dataset. The visualized results confirm the superior performance of the Gradient Boosting model, showing high alignment between its predictions and actual website labels compared to KNN and the Ridge classifier. Second, it is feasible to achieve high website-level traffic classification accuracy without the need for individual packet payload inspection, highlighting the efficacy of ML techniques in the modern domain of encrypted traffic.

This study emphasizes the ability of machine learning to enhance network monitoring and analysis. Accurate classification based on traffic statistics plays an important role in network management, resource optimization, and

cybersecurity by offering insights into application performance at a detailed level. While these results are promising, it could be worth looking at larger datasets in the future by utilizing automated collection, implementing stricter controls during data capture to minimize variability, as well as comparing a greater variety of ML models. This project overall remains a strong proof-of-concept for the application of ML models in statistical traffic analysis toward detailed network insights.

REFERENCES

[1] *1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/ensemble.html

[2] GeeksforGeeks. (2025, March 11). *Gradient boosting in ML*. GeeksforGeeks. https://www.geeksforgeeks.org/ml-gradient-boosting/

[3] Minhas, M. S. (2025, March 5). *Gradient descent unraveled*. Towards Data Science. https://towardsdatascience.com/gradient-descent-unraveled-3274c895d12d-2/

[4] Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, *7*. https://doi.org/10.3389/fnbot.2013.00021

[5] Salau, A. O., & Beyene, M. M. (2024). Software-defined networking-based network traffic classification using machine learning techniques. *Scientific Reports*, *14*(1). https://doi.org/10.1038/s41598-024-70983-6

[6] Azab, A., Khasawneh, M., Alrabaee, S., Choo, K. R., & Sarsour, M. (2022). Network traffic classification: Techniques, datasets, and challenges. Digital Communications and Networks, 10(3), 676–692. https://doi.org/10.1016/j.dcan.2022.09.009

[7] Bozkır, R., CiCiOğlu, M., Çalhan, A., & Toğay, C. (2023a). A new platform for machine-learning-based network traffic classification. Computer Communications, 208, 1–14. https://doi.org/10.1016/j.comcom.2023.05.010

[8] Kalwar, J. H., & Bhatti, S. (2024). Deep Learning Approaches for Network Traffic Classification in the Internet of Things (IoT): a survey. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2402.00920

[9] Mahmood, M., Khalil, S. A., Shah, S. J., & Ahmad, M. (2023). Network traffic classification techniques and comparative evaluation of machine learning models. Zenodo (CERN European Organization for Nuclear Research). https://doi.org/10.5281/zenodo.8098744

[10] 1.6. Nearest neighbors. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/neighbors.html

[11] RidgeClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html

[12] Wireshark Foundation. 2024. Wireshark. https://www.wireshark.org