CSCI 434 COLL 400 Project

Evan Abney
etabney@wm.edu
William & Mary
Williamsburg, Virginia, USA

Jinyan Kuang
jkuang@wm.edu
William & Mary
Williamsburg, Virginia, USA

Hosanna Root
hzroot@wm.edu
William & Mary
Williamsburg, Virginia, USA

Abhayprad Jha
ajha03@wm.edu
William & Mary
Williamsburg, Virginia, USA

Krishika Pudasaini
kpudasaini@wm.edu
William & Mary
Williamsburg, Virginia, USA

ABSTRACT

This paper explores various options for the use of machine learning models to classify network traffic by its originating website, using features of the traffic itself without requiring the actual contents. We collected a dataset of packet captures from four popular and widely-used websites: ChatGPT, LinkedIn, Reddit, and Wikipedia; and tested five machine learning models to classify them: Linear Regression, Logistic Regression, Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Random Forest. Our evaluations showed the best results from Random Forest, achieving highest accuracy (0.92) and Macro F1 scores (0.93). The project contributes to the field of network traffic classification by providing information on machine learning models that show promise in field-specific applications, as well as proposing opportunities for further development on this research.

KEYWORDS

Network Traffic Classification, Machine Learning, Packet Capture

1 INTRODUCTION

Network traffic classification has many useful applications in the fields of cybersecurity and quality of network service [11]. As the Internet continues to grow and traffic continues to increase year by year globally, network traffic classification continues to become increasingly important to make sense of this expanding usage. This project explores methods for network classification by originating website using machine learning, with the goal of identifying models that show high accuracy and thus promise for further development.

Accurate classification of network traffic is highly beneficial in cybersecurity, as analysis can reveal unusual traffic that enable the detection of suspicious behavior. It is also useful for network management and quality of service, as understanding patterns of traffic allow for more efficient optimization.

We collected a dataset of traffic records from four popular websites to conduct our analysis: LinkedIn, Wikipedia, Reddit, and ChatGPT. This data was split into training, validation, and testing sets in a 6:2:2 ratio, respectively. We evaluated five machine learning models for their effectiveness in classifying network traffic data: Linear Regression, Logistic Regression, Support Vector Machine (SVM), Convolutional Neural Network (CNN), and Random Forest. Among them, Random Forest achieved the highest performance, with an accuracy of 0.92 and a Macro F1 score of 0.93. These results highlight the practical value of machine learning models in network traffic classification, demonstrating both their effectiveness and reliability.

In summary, this project contributes to the field of network traffic classification by highlighting the effectiveness of integrating machine learning models into such analysis, demonstrating potential for advancements in the fields of network security, performance optimization, and management.

2 PROPOSED METHOD

In this section, we present our methodology for collecting and analyzing network traffic data to classify it based on its originating website. The full implementation is available on GitHub.[1]

2.1 Data Preparation

To collect and analyze network traffic data and classify it by originating website, we follow a systematic data preparation process as shown in Fig 1.

2.1.1 Tool Selection. In this project, we use Wireshark (as shown in Fig 2 (a)), a packet capture tool considered to be an industry standard, to handle the task of collecting network traffic data. Wireshark is ideal for this task for multiple reasons. First, Wireshark allows for the collection of detailed information on an individual packet level, including packet

[1] https://github.com/DurinMMII/coll400.git

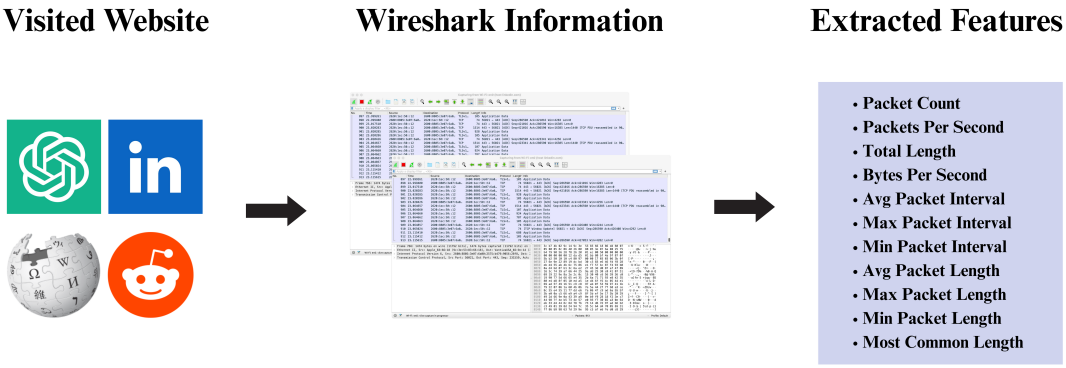
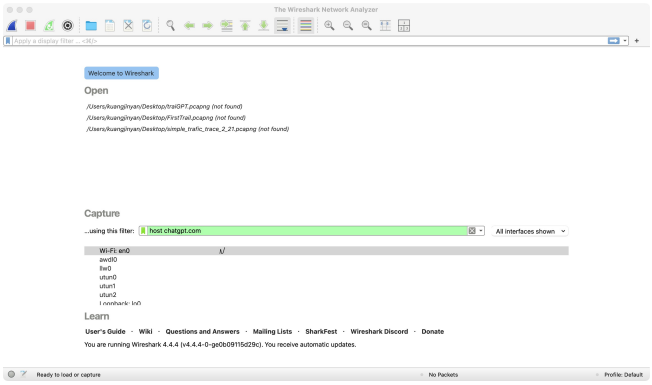
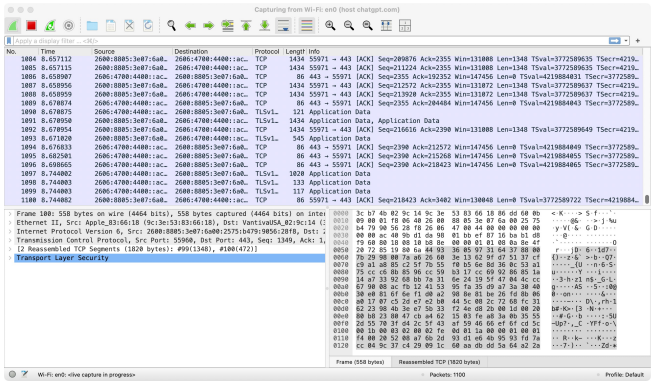


Figure 1: The pipeline of data preparation.



(a) The interface of Wireshark, demonstrating its comprehensive design. This is a sample to trace CahtGPT data



(b) Sample of a data record, including 1099 packets with Time, Source, Destination, Protocol, and Length.

Figure 2: Visualization of the Wireshark tool and recorded packet details.

sizes, contents, times sent and received, and more. These details are essential for our task, providing an array of features to feed into machine learning models to enable them to classify data. Second, Wireshark supports a wide variety of protocols and formats, rendering it extremely useful for collecting comprehensive data on a broad spectrum of Internet use.

2.1.2 Data Collection. We begin the process of data collection by configuring Wireshark to capture traffic from the specific websites we intend to study: Wikipedia, ChatGPT, LinkedIn, and Reddit. These are all popular sites that provide different services: an online encyclopedia, a large-language model chatbot, a business-oriented social media platform, and a forum-based social media platform. These different sites were selected for their popularity and broad use among the general public, as well as their differing usage and formats likely to generate distinct traffic patterns. We then visit the sites, navigating them in patterns similar to that of typical usage, for varying lengths of time. Each such session results in a data record containing information about each packet going to or from the target website (as shown in Fig 2 (b)). We repeat this process thirty times for each website, gathering a comprehensive set of raw data for use by our machine learning models.

2.1.3 Feature Engineering. After collecting our data set, we proceed to process this data in a way that will render it usable by our machine learning models. Exporting each full capture record as a .csv file, we use a Python script to automate the process of extracting key features from the data (as shown in Fig 3). Specifically, we extract:

chatGPT_summary

Packet Count	Pkts_per_sec	Total Length	Bytes_per_sec	Norm Avg Interval	Norm Max Interval	Norm Min Interval	Avg Length (bytes)	Max Length (bytes)	Min Length (bytes)	Most Common Length (bytes)	Label
1157	26.156206629330100	905918	20480.015900803400	-1.153921249210032E-16	15.593646606713900	-0.12053908575515900	782.9887604449440	1434	86	1434	ChatGPT
1369	26.59772370890780	1009091	19605.2035172721	8.782164404404422E-17	27.25639002413930	-0.11008275075454000	737.1008035062090	1434	86	1434	ChatGPT
2057	14.18093823789830	1439098	9921.127786233790	-9.44175563500528E-17	22.104896576841800	-0.10421889556043200	699.6101118133200	1434	86	1434	ChatGPT
659	14.258248997449300	421478	9119.1779528785	-8.018746085913460E-17	12.7141754483122000	-0.1761359443484350	639.5720789074360	1434	86	1434	ChatGPT
1582	11.8494489609254000	847846	6350.510685034620	-6.767725743089780E-17	13.821244220511700	-0.23271709140567900	535.9329962073330	1434	86	86	ChatGPT
412	15.297862565562200	253578	9415.537363228480	5.064886061246150E-18	7.615171153687330	-0.2036180260402350	615.4805825242720	1434	86	1434	ChatGPT
1502	43.41598094275460	1157428	33455.973362590300	1.865781119894534E-17	19.175159900402000	-0.13851668272678400	770.5912117177100	1434	86	1434	ChatGPT
1587	22.111716584807300	1087830	15156.76663670510	1.423829528428480E-16	23.99417747801690	-0.16654248347673200	685.4631379962190	1434	86	1434	ChatGPT
1982	58.13666816056200	973849	28565.255374114600	3.021168485612350E-16	18.226033267907400	-0.21801510453824100	491.3466195761860	1434	86	86	ChatGPT
512	12.06489471223240	304259	7169.634375486530	-1.906498442482530E-17	17.156322812587700	-0.12072518737228200	594.255859375	1434	86	86	ChatGPT
927	50.26712339797950	762014	41320.65994497080	-1.541243734509330E-16	26.064733304192200	-0.11297758732205200	822.0215749730310	1434	86	1434	ChatGPT
3537	77.80742931820890	1481954	32600.234975356800	4.085627010162570E-17	36.97721829297590	-0.16909145303977600	418.9861464517950	1434	86	86	ChatGPT
1104	29.953382606466800	858374	23289.13482014800	8.488978634571550E-17	29.308692555329600	-0.09595878907400570	777.5126811594200	1434	86	1434	ChatGPT
1313	30.366261284402200	953966	22062.742431406000	1.762648292907170E-16	14.397827196901900	-0.18494650047422800	726.5544554455450	1434	86	1434	ChatGPT
1042	54.8129822602149	832833	43810.03882410900	1.589346649805610E-16	13.73939912400370	-0.2006441100078470	799.2639155470250	1434	86	1434	ChatGPT
1552	36.71275157815160	1170761	27694.495973188300	1.57836348761995E-16	22.623264209046900	-0.13298343623674900	754.3563144329900	1434	86	1434	ChatGPT
1158	15.391398069270100	609851	8105.750866962360	1.826062589335936E-16	16.210617840893800	-0.19705838220707000	526.6416234887740	1434	86	86	ChatGPT
867	10.231381210973500	573266	6765.055341741590	-1.259577507730040E-17	22.587934300830700	-0.09259720239456570	661.20645905421	1434	86	1434	ChatGPT
733	15.169971982117200	408329	8450.667789203200	1.550823828796750E-16	16.516656138669800	-0.1552936096064500	557.0654843110510	1434	86	86	ChatGPT
1221	34.993195385437900	964553	27643.56395463580	-2.362522747417960E-16	18.147368344766500	-0.11571903139791500	789.969696969697	1434	86	1434	ChatGPT
1239	11.473838652362000	759424	7032.694467095530	5.373999575982430E-17	27.355616837778400	-0.10361960011833100	612.9330104923330	1514	66	66	ChatGPT
1122	8.973515628357200	498032	3983.153240126550	4.795945581398150E-17	26.874364919475100	-0.10433104969362900	443.8787878787880	1514	66	66	ChatGPT
1305	9.90740012412568	663579	5037.810472771800	8.967349698515690E-17	28.819726733280100	-0.12668025610742500	508.4896551724140	1514	66	66	ChatGPT
1255	6.688726103120300	605148	3225.234441315570	9.008388577002370E-18	22.771170698632300	-0.11754178306416600	482.18964143426300	1514	66	66	ChatGPT
1152	7.395748831125010	617800	3966.2271075252000	9.766297240946750E-18	26.764307842202500	-0.129173344744181	536.2847222222220	1514	66	66	ChatGPT
4684	29.72456094780080	2562172	16259.487141919000	-5.102450362480230E-17	56.219961053044000	-0.07760549751267430	547.00512382579	1514	66	66	ChatGPT
931	8.581941865022450	418791	3860.408180015700	5.730183352904030E-18	14.099407723324900	-0.17847684856992700	449.82921589688500	1514	66	66	ChatGPT
1824	7.093798564348180	935951	3640.048168914610	1.084796907632230E-17	31.02394807322000	-0.08534529920949690	513.1310307017540	1514	66	66	ChatGPT
1910	19.095177894726200	1437125	14367.621744480900	-1.831010170326990E-16	27.650425289739700	-0.08441329289018650	752.4214659685860	1514	66	1454	ChatGPT
1328	5.098146733830450 0	652603	2505.3206723930400	-4.915267723943330E-19	22.55976733925440	-0.10906043954548100	491.417921686747	1514	66	66	ChatGPT

Figure 3: The Sample Processed Data

- **Total Packet Count:** The total number of packets in the capture, summarizing the overall communication volume exchanged with the site.
- **Packets per Second:** The average number of packets transmitted each second, capturing the traffic intensity during interactions with the site. Higher values indicate data-intensive, more frequent communication.
- **Total Length:** The cumulative size (headers+payloads) of all packets, giving the total amount of data transferred during the session.
- **Bytes per Second:** The mean number of bytes transferred per second. This reflects session-level data throughput and helps gauge bandwidth usage.

Before computing the three interval-based features below, each raw inter-arrival time t_i (the gap between consecutive packets) is standardized with the z-score transformation

$$z_i = \frac{t_i - \mu_t}{\sigma_t},$$

where μ_t and σ_t are, respectively, the mean and standard deviation of all t_i values in the capture. Expressing intervals in standard-deviation units removes scale effects and makes them directly comparable across different traces.

- **Average Packet Interval:** The mean of the normalized gaps. Larger values point to more irregular or bursty traffic, whereas values near 0 suggest a steady rhythm.
- **Minimum Packet Interval:** The smallest z-score gap, revealing the tightest burst of packets and the best-case responsiveness.
- **Maximum Packet Interval:** The largest z-score gap, highlighting the longest pause (periods of inactivity or latency spikes).

- **Average Packet Length:** The mean size of packets (in bytes). Smaller averages hint at lighter traffic, like text-focused content; while larger averages may suggest data-intensive, media-rich content.
- **Minimum Packet Length:** The smallest packet size observed, reflecting lightweight communications such as ACKs or small requests.
- **Maximum Packet Length:** The largest packet size observed, revealing peak data-transfer demands (e.g., image or video segments or large files).
- **Most Common Length:** The packet size that appears most frequently, profiling the "typical" packet and characterizing the site's dominant content type.

The insights provided by these data points are useful for the classification of host websites, as they hold information about the traffic that can convey suggestions about the underlying data, common formats and approaches used by the site, and more.

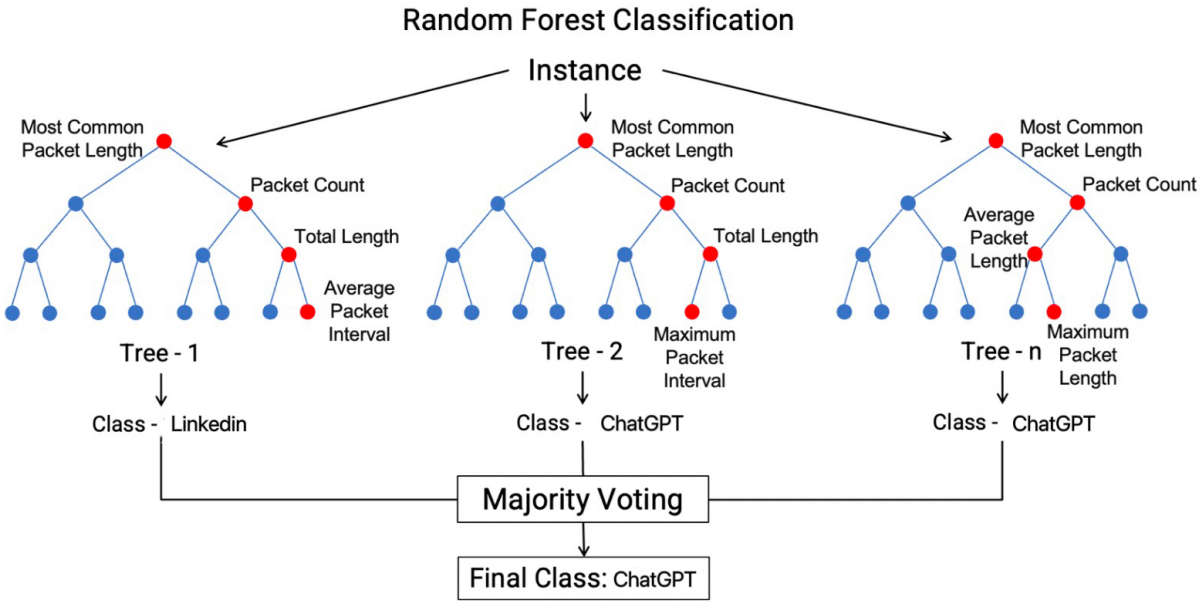


Figure 4: Example of majority voting in Random Forest.

2.2 Employed Method

2.2.1 SVM. Support Vector Machines (SVMs) are supervised learning models used for classification and regression tasks[2]. They work by finding a decision boundary that separates different classes in a dataset [5]. This boundary is defined by a linear equation involving weights ($\mathbf{w}$), feature vectors ($\mathbf{v}$), and a bias term (b). The objective is to maximize the margin—the distance between the decision boundary and the closest data points from each class—while minimizing classification error.

The regularization parameter C controls the trade-off between maximizing the margin and minimizing classification errors. A larger C prioritizes correct classification, potentially leading to a narrower margin and overfitting. Conversely, a smaller C allows for a wider margin but tolerates more misclassification. SVM is particularly well-suited for smaller datasets and performs well in high-dimensional feature spaces, making it a natural choice for this project, which uses a small, manually-collected dataset with twelve different features.

We tested four kernel types: linear, polynomial, radial basis function (RBF), and sigmoid. Several values of C were explored, ranging from 30 to 60.

SVMs are powerful classifiers due to their ability to create complex decision boundaries and generalize well in many settings [3]. However, they require careful tuning to avoid overfitting, especially when using non-linear kernels or high regularization values.

2.2.2 Random Forest. Random Forest is an ensemble learning method primarily used for classification and regression tasks [1]. It constructs a collection of decision trees that come to independent conclusions, then aggregate their outputs to make a final decision (as shown in Fig 4). It uses majority-voting mechanism to improves both prediction accuracy and model robustness. Each tree is trained on a different "bootstrapped" sample- a randomly selected sample from the training data. This collection of trees is then run on other data, where each tree predicts a class label (in our case, the originating website) independently of all other trees in the model. Once this process is complete, the label that received the most votes among all trees is returned, as the model's final prediction. Aggregating these results from multiple trees helps to reduce the risk of overfitting, which is a common flaw when using single decision trees alone[8], and gives a more stable and reliable model that is less vulnerable to being thrown off by complex feature sets or outliers in the data. Similar to SVM, Random Forest is well-suited for tasks involving high-dimensional dataspaces, making it a natural choice for our project. Its strong resistance to overfitting is particularly advantageous given the relatively small size of our dataset.

2.2.3 CNN. Convolutional Neural Networks (CNNs) are a class of deep learning models that excel at capturing spatial or sequential patterns in structured data [7]. CNNs consist of an input layer where data is entered, followed by a set of "hidden layers" that each operate on the data before passing

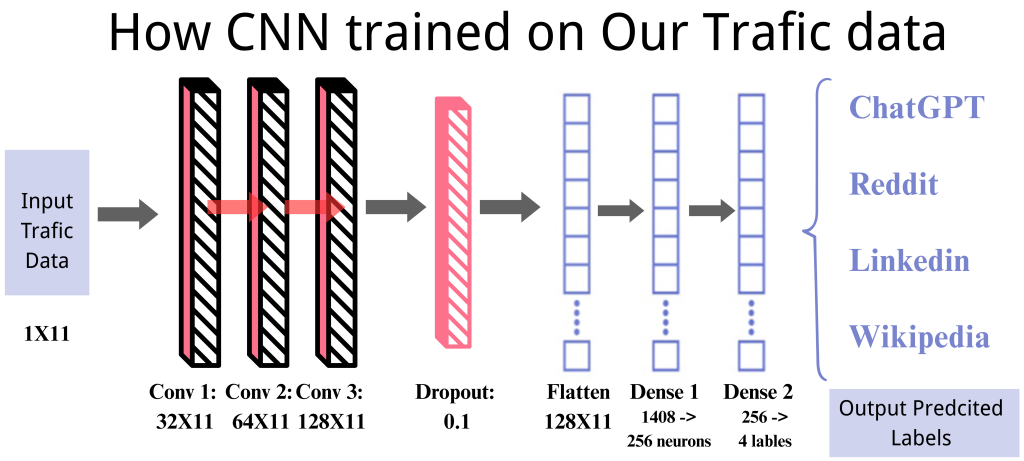


Figure 5: CNN architecture for classifying website traffic.

it forward to the next layer, and concluding with an output layer where the final result is determined. While CNNs are widely used in image recognition, they have also demonstrated ability in processing one-dimensional signals such as time series or network traffic flows[6]. The model used for this project, Conv1D, is designed to automatically learn temporal dependencies and hierarchical feature representations from the input vector, without relying on hand-crafted rules. In our task, we employ a one-dimensional CNN (Conv1D) to classify website traffic based on 11 features extracted from packet flows (as shown in Fig 5). The overall architecture of the model consists of three "blocks" of filters (32, then 64, then 128), each block followed by batch normalization, ReLU activation, and dropout. After these blocks comes a flattening layer, in which the feature map is flattened into a many-dimensional vector, after which two dense layers reduce the feature space and then output the predicted class. CNNs have multiple advantages: the combination of convolutional layers and regularization techniques help mitigate overfitting, and improve generalization to unseen patterns. Disadvantages exist as well, however: they generally need large datasets, which could present an issue with the size of the datasets collected for this project.

2.2.4 Baselines. We compared two baselines:

- **Linear Regression** is a foundational supervised learning technique that models the relationship between input features and a continuous output by minimizing the mean squared error. It is widely used for both prediction and interpretation of linear relationships [10]. In this study, we included Linear Regression to explore how well a simple linear fit could capture patterns in network traffic data.
- **Logistic Regression**, in contrast, is a widely adopted statistical model specifically designed for binary and multi-class classification tasks. It estimates class probabilities by applying the logistic (sigmoid) function to a linear combination of input features [4]. Due to its simplicity, interpretability, and strong theoretical grounding, it remains a common benchmark in classification problems.

  Together, these two models served as baseline approaches in our evaluation, providing a reference point against which we compared the performance of more advanced machine learning methods such as SVM, CNN, and Random Forest.

3 EVALUATION

3.1 Dataset

After processing the collected data, we obtained the dataset shown in Table 1. The train, validation, and test sets were split with a ratio of 6:2:2.

Table 1: Details of Dataset

Label	Size of Processed Data
ChatGPT	30
LinkedIn	30
Wikipedia	30
Reddit	30
Total	**120**

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.

Accuracy. Accuracy is defined as the proportion of correctly classified instances (true positives and true negatives) out of the total number of samples:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}} \tag{1}$$

Precision. Precision measures how many of the predicted positive samples are actually positive:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

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

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3}$$

Macro F1 Score. The Macro F1 Score is computed by first calculating the F1 score for each class using the harmonic mean of its precision and recall, and then taking the average across all classes:

$$\text{Macro F1} = \frac{1}{n} \sum_{i=1}^{n} \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{4}$$

3.3 Evaluation Results

Table 2: Comparison of Model Performance

Model	Accuracy	Macro F1 Score
Linear Regression	0.38	0.32
Logistic Regression	0.80	0.80
CNN	0.46	0.31
SVM	0.88	0.88
Random Forest	**0.92**	**0.93**

The results in Table 2 demonstrate that the Random Forest model outperforms all other models, achieving the highest accuracy (0.92) and Macro F1 score (0.93). Compared to Linear Logistic Regression, CNN, and SVM, Random Forest is more effective at capturing complex, nonlinear patterns in the data. Its ensemble structure also offers greater robustness against overfitting, which is particularly beneficial given the relatively small size of our dataset. In contrast, the CNN model performs poorly (Macro F1 score of 0.46), likely due to its reliance on large training sets and sensitivity to limited data, which hinders its ability to generalize effectively in this context.

3.4 Visualization Results

To evaluate how each model's predictions align with actual values, we visualize the predictions as compared to the real values for each of our five models: Linear Regression (Fig 6 (a)), Logistic Regression (Fig 7 (a)), CNN (Fig 8 (a)), SVM (Fig 9 (a)), and Random Forest (Fig 10 (a)). These visualizations show the closest alignment to real values for Random Forest, reinforcing its superior performance observed in the raw data results.

We also present the confusion matrices for each model—Linear Regression (Fig 6 (b)), Logistic Regression (Fig 7 (b)), CNN (Fig 8 (b)), SVM (Fig 9 (b)), and Random Forest (Fig 10 (b))—to illustrate the relationship between predicted and actual website categories. These visualizations help reveal how accurately each model classifies traffic across the four website categories. Also, the Random Forest model demonstrates strong overall performance, although it misclassified two instances in the ChatGPT category.

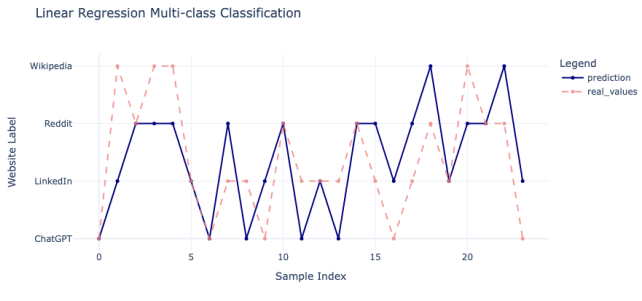
4 DISCUSSION & FUTURE WORK

4.1 Interpretation of Results

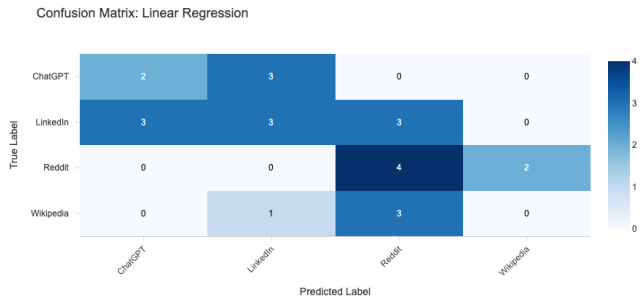
Five models were tested: Linear Regression, Logistic Regression, SVM, CNN, and Random Forest. The results indicate the best results from Random Forest, with an accuracy of 0.92 and a Macro F1 score of 0.93. SVM also achieved strong results, with an accuracy and a Macro F1 score both equaling 0.88. While SVM did not perform as well as Random Forest, its performance was still sufficient to suggest potential in the field of network traffic classification. Logistic Regression, with both Accuracy and Macro F1 score at 0.80, was not as accurate as SVM and Random Forest, but proved suitable as a simple method with decently accurate performance to compare other models to. CNN was remarkably unsuccessful compared to the other non-regression models, with an accuracy of 0.46 and a Macro F1 of 0.31: Linear Regression was the only model that performed worse, with a comparable Macro F1 score of 0.32 but a lower accuracy of 0.38. These results highlight the strengths of SVM and Random Forest in tasks of network traffic classification. This comparison and evaluation contributes to understanding of the use of machine learning models to undertake such tasks, and indicates promising avenues for future work.

4.2 Future Work

While the end result with Random Forest proved to be successful, demonstrating an accurate and reliable model for network traffic classification, certain areas do stand out as options to improve upon the results of this project.



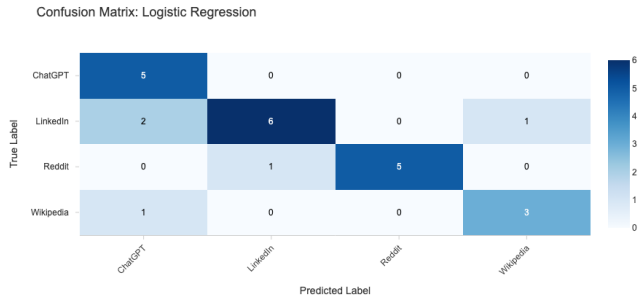
(a) Predictions vs. actual values for Linear Regression.



(b) Confusion Matrix of Linear Regression

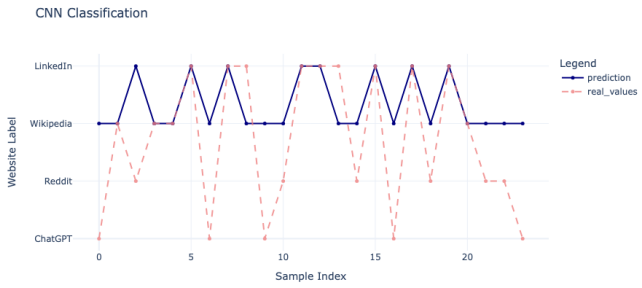
Figure 6: Visualization of the Results of Linear Regression

(a) Predictions vs. actual values for Logistic Regression

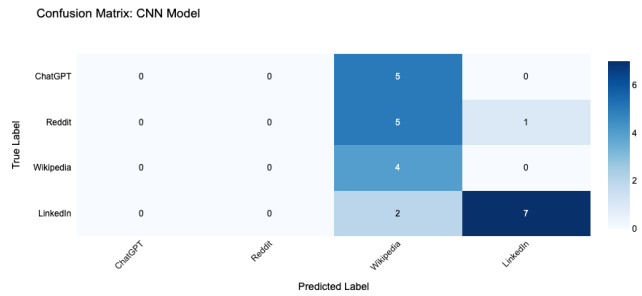


(b) Confusion Matrix of Logistic Regression

Figure 7: Visualization of the Results of Logistic Regression



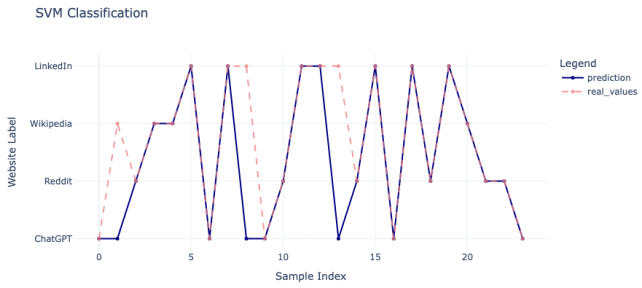
(a) Predictions vs. actual values for CNN



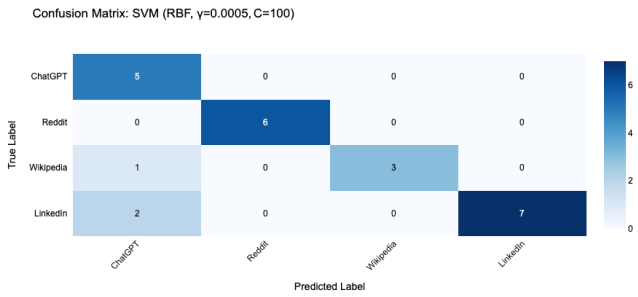
(b) Confusion Matrix of CNN

Figure 8: Visualization of the Results of CNN

- **Dataset Size.** The relatively small size of the dataset, limited by the time and manpower required to manually collect data, might limit the model's ability to generalize to data less similar to the limited set it was trained on, with an elevated risk of overfitting.
- **Model Selection.** While this project included five models and found strong results of >0.85 accuracy and Macro F1 score from two of them, many more models exist, some of which could prove even more effective than Random Forest or SVM.
- **Real-time Data.** This project's methodology focused solely on capture files from static browsing sessions,

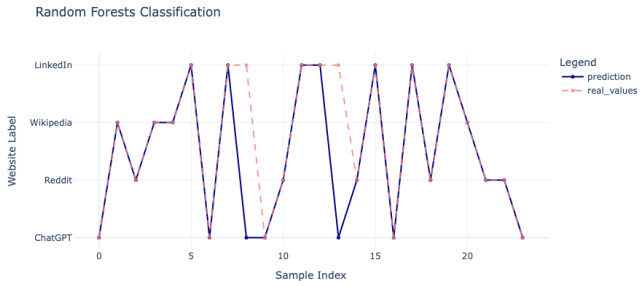


(a) Predictions vs. actual values for SVM

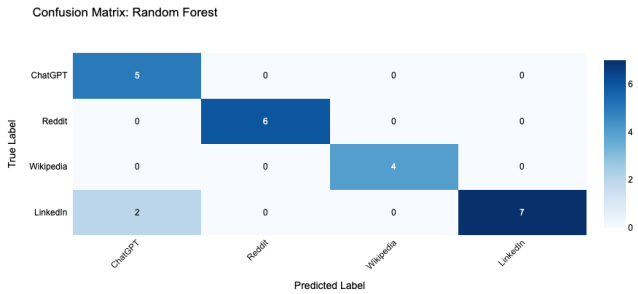


(b) Confusion Matrix of SVM

Figure 9: Visualization of the Results of SVM



(a) Predictions vs. actual values for Random Forest



(b) Confusion Matrix of Random Forest

Figure 10: Visualization of the Results of Random Forest

while real-life applications of network traffic classification often involve rapidly accumulating data and dynamic traffic.

To address these challenges and continue building on the groundwork laid by this project, we propose the following:

- **Larger Dataset.** Expanding the size of the dataset used would help ensure that the models being tested are more reliable and more capable of generalizing to other data. We propose exploring options for the automation of data collection, in order to reduce the time human researchers need to spend collecting data by hand and facilitate such a larger database. This may particularly help the CNN model to perform better, as this model typically works best with a large dataset that we were unable to provide.
- **Advanced Model.** There is growing interest in applying large language models (LLMs) such as BERT to network traffic analysis tasks [9]. In future work, we propose the exploration of more advanced approaches beyond the traditional machine learning models used in this study. These include multilayer perceptrons (MLPs), improved neural network architectures, enhancements to our CNN model with larger datasets, and potentially the integration of LLM-based methods for traffic classification. As demonstrated in recent benchmarks, foundation models hold significant promise for advancing accuracy and generalizability in this domain.
- **Real-time Processing.** Expanding the model to handle real-time data streams would enhance its practicality and render it far easier to use in dynamic, real-world network environments. This could prove particularly useful in network security contexts, as such data processing could allow the model to alert administrators and/or take direct action in response to suspicious activity in real time.

5 CONCLUSION

In this project, we developed and evaluated a model for classifying network traffic by its originating website. Using a dataset collected from four popular websites with varying uses, we employed five different machine learning models:

Linear Regression, Logistic Regression, Convolutional Neural Networks (CNN), Support Vector Machine (SVM), and Random Forest. Through comprehensive testing and evaluation, the Random Forest model was found to be the most effective, while SVM was a close second, with both achieving high accuracy and Macro F1 scores.

This work demonstrates the feasibility of the use of machine learning models for network traffic classification, and provides groundwork for determining which models may be stronger candidates for such applications: including Random Forest and SVM. As network traffic classification is critical for network maintenance, security, and performance optimization, this research has broad practical applications in these fields, enabling far more efficient methods than manual data classification.

The main insights include: 1) The Random Forest model provides the best performance and accuracy of the models tested, with SVM proving similarly useful, 2) The visualizations provided in this project help to illuminate and reinforce its findings, graphically demonstrating each model's ability to accurately classify network traffic, 3) A larger dataset and a broader array of models to test provides significant promise for expanding upon the conclusions of this project.

This project serves as a foundation for further developments in the field of network traffic analysis, providing practical guidance for the integration of machine learning models into real-world network systems to enhance their security, enable their optimization, and improve their management.

REFERENCES

[1] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[2] Richard G Brereton and Gavin R Lloyd. 2010. Support Vector Machines for classification and regression. *Analyst* 135, 2 (2010), 230–267.

[3] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.

[4] David W. Hosmer, Stanley Lemeshow, and Rodney X. Sturdivant. 2013. *Applied Logistic Regression* (3rd ed.). Wiley.

[5] S. Huang, N. Cai, P. P. Pacheco, S. Narrandes, Y. Wang, and W. Xu. 2018. Applications of Support Vector Machine (SVM) Learning in Cancer Genomics. *Cancer Genomics & Proteomics* 15, 1 (Jan–Feb 2018), 41–51. https://doi.org/10.21873/cgp.20063

[6] Serkan Kiranyaz, Turker Ince, Osama Abdeljaber, Onur Avci, Moncef Gabbouj, and Daniel J. Inman. 2021. 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing* 151 (2021), 107398. https://doi.org/10.1016/j.ymssp.2020.107398

[7] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[8] Andy Liaw and Matthew Wiener. 2002. Classification and Regression by randomForest. *R News* 2, 3 (2002), 18–22. https://cran.r-project.org/doc/Rnews/.

[9] Chen Qian, Xiaochang Li, Qineng Wang, Gang Zhou, and Huajie Shao. 2024. NetBench: A Large-Scale and Comprehensive Network Traffic Benchmark Dataset for Foundation Models. In *Proceedings of the 2024 IEEE International Workshop on Foundation Models for Cyber-Physical Systems & Internet of Things (FMSys)*. IEEE Computer Society, Los Alamitos, CA, USA, 20–25. https://doi.org/10.1109/FMSys62467.2024.00008 Dataset available at https://github.com/WM-JayLab/NetBench.

[10] George A.F. Seber and Alan J. Lee. 2012. *Linear Regression Analysis* (2nd ed.). Wiley.

[11] Amin Shahraki, Mahmoud Abbasi, Amir Taherkordi, and Anca Delia Jurcut. 2021. Active Learning for Network Traffic Classification: A Technical Study. *arXiv preprint arXiv:2106.06933* (2021).