CSCI434/534 COLL 400 Capstone Machine Learning Web Traffic Analysis Project

Connor MacKinnon
College of William & Mary
Williamsburg, Virginia, USA
cdmackinnon@wm.edu

Thomas Eby
College of William & Mary
Williamsburg, Virginia, USA
tgeby@wm.edu

Abstract

Our objective in this project was to identify the source of network traffic. This approach could be used for an institution to block specific website traffic from their wireless network. This was done by analyzing flows of incoming and outgoing packet data. We opted to do our classification of websites: Weather.com, Discord.com, Wikipedia.org, GitHub.com, and LinkedIn.com. We performed logistic regression for a baseline and then tested Support Vector Machines (SVM) and Decision Trees for improved accuracy. Decision trees scored the highest on testing data with an accuracy of 96%. This implies that a decision tree or random forest could be effectively used for website classification and web traffic supervision.

1 Introduction

In educational or professional settings, an organization may wish to restrict access to certain websites deemed inappropriate or off-task. To enforce such an organization's policy, network administrators must be able to determine the source of some stream of network traffic, and this is done by utilizing machine learning classification models. We sought to create a resource to identify the destination of user traffic from the flow of packets.

We identified three issues that must be accounted for when building such a model. First, end-to-end data encryption. Our proposed method accounts for some forms of encryption by examining different IP datagram headers but not using a datagram's payload to inform our model. Instead, we utilize headers such as the *type of service*, *timestamp*, and *length*. The second threat is the interference of VPNs, but it has been shown in [1] that network administrators are already capable of detecting and blocking the use of VPNs on their networks. The final potential complication that we identified is peer-to-peer connections. A common form of website that is found inappropriate for use on school networks is online games. Because of this, it is likely that peer-to-peer traffic will need to be classified too, so the source IP address of incoming network traffic cannot be relied on to inform one's model. Our proposed design effectively mitigates this by not using the source IP addresses to inform our model.

Additionally, our tool could be useful in gathering analytics on the usage of a network. Perhaps an administrator of a network would like to collect information on the most commonly accessed websites to better understand their user base or answer other questions they may have. This tool could be modified to identify the traffic destination and increase a counter on the frequency of that website's access. The lightweight code for generating our models allows for it to be easily applicable to any network usage-related interests.

Our models were built on data from the following set of websites: Weather.com, Discord.com, Wikipedia.org, GitHub.com, and LinkedIn.com. Data collection was done using the Wireshark packet captures. To assemble our dataset we used this software to generate ten different streams of network traffic for each website and capture around 30,000 packets each time. After performing a packet capture we preprocessed the data to extract a range of identifying features as a means to differentiate between the websites. Then we assembled one CSV of all these statistics for each observation so that we may fit models and search for patterns later.

2 Proposed Method

2.1 Overview

Our preprocessing and model generation code is all available in our public GitHub repository: https://github.com/cdmackinnon/NetworkTrafficAnalyzer.

2.2 Data Collection

In order to build our models, we first needed to gather network traffic data from our chosen websites. To do this we used the Wireshark software by setting the host filter to the desired website, setting the software to begin data collection, and then visiting the website in a web browser. In that browser we clicked on various buttons and links until Wireshark showed that we had collected around 30,000 packets before pausing data collection. Finally, we exported the data as a CSV file named by the corresponding website (Fig. 1). We repeated this process ten times for each website with the only variation being that we saved the CSV files with identification numbers (1-9) appended to the website name when naming subsequent data for the same website. After all 50 streams of network traffic were collected, the CSV files were placed in a directory labeled 'Data' in our repository.

2.3 Data Preprocessing

Preprocessing our data was done with Python and the Pandas library. Once the preprocessing part of our program is linked to the data folder described above, it reads the CSV files

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.63	162.159.135.232	QUIC	1292	Initial, DCID=0e11ae1ec68da101, PKN: 1, CRYPTO
2	0.000007	192.168.0.63	162.159.135.232	QUIC	1292	Initial, DCID=0e11ae1ec68da101, PKN: 2, PING, PADDING, CRYPTO, PADDING, PING, PING, PADDING, PING
3	0.000214	192.168.0.63	162.159.135.232	QUIC	118	0-RTT, DCID=0e11ae1ec68da101
4	0.016030	162.159.135.232	192.168.0.63	QUIC	1242	Initial, SCID=01992d234b85496bd1998e236a8540472e4616bb, PKN: 0, ACK
5	0.016481	162.159.135.232	192.168.0.63	QUIC	1242	Initial, SCID=01992d234b85496bd1998e236a8540472e4616bb, PKN: 1, ACK, CRYPTO
6	0.016482	162.159.135.232	192.168.0.63	QUIC	1242	Handshake, SCID=01992d234b85496bd1998e236a8540472e4616bb
7	0.016841	162.159.135.232	192.168.0.63	QUIC	66	Protected Payload (KP0)
8	0.016841	162.159.135.232	192.168.0.63	QUIC	66	Protected Payload (KP0)
9	0.016841	162.159.135.232	192.168.0.63	QUIC	91	Protected Payload (KP0)
10	0.017043	192.168.0.63	162.159.135.232	QUIC	1292	Handshake, DCID=01992d234b85496bd1998e236a8540472e4616bb
11	0.017070	192.168.0.63	162.159.135.232	QUIC	85	Protected Payload (KP0), DCID=01992d234b85496bd1998e236a8540472e4616bb
12	0.017163	192.168.0.63	162.159.135.232	QUIC	89	Protected Payload (KP0), DCID=01992d234b85496bd1998e236a8540472e4616bb

Figure 1. Sample CSV Generated by Wireshark

statistics_output

packet_mean_timing	packet_median_timing	packet_std_timing	packet_mode_length	packet_mean_length	packet_median_length	packet_std_length	average_packets_per_second	average_bytes_per_second	TCP_percentage	TLS_percentage	UDP_percentage	QUIC_percentage	Website
0.007441593166468020	0.000112000000014440 0	0.047922179838914500	74	582.4583002382840	135.0	606.904972183321 0	134.43321557524700	78301.74223952500	0.6687847498014300	0.0	0.0	0.0	LinkedIn
0.002738568490878940	1.000000000139 78E-06	0.07432858425076740	1294	1109.5974801061000	1294.0	420.5710756009350	365.2753905152420	405308.6528604840	0.0	0.0	0.9512599469496020	0.0487400530503979 0	WhatsApp
0.005609889208042680	0.000187999999996330	0.018171832172417300	74	557.0849056603770	112.0	590.229410697726	178.32978575556100	99344.83187407200	0.6337161607875310	0.0	0.0	0.0	LinkedIn
0.006187548363095200	0.000101000000000035100	0.03679015984884460	74	662.4767571587950	303.0	632.7465742547690	151.14202724682100	100128.08008088000	0.6757158795091110	0.0	0.0	0.0	LinkedIn
0.005621452862530000	0.000140000000000029	0.027761269060065800	74	567.2429746401640	122.0	601.0138180930 80	177.95093945045500	100941.4202338880	0.636737491432488	0.0	0.0	0.0	LinkedIn
0.011576218450479200	1.000000000102796E-06	0.12642446795723900	1242	923.5165668662680	1242.0	477.4362332856760	86.41849368184220	79808.91059880910	0.005189620758483030	0.0	0.9800399201596810	0.0071856287425149 7	Discord
0.006855177573822830	0.000149999999996500	0.03347911164047700	1494	583.9262066214600	235.0	611.0450814135100	145.93335205766300	85214.30868658530	0.6190666134822500	0.0	0.0	0.0	LinkedIn
0.005530489385898410	0.000164499999999990100	0.026980863602011700	74	551.1254262978400	117.0	585.3294347534090	180.8843675945960	99689.97420118720	0.6384994316028800	0.0	0.0	0.0	LinkedIn
0.004931455830388690	9.999999992516E-07	0.05766367998736590	1242	928.1542386185240	1242.0	479.3239965541510 0	202.85949082267800	188284.89625106400	0.000784929356379280	0.0	0.9992150706436420	0.0	Discord

Figure 2. A Sample of Processed Data CSV

one by one into DataFrames which are then analyzed and summarized into one-row DataFrames containing statistics about datagram length, protocol usage, and the time elapsed between datagram arrivals as their feature values. Each one-row DataFrame also holds a target column that holds the label of the corresponding website obtained by parsing the CSV's file name. Each of these one-row DataFrames is concatenated into one DataFrame summarizing all streams of network traffic. Finally, that DataFrame is exported into a CSV file for later use (Fig. 2).

The statistics we used are packets per second, average bytes per second, percentages for the usage of different transport layer protocols, mode packet length, mean packet length, median packet length, standard deviation packet length, mean packet timing, median packet timing, and standard deviation packet timing. These statistics are generated from the time, protocol, and length columns of the data exported from Wireshark. The time column contains timestamps measured in seconds since the first packet's arrival. These timestamps are generated by Wireshark when each packet arrives [2]. The protocol values indicate the transport layer protocol used to deliver the associated packet. The length values indicate the associated datagram's length in bytes.

2.4 Packet Length Statistics

This includes the mean, median, and mode of the packet lengths. This is looking at the aggregate of all packets in a given capture. These metrics are then calculated to show the behavior of a site. These metrics have some covariance because of their related nature. For example, the mean is related to standard deviation because it is used in calculating it. However, these related metrics are still important to include because they capture slightly different aspects of the data that can vary across websites. Figure 3 and Figure 4 below show the contrast of these two features across each of the websites for all points in the data set. The dispersion of the points by class shows the differences in website tendencies.

2.5 Packet Timing Statistics

This includes the mean, median, and standard deviation of the difference in timing between packets. This metric focuses on the aggregate of time elapsed between consecutive packets. Figure 5 and 6 display the same standard deviation and mean metrics. These figures show dispersion and serve as validation for using our metrics.

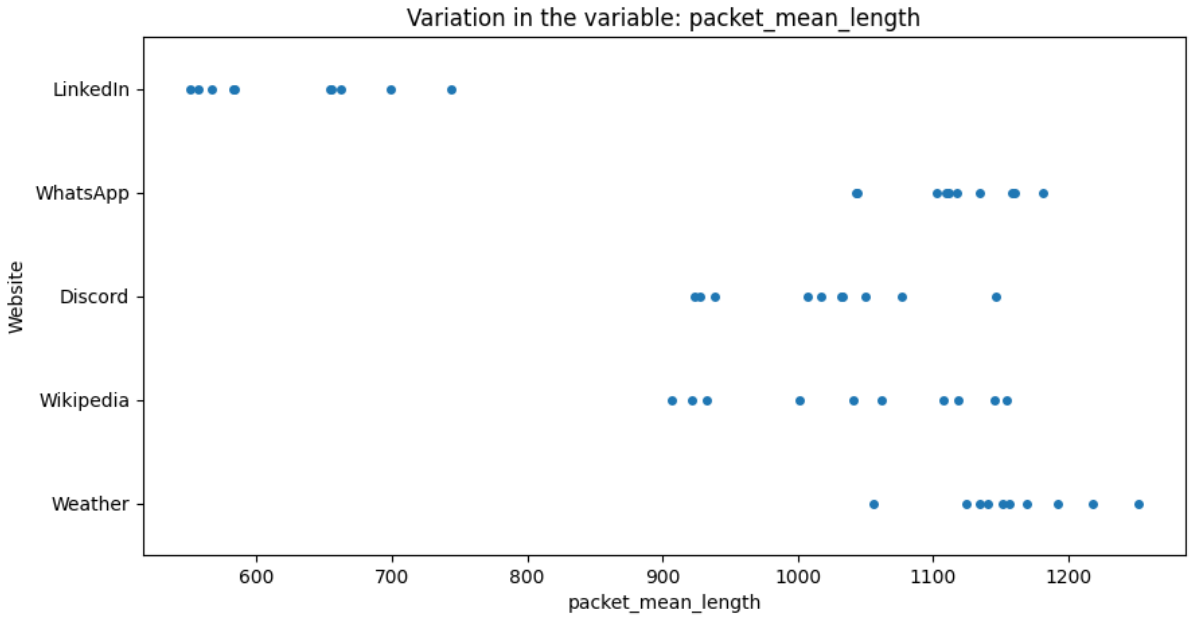

Figure 3. Mean of Website Packet Lengths

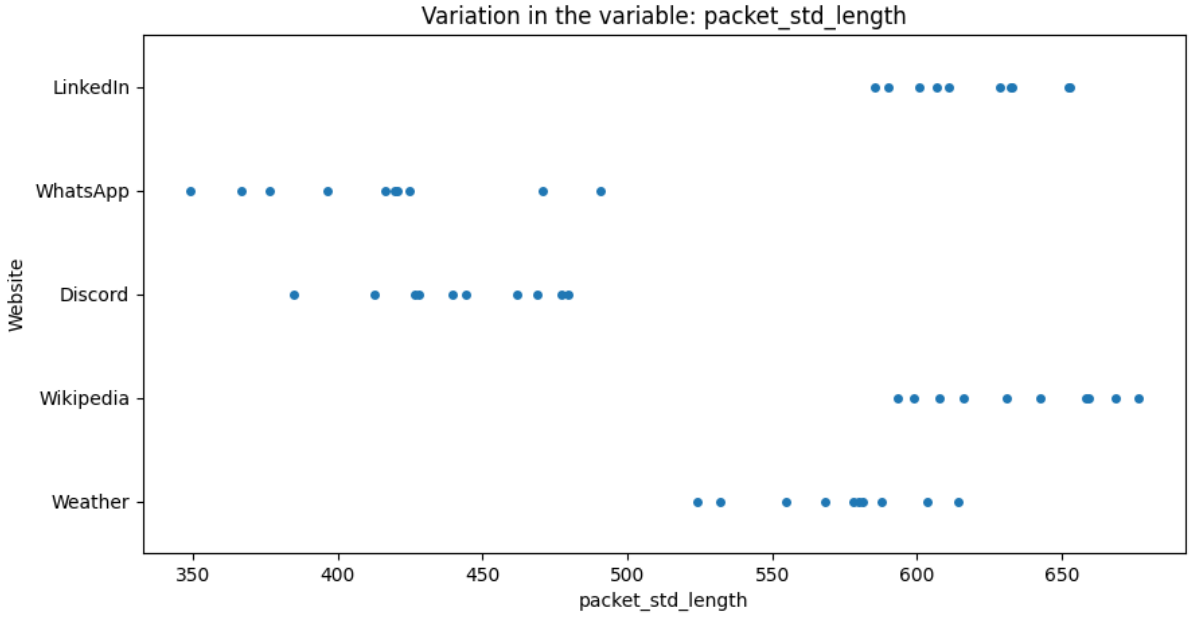

Figure 4. Standard Deviation of Website Packet Lengths

2.6 Extra Statistics

The remaining stats include protocol distribution, packets per second, and bytes per second. *Average packets per second* differs from packet timing because it considers the overall duration of the connection and how many packets are sent in total. Similarly, the *average bytes per second* considers the total number of bytes and the time elapsed over the entire capture. The protocol distribution looks at the types of transmissions a website uses over a given capture. The stacked

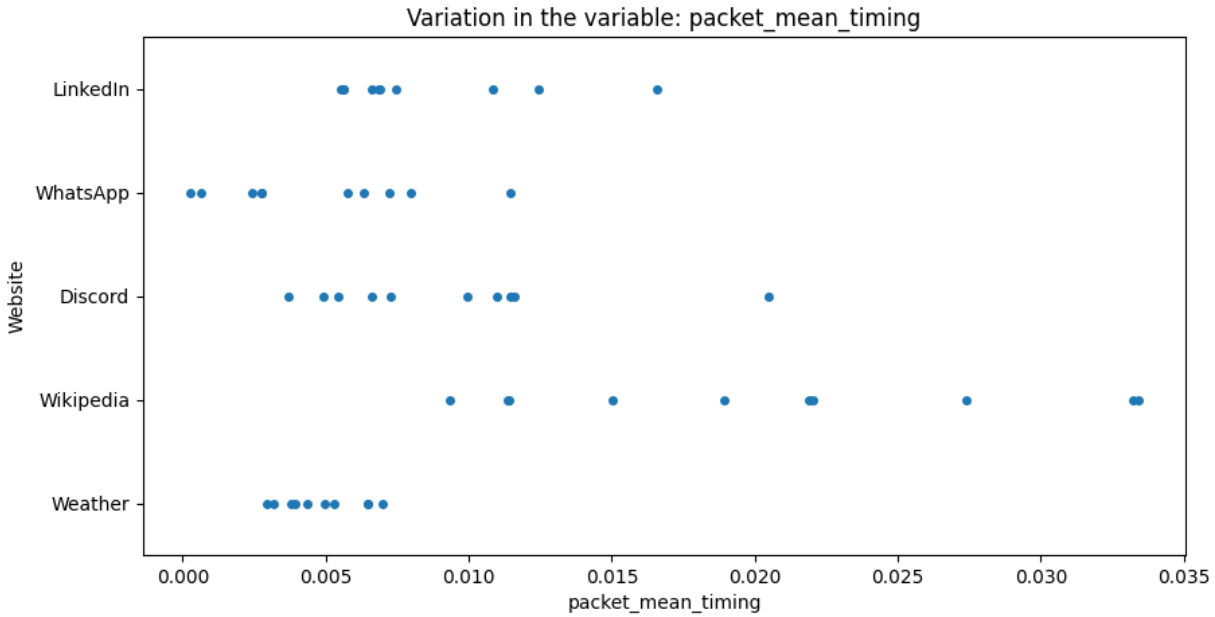

Figure 5. Mean of Website Packet Timing

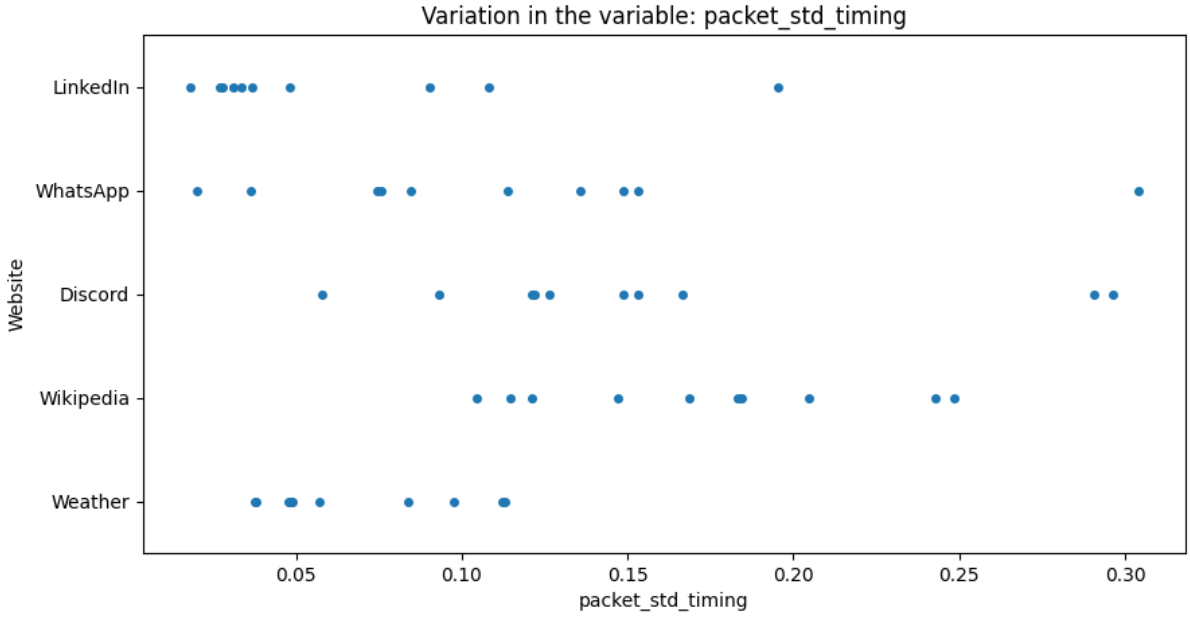

Figure 6. Standard Deviation of Website Packet Timing

bar plot Figure 7 shows the mean of the different proportions in each website. The protocol percentages differ for every observation, but this shows the average and justifies the usage of this feature as a differentiating factor.

After creating a cohesive CSV that condensed every packet capture's statistics we decided to design our baseline model using multinomial logistic regression. This is effectively logistic regression which creates a logistic curve based on weighted coefficients found using gradient descent. These

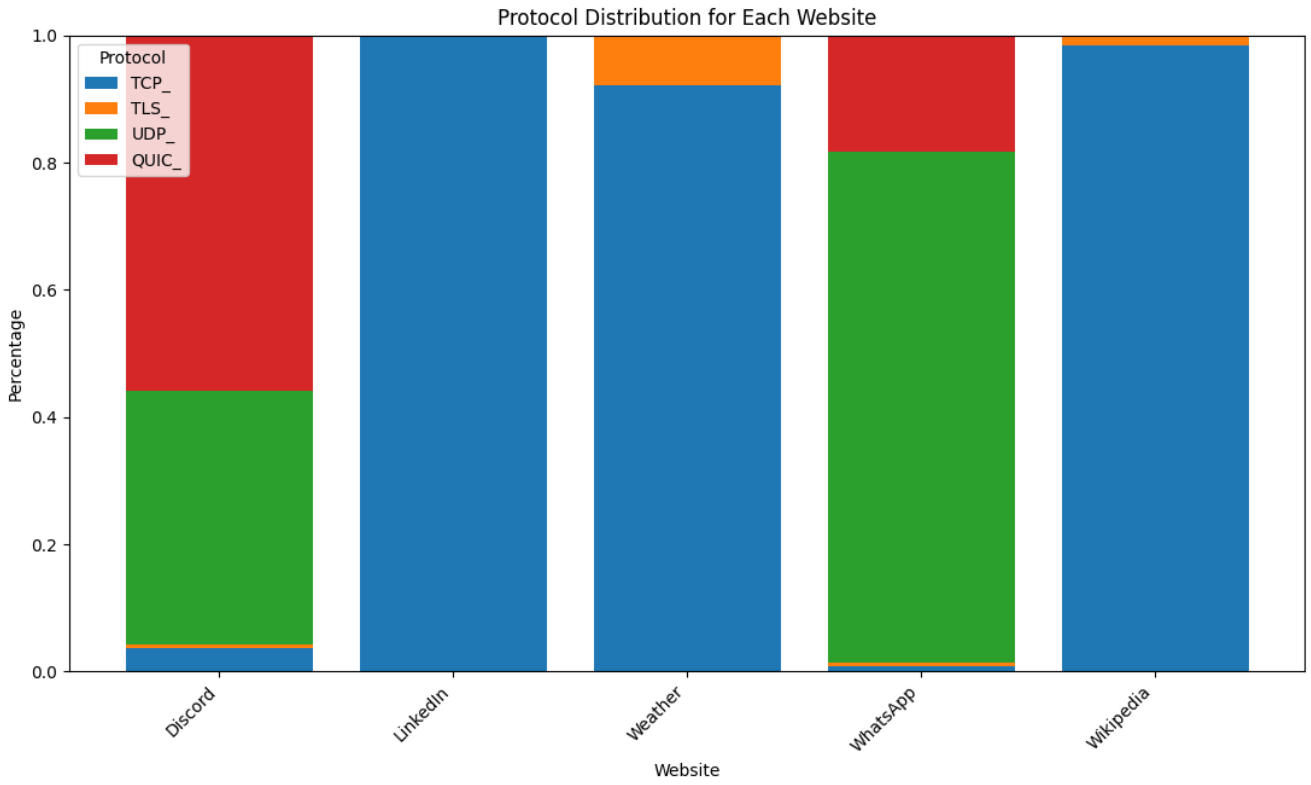

Figure 7. Average Percentage of Protocol Used Per Website

coefficients correspond to the different features of a data set. The difference for multinomial logistic regression is that there are several sets of estimates for each class and the ultimate prediction is made of the highest predicted probability. We also must scale the data for this method to help balance the weighting of features.

Using this methodology along with K-fold cross-validation to iterate through different groupings of training and testing data for training the model, we were able to get an average testing accuracy of 88% over 50 folds. This is a high starting metric to beat, but we found even more success with SVM and decision trees later.

The next method we used was multiclass Support Vector Machines (SVM). This aims to create hyperplanes in the 13-dimensional feature set and to maximize the margins between each of the five website classes. Maximizing this margin allows for the greatest distinction between websites. We also optimized the hyperparameter C which controlled the strength of the regularization term for drawing the boundaries. This term is designed to shrink the coefficient values of the boundaries.

After performing 50-fold validation on the model we found an average testing score of 92%. We wanted to visualize how well the SVM was working, so we decided to perform PCA and reduce the 13-dimension feature set to two features. These two features capture as much variance of the original data set as possible and allow us to get a better idea of how the decision boundaries appear.

Finally, we attempted to use decision trees to classify the packet flow traffic. Decision trees use recursion to repeatedly split and divide the data set into its classes based upon different feature thresholds. This does not require any regularization or scaling of the data because the features are considered in isolation on each branch of the tree. We correctly suspected this model would perform the best because it selects features that remove the most uncertainty at each level of the tree and lends itself to solving multiclass problems well.

We performed 50-fold cross-validation and received a testing accuracy of 96%. Many of these trees were performing nearly flawlessly and had a maximum depth of 3 meaning they were not overfit. An overfit decision tree makes too many specific decisions to classify every data point correctly. We had intended to try the random forest model if the decision tree was overfitted because this forest model would allow for smaller trees using bagging comprised of bootstrapping and aggregation, but our results consistently fit appropriately.

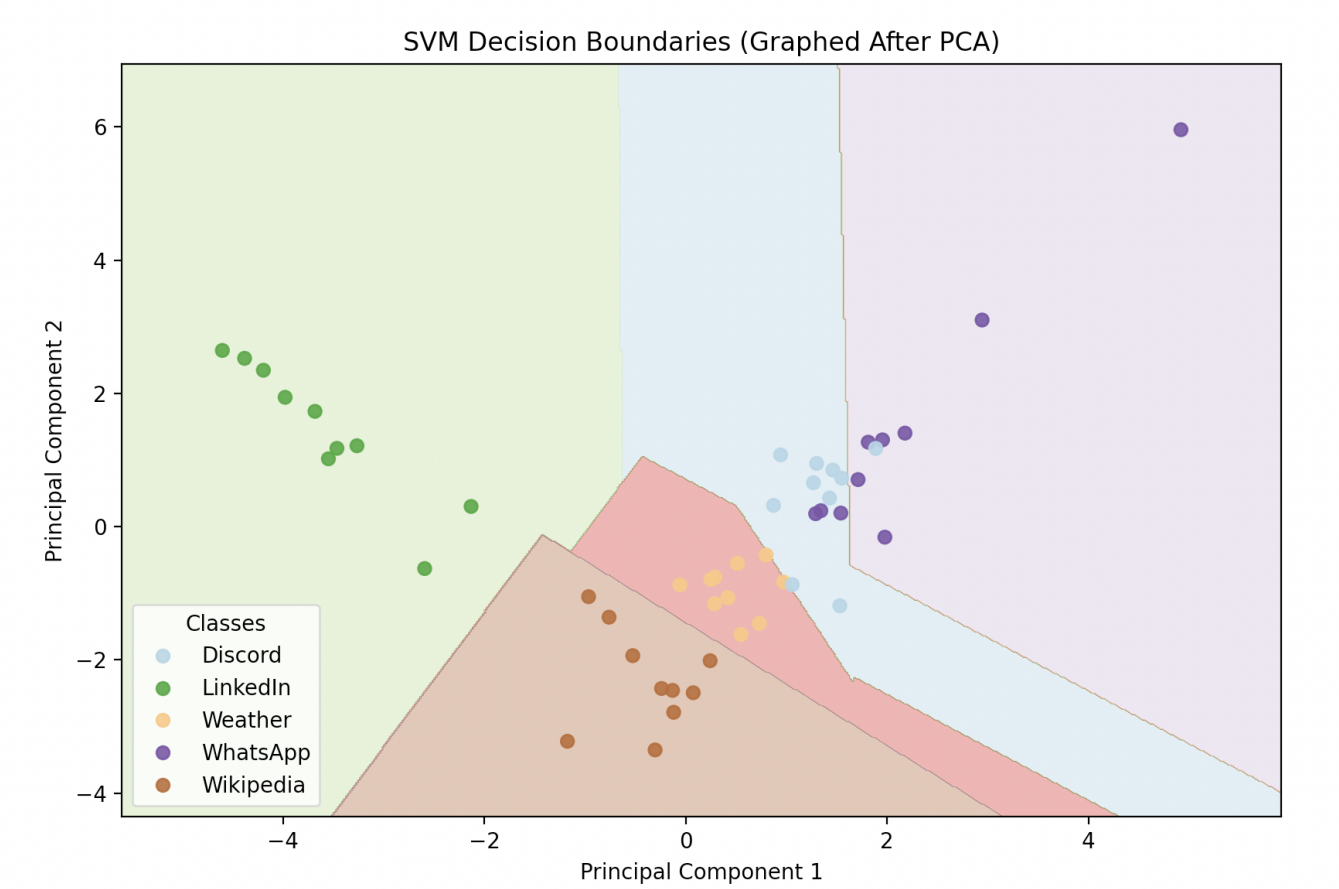

Figure 8. SVM Decision Boundaries In 2 Dimensions After PCA

3 Evaluation

In this section we review our methods for evaluating the efficacy of our models and present the resulting evaluation metrics.

3.1 Dataset

Our data is well-balanced with 10 observations per website for a total of 50 data points. Each observation contains summary statistics for a flow of roughly 30,000 packets.

3.2 Hyperparameter Optimization

For SVM we optimized the hyperparameter that controls the regularization of the model. What this means is that the larger the C value the more the model is penalized for a larger margin or misclassifications. We optimized this value by iterating through a range of C values with training and testing data and checking for the highest testing accuracy score. This value has to be balanced to ensure reliability within the model. We used it throughout the rest of the SVM process.

3.3 Evaluation Metrics

We evaluated our models based on their average internal and external classification accuracy. Accuracy is calculated by summing the total amount of correct classifications and dividing that sum by the total number of observations. Internal accuracy refers to the ability of a model to accurately classify the data it was trained on. External accuracy refers to the ability of a model to accurately classify data it was not trained on.

3.4 Evaluation Process

We utilized KFold cross-validation to calculate the average internal and external accuracies of our models. Specifically, we used KFold with 5 and 50 folds. The way this works is by splitting the data into evenly sized folds; since our dataset has 50 observations, KFold with 5 folds produces 5 sets of 10 observations and with 50 folds, 50 sets are produced with only 1 observation in each. The cross-validation process creates 1 model per fold, and uses each fold as the testing data for one of those models. The remainder of the data is used to build the models. To obtain the internal validation of a model, it tries to classify its testing observations. The

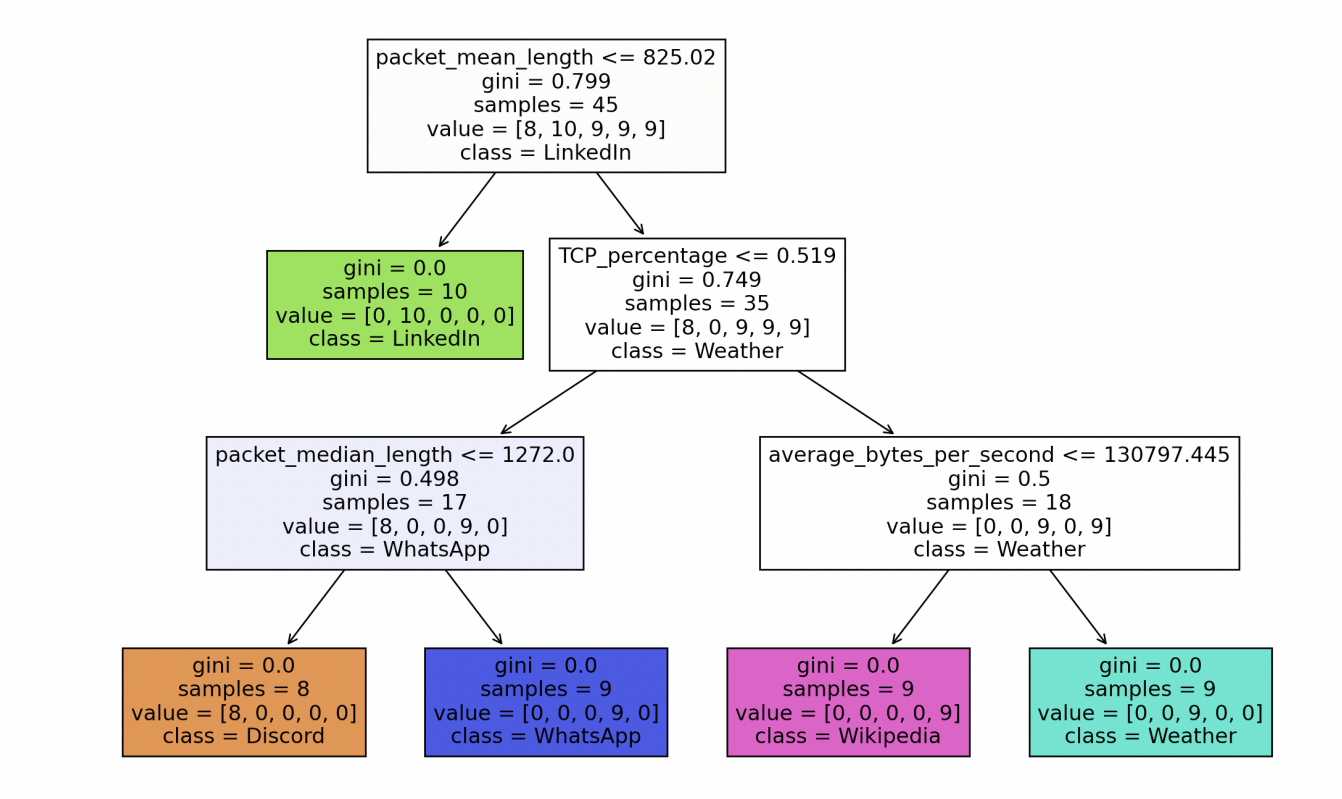

Figure 9. Figure 9: A Perfect Accuracy Decision Tree

	Average Internal Accuracy	Average External Accuracy
LR 5-Fold	0.945	0.86
LR 50-Fold	0.9412244897959183	0.88
SVM 5-Fold	0.9949999999999999	0.9
SVM 50-Fold	1.0	0.92
DTC 5-Fold	1.0	0.96
DTC 50-Fold	1.0	0.96

Figure 10. Cross-Validation Accuracies

resulting classifications are compared with the actual classes to calculate an accuracy score. Similarly, to calculate the external validity the model tries to classify the testing fold's observations, and the predicted classes are compared to the expected values to calculate the accuracy value. The internal and external accuracies are stored in their corresponding lists, and after all folds have been evaluated, both lists of accuracy values have their values averaged to obtain the final internal and external accuracy metrics. These values are displayed in Figure 10 above. We are very satisfied with our results and feel that our project met its objectives. Specifically the accuracy of the decision tree at 96% and SVM at 92% indicates our model is correct well above the majority of the time.

3.5 Strengths and Weaknesses

Each of our models had its set of advantages and disadvantages. One of our primary concerns was the risk of overfitting the model to our data. While 50 data points, each representing aggregates of thousands of other data points, provide some reliability it is always a concern to make your model too closely tailored to the training data so that it might not transfer and identify new data quite as well

Logistic regression and SVM prevent overfitting via regularization and controlling how strict their threshold is for predicting a class. If they're too strictly taught the training data then their decisions capture too much of the specific set of variances appearing in the training data.

Decision trees on the other hand can quickly become overfit if the depth of the tree grows too large. This would mean that minor details in the data set are being over-scrutinized. Fortunately, our trees remained around a depth of three and never had this issue. We theorize that this could be attributed to the relatively small number of website classes. With more classes, the tree might need to make finer distinctions between each, potentially leading to overfitting.

Another potential weakness within our approach is the existence of multicollinearity due to the close relations of the metrics we selected as features. If there are strong relationships between the features, or if new data is added and not checked accordingly, the model could suffer in accuracy.

4 Discussion & Future Work

The motivation behind our project was to be able to identify if some network traffic data originated from one of our chosen websites, and with our primary model having an overall accuracy of ~96% after rigorous testing shows that we have made significant progress toward a robust method of determining the source of network traffic. A major limitation we faced was our method of collecting data. Since we built our models based on flows of data, it took a significant amount of time to collect a single datapoint. A potential solution to this would be to develop a method of collecting the data automatically by setting up a script to click on certain coordinates of a computer screen and enter keystrokes to activate Wireshark, visit the website, and export the data. An interesting direction we could take our research in the future is to see how our processed statistics fare against forms of data encryption that introduce additional variation in packet sizes by padding data.

A question that we still have is how many websites our model might be able to predict as well as if it is able to identify the different types of websites if given the proper amount of data. We only have 5 websites and this works well with the decision boundaries of SVM and the branches of decision trees, however, the complexity of this may grow multiplicatively with more websites to identify. We wonder if a neural network or more complex model is more suitable to grow in this way. Another consideration we had is how well we can classify types of websites. Currently, we can identify specific websites that are getting traffic, but could our model identify, for example, retail sites, social media sites, etc?

Another question that remains is whether or not our data set had any bias. When collecting the packets and ending the capture sometimes there might only be 25,000 packets or other times 50,000. Does this level of variance introduce any bias into the data set? There could also be extraneous factors such as the time of access on a website. Not all of these data points were gathered at exactly the same time and a server speed can be incredibly dynamic. The model could've potentially learned the signature for a data set collected during a busy time of day when the site would be categorized as much slower. If given the computational power and the proper data, seeing how the models fared against 1,000 data points with 100,000 packets each could test the overall efficacy.

5 Conclusion

In conclusion, our study has shown that machine learning models are an effective way to classify web traffic to different websites based on packet data. Using logistic regression, support vector machines, and decision trees we achieved reliable accuracy metrics for all three methods. This paper shows promising results in gathering web data from packets and would benefit from future work in data collection automation, handling biases, and scaling the model to increase its use cases and robustness.

6 REFERENCES

1. "VPN Blocker, Types & How To Avoid VPN Blocks," Fortinet.com. Retrieved May 5, 2024 from: https://www.fortinet.com/resources/cyberglossary/vpn-blocker.
2. R. Sharpe, E. Warnicke, and U. Lamping, "Wireshark User's Guide," Wireshark.org. Retrieved May 5, 2024 from: https://www.wireshark.org/docs/wsug_html_chunked/ChAdvTimestamps.html.
3. IBM. What is logistic regression? Retrieved May 5, 2024 from: https://www.ibm.com/topics/logistic-regression.
4. IBM. 2023. What are support vector machines (SVMs)? Retrieved May 5, 2024 from: https://www.ibm.com/topics/support-vector-machine.
5. IBM. What is a decision tree? Retrieved May 5, 2024 from: https://www.ibm.com/topics/decision-trees.