Classifying network traffic of social networks of different types

Tyler Pringle
Department of Computer Science
College of William & Mary
Williamsburg, VA, USA
tmpringle@wm.edu

ABSTRACT

In this paper, I discuss the research I undertook to develop a model for the purpose of network traffic classification. I approached this problem from the angle of social networks, seeking to create a model that was able to differentiate traffic that came from social networking sites in different categories. To this end, I utilized a stream-based approach to network classification and compiled a dataset of network traffic streams from Glassdoor, X, and Stack Overflow. The model I chose to handle this problem is a decision tree classifier implemented through Python and the scikit-learn library. After detailing the model (alongside a baseline logistic regression classifier), I discuss the potential applications of my results and future work that could be done to build off of my research.

KEYWORDS

Network traffic classification, decision tree, social networks

1 Introduction

As the Internet has grown into a network of networks that is ever-increasing in size over the years, we deal with an ever-increasing amount of network traffic that goes through our devices every single day. And yet, as the Internet has grown, traffic has also begun to consolidate towards just a handful of large websites and applications [1]. As this consolidation continues, what we want to better understand is this: what hubs on the Internet are traffic mainly going to and from? How are users choosing to connect in this age of large social media networks?

Although these are not questions that this paper will attempt to answer, we do need to understand how to even try to tackle and answer these questions in the first place. We need to classify the traffic that is going to/from different kinds of large sites and find out what differentiates the traffic that goes to/from one site from the traffic going to/from another. We need methods for network traffic classification.

1.1 Project Overview

1.1.1 Dataset. Before even getting into how to classify network traffic, I needed to decide what kind of sources I wanted to base my data on. Since we are tackling this network traffic classification problem from the angle of which social networks are getting the most traffic, I endeavored to include different kinds of social media networks. I knew that although my dataset should include an obvious example of a major social media site, such as Instagram, X, or Facebook. However, I felt it would be best for the dataset to be more diverse and not only include sources of this kind of social network. Rather, I wanted to focus on social networks with different kinds of purposes (e.g. for work or for getting questions answered). Thus, I settled on three sites from which to classify network traffic: **Glassdoor** (career networking), **X** (social media), and **Stack Overflow** (Q&A for programming questions). Beyond just seeing if I could classify the traffic, I wanted to see what features of the traffic differentiated, say, a site like Glassdoor from a site like X.

Figure 1: Websites used in dataset

Dataset Format. The final dataset was a summary of packet streams. A stream, in this case, is a continuous capture of network traffic represented as a list of packets. To be more specific, each data sample was associated with its own continuous capture of network traffic from one of the above websites. A data sample included summarization data about the network capture, such as the number of packets included in the capture and the average length (in bytes) of a packet in the capture.

1.1.2 Methodology. Once packet streams were captured and summarized, these streams would be fed into an AI model that would train and test itself on the dataset.

However, it should be noted that at the start of the research project, I actually set about to create a model that classified the origin of individual packets rather than the origin of a stream of packets. This start ended up having quite an influence on the models I would choose to develop and analyze, as the packet-based models and stream-based models had significant differences in effectiveness. Nonetheless, in both the packet- and stream-based approaches, I utilized a logistic regression classifier as my baseline model and a decision tree classifier as my main model to develop and test on. I settled on a decision tree as my main model because I wanted the model to be more easily interpretable; that is, I wanted to know *how* the model classifies the origin of network traffic the way it does.

1.1.3 Summary of Results. Surprisingly, with the final stream-based method of traffic classification, both the logistic regression and decision tree classifiers achieved **100% accuracy** and a **1.0 macro F1 score** on the testing dataset. In other words, both models, using the stream-based approach, correctly identified the origin website of each respective packet in the testing datasets the models worked with. Additionally, I have concluded that there were two extracted features in the dataset that seem to uniquely identify the origin of a stream of packets among the three websites analyzed. These two features were the **most common packet length** and the **average packet length** of the stream. I will delve deeper into how I reached this conclusion later in this paper.

1.1.4 Novelty of Approach. The choice of the main model to use, the decision tree, was not particularly novel. Both the decision tree model and the related random forest model (which utilizes a group of decision trees to make predictions) are often cited as top models for use in multiclass classification problems. However, the inclusion of just the decision tree model rather than the random forest model is not insignificant, as the random forest is seen as having higher model accuracy due to the use of multiple decision trees [2]. I chose not to pursue a random forest model after seeing the effectiveness of the more basic decision tree classifier.

2 Proposed Methodology

From a bird's-eye view, this was the methodology of the project: for each website, I captured **15** streams of network traffic data using Wireshark (for a total of **45** streams). I then exported the data to CSV format and created a Python program that uses the Pandas library [3] to automatically summarize each stream with important features, such as the most common packet length and average packet length in a particular stream. Once these features were extracted, I researched which multiclass classifiers to use, and I settled on logistic regression as a baseline and a decision tree model as the main model for the project. I implemented these models in Python through the scikit-learn library [4] and developed them with the dataset I compiled using a training/validation/testing split of 8:1:1 (80% of the dataset was used for training the models, 10% for validation, and 10% for testing).

2.1 Data Collection

2.1.1 Initial Data Collection. Wireshark was utilized to capture traffic going between my laptop and each of the three websites I chose to analyze. During each stream capture, I used a capture filter (i.e. `host glassdoor.com`) to exclusively capture network traffic that was going to or from Glassdoor's website. In the dataset, each website has **15** network traffic streams, for a total of **45** streams of network data. Each traffic stream records about 30 seconds of continuous navigation through one website, which amounted to me continuously clicking on links to access new parts of each website. Each stream contains anywhere from around 300 packets to almost 12,000 packets, though most streams contain less than 1,000 packets. A sample of an individual network stream can be found in **Figure 2**. Each packet in a stream contains its order in the stream, the time it was captured (since the start of the capture), the source and destination IP addresses of the packet, the protocol (i.e. TCP, UDP) of the packet, the length of the packet in bytes, and extra information that Wireshark adds on to packets to provide extra context (such as the source and destination ports of the packet).

2.1.2 Extracted Features and Final Dataset. Once the network traffic streams were captured, I created a Python program to automatically summarize important features about an individual stream and place this information in a CSV file alongside stream summaries from the same website. The Python program handles one website per run and takes in the CSV files representing the ~30-second stream of packets one at a time. I manually labeled which website the network stream came from, and the code extracted these nine features from each stream:

1. **Packet Count** - the number of packets captured in an individual stream
2. **Total Length** - the sum of the length of every packet (in bytes) in an individual stream
3. **Average Packet Interval** - the average amount of time in between two packets being captured
4. **Maximum Packet Interval** - the largest amount of time in between two packets being captured
5. **Minimum Packet Interval** - the smallest amount of time in between two packets being captured
6. **Average Packet Length** - the average length (in bytes) of a packet in an individual stream

No.	Time	Source IP	Destination IP	Protocol	Length	Info
1	0	192.168.0.183	104.17.64.70	UDP	1292	52377 > 443 Len=1250
2	0.000125	192.168.0.183	104.17.64.70	UDP	1292	52377 > 443 Len=1250
3	0.000159	192.168.0.183	104.17.64.70	UDP	393	52377 > 443 Len=351
4	0.026158	104.17.64.70	192.168.0.183	UDP	68	443 > 52377 Len=26
5	0.05115	104.17.64.70	192.168.0.183	UDP	334	443 > 52377 Len=292

Figure 2: A sample of some of the data in a Wireshark capture of network traffic (a packet stream) from Glassdoor

7. **Maximum Packet Length** - the smallest length (in bytes) of a packet in an individual stream
8. **Minimum Packet Length** - the smallest length (in bytes) of a packet in an individual stream
9. **Most Common Packet Length** - the most frequently appearing packet length (in bytes) in an individual stream

Justification for Extracted Features. Different websites and applications are likely to send and receive differing amounts of information/traffic in order to operate for a single user. This is what makes the features related to packet count and length (*total length, average/maximum/minimum packet length, most common packet length*) crucial. All these metrics are likely to vary based on the type of information that is being sent to and from the client device.

A site that includes a lot of images and videos (like X) will likely have larger average packet lengths compared to a site that primarily consists of text (like Stack Overflow). Or, at the very least, the streams from a site like X will likely have a larger total length due to more information being set to render images and videos. This comparison is fair, as each stream was captured over a similar amount of time. The features based on packet intervals (*average/max./min. packet interval*) can also be important, as certain websites may have a tendency to send packets in quicker succession than others.

Once each website's streams were summarized in its own CSV file (with 3 files total), I manually combined these 3 CSVs into a single CSV to create the final 45-stream dataset. The last modification I made to this final CSV for the dataset was changing the labels representing which website each stream came from. Instead of strings representing the origin website, I used numbers to identify which website a stream was from. Specifically, I used **1** for **Glassdoor**, **2** for **X**, and **3** for **Stack Overflow**. A sample of the final dataset can be seen in **Figure 3**.

2.2 The Models

As mentioned previously, this project utilizes a logistic regression classifier (as a baseline) and a decision tree classifier to classify the streams of network traffic. Both of these models were implemented using the scikit-learn library in Python.

2.2.1 Choosing the Models. The guidelines for this project encouraged us to use a simple linear classifier that was easy to implement. Because scikit-learn provides a **logistic regression classifier** [5] that is ready for use without even passing in any parameters, it made sense to use this model as my baseline. The choice of the **decision tree classifier** as my main model came out of my desire to make my model as interpretable as possible. Decision tree models are known for being interpretable, as their structure can be summarized in a tree diagram that identifies how the model makes decisions. This advantage in interpretability is something that the decision tree model has even over the random forest model. As mentioned in [6], while a random forest gains predictive power by utilizing multiple decision trees, the model loses out on the interpretability of a single decision tree in the process. Ultimately, this is why I settled on the more simple decision tree classifier, as it made it easier to figure out how the model classified traffic the way it did in the end.

2.2.2 Technical Specifications. As I mentioned above, the scikit-learn Python library was used to develop both AI classification models. The logistic regression classifier is based on the `sklearn.linear_model.LogisticRegression` class. In fact, the default parameters were used, as I did

Label	Packet Count	Total Length	Avg. Packet Interval	Max. Packet Interval	Min. Packet Interval	Avg. Packet Length	Max. Packet Length	Min. Packet Length	Most Common Packet Length
Glassdoor	1974	2088574	0.0146	11.412	0.000011	1058.0415	1514	54	1242
Glassdoor	451	299928	0.0834	14.686	0.000024	665.0288	1292	54	1292
X	541	487314	0.0661	2.993	0.000026	900.765	1414	54	1414
X	631	629150	0.0668	2.833	0.000025	997.0681	1414	54	1414
Stack Overflow	509	387878	0.0533	5.169	0.000022	762.0393	1514	54	1514
Stack Overflow	417	281669	0.0845	6.345	0.000025	675.465	1514	54	1514

Figure 3: A sample of the final dataset, representing information about 6 traffic streams, 2 from each website (note that in the final dataset, the string labels have been replaced with categorical numbers)

not want to attempt to improve the baseline model. The only significant modification I made when running the logistic regression model was actually to the dataset rather than the model itself. I used the `sklearn.preprocessing.StandardScaler` class to standardize the features in the dataset into standard normal distributions.

The decision tree classifier is based on the `sklearn.tree.DecisionTreeClassifier` class. The `sklearn.tree` module also provides the useful show_tree function, which represents the model as a simple tree diagram that we can use to understand how the model works. The exact tree diagram for the model used in the project can be found in **Figure 4**. This model was not actually modified too much either; the only parameter I passed in was `random_state=0`, which made the fitting behavior of the model deterministic rather than unique on each run of the model. Additionally, I chose *not* to standardize the dataset when running the decision tree in order to make the model even more interpretable.

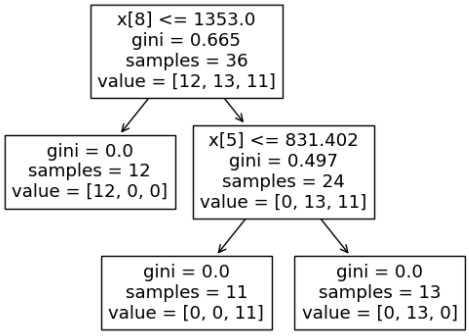

Figure 4. Diagram of the fully-trained decision tree classifier

Interpreting the decision tree diagram. In a decision tree, the tree starts with the root node. Before the model is fitted to the data, the root node represents all of the data in the dataset. Once the model is trained, though, we get a tree like the one we see above. In a multiclass classification such as this one, each leaf node represents a subset corresponding to one of the classes. In this case, these classes would be Glassdoor, X, and Stack Overflow, the origin websites of the network traffic. The internal nodes, or decision nodes, determine how the model chooses to split up the overall dataset into subsets/classes. In the representation, the decision criteria are the first lines in the internal/decision nodes, that is, `x[8] <= 1353.0` and `x[5] <= 831.402`. Essentially, the trained

model takes in a data point and analyzes it based on the decision criteria to decide which root node/subset/class the data point belongs to.

Let us analyze the first decision criterion, `x[8] <= 1353.0`. x represents the final dataset with the labels removed, so just the summaries of the traffic streams without identification of the origin website. In this case, `x[8]` identifies the data for the 9th feature (not including the label) since we are indexing from 0. If we look at the sample of the final dataset in **Figure 3** again, we see that the 9th feature is the most common packet length.

Again, the entire criterion is `x[8] <= 1353.0`. So, what this decision tree diagram is telling us is that, once we pass in a summary of a network stream to predict its class, the model first checks whether the most common packet length of the network stream is less than or equal to 1353. If this is true, then the model takes us to the left branch, where we stop at the leaf node. This means the network stream is now classified. We can even tell from the tree diagram which class the left-most leaf node represents: the `value` array of a node represents the number of each class that is represented in that node, while the index of the array indicates which class the count is for. The first index/position thus shows how many of class 1 (the Glassdoor class) are in the subset for the node. As I will discuss later, this model has been 100% accurate with the data I fed into it, so we know for sure that the leftmost leaf node is the node for the Glassdoor class.

To sum up, from just the simple tree diagram, we can conclude that Glassdoor network traffic streams are characterized by having the most common packet lengths be less than or equal to 1353.

2.2.3 Training the Models. There is an 8:1:1 training/validation/testing split in the dataset, meaning that 80% of the dataset is used for training, while 10% is used for validation, and another 10% is used for testing. This split was done automatically through the scikit-learn library using the `train_test_split` function that is a part of the `sklearn.model_selection` module. I used the same random seed (`random_state=42`) when doing the dataset split for both the logistic regression and decision tree models, so both models were trained and tested on the exact same data. The validation datasets are the same for both models, but I did not validate the logistic regression model. This is because I wanted to modify the logistic regression model as little as possible to keep it as a baseline for the performance of the decision tree model. Ultimately, though, the validation just acted as a secondary testing stage for the decision tree model, as I did not make any modifications to the model after validating the dataset.

3 Evaluation

3.1 Testing Methodology

There are **5** network streams out of the 45 total that comprise the testing dataset. These exact data points can be found in **Figure 5**. The dataset was scaled for the logistic regression model, but *not* for the decision tree model. I could have scaled the data for the decision tree, but it would have made the model less interpretable. We would not quickly be able to determine how the original values in the final dataset would be handled in the decision tree algorithm if all the data was standardized.

Label	Packet Count	Total Length	Avg. Packet Interval	Max. Packet Interval	Min. Packet Interval	Avg. Packet Length	Max. Packet Length	Min. Packet Length	Most Common Packet Length
Glassdoor	913	702620	0.0429	3.205	0.000023	769.573	1292	65	1242
X	678	690483	0.0660	3.353	0.000036	1018.412	1414	54	1414
Stack Overflow	606	412975	0.0512	3.743	0.000023	681.477	1514	54	1514
Stack Overflow	629	436184	0.0478	3.372	0.000027	693.456	1514	54	1514
Glassdoor	5609	5827682	0.00526	3.205	0.000006	1038.988	1242	65	1242

Figure 5. The testing dataset (note that in the final dataset, the string labels have been replaced with categorical numbers)

3.2 Results

3.2.1 Accuracy Metrics Used. The two metrics used to measure the effectiveness of the models are **accuracy** and the **macro F1 score**, which are defined below:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

where TP = true positive, FP = false positive, TN = true negative, and FN = false negative. Accuracy models the number of correct predictions among all predictions made, while the macro F1 score represents the amount of times a model made correct predictions across the whole dataset.

3.2.2 Results on Testing Data. Both models made predictions on the 5-stream testing dataset. These are the accuracy results of those predictions:

Evaluation Results on Testing Data
(testing data is 10% of entire dataset)

Model	Accuracy	Macro F1 Score
Logistic Regression	1.0	1.0
Decision Tree	**1.0**	**1.0**

In essence, all the classifications that the models gave the five network streams were correct. We can also look at the confusion matrices for both the logistic regression and decision tree models in **Figures 6 and 7**, respectively.

I would say that this project succeeded in achieving its goals. Not only do I have a model that effectively classifies the origin of network traffic among three social media networks, I also know which features the decision tree model identified as being the most important identifying characteristics of a network stream.

4 Discussion

4.1 Analysis of Results

4.1.1 Insights on the dataset. The 100% accuracy was not at all what I was expecting when I switched to a stream-based approach to classifying network traffic.

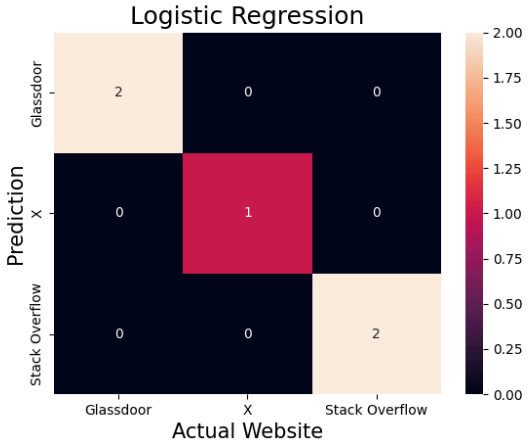

Figure 6. Confusion matrix for the logistic regression classifier's predictions on testing data (using stream-based approach)

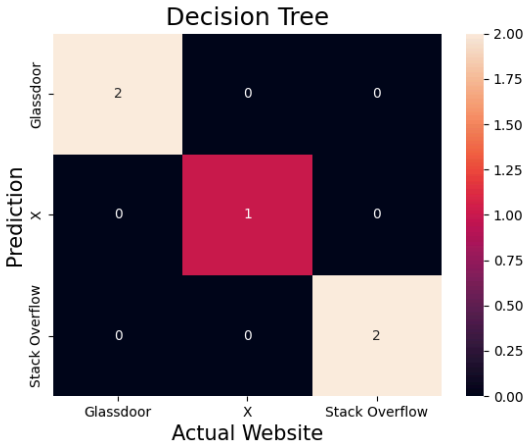

Figure 7. Confusion matrix for the decision tree classifier's predictions on testing data (using stream-based approach)

Early on, when the model was classifying the origin of individual packets based on two extracted features, the protocol and the length, the testing accuracy and macro F1 scores looked more like this:

Results on Testing Data w/ Old Packet-based Approach

Model	Accuracy	Macro F1 Score
Logistic Regression	0.78	0.75
Decision Tree	**0.88**	**0.88**

Most influential extracted features. The decision tree model was fairly accurate in identifying the origins of

individual packets, but certainly not 100% effective on the testing data (which included 602 individual packets). One may compare the packet-based approach and the stream-based approach and note that the packet-based approach only includes 2 features in the dataset while the stream-based approach includes 9. Generally, more dataset features allow a model to be more accurate. While this is fair to point out, we must also note that the final decision tree model for the stream-based approach only utilizes two features to classify streams of network traffic: the most common packet length and the average packet length of a stream. The two features (protocol and length) of the packet-based approach do not necessarily identify a packet's origin uniquely, while two features (most common packet length and average packet length) of the stream-based approach appear to identify a network stream's origin uniquely, at least among the 3 websites I chose to analyze.

As discussed earlier on in the paper, we can analyze the tree diagram for the model (**Figure 4**) and deduce that the tree classifies network streams as being from Glassdoor if their most common packet length is ≤ 1353 bytes. Additionally, we can further analyze the tree by looking at how it classifies a network stream if its most common packet length is > 1353 bytes. In this case, the decision tree checks the stream's 6th feature, its average packet length, and checks to see if this length is less than or equal to 831.402. Thus, if a stream's most common packet length is > 1353 bytes and its average packet length is ≤ 831.402, then the stream goes to the middle leaf node, which represents the class with label 3, Stack Overflow. Conversely, if a stream's most common packet length is still > 1353 bytes but its average packet length is > 831.402, then the stream is classified as a traffic stream from X.

In short, these are the identifying characteristics of each website's network streams:

1. **Glassdoor** - Most common packet length ≤ 1353
2. **X** - Most common packet length > 1353 and average packet length > 831.402
3. **Stack Overflow** - Most common packet length > 1353 and average packet length ≤ 831.402

We can verify these classifications as accurate with a glance at **Figure 3** and by looking through all 45 streams in the final dataset. All the streams correspond to this simple classification based on just two extracted features.

4.1.2 Applications and Relevance to Problem. These results imply that the most common packet length and the average packet length could potentially be powerful identifiers not just for network streams related to Glassdoor, X, or Stack Overflow but for network streams related to all kinds of social media sites and applications. While I could see a model like this becoming less effective at identifying the origins of individual network streams if we expanded the classification problem to many more sites/applications, I believe the model could stay effective by making the classes of social networks more categorical. After all, I chose Glassdoor, X, and Stack Overflow because they are social networks that I consider to be in different categories (career networking, social media, and Q&A, respectively). This project's results indicate that a social media network like X, which would likely allow a lot of photo and video, is also likely to have network traffic streams with a relatively high most common packet length and average packet length (due to the larger size of photo and video compared to just text).

4.2 Recommendations for Future Work

4.2.1 Limitations of Project. I would have liked to include more social networks in the dataset. It could be that the effectiveness of the model drops off once we increase the amount of classes in this classification problem, and it would be interesting to see just how effective a decision tree model is as the number of classes increases. Or, at the very least, if the decision tree structure stays effective, I would like to see what extracted features are used to classify network traffic. Are the most common packet length and the average packet length still the most important identifiers for a network stream as you consider more social networks? This is a question that I would love to see explored.

4.2.2 Modifying the Dataset. As I mentioned near the end of the analysis of the results, I believe the model may need to shift to classifying network streams into different categories of social networks. Although I do not have any particular suggestions on which categories to choose, I would suggest dividing categories based on the types of media (e.g. text, photo, video, etc.) that are usually included in that category. My work here seems to imply that that factor has an effect on the packet lengths, at the very least. My hypothesis is that if one starts to include more social networks in the dataset, there will begin to be websites that have very similar features in their network streams. I would further hypothesize that social networks with similar features in their streams would likely be part of the same category.

5 Conclusion

Although the Internet is sure to evolve over the next few decades, perhaps into a form completely unrecognizable to us in the present, it is still important for us to understand how Internet users are connecting in our current age. In this age of consolidated networks, we

already have a general idea of where most users are spending most of their time. However, if we want a more exact picture, this is where network traffic classification can come in. It is useful to know exactly where (or at least how, if we are dealing with categories instead of sites/applications) users are spending their time, especially if you are a marketer or researcher trying to extend your reach out to as many users as possible. Further, as the number of Internet users inevitably increases, it becomes more and more pragmatic to utilize machine-learning models that classify this traffic.

This research concludes that a stream-based approach to network traffic classification utilizing a decision tree with the most common packet length and average packet length being primary features in the dataset seems to be a good foundation for further research into improving network traffic classifiers. Refining the dataset by either including more social networks or switching to a more categorical approach should better reveal the true effectiveness of this approach.

In the end, it is my hope that this work acts as a tool for other researchers to determine where to start when it comes to tackling the problem of network traffic classification.

REFERENCES

[1] P. X. McCarthy, X. Gong, S. Eghbal, D. S. Falster, and M.-A. Rizoiu, "Evolution of diversity and dominance of companies in online activity," *PLOS ONE*, vol. 16, no. 4, p. e0249993, Apr. 2021, doi: 10.1371/journal.pone.0249993.

[2] N. T. Flores and O. Zhenchuk, "4 Most Popular Machine Learning Classification Algorithms | data-science-ua.com," Data Science UA. Accessed: May 03, 2024. [Online]. Available: https://data-science-ua.com/blog/4-most-popular-machine-learning-classification-algorithms/

[3] "pandas - Python Data Analysis Library." Accessed: May 04, 2024. [Online]. Available: https://pandas.pydata.org/

[4] "scikit-learn: machine learning in Python — scikit-learn 1.4.2 documentation." Accessed: May 04, 2024. [Online]. Available: https://scikit-learn.org/stable/

[5] "sklearn.linear_model.LogisticRegression," scikit-learn. Accessed: May 04, 2024. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[6] A. Sharma, "Random Forest vs Decision Tree | Which Is Right for You?," Analytics Vidhya. Accessed: May 05, 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/