Part II:
Machine Learning Based Analysis and Solutions for Webpage Classification

Chapter 4: Identifying Malicious Webpages Using Conventional ML

Chapter 5: Detection of Malicious Webpages Using Deep Learning: Structured Data

Chapter 6: Detection of Malicious Webpages Using Deep Learning: Unstructured Data

Chapter 4: Identifying Malicious Webpages Using Conventional Machine Learning

4.1 Introduction

In Part I, we discussed data collection and pre-processing of malicious webpages dataset. In this chapter, we describe the classification process for detecting malicious webpages and analyzing attributes to be selected for this task.

In this chapter, we use Conventional ML to classify webpages as either benign or malicious. The ML literature did not have the concept of shallow ML until deep learning became prevalent. The word 'Shallow ML', also called 'Conventional ML', was coined to differentiate it from the new deep learning technology. We'll discuss about deep learning in subsequent chapters. In this chapter, we use Shallow ML (hereinafter referred to as Conventional ML) for malicious webpage classification. It is important to note that Conventional ML techniques require extensive pre-processing for feature extraction. Deep learning has more or less eliminated feature engineering requirements and is capable of automatically extracting features from raw data. Conventional ML based classification comprises a variety of supervised algorithms like Naive Bayes [97], Decision Trees (ID3 [98], C4.5 [56], CART [99], etc.), Random Forest [100] (which uses ensemble learning with numerous Decision Trees), and Support Vector Machine [101] (SVM is Kernel-based classification algorithm), etc. In this chapter, we have used Naive Bayes [[97], Decision Tree (C4.5) [56], SVM (with RBF) [101] and Random Forest [100]. The idea behind choosing these algorithms was to cover the main categories of classification algorithms, viz., entropy based algorithm, probabilistic classifier, maximal margin kernel-based classifier, and ensemble classifier

The design of the conventional ML pipeline to detect malicious websites is shown in **Figure 4.1**. The steps involved in such a process are data collection

(crawling websites), feature extraction from raw data and its pre-processing, creation of training and test datasets, training of the classifier, and testing the model. The crawling process for data collection using MalCrawler has already been described in Chapter 2.

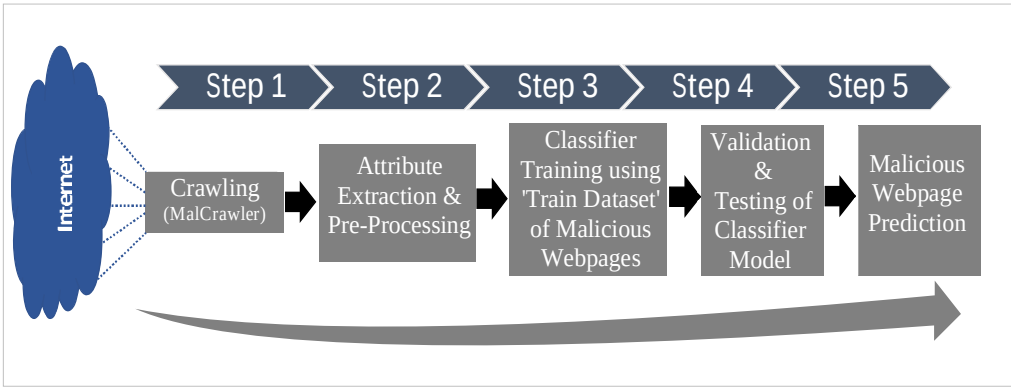

Figure 4.1: Malicious Website Detection Process Using Conventional ML

One essential facet in Conventional ML is handcrafting the right set of attributes which can be used for classification. A correct selection of attributes ensures better classification accuracy and reduces computational requirements. This chapter focuses on identifying attributes which can classify malicious webpages more effectively and efficiently. A set of 25 attributes have been identified for malicious website detection. These attributes have been analyzed with respect to classification accuracy and computational requirements (during extraction and pre-processing). Based on this analysis, the most suitable subset of attributes is recommended. The chapter mainly focuses on identifying the most relevant attributes for detecting malicious webpages using ML. While results are presented using metrics like overall prediction accuracy, precision, F1-score, etc., we have delved more into the inter se performance of attributes for achieving the best prediction.

Various tools and platforms have been used as part of the research described in this chapter. MalCrawler [67] has been used to crawl websites (refer to Chapter 2 for details of MalCrawler). Extraction and pre-processing have been done using HTML Unit Browser Emulator [53] and custom-written Java code. Rhino JavaScript Emulator [54] has been used for running the JavaScript code in a sandboxed environment. Machine learning was carried out

on the WEKA platform [55]. Details of these software tools are given in Appendix C of the thesis.

4.2 Related Work

Significant research has been carried out on malicious webpage detection using ML techniques. However, the available literature has used a very restrictive set of attributes. Ma et al., in their work on detecting malicious websites from suspicious URLs, have considered only URL related attributes [102]. While Wressnegger et al. have discussed flash-based malware [103], Mavrommatis et al. have discussed iFrames [104], Ganesh et al. have analyzed Java Applet based malware [105], Mao et al. have discussed HTML content-based analysis of malicious websites [106], Cova et al. have examined few JavaScript-based attributes [107], and Gorji et al. have discussed obfuscated JavaScript code based attacks [108]. We find that a holistic analysis of all possible attributes, which can play a critical role in detecting malicious websites, is lacking. An effort has been made in this chapter to address this gap.

Apart from analyzing the attributes, we will also rank them based on two aspects. Firstly, with respect to their classification accuracy in malicious website detection. Secondly, with respect to their requirement of computational resources during extraction and pre-processing. Attribute selection is an important aspect, which improves classification results by reducing both overfitting and underfitting. Hall et al. have discussed few attribute selection techniques applicable to all data mining processes [109]. We have used two attribute selection techniques for ML to ensure better selection, viz., 'Gain Ratio method' and '10-Fold Cross Validation' of individual attributes.

4.3 Attributes for Classification

Various attributes which can be used to detect malicious websites were considered. Amongst the list of all possible attributes, twenty five most suitable attributes were selected for further analysis. The attributes considered for detecting malicious websites are shown below in **Table 4.1**.

Table 4.1: List of Attributes Considered- Conventional ML

No	Attribute Category	Attribute Name	Attribute Description
A1	Location of the Website	Geographical Location	The IP address is used to find the geographical location of the webserver using the Geo IP database [73].
A2	URL Properties	URL	Keywords extracted from the URL are analyzed.
A3	-do-	HTTPS Enabled Website	Whether the website uses secure HTTPS or unsecured HTTP.
A4	-do-	Domain Name	Keywords are extracted from domain name and analyzed.
A5	-do-	DNS WHOIS Information	It is checked whether the DNS record of the website exists.
A6	-do-	Redirection from the website	It is checked whether the website redirects or not.
A7	Website Behavior	Cloaking	Cloaking is the phenomenon in which the website shows different pages based on the client platform. It is checked whether a website is cloaking or not.
A8	Web Page Semantics (HTML & Content)	Presence of iFrame on the webpage	The presence of iFrame is checked.
A9	-do-	Number of Applet Tags	The number of these tags is recorded.
A10	-do-	Number of Flash Script Tags	-do-
A11	-do-	Top 10 keywords of the page content	These keywords are analyzed against a set of keywords generally found on malicious websites.
A12	-do-	Meta Tag Values	Keywords are extracted from Meta Tags and compared as in A11.
A13	-do-	HTML Title Tag Values	Keywords are extracted from HTML and compared as in A11.
A14	Java Script Behavior and semantics	Redirection of URL using document. location()	The presence of redirection using JavaScript is checked.

No	Attribute Category	Attribute Name	Attribute Description
A15	-do-	XML HTTP Request (XHR)	It is checked if XHR is meant for the same or different domain.
A16	-do-	Communication with Flash components	It is checked if the Flash component communicates with the browser.
A17	-do-	Communication with Applet Components	It is checked if the Applet component communicates with the browser.
A18	-do-	URL in XHR	It is checked whether the URL in XHR is encoded or not.
A19	JavaScript Code Content	Presence of eval() function	Presence of JavaScript eval() function is checked.
A20	-do-	Presence of unescape() function	Presence of JavaScript unescape() function is checked.
A21	-do-	Presence of Popups using Window.open() function	Checks the presence of Window.open() JavaScript function.
A22	-do-	Presence of find () function	Checks the presence of JavaScript find() function.
A23	-do-	Presence of obfuscated code	Obfuscated code is the encrypted JavaScript code. The presence of such code is checked.
A24	-do-	Size of JavaScript Code	The size of JavaScript code length is recorded.
A25	-do-	Size of Obfuscated Code	The size of Obfuscated JavaScript is recorded.

4.4 Attribute Extraction and Pre-processing

The details of the attributes listed in the previous section and the details of extraction and pre-processing of these attributes are given in the succeeding paragraphs. MalCrawler [67] has been used for crawling websites. Wherever a known list of malicious sites was required (e.g., when providing an initial seed for crawling to the MalCrawler), the Malware Domain list [110] was utilized.

For labeling the dataset of malicious webpages, Google Safe Browsing API was used [61].

Location Region- Based on IP Address. The IP address of the website is detected and recorded by the crawler. Geographical location is then determined from the IP address using the GeoIP database [73]. The geographical location is stored as country name, a nominal type attribute.

URL. The URL is extracted by the crawler. A bag-of-words representation of the URL is generated and then compared with the top 100 words found in URLs of malicious websites [102]. The number of the matches is then stored as a numerical type attribute.

HTTPS Enabled Website. HTTPS is more secure than HTTP, and websites running on HTTPS are less likely to host malicious content [111]. Therefore, during crawling, we have checked whether the website uses HTTP or HTTPS. This value is then stored as a Boolean attribute.

Domain Name. The Domain Name is known to have been used as an attribute in detecting malicious websites using ML [112]. Here, the domain name is extracted by the crawler. Keywords are extracted from the domain name and are then compared with few common keywords linked to malicious behavior. The number of matches is then stored as a numerical type attribute.

DNS WHOIS Information. The DNS WHOIS information provides the registration details of the website. The presence and absence of fields in DNS WHOIS information (e.g., address of domain owner) has been found to be linked to maliciousness [113]. The presence and absence of this information is stored as a Boolean attribute.

Redirection from Website. Redirection has often been linked to malicious behavior [114]. MalCrawler is capable of redirection detection. The detection results are stored as a Boolean attribute.

Cloaking. Cloaking is a phenomenon in which the website shows different pages based on the client platform. Malcrawler [67] is used to detect cloaking actions by a website by serving different HTTP requests having

various user-agent fields in the HTTP header. The presence of cloaking has been found to be linked to maliciousness [115] [116]. The presence and absence of cloaking is saved as a Boolean attribute.

Presence of iFrame on Web Page. The iFrame HTML tag is known to have been utilized to download malicious JavaScript exploits in numerous attack vectors [104]. The presence and absence of the iFrame tag has been checked by parsing websites. Parsing is the process of reading the complete HTML content, including the JavaScripts. Customized libraries in Java have been used in conjunction with MalCrawler [67]. The presence or absence is stored as a Boolean attribute.

Number of Applet Tags. Malicious exploits through Java Applets have been reported on many websites [105]. Thus, its presence is recorded by parsing the webpage and is stored as a Boolean attribute.

Number of Flash Script Tags. Like Java Applets, numerous exploit injection cases through Flash Scripts have been reported [103] [117]. Thus, its presence is recorded by parsing the website. The value is stored as a Boolean attribute.

Top 10 Keywords of Page Content. The website is parsed to extract the Term Frequency - Inverse Document Frequency (TF-IDF) (Note: TF-IDF is a statistical method showing the importance of a word in a document). The TF-IDF of this website is compared against the TF-IDF of know malicious websites [107] [118]. The match of the top 10 TF-IDF keywords of the website against the TF-IDF of malicious websites is saved as a numerical value attribute.

Meta Tag Values. The TF-IDF of the meta tag is computed separately. This TF-IDF is compared against the TF-IDF of known malicious websites [118]. The value of this match is saved as a numerical value attribute.

HTML Title Tag Values. The bag-of-words is extracted from the HTML title tag. It is compared against the bag-of-words of malicious websites [118]. The match of this comparison is saved as a numerical attribute.

Redirection Using document.location(). The presence of JavaScript function document.location() is checked by parsing the website. This function has been found to be linked to malicious redirects [108][114]. The presence or absence of this function is saved as a Boolean attribute.

XML HTTP Request (XHR). XHR is the core of AJAX technology (AJAX -Asynchronous JavaScript and XML is a technology to create asynchronous web applications). However, XHR can be used to inject exploits [119]. The website is parsed to take a count of the number of XHR instances. This value is saved as a numerical attribute.

Communication with Flash Components. Flash components communicating with the browser are indicative of exploitative behavior. This feature is checked using the HTML Unit Browser Emulator [103][117]. The presence or absence of such behavior is stored as a Boolean attribute.

Communication with Java Applets. Like the Flash component, the communication of Applet with the browser can be used to inject exploits [105]. This is also checked using the HTML Unit Browser Emulator. This behavior is stored as a Boolean value.

URL in XHR. A URL in XHR outside the domain is an indicator of malicious behavior [119]. This can be analyzed using the HTML Unit Browser Emulator. The presence or absence of this behavior is recorded as a Boolean value.

Presence of eval() Function. Malicious websites use the eval() function to generate malicious code at runtime to thwart detection [108]. The JavaScript code on the website is parsed to detect the eval() function. The number of eval() functions in JavaScript code is stored as a numerical attribute.

Presence of unescape() Function. Hackers generally encode malicious code and use the unescape() function to decode it. Thus, the number of unescape() function calls in JavaScript is a strong indicator of malicious activity [120]. For our ML analysis, the number of unescape() functions in JavaScript code is stored as a numerical attribute.

Presence of Windows.open() Popups. The JavaScript Windows.open() Popups are used for ads and also to inject exploits [121]. For our ML analysis, the presence or absence of Popup is stored as a Boolean value.

Presence of find() Function. The find() JavaScript is used along with unescape() and eval() to decrypt malicious code at runtime [120]. Thus, the occurrences of the find() function in JavaScript code is noted and stored as a numerical attribute.

Presence of Obfuscated Code. Obfuscated code is the encrypted JavaScript code. Generally, obfuscation is done to thwart the detection of malicious code [122]. Thus, its presence is a strong indicator of malicious activity. The presence or absence of obfuscated code is recorded as a Boolean attribute.

Size of JavaScript Code. Generally, malicious JavaScript code is of relatively large size [122]. Thus, the size of JavaScript code is a good indicator of maliciousness. We have used it as a numerical attribute with a value equal to the size of JavaScript code in KiloBytes (KBs).

Size of Obfuscated Code. Large obfuscated code indicates the presence of an exploit [120]. Thus, the size of obfuscated code (in KB) is captured and stored as a numerical value for ML.

4.5 Experimental Setup and Evaluation Metrics

Classification was carried out using four algorithms - C4.5, Naive Bayes, SVM and Random Forest. These four algorithms were chosen as they represent four different approaches to classification. While C4.5 uses a Decision Tree, Naive Bayes is a probabilistic generative model, SVM is a kernel-based (RBF kernel) maximal margin learner, and Random Forest is an ensemble learning algorithm. The SVM classifier was chosen with the Radial Basis Function (RBF) kernel in order to capture the non-linear relationship in the dataset. WEKA [55] data mining software was used for training and running the classifiers. The training dataset was collected using MalCrawler and customized Java code. The initial seed for crawling in MalCrawler was created

using Malware Domain List [110]. While MalCralwer is designed to seek more malicious webpages than benign, an initial malicious seed improves this performance further. Google Safe Browsing API [61] was used to prepare the class labels in the dataset- '1' for benign and '0' for malicious webpages. A copy of the dataset that has been created for this research has been published online [123]. This dataset has an imbalance of classes; there are more benign classes than malicious. This imbalance reflects the real web from where this data has been collected, wherein malicious webpages represent a very small fraction of the total number of webpages. Class imbalance is known to create a bias towards the majority class. Hence, for the classification task, we have oversampled the minority class to reduce the imbalance. We used the SMOTE (Synthetic Minority Oversampling Technique) for oversampling the malicious samples [124].

For checking the attribute predictability, two techniques were used- The Gain Ratio method and '10-fold cross-validation. Firstly, as we were looking to rank the attributes based on their individual ability to predict malicious websites, we used the Gain Ratio method [109] of attribute selection. In this method, each attribute A_i is assigned a score based on the information gain between itself and the class. If C is the class and A is the attribute, equations (4.1) and (4.2) below give Entropy H before and after observing the attribute.

$$H(C) = -\sum_{c \epsilon C} p(c) log_2 p(c) \qquad (4.1)$$

$$H(C/A) = -\sum_{a \epsilon A} p(a) \sum_{c \epsilon C} p(c/a) log_2 p(c/a) \qquad (4.2)$$

The Entropy $H(c/A)$ was computed for all 25 attributes and is shown in **Table 4.2** (refer to the second column shown in red color). Interestingly, the Entropy of the entire dataset $H(c)$ was as low as **0.156**, which is due to class imbalance (malicious webpages were a mere 2.27% of the entire dataset). Secondly, 10-fold cross-validation was run, one attribute at a time, to assess each attribute's ability to predict malicious websites (Only three classification algorithms were used for cross-validation, viz., C4.5, Naive Bayes and SVM). The confusion matrix produced by this 10-fold cross-validation is given in

Table 4.2 (Refer columns 3-18). The table elucidates an attribute's overall ability to predict malicious websites using the two techniques mentioned above.

Table 4.2: Attribute Selection using 'Gain Ratio' & '10-Fold Cross Validation'

Attribute	Gain Ratio	Ten Fold Cross Validation																Overall Relative Rank
	H(C/A)	C 4.5 Classifier				Naive Bayes Classifier				SVM Classifier				Accuracy				
#		TNR	FPR	TPR	FNR	TNR	FPR	TPR	FNR	TNR	FPR	TPR	FNR	C4.5	Naive Bayes	SVM	Average Accuracy	
A1	0.05888922	0.34	0.66	0.46	0.54	0.33	0.67	0.37	0.63	0.34	0.66	0.42	0.58	0.40	0.35	0.38	0.38	24
A2	0.07243895	0.39	0.61	0.51	0.49	0.42	0.58	0.52	0.48	0.44	0.56	0.50	0.50	0.45	0.47	0.47	0.46	22
A3	0.11386985	0.71	0.29	0.75	0.25	0.67	0.33	0.77	0.23	0.72	0.28	0.75	0.25	0.73	0.72	0.74	0.73	13
A4	0.07921381	0.49	0.51	0.51	0.49	0.49	0.51	0.51	0.49	0.48	0.52	0.56	0.44	0.50	0.50	0.52	0.51	21
A5	0.11360928	0.66	0.34	0.76	0.24	0.65	0.35	0.79	0.21	0.73	0.27	0.77	0.23	0.71	0.72	0.75	0.73	14
A6	0.14519058	0.88	0.12	0.96	0.04	0.86	0.14	0.99	0.01	0.91	0.09	0.97	0.03	0.92	0.93	0.94	0.93	3
A7	0.14904704	0.95	0.05	0.95	0.05	0.90	0.10	1.00	0.00	0.92	0.08	1.00	0.00	0.95	0.95	0.96	0.95	1
A8	0.14670189	0.90	0.10	0.97	0.03	0.87	0.13	1.01	-0.01	0.93	0.08	0.96	0.05	0.94	0.94	0.94	0.94	2
A9	0.04586063	0.20	0.80	0.30	0.70	0.27	0.74	0.36	0.65	0.29	0.71	0.35	0.65	0.25	0.31	0.32	0.29	25
A10	0.09536926	0.56	0.44	0.64	0.36	0.55	0.45	0.65	0.35	0.58	0.42	0.68	0.32	0.60	0.60	0.63	0.61	18
A11	0.11829957	0.75	0.25	0.77	0.23	0.74	0.26	0.76	0.24	0.75	0.25	0.77	0.23	0.76	0.75	0.76	0.76	12
A12	0.08286182	0.49	0.51	0.55	0.45	0.44	0.56	0.56	0.44	0.52	0.48	0.62	0.38	0.52	0.50	0.57	0.53	20
A13	0.06357951	0.38	0.62	0.42	0.58	0.40	0.60	0.40	0.60	0.41	0.59	0.43	0.57	0.40	0.40	0.42	0.41	23
A14	0.12142644	0.74	0.26	0.80	0.20	0.78	0.22	0.78	0.22	0.77	0.23	0.79	0.21	0.77	0.78	0.78	0.78	11
A15	0.09015783	0.50	0.50	0.60	0.40	0.56	0.44	0.62	0.38	0.56	0.44	0.62	0.38	0.55	0.59	0.59	0.58	19
A16	0.11048242	0.64	0.36	0.72	0.28	0.70	0.30	0.72	0.28	0.68	0.32	0.78	0.22	0.68	0.71	0.73	0.71	15
A17	0.10579213	0.57	0.43	0.69	0.31	0.67	0.34	0.70	0.31	0.69	0.31	0.75	0.25	0.63	0.68	0.72	0.68	16
A18	0.10110184	0.58	0.43	0.69	0.32	0.58	0.42	0.66	0.34	0.68	0.32	0.70	0.30	0.63	0.62	0.69	0.65	17
A19	0.13237045	0.84	0.16	0.86	0.14	0.83	0.17	0.85	0.15	0.85	0.15	0.85	0.15	0.85	0.84	0.85	0.85	7
A20	0.13601845	0.84	0.16	0.90	0.10	0.84	0.16	0.90	0.10	0.84	0.16	0.90	0.10	0.87	0.87	0.87	0.87	6
A21	0.13862417	0.88	0.13	0.89	0.12	0.85	0.15	0.91	0.09	0.86	0.14	0.94	0.06	0.88	0.88	0.90	0.89	5
A22	0.12955627	0.77	0.23	0.89	0.11	0.77	0.23	0.87	0.13	0.80	0.20	0.87	0.13	0.83	0.82	0.84	0.83	8
A23	0.12403215	0.73	0.27	0.85	0.15	0.75	0.25	0.83	0.17	0.78	0.22	0.82	0.18	0.79	0.79	0.80	0.79	10
A24	0.12611673	0.77	0.23	0.85	0.15	0.76	0.25	0.85	0.16	0.74	0.26	0.88	0.12	0.81	0.80	0.81	0.81	9
A25	0.14023972	0.84	0.16	0.93	0.07	0.86	0.14	0.92	0.08	0.86	0.15	0.98	0.02	0.89	0.89	0.92	0.90	4

Note:- Overall rank is based on both Info Gain [H(C/A)] & Accuracy achieved by attribute in 10 Fold Cross Validation

The accuracy achieved, using a single attribute at a time, while carrying out cross-validation, is plotted in **Figure 4.2**.

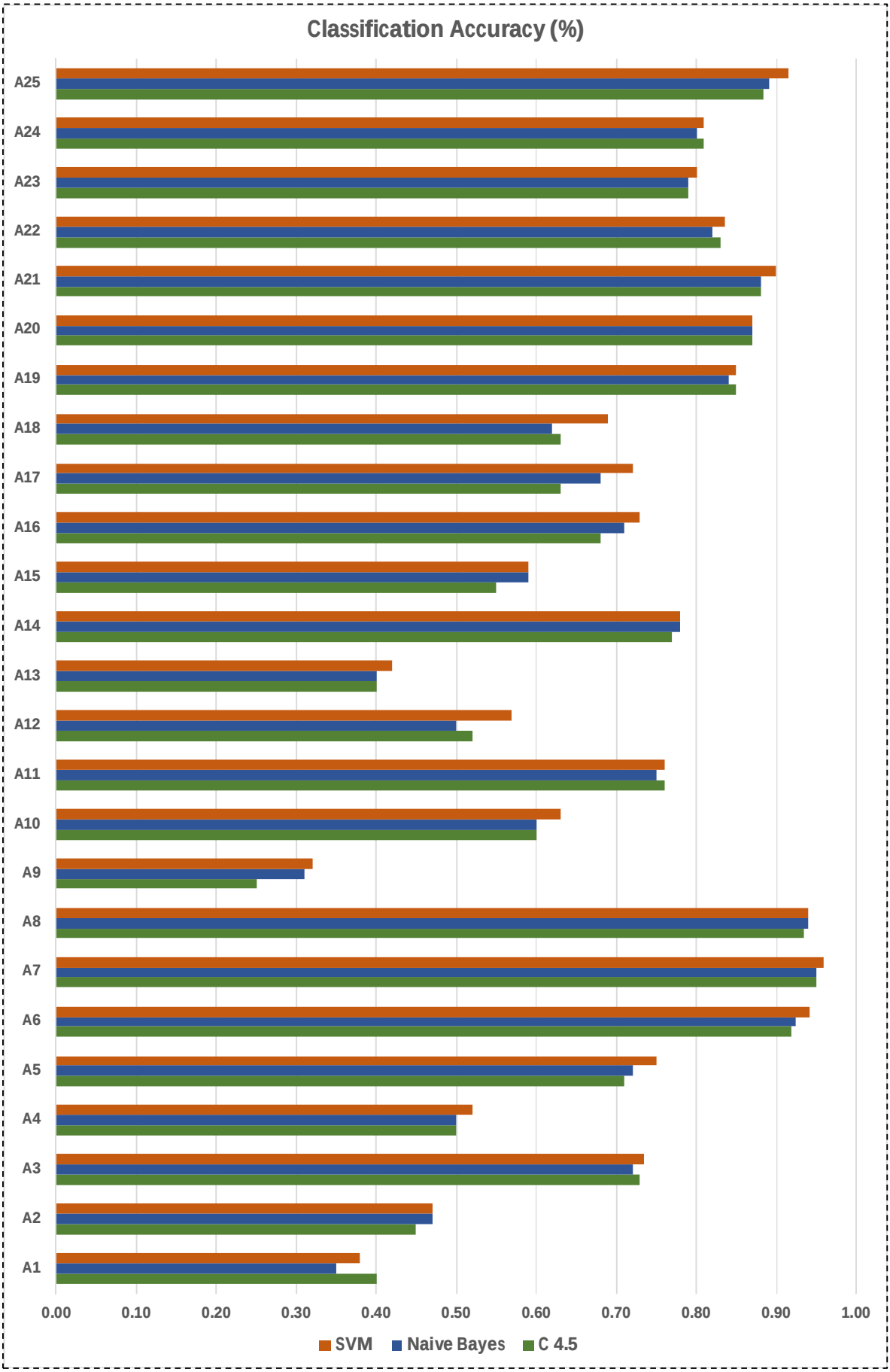

Figure 4.2: Classification Accuracy of Attributes

Apart from the 10-fold cross-validation which was carried out for attribute selection, the dataset was trained with one million samples and thereafter tested on 0.564 million samples using all four ML algorithms mentioned earlier, viz., C4.5, Naive Bayes, SVM and Random Forest. The classification results from these four algorithms (with all 25 attributes) on the test dataset is shown in **Table 4.3**.

Table 4.3: Classification Results with Conventional ML Algorithms (all 25 attributes used)

Metrics	Classifiers			
	C 4.5	Naive Bayes	SVM	RF
Accuracy	0.979	0.981	0.986	0.983
Recall (TPR, Sensitivity)	0.9740	0.9800	0.9830	0.9810
Specificity (TNR, Selectivity)	0.9840	0.9820	0.9890	0.9850
Precision (PPV)	0.5857	0.5584	0.6749	0.6030
NPV	0.9994	0.9995	0.9996	0.9996
F1 Score	0.7315	0.7114	0.8003	0.7469
*Note: Positive class represents the 'Malicious label'.				

The meaning of the attributes used is summarized in **Table 4.4** for ready reference.

Table 4.4: Meaning of the Classification Metrics Used

Accuracy is the ratio of correctly classified samples over the total number of samples	$= \frac{TP + TN}{TP + FP + TN + FN}$
Recall is the ratio of how many were correctly classified as positive to how many were actually positive	$= \frac{TP}{TP + FN}$
Specificity is the ratio of how many were correctly classified as negative to how many were actually negative.	$= \frac{TN}{TN + FP}$
Precision is the ratio of how many were correctly classified as positives out of all positives.	$= \frac{TP}{TP + FP}$
Negative Predictive Value (NPV) is the ratio of how many were correctly classified as negatives out of all negatives.	$= \frac{TN}{TN + FN}$
F1 Score is the harmonic mean of precison and recall. It gives a measure of the model's classification ability.	$= \frac{2 * Precision * Recall}{Precision + Recall}$

The confusion matrices generated from the test results with all four classification algorithms (all 25 attributes are used) are given in **Figure 4.3**.

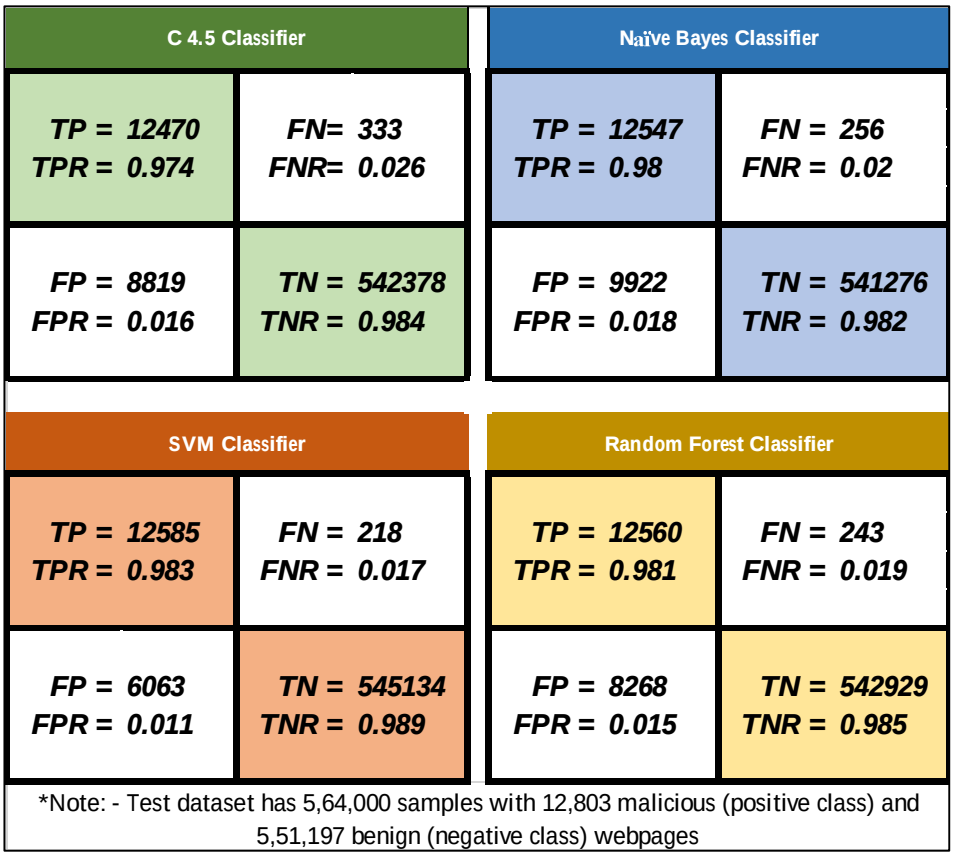

Figure 4.3: Confusion Matrices using Conventional ML Algorithms (All 25 Attributes used)

4.6 Analysis of Results

The results obtained are analyzed to find the most suitable set of attributes for detecting malicious webpages.

4.6.1 Classification Accuracy of Attributes

The comparison of the 25 attributes considered for detecting malicious webpages using ML was shown graphically in **Figure 4.2**. The bar graph showed the classification accuracy for each attribute, running 10-fold cross-validation, using the three classification algorithms- C4.5, Naive Bayes, and SVM. It emerges from the graph that few attributes (A7, A8, A6, A25, A21, A20, A22, A24, A23) are better predictors and contribute more towards classification accuracy.

4.6.2 Computational Resources Used

The computational resources (memory & CPU cycles) utilized for attribute extraction and pre-processing is an essential factor in ranking the attribute, especially when you want to deploy your solution in a near-real-time or real-time environment. The computational resources used by attributes were assessed using the Netbeans Profiler when running the extraction and pre-processing java code. Netbeans is a software development platform for Java. Netbeans Profiler was used to measure the CPU cycles & memory utilization while running the Java code for Attribute extraction & Pre-processing. The values obtained were normalized to show on a scale of 0 to 1. The result obtained is shown in **Figure 4.4**.

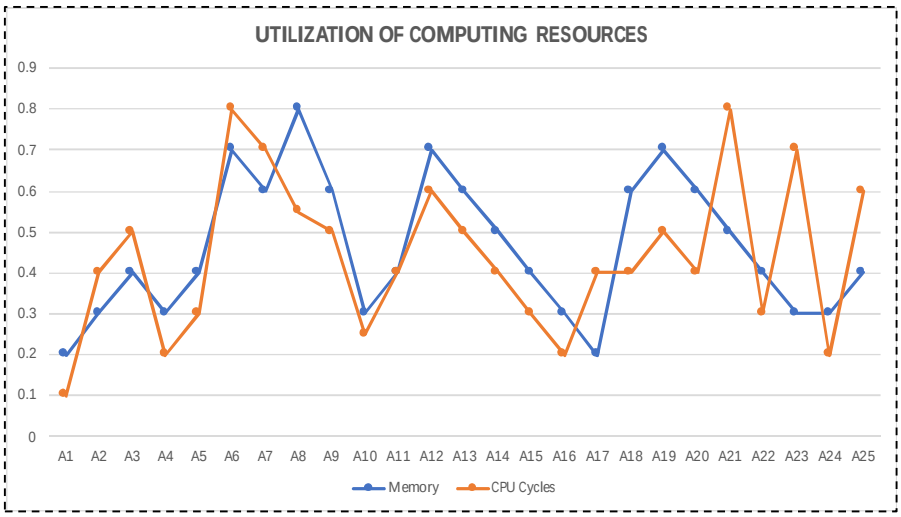

Figure 4.4: Computation Resources Used by Attributes

4.6.3 The Top Attributes

The top attributes to predict a malicious website based on the classification accuracy are - A7 (Cloaking), A8 (presence of iFrame), A6 (redirection from website), A25 (size of obfuscated code), and A21 (Popups using Window.open()). However, when we also consider the complexities and resources during the extraction and pre-processing of attributes, we find that suitable attributes for scalable commercial ML web security applications are - A25 (size of obfuscated code), A24 (JavaScript length), A5 (WHOIS), A3 (HTTPS), A4 (domain name), A2 (URL) and A1 (geographic location). The 10-fold cross-validation result using the top five attributes is given below in **Figure 4.5**.

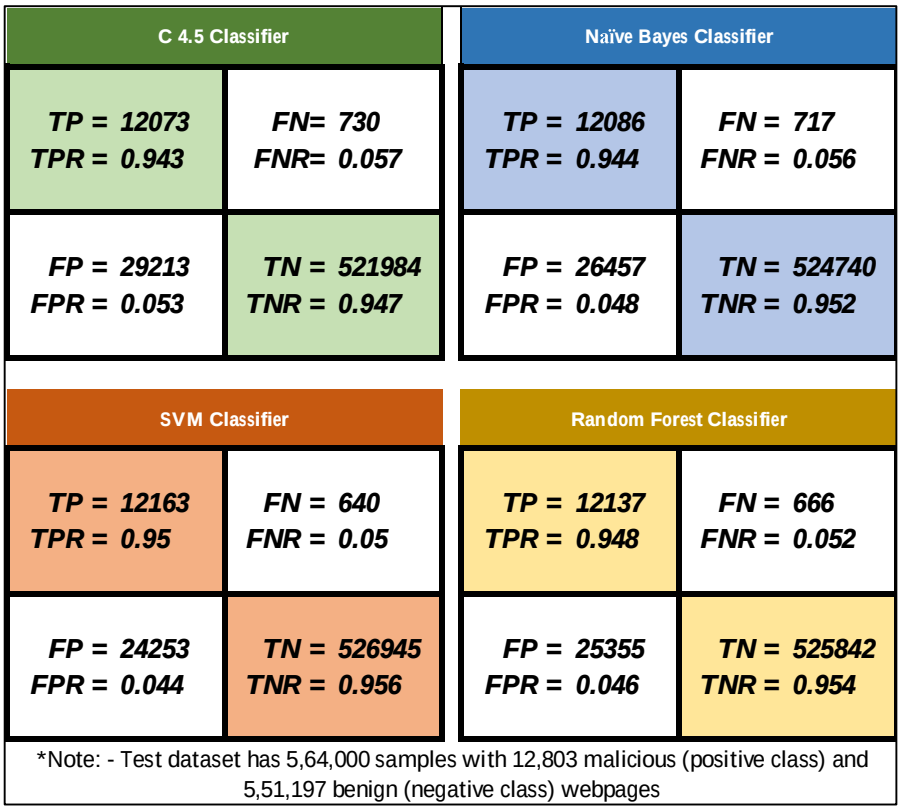

Figure 4.5: Confusion Matrix- Conventional ML Algorithms with Top Five Attributes

4.7 Conclusion

This chapter compares the attributes used for their effectiveness in detecting malicious webpages using ML. A total of 25 attributes were considered, which are generally used to detect malicious websites. These attributes were analyzed with respect to the computational resources required for extraction and pre-processing and the classification accuracy while predicting malicious webpages. The analysis was carried out using four different categories of Conventional ML algorithms Decision Trees (C4.5), Probabilistic classifier (Naive Bayes), Kernel-Based Classifier (SVM with non-linear RBF kernel), and Ensemble Leaner (Random Forest). Based on the analysis, the best attributes were identified for detecting malicious websites. Previous studies related to detecting malicious webpages using ML had considered few attributes and had not compared the relative importance of using an attribute with respect to others. The comparative and analytical model discussed in this chapter may also be used by researchers to carry out other forms of malware analysis that use ML. The classification accuracy achieved through Conventional ML algorithms, though better than the accuracy reported

in the literature, was not high enough for commercial deployment. Also, there is scope for improving Precision and F1-score, especially since the dataset is imbalanced. In later chapters of the thesis, we have explored deep learning to improve these results further. Notwithstanding the shortcomings of classification results' metrics, this research has successfully utilized conventional ML techniques (including the Gain Ratio method) to select most suitable attributes for web security tasks.