Data Mining for Business (BUDT758T)

Project Title: Malicious URL Detection Using Machine Learning

Team Members: Alekya Ghanta
Yanpu Wang
Xingjian Qin
Weian Shi
Jie Gao

ORIGINAL WORK STATEMENT

We the undersigned certify that the actual composition of this proposal was done by us and is original work.

	Type Name	**Signature**
Contact Author	Alekya Ghanta	Alekya Ghanta
	Yanpu Wang	Yanpu Wang
	Xingjian Qin	Xingjian Qin
	Weian Shi	Weian Shi
	Jie Gao	Jie Gao

I. Executive Summary

Our project is to build a classification model to accurately determine whether a given URL is safe, phishing, or malware.

Modified or compromised URLs employed for cyber attacks are known as malicious URLs. Mostly, malicious URLs steal financial information by injecting trojans, malware, virus, spam, and other malicious actions to achieve their purpose. It could be more dangerous if confidential information was leaked like critical business strategies. Our project is based on the growing malicious URLs since the covid-19 pandemic. This analysis is of importance for society given that our current life can no longer exist without the information system. Our objective is to leverage the models we have developed to assist individuals dealing with problems caused by harmful URLs and find ways to resolve these issues.

II. Data Description

The raw data were gathered by web scraping. To ensure a comprehensive dataset, URLs from both phishing and malware categories were gathered, along with a substantial number of safe URLs.We collected phish and safe URLs from PhishTank (an open source website that verifies URLs to be phishing or safe). We also collected Malware URLs from URLhaus (an open source database with a collection of malware URLs). Therefore, all the URLs collected could be divided into three categories: malware, phishing and safe. We assigned labels to the URLs, which would serve as the target variable in the prediction models.

Website links:

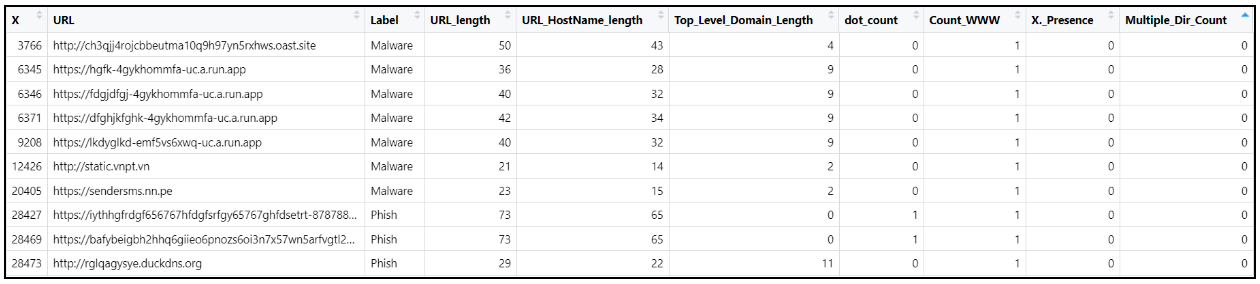
https://www.phishtank.com/index.php

https://urlhaus.abuse.ch/browse/

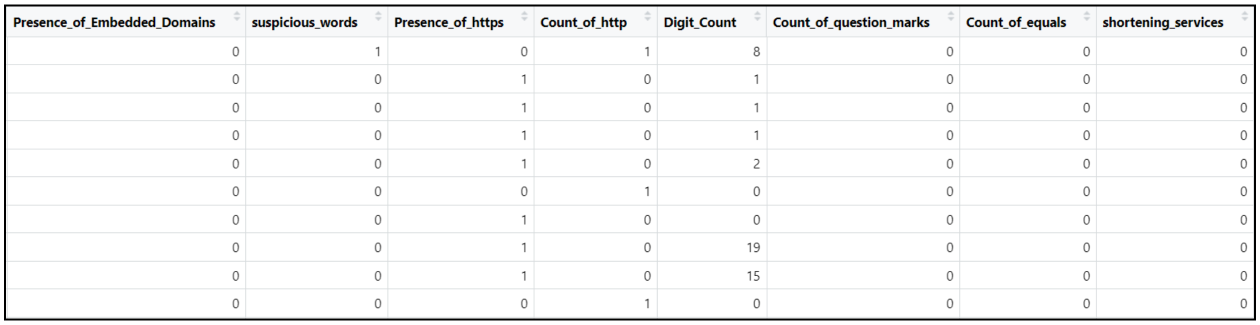
We then extracted a list of features from the raw data by feature engineering the raw URLs, for further model training based on the characteristics of the URLs, such as the length of the URL, count of www and the presence of suspicious words. The 15 features that we finally got contain not only features based on length including URL length, hostname length and length of top-level domain, but also features based on abnormality including presence of multiple domains, presence of multiple www's and presence of multiple directories, etc. The model is trained using features that rely on special characters such as the number of question marks, the number of equals signs, and the presence of the "@" symbol.

The data frame contains 77303 rows and 18 columns. The sample size *n* of the data is 77303. The number of variables *k* is 18. The definitions of the variables can be found in "Feature Codelist" in the appendix (a), including what is measured by each variable, whether it is treated as a numerical or categorical variable and how it is encoded.

Here is a small sample of observations from the data:

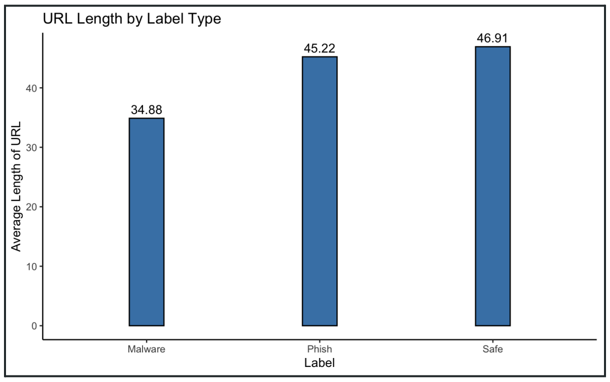


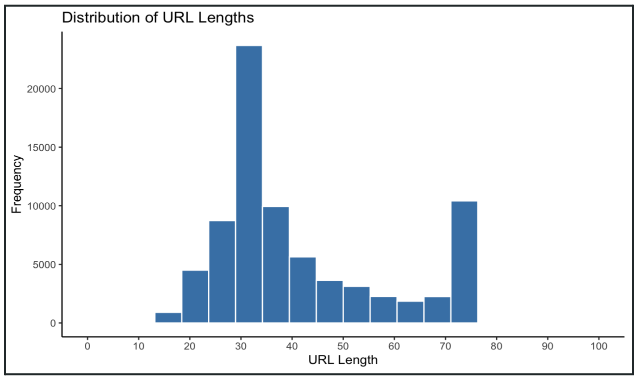
X	URL	Label	URL_length	URL_HostName_length	Top_Level_Domain_Length	dot_count	Count_WWW	X._Presence	Multiple_Dir_Count
3766	http://ch3qjj4rojcbbeutma10q9h97yn5rxhws.oast.site	Malware	50	43	4	0	1	0	0
6345	https://hgfk-4gykhommfa-uc.a.run.app	Malware	36	28	9	0	1	0	0
6346	https://fdgjdfgj-4gykhommfa-uc.a.run.app	Malware	40	32	9	0	1	0	0
6371	https://dfghjkfghk-4gykhommfa-uc.a.run.app	Malware	42	34	9	0	1	0	0
9208	https://lkdyglkd-emf5vs6xwq-uc.a.run.app	Malware	40	32	9	0	1	0	0
12426	http://static.vnpt.vn	Malware	21	14	2	0	1	0	0
20405	https://sendersms.nn.pe	Malware	23	15	2	0	1	0	0
28427	https://iythhgfrdgf656767hfdgfsrfgy65767ghfdsetrt-878788...	Phish	73	65	0	1	1	0	0
28469	https://bafybeigbh2hhq6giieo6pnozs6oi3n7x57wn5arfvgtl2...	Phish	73	65	0	1	1	0	0
28473	http://rglqagysye.duckdns.org	Phish	29	22	11	0	1	0	0



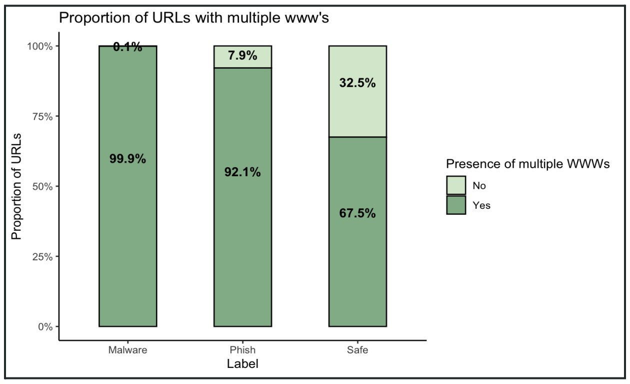
Presence_of_Embedded_Domains	suspicious_words	Presence_of_https	Count_of_http	Digit_Count	Count_of_question_marks	Count_of_equals	shortening_services
0	1	0	1	8	0	0	0
0	0	1	0	1	0	0	0
0	0	1	0	1	0	0	0
0	0	1	0	1	0	0	0
0	0	1	0	2	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	19	0	0	0
0	0	1	0	15	0	0	0
0	0	0	1	0	0	0	0

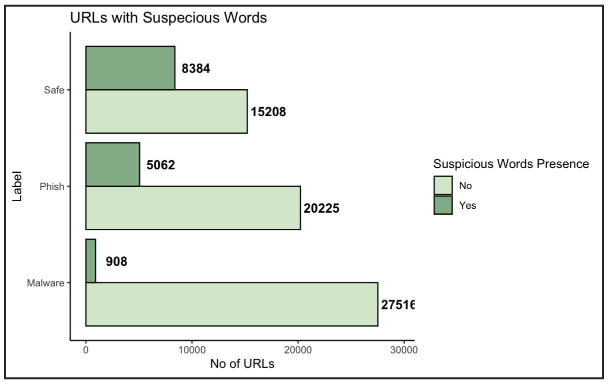
Exploratory Data Analysis:



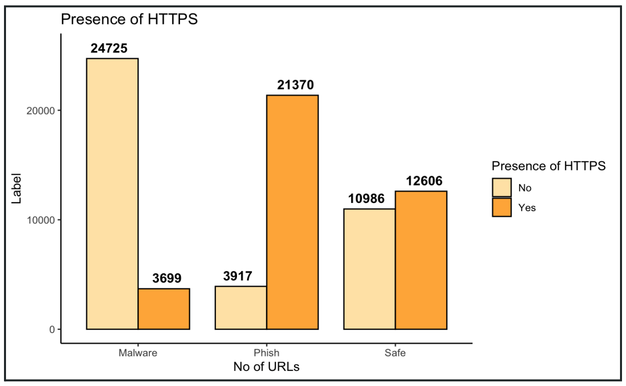


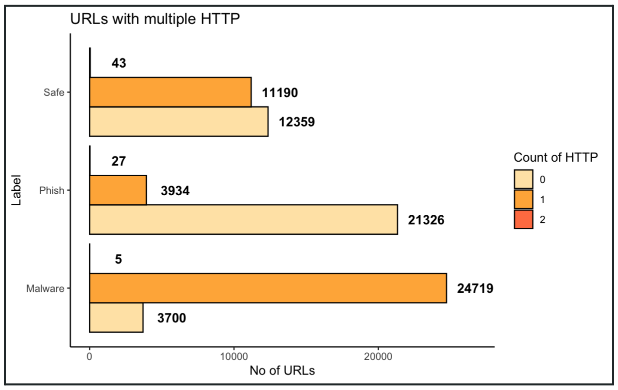
We conducted exploratory data analysis by data visualization to investigate and summarize the characteristics of the data. It is noteworthy that the average length of safe URLs is the highest, unlike our usual expectations. Most of the URL lengths fall in the 30-35 length bin.





99% of the malware URLs and 92.1% phish URLs have multiple www's. In comparison, only 67.5% of the safe URLs have multiple www's, which indicates this feature can help detect malicious URLs. In terms of presence of suspicious words, phish URLs are more likely to have suspicious words compared to malware URLs





Most of the malware URLs and phish URLs have no Secure Protocol, which requires user credentials. By contrast, more than half of the safe URLs have Secure Protocol, implying a safe environment for transactions. Most malware URLs have one or multiple HTTP in URL.

III. Research Questions

The research question for this project is how to build up (an) efficient and accurate classification model(s) to distinguish different types of URLs, including the phishing URLs, malware URLs and the safe ones. Before building the model, features of the URLs like the length of the URL and the existence of some special character strings must be extracted for better model construction. The model(s) should make classification according to the URLs' features, and assign labels for the URLs to be identified with high accuracy.

IV. Methodology

We have applied various techniques for different parts of the data mining process:

1. For data collection, we applied Web Scraping using BeautifulSoup library, to directly and selectively get URLs from the two websites providing us with a list of phishing and malware URLs. This task is accomplished in Python. (See Appendix for the code sample, completed code is enclosed)

2. For feature engineering, we extracted features like the length of a certain domain or the existence of a certain set of character strings and made them into categorical and numerical features. These features can improve the efficiency of the classification model. This task is accomplished in Python. (See Appendix for the code sample, completed code is enclosed)

3. For exploratory data analysis, we used the ggplot2 package to do data visualization and gain insights about the distribution of the features. This task is accomplished in R. (See Appendix for the code sample, completed code is enclosed)

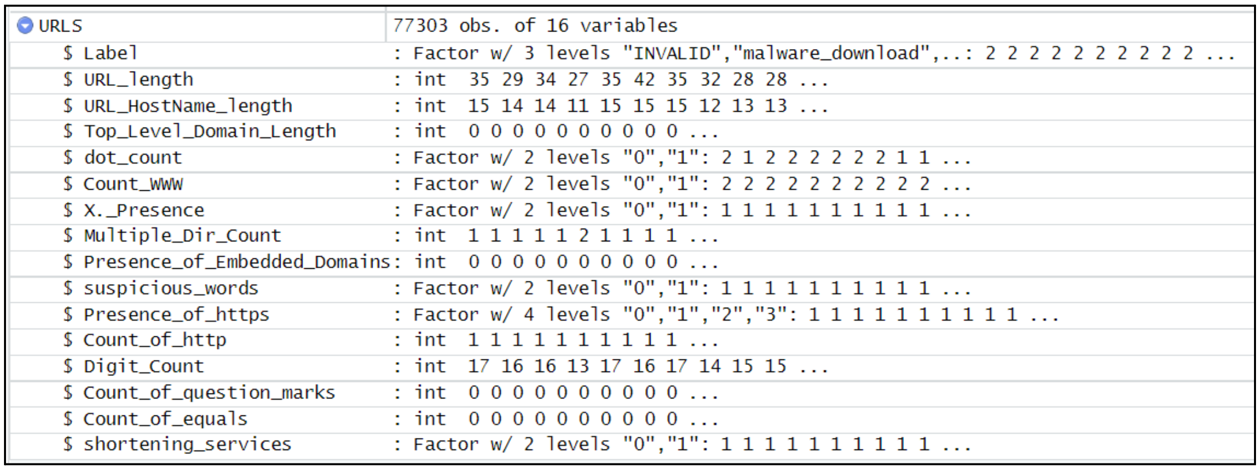
4. For modeling, we applied various Machine Learning techniques including Naive Bayes model, Decision Tree, Gradient Boosting Model, Bagging and Random Forest. Considering we have 15 parameters in total –which is a large number–and the majority of them are categorical, techniques like simple linear regression and logistic regression would not perform well under this circumstance. Therefore, we have chosen to experiment with

prediction models from Naive Bayes to Random Forest, which are capable of handling high dimensions of parameters and are also easy to explain and interpret. For the Decision Tree, we have pruned the tree with cross-validation. For the parameter tuning part of Random Forest, we have selected the number of subset parameters as 8 by multiple attempts. This task is accomplished in R. (See Appendix for the code sample, completed code is enclosed)

V. Results and Finding

We accomplished the goal of building an accurate and efficient model mainly by three data mining procedures: Web Scraping, Feature Engineering and Model Selection and Tuning.

For Web Scraping and Features Engineering, we obtained 77303 URLs from two websites and extracted 15 features from each URL.



```
URLS                               77303 obs. of 16 variables
  $ Label                          : Factor w/ 3 levels "INVALID","malware_download",..: 2 2 2 2 2 2 2 2 2 2 ...
  $ URL_length                     : int  35 29 34 27 35 42 35 32 28 28 ...
  $ URL_HostName_length            : int  15 14 14 11 15 15 15 12 13 13 ...
  $ Top_Level_Domain_Length        : int  0 0 0 0 0 0 0 0 0 0 ...
  $ dot_count                      : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 2 1 1 ...
  $ Count_WWW                      : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
  $ X._Presence                    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
  $ Multiple_Dir_Count             : int  1 1 1 1 1 2 1 1 1 1 ...
  $ Presence_of_Embedded_Domains   : int  0 0 0 0 0 0 0 0 0 0 ...
  $ suspicious_words               : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
  $ Presence_of_https              : Factor w/ 4 levels "0","1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
  $ Count_of_http                  : int  1 1 1 1 1 1 1 1 1 1 ...
  $ Digit_Count                    : int  17 16 16 13 17 16 17 14 15 15 ...
  $ Count_of_question_marks        : int  0 0 0 0 0 0 0 0 0 0 ...
  $ Count_of_equals                : int  0 0 0 0 0 0 0 0 0 0 ...
  $ shortening_services            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

For Model Selection and Tuning, we have applied five models including Naive Bayes model, Decision Tree, Gradient Boosting Model, Bagging and Random Forest. The accuracies for the five models vary from 72.2% (from Naive Bayes model) to 89.7% (From Random Forest model). Here are the performance characteristics for the best performed Random Forest model and the model comparison matrix, which includes metrics like sensitivity, specificity and balanced accuracies etc:

Random Forest: (Best Performing Model)

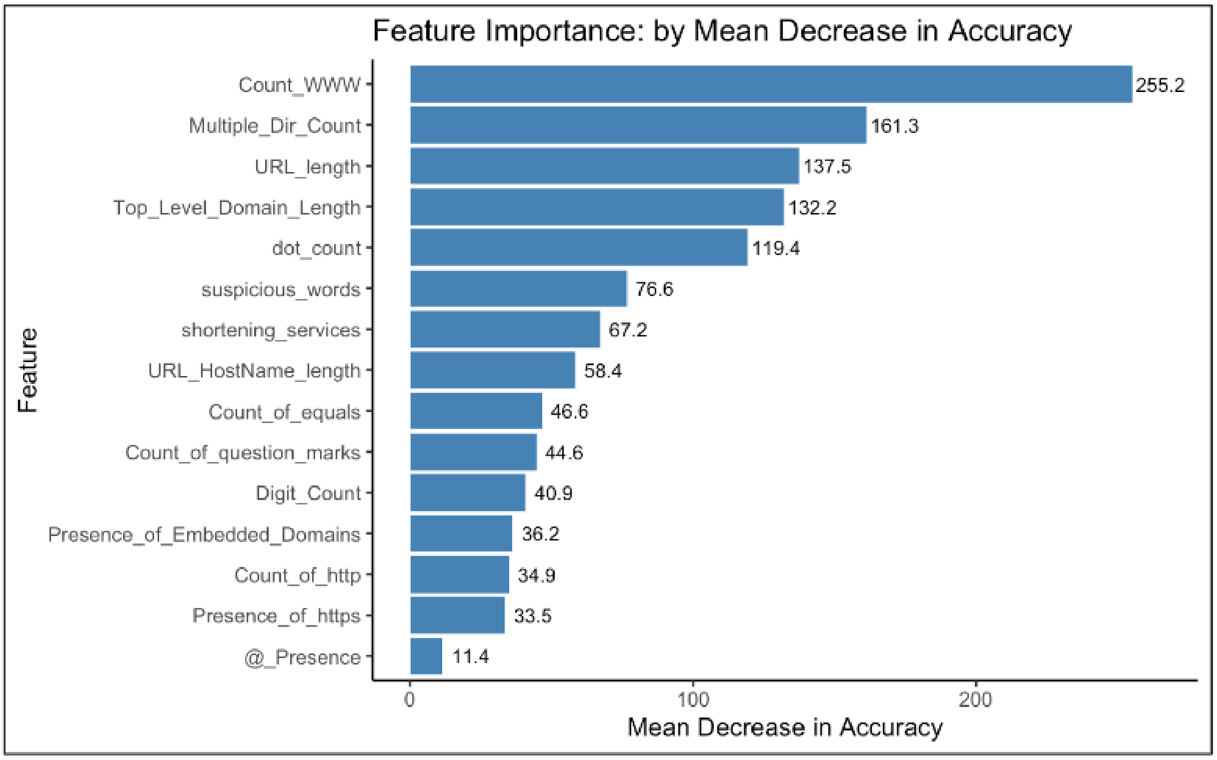
Metric (m=8)	*Malware*	*Phish*	*Safe*
Sensitivity	0.98	0.85	0.84
Specificity	0.98	0.94	0.93
Positive Pred Value	0.96	0.87	0.84
Negative Pred Value	0.99	0.93	0.93
Prevalence	0.37	0.33	0.30
Detection Rate	0.36	0.28	0.25
Detection Prevalence	0.37	0.32	0.30
Balanced Accuracy	0.98	0.89	0.89

Model Comparison:

Model	*Accuracy*	*FPR (Classifying Safe URL as Unsafe)*	*FNR (Classifying Unsafe URL as Safe)*
Naive Bayes	72.2%	56.3%	6.3%
Decision Tree	74.6%	48.4%	10.3%
Gradient Boosting	81.7%	23.8%	11.1%
Random Forest - Bagging	89.3%	16.7%	7.1%
Random Forest (m=8)	89.7%	15.8%	6.7%

We can tell from the performance of the Random Forest model that it is a relatively balanced and accurate model with high sensitivity, high specificity and the balanced accuracies for all three labels beyond 88%. According to the model performance comparison matrix, two Random Forest models are the best-performed ones with high accuracy, FPR and FNR. However, Naive Bayes can also be considered to be an important model keeping in mind that minimizing the False Negative Rates is crucial for our study.

Below is the feature importance generated by Random Forest Model based on the Mean Decrease in Accuracy (MDA) and the Mean Decrease in Gini :
(see Feature Importance Rank for the Mean Decrease in Gini in Appendix)



Count of WWW has the highest MDA, followed by Count of multiple directories and the URL length.

VI. Conclusion

In this project, we focus on developing an accurate and efficient machine learning model to classify URLs as phishing, malware, or safe ones. We collected phishing and safe URLs from PhishTank, and Malware URLs from URLhaus.

During the data preprocessing stage, we extracted 15 features from the raw URLs, focusing on factors such as length, abnormality, and the presence of special characters. Exploratory data analysis revealed interesting patterns, such as the average length of safe URLs being higher than expected and malware URLs having multiple WWWs. Phishing URLs often contained dubious words compared to malware URLs.

We built and compared five prediction models including Naive Bayes, Decision Tree, Gradient Boosting, Bagging and Random Forest. The Random Forest model achieved the highest accuracy of 89.7%, outperforming the other models. It showed good performance according to its high accuracy and low false positive and false negative rates compared to other models.

In terms of the feature importance, according to the the rank of the Mean Decrease in Accuracy (MDA) and the Mean Decrease in Gini Random that the Random Forest model generated, features like count of WWWs, top-level domain length and URL length have high classification capacity, which can be therefore popularized to the public as common sense to identify phishing and malware URLs.

Overall, our finding suggests that we could apply the Random Forest Model we have built and tuned to effectively and accurately classify phishing, malware, and safe URLs.

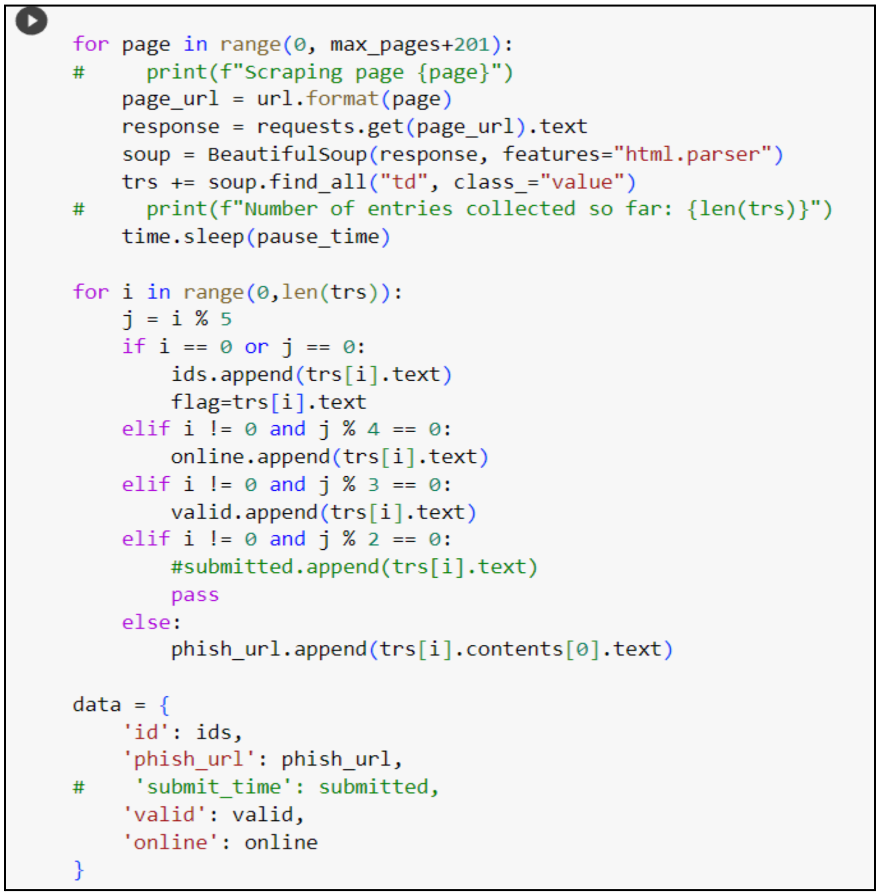
VII. Appendix

a. Feature Codelist:

Var. #	Variable Name	Description	Variable Type	Code Assignment
1	X	Observation No.	Numerical	
2	URL	Raw URL	Character	
3	Label	Label assigned to different types of URLs	Categorical	Malware
				Phish
				Safe
4	URL_length	Length of URL	Numerical	
5	URL_HostName_length	Length of hostname	Numerical	
6	Top_Level_Domain_Length	Length of top-level domain	Numerical	
7	dot_count	Count of "." (multiple domains)	Categorical	0: else
				1: >3
8	Count_WWW	Count of www	Categorical	0: 1
				1: 0 or >=1
9	@_Presence	Presence of "@"	Categorical	0: else
				1: >0
10	Multiple_Dir_Count	Count of multiple directories	Numerical	
11	Presence_of_Embedded_Domains	Count of embedded domains	Numerical	
12	suspicious_words	Presence of suspicious words	Binary	0: No
				1: Yes
13	Presence_of_https	Presence of "https"	Binary	0: No
				1: Yes
14	Count_of_http	Count of "http"	Numerical	
15	Digit_Count	Count of digits	Numerical	
16	Count_of_question_marks	Count of "?"	Numerical	
17	Count_of_equals	Count of "="	Numerical	

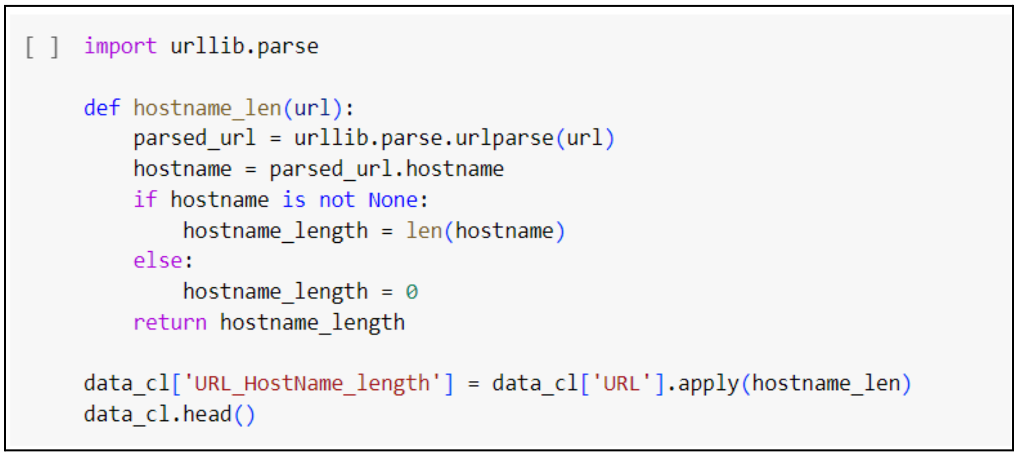
18	shortening_services	Use URL shortening services	Categorical	0: No
				1: Yes

b. Sample code for Web Scraping:



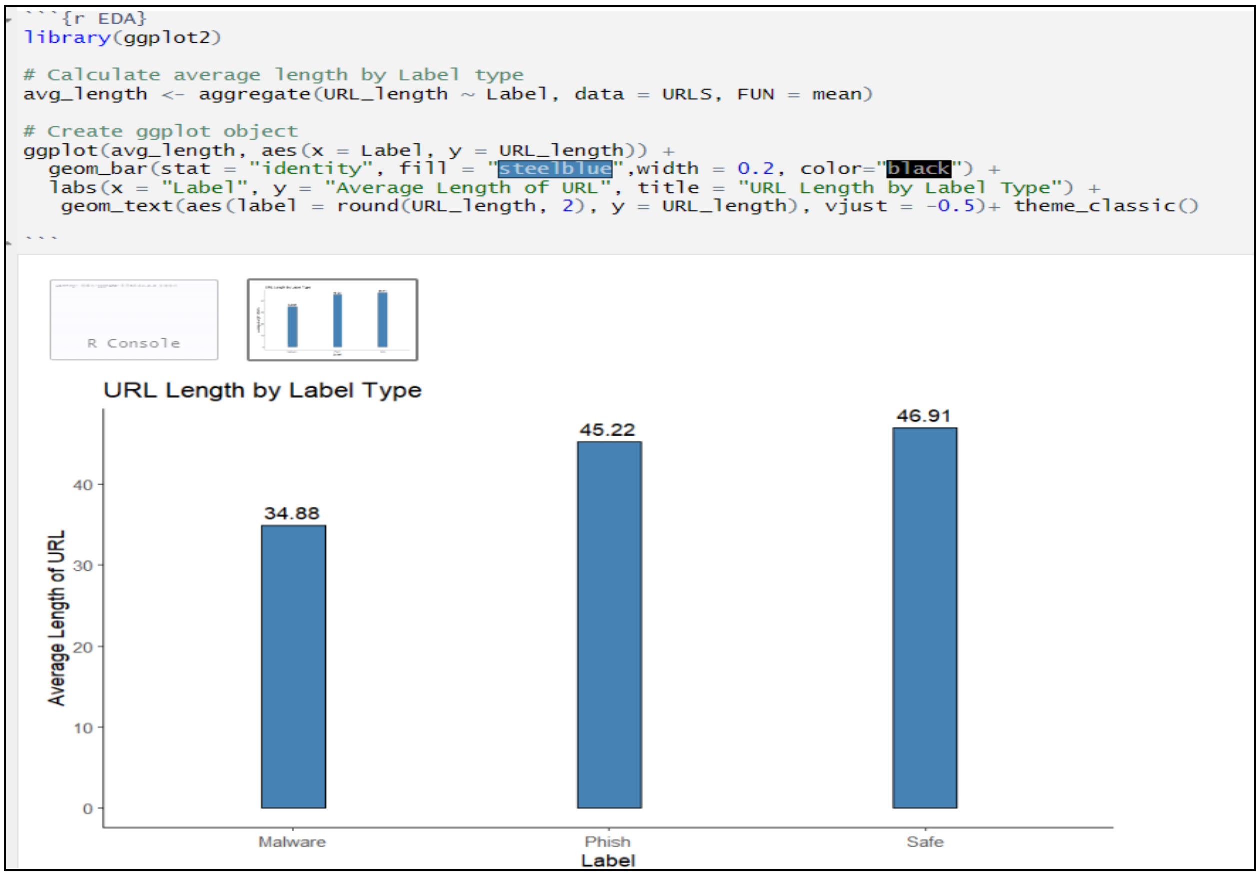
```
for page in range(0, max_pages+201):
#     print(f"Scraping page {page}")
    page_url = url.format(page)
    response = requests.get(page_url).text
    soup = BeautifulSoup(response, features="html.parser")
    trs += soup.find_all("td", class_="value")
#     print(f"Number of entries collected so far: {len(trs)}")
    time.sleep(pause_time)

for i in range(0,len(trs)):
    j = i % 5
    if i == 0 or j == 0:
        ids.append(trs[i].text)
        flag=trs[i].text
    elif i != 0 and j % 4 == 0:
        online.append(trs[i].text)
    elif i != 0 and j % 3 == 0:
        valid.append(trs[i].text)
    elif i != 0 and j % 2 == 0:
        #submitted.append(trs[i].text)
        pass
    else:
        phish_url.append(trs[i].contents[0].text)

data = {
    'id': ids,
    'phish_url': phish_url,
#     'submit_time': submitted,
    'valid': valid,
    'online': online
}
```

c. Sample code for Feature Engineering:



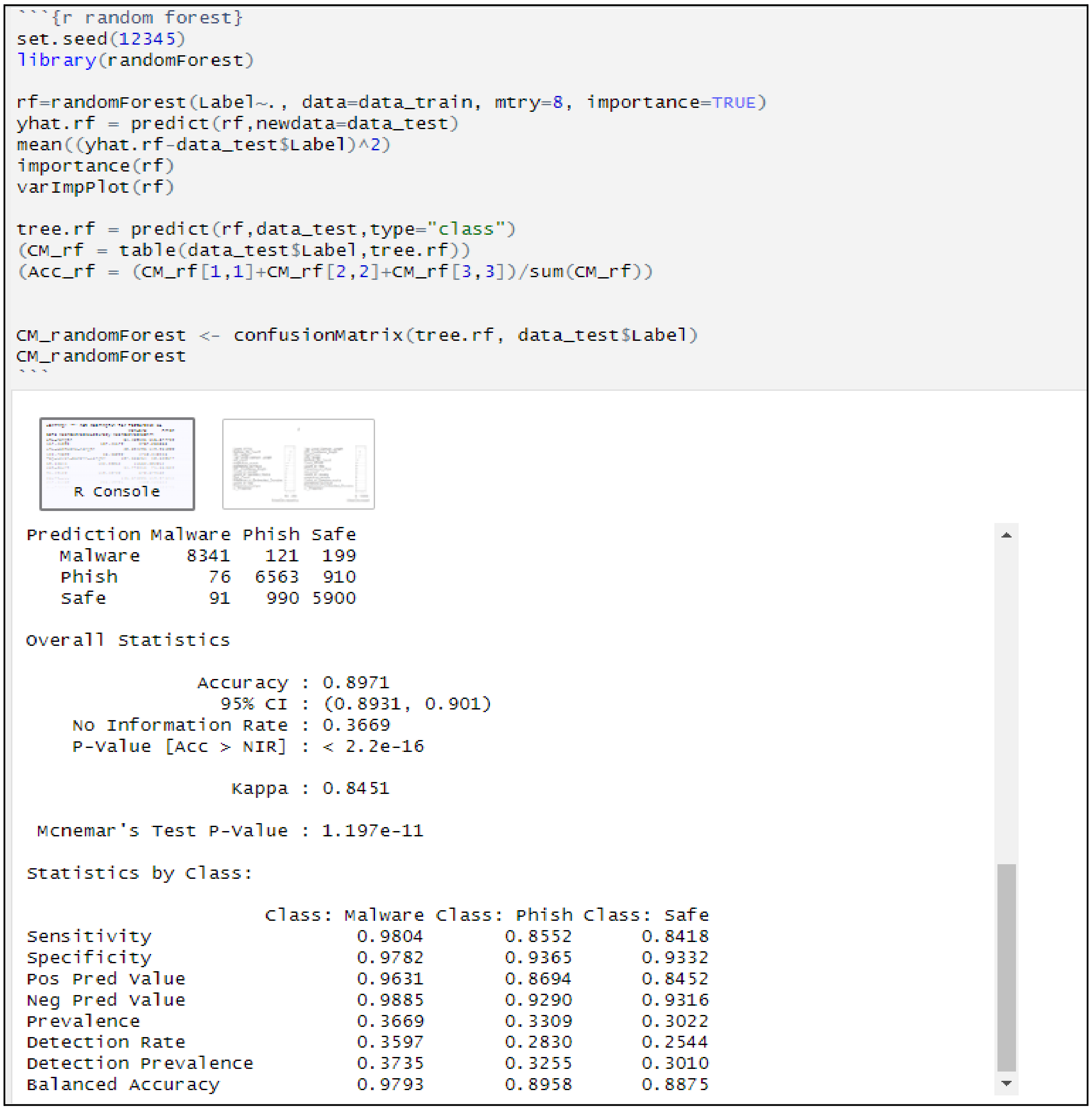
```
[ ]  import urllib.parse

     def hostname_len(url):
         parsed_url = urllib.parse.urlparse(url)
         hostname = parsed_url.hostname
         if hostname is not None:
             hostname_length = len(hostname)
         else:
             hostname_length = 0
         return hostname_length

     data_cl['URL_HostName_length'] = data_cl['URL'].apply(hostname_len)
     data_cl.head()
```

d. Sample code for Exploratory Data Analysis:



~~~
```{r EDA}
library(ggplot2)

# Calculate average length by Label type
avg_length <- aggregate(URL_length ~ Label, data = URLS, FUN = mean)

# Create ggplot object
ggplot(avg_length, aes(x = Label, y = URL_length)) +
  geom_bar(stat = "identity", fill = "steelblue",width = 0.2, color="black") +
  labs(x = "Label", y = "Average Length of URL", title = "URL Length by Label Type") +
   geom_text(aes(label = round(URL_length, 2), y = URL_length), vjust = -0.5)+ theme_classic()

```
~~~

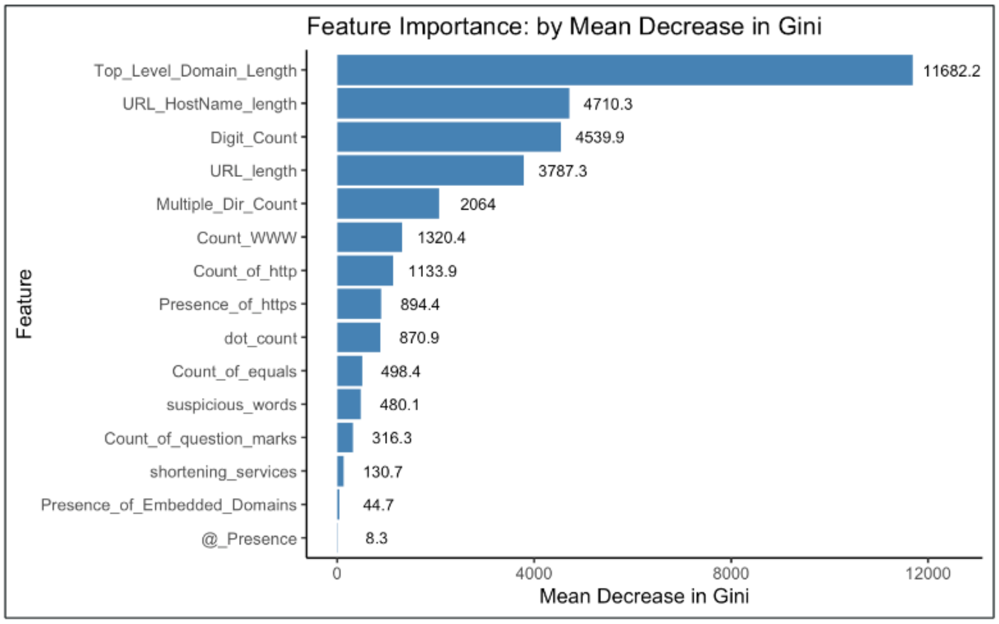
e. Sample code for Modeling:



~~~
```{r random forest}
set.seed(12345)
library(randomForest)

rf=randomForest(Label~., data=data_train, mtry=8, importance=TRUE)
yhat.rf = predict(rf,newdata=data_test)
mean((yhat.rf-data_test$Label)^2)
importance(rf)
varImpPlot(rf)

tree.rf = predict(rf,data_test,type="class")
(CM_rf = table(data_test$Label,tree.rf))
(Acc_rf = (CM_rf[1,1]+CM_rf[2,2]+CM_rf[3,3])/sum(CM_rf))


CM_randomForest <- confusionMatrix(tree.rf, data_test$Label)
CM_randomForest
```
~~~

```
Prediction Malware Phish Safe
   Malware    8341   121  199
   Phish        76  6563  910
   Safe         91   990 5900

Overall Statistics

               Accuracy : 0.8971
                 95% CI : (0.8931, 0.901)
    No Information Rate : 0.3669
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8451

 Mcnemar's Test P-Value : 1.197e-11

Statistics by Class:

                     Class: Malware Class: Phish Class: Safe
Sensitivity                  0.9804       0.8552      0.8418
Specificity                  0.9782       0.9365      0.9332
Pos Pred Value               0.9631       0.8694      0.8452
Neg Pred Value               0.9885       0.9290      0.9316
Prevalence                   0.3669       0.3309      0.3022
Detection Rate               0.3597       0.2830      0.2544
Detection Prevalence         0.3735       0.3255      0.3010
Balanced Accuracy            0.9793       0.8958      0.8875
```

f. Rank for Mean Decrease in Gini:


Feature Importance: by Mean Decrease in Gini
Top_Level_Domain_Length
URL_HostName_length
Digit_Count
URL_length
Multiple_Dir_Count
Count_WWW
Count_of_http
Presence_of_https
dot_count
Count_of_equals
suspicious_words
Count_of_question_marks
shortening_services
Presence_of_Embedded_Domains
@_Presence
Feature
Mean Decrease in Gini