Data Programming & Analytics Final Project
Data Analytics - Cyber Security, Malicious Websites

Written by
Daniel W. Lee, Yuzi Liu, Ziwei Liang, Maryna Pavlenko

Guided by
Dr. Tingting Nian

Winter 2019

Introduction

One of the biggest threat in the cyber world is malicious websites, and most people are unaware of the fact that we don't have to intentionally download, visit, or click on a malicious attachment in order to compromise computer security. The malicious website comprises computer security by doing nothing but just instigating to visit a website. It often looks like a legitimate website and asks a user to install a software that the computer appears to need. Once the malicious attachments are installed, it will do everything to disrupt computer operations, gather personal information, and in the worst-case scenario, it will gain total access of the machine. In today's generation, many malware control software allow users to prevent visiting this type of websites by spotting a potential threat and warning a user before the visit. Our project's objective is to evaluate different classification models to predict malicious and benign websites.

In this project, we attempt to answer the following questions:

1. How to detect a malicious website?
2. What are the characteristics of malicious websites?

The tools used in the project include Python, R, and Tableau. Our team performed a substantive analytical research to solve this classification problem and build a model upon it to predict malicious and benign websites.

Data Overview

Our team chose a dataset from Kaggle for the project, and it contains the following variables:

URL: The address that the website located.
URL_LENGTH: The length of the address that the website located.
NUMBER_SPECIAL_CHARACTERS: The number of special characters that are included in the address that the website is located.
CHARSET: Character encoding method
SERVER: The server that host the website
CONTENT_LENGTH: The length of the content on the website
WHOIS_COUNTRY: The country where the website is hosted.
WHOIS_STATEPRO: The state in the country where the website is hosted.
WHOIS_REGDATE: The date that the website is registered.
WHOIS_UPDATE_DATE: The last date that the website was updated.
TCP_CONVERSATION_EXCHANGE: The number that the server receive conversation through the transmission control protocol
DIST_REMOTE_TCP_PORT: The number of transmission control protocol that the website has.
REMOTE_IPS: The number of IP address that the website has remote access.

APP_BYTES: The size of the application that is located in the website.
SOUCE_APP_PACKETS: The unit of data from the source application on the website.
REMOTE_APP_PACKETS: The unit of data from the remote application on the website.
SOURCE_APP_BYTES: The size of data from the source application on the website.
REMOTE_APP_BYTES: The size of data from the remote application on the website.
APP_PACKETS: The unit of data from the applications on the website.
DNS_QUERY_TIMES: The time that domain name system took to run the query.
TYPE: Numeric variable that provide type initial classification for the future usage.

```
In [4]: #quick overview of data
        dataset.head()
```

Out[4]:

	URL	URL_LENGTH	NUMBER_SPECIAL_CHARACTERS	CHARSET	SERVER	CONTENT_LENGTH	WHOI
0	M0_109	16	7	iso-8859-1	nginx	263.0	None
1	B0_2314	16	6	UTF-8	Apache/2.4.10	15087.0	None
2	B0_911	16	6	us-ascii	Microsoft-HTTPAPI/2.0	324.0	None
3	B0_113	17	6	ISO-8859-1	nginx	162.0	US
4	B0_403	17	6	UTF-8	None	124140.0	US

5 rows × 21 columns

Preparation - Data Pre-Processing

Classifying websites may sound very easy, but there are many variables that the engineers should take into consideration especially when dealing with the web-based dataset. The numpy and pandas are both very helpful analytics libraries in Python 3 environment; our team heavily utilized these libraries in this project. By running some input and description functions, we were able to get a general overview and statistical summary of the data. It is important to look into these variables and functions before moving forward as the future data modeling will deal with the critical variables selection. The following illustration demonstrates how we use python related functions to study the basic statistical summary of our dataset:

Descriptive statistics

```
dataset.describe()
```

	URL_LENGTH	NUMBER_SPECIAL_CHARACTERS	CONTENT_LENGTH	TCP_CONVERSATION_EXCHANGE	DIST_REMOTE_TCP_PORT	REMOTE_IPS	APP_
count	1781.000000	1781.000000	1781.000000	1781.000000	1781.000000	1781.000000	1.7810
mean	56.961258	11.111735	13497.243964	16.261089	5.472768	3.060640	2.9823
std	27.555586	4.549896	38415.552697	40.500975	21.807327	3.386975	5.6050
min	16.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	39.000000	8.000000	603.000000	0.000000	0.000000	0.000000	0.0000
50%	49.000000	10.000000	4714.750000	7.000000	0.000000	2.000000	6.7200
75%	68.000000	13.000000	12578.500000	22.000000	5.000000	5.000000	2.3280
max	249.000000	43.000000	649263.000000	1194.000000	708.000000	17.000000	2.3629

1) Handling Missing Values

In the dataset, our team noticed that the variable URL is absolutely unique, which is something that we do not want to include for training the model. In addition, at this early stage, we also look into any null or empty value. Null or empty value is everywhere as not all the things are accessible on any particular website, and those values are going to be a huge problem with the model that we will be building and training. So we are using interpolate function to interpolate missing data:

```
dataset = dataset.interpolate()
print(dataset.isnull().sum())
```

```
URL_LENGTH                 0
NUMBER_SPECIAL_CHARACTERS  0
CHARSET                    0
SERVER                     1
CONTENT_LENGTH             0
WHOIS_COUNTRY              0
WHOIS_STATEPRO             0
TCP_CONVERSATION_EXCHANGE  0
DIST_REMOTE_TCP_PORT       0
REMOTE_IPS                 0
APP_BYTES                  0
SOURCE_APP_PACKETS         0
REMOTE_APP_PACKETS         0
SOURCE_APP_BYTES           0
REMOTE_APP_BYTES           0
APP_PACKETS                0
DNS_QUERY_TIMES            0
Type                       0
dtype: int64
```

2) Handling Categorical Values

First of all, we grouped values with the least count into one bin “other” to reduce number of unique values.

```
dataset["WHOIS_COUNTRY"].value_counts()
```

```
US       808
ES        63
CA        57
Other     48
AU        23
GB        17
PA        11
IN         8
JP         7
CN         4
Name: WHOIS_COUNTRY, dtype: int64
```

We also cleaned the data for the duplicate values in WHOIS_COUNTRY and WHOIS_STATEPRO. The latter helped to reduce the number of unique values in the categorical variables which facilitated further analysis and model building:

```
#Handling duplicate values for countries
def replace(x):
    if x == "[u'GB'; u'UK']"or x=="United Kingdom" or x=="UK":
        return "GB"
    elif x == "Cyprus":
        return "CY"
    elif x == "us":
        return "US"
    elif x == "ru":
        return "RU"
    elif x == "se":
        return "SE"
    else:
        return x

dataset["WHOIS_COUNTRY"] = list(map(lambda x: replace(x), dataset["WHOIS_COUNTRY"]))
```

Another important step in our data processing was to choose the appropriate method to convert our categorical variables into numeric. Some of the machine learning and data analytics methods have very tight rules on the type of a variable. For instance, it would be more efficient if we use numeric variables to run a random forest and any kind of decision tree model. Therefore, we looked at different approaches and selected binary encoding as the method that fitted our data the most. It is a combination of label encoding and one hot encoding to create a binary column that meets our needs for further analysis. This method encodes the data in fewer dimensions that one-hot encoding, and it doesn't give any weight to any values like label encoding does.

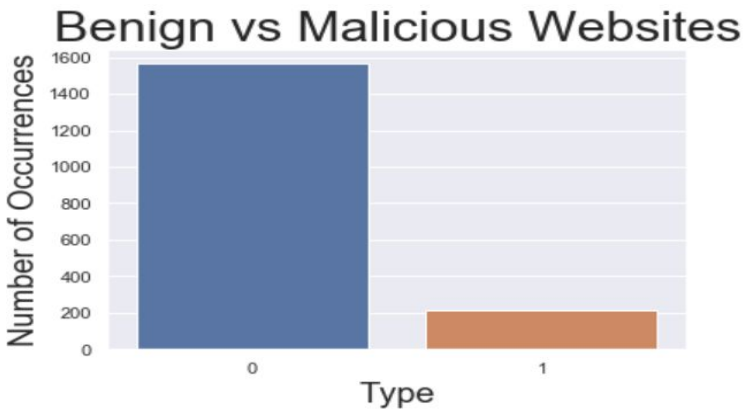
```
#binary encoding of categirical variables
dataset_ce = dataset.copy()
encoder = ce.BinaryEncoder(cols=['WHOIS_STATEPRO','WHOIS_COUNTRY', 'CHARSET', 'SERVER'])
df_binary = encoder.fit_transform(dataset_ce)
df_binary.head()
```

	WHOIS_STATEPRO_0	WHOIS_STATEPRO_1	WHOIS_STATEPRO_2	WHOIS_STATEPRO_3	WHOIS_STATEPRO_4	WHOIS_STATEPRO_5	WHOIS_COUNTRY_0
0	0	0	0	0	0	1	0
1	0	0	0	0	0	1	0
2	0	0	0	0	0	1	0
3	0	0	0	0	1	0	0
4	0	0	0	0	1	0	0

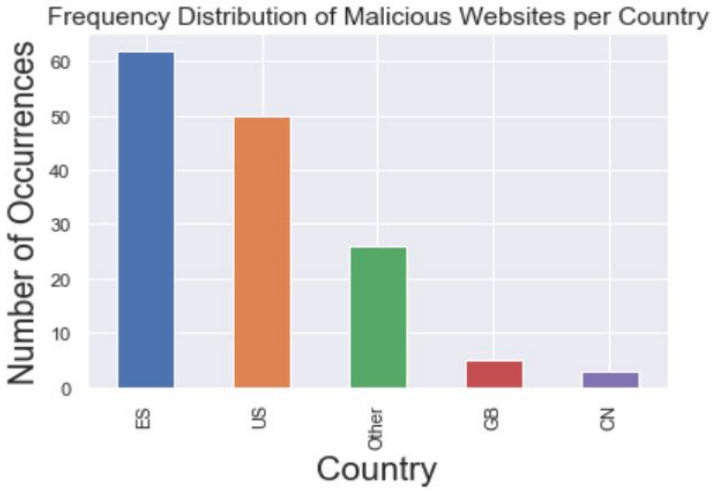
The data cleaning process is completed at this point. We have a set of variables to train, which enables us to move to the next step.

Visualizations

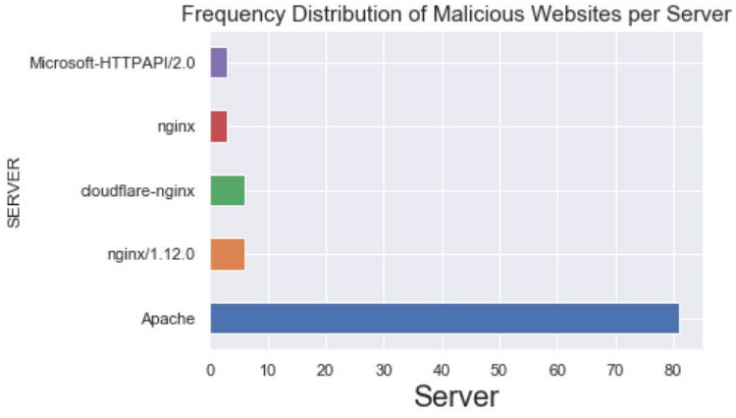
The team additionally included various visualizations into the analysis to gain a better understanding of the dataset.



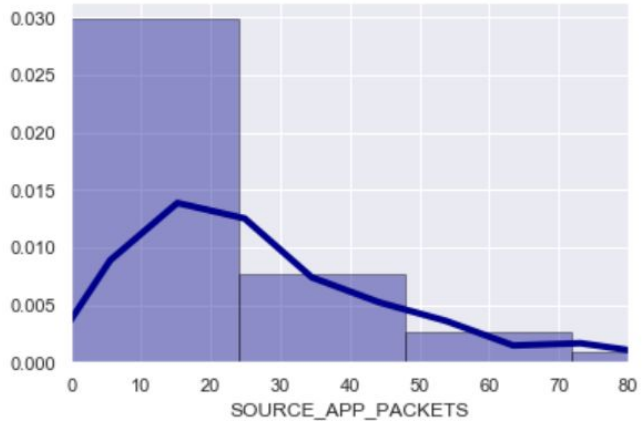
Malicious websites comprise 12% of all data.



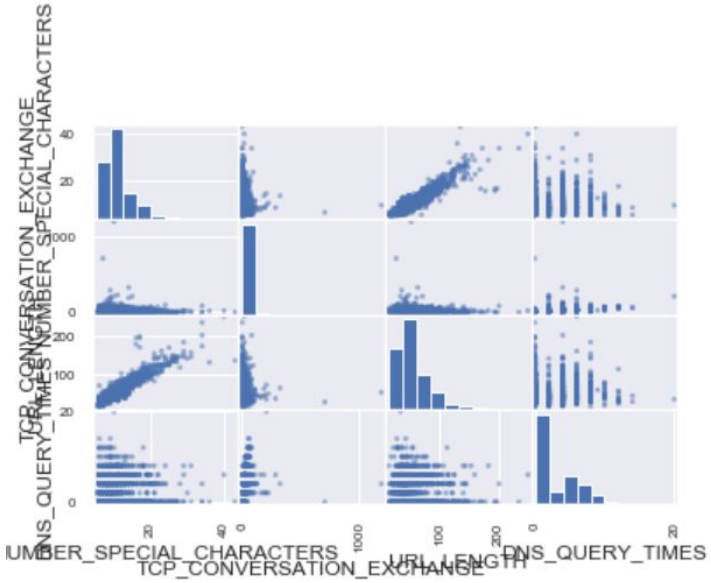
Based on this dataset, the highest amount of malware is observed in Spain and the United States.



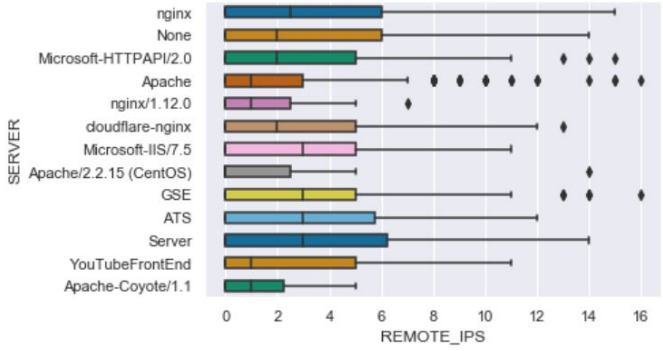
This illustration demonstrates that majority of malicious websites appear on Apache servers.



Right skewed distribution of Source App packets with the mean value in the range 14 to 18.



This scatterplot matrix revealed correlation between some variables.



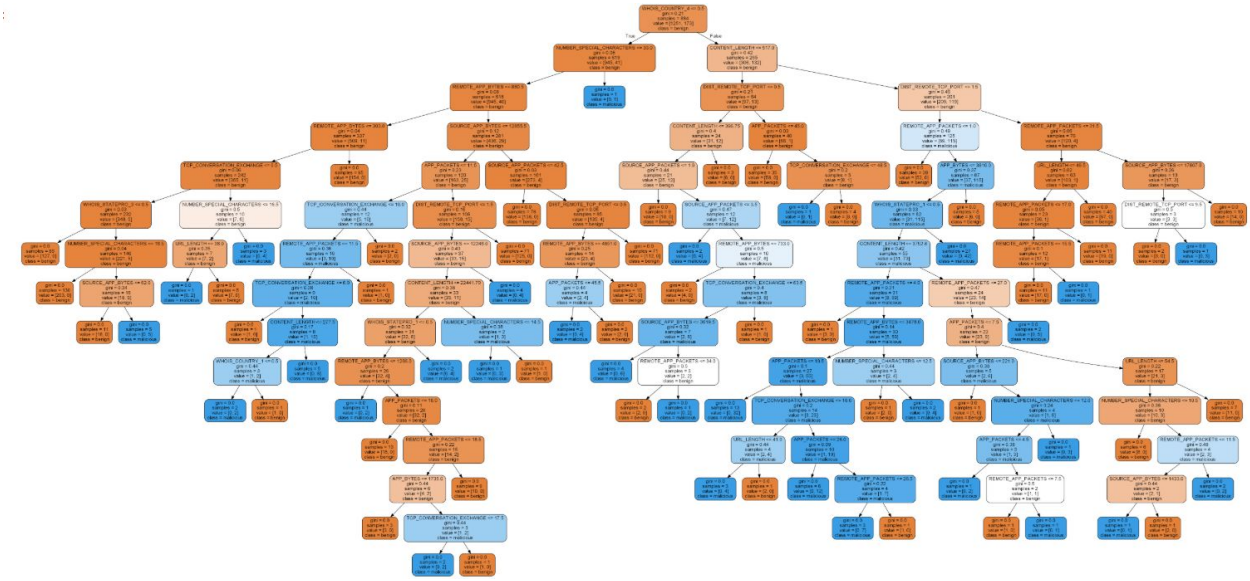
We can observe outliers of the variable and its quantile values per server.

Data Modeling

We split the dataset into the training and testing sets. The split ratio is set for 80:20, our team uses these two datasets to train and test our model then select the best one to move forward.

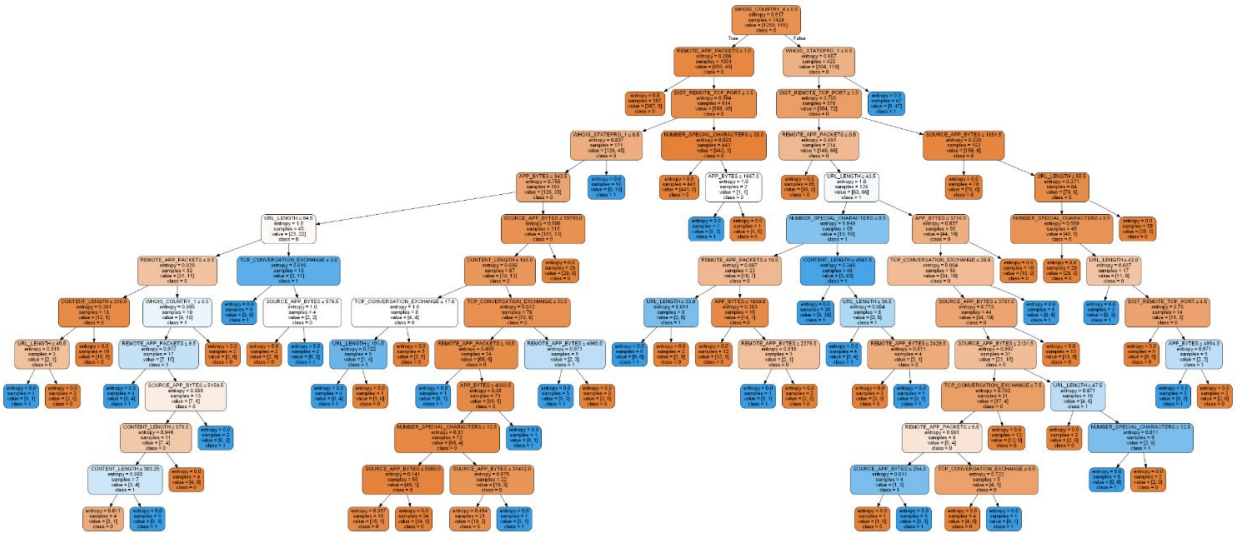
1) Random Forest

The project is now at the stage where our team needs to train the data by various machine learning methods in order to use the test data to predict and perform analysis, in this case, website classification. We made the assumption that random forest, decision tree, and Xgboost would be the best fit, therefore, we will mainly focus on these models while testing other machine learning techniques to make sure we select the best one. We trained the data with Random Forest Regressor by using Python data analytics library sklearn.model_selection, we selected the variables based on the feature importance. This provides us a better selection of the variables for the random forest model. This method yielded the accuracy of 94% on website classification.



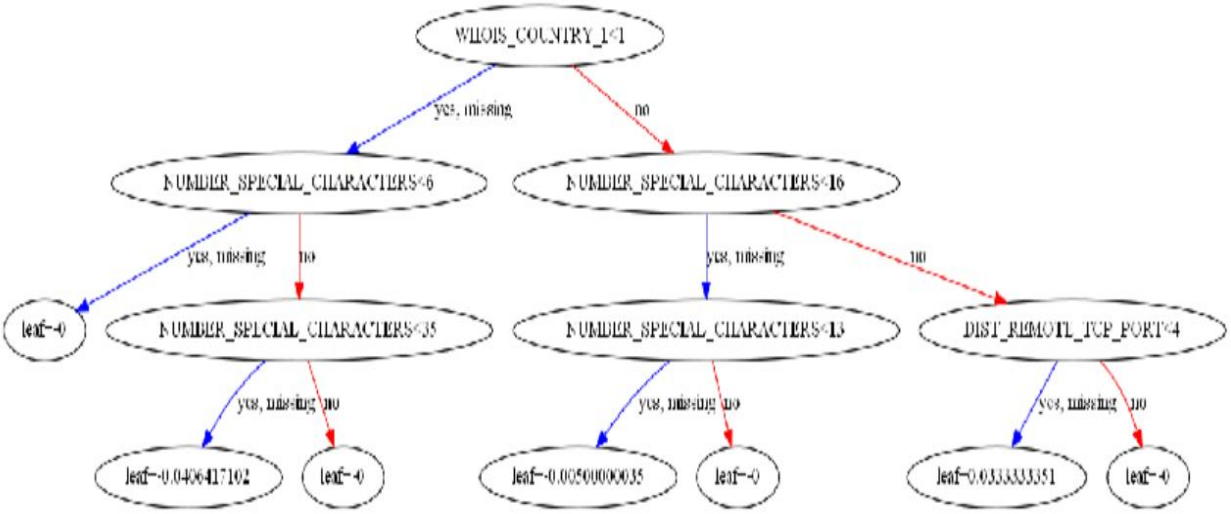
2) Decision Tree

Improving the dataset or trying other approach might or might not increase the accuracy of the machine learning method, but it's also worth to give it a try based on data science practice. In this case, our team decided to test decision tree approach, one of the commonly used machine learning methods. We noticed that the accuracy increased insignificantly by less than 1%.



3) Xgboost

We also wanted to try if other approaches can give us a better result, Xgboost is the next model that we tested. Our team converted the dataset into an optimized data structure, then instantiated the Xgboost regressor object from its library. We then fitted the regressor to the training set and made predictions on the test set. As a result, we reached the RMSE of 0.28 (the lower the better). After careful consideration, in order to build a more robust model, we then performed k-fold cross validation where all the entries in the original training dataset were used for both training as well as validation. Our team eventually received lower RMSE of 0.22.



Having tested various machine learning methods, we concluded that pruned version of decision tree produced the best accuracy rate so far.

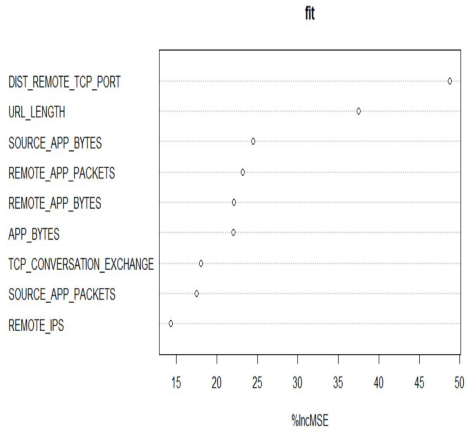
Additionally, we tried other machine learning models to predict malicious websites such as support vector machine, general linear model, and naive Bayes, but they were not as close as

decision tree and random forest. Nevertheless, these models did provide some insight to the data too.

Other Methods - Evaluation

1) Support Vector Machine

The support vector machine generated 427 support vectors, which indicates that the variables do associate with each other and have relationships. However, looking into the fitness of the model, it does not fit so well as to compare with the decision tree. Also the gamma and epsilon are significantly low to compete with other methods.




```
> fit <-svm(y_train ~ ., data = train_cap)# try using the train_cap and train_other
> summary(fit)

Call:
svm(formula = y_train ~ ., data = train_cap)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.1111111
    epsilon:  0.1


Number of Support Vectors:  427
```

2) Logistic Regression

The model turned out not as good as random forest and decision tree. The p-values for some of the variables are quite high, which indicates that the variables have a low level of significance.

```
> predicted= predict(fit,x_test)
> logistic <- glm(y_train ~ ., data = train_cap,family='binomial')
> summary(logistic)

Call:
glm(formula = y_train ~ ., family = "binomial", data = train_cap)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8037  -0.2734  -0.1904  -0.1201   3.9481

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)               -2.2645     0.9890  -2.290   0.0220 *
REMOTE_APP_PACKETS         3.8553     0.6196   6.222 4.90e-10 ***
URL_LENGTH                -0.4395     0.2440  -1.801   0.0717 .
SOURCE_APP_PACKETS         0.1241     1.1820   0.105   0.9164
REMOTE_APP_BYTES           0.4083     0.7794   0.524   0.6004
SOURCE_APP_BYTES           0.2236     0.1431   1.562   0.1183
APP_BYTES                 -0.2506     0.7574  -0.331   0.7408
REMOTE_IPS                 0.4555     0.2760   1.650   0.0988 .
DIST_REMOTE_TCP_PORT      -1.5057     0.1772  -8.497   < 2e-16 ***
TCP_CONVERSATION_EXCHANGE -4.4254     1.1116  -3.981 6.86e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1316.04  on 1780  degrees of freedom
Residual deviance:  781.97  on 1771  degrees of freedom
AIC: 801.97

Number of Fisher Scoring iterations: 7
```

3) Naive Bayes

After running the Naive Bayes, one of the most common machine learning models that generated a table with the values of true-positive, true negative, false positive, and false negative. We can see that Naive Bayes appeared to be not the best approach to solve our classification and prediction problem. The summary of fitness does not give our team much information to further develop the model. Also, the apriori is the unit of probability of estimation, which is the value and numerator that we wished to maximize based on the 21 tables provided through naive bayes classifier.

```
> fit <-naiveBayes(y_train ~ ., data = train_combine)#read and try to find parameters
> summary(fit)
        Length Class  Mode
apriori  2     table  numeric
tables  21     -none- list
levels   0     -none- NULL
call     4     -none- call
```

Website Classification

At the end of this project we were able to reach the training accuracy 100%, and testing accuracy of 95.24%. The websites were categorized into four big groups based on the following variables: Number_of_Specical_Character, State, Country, Remote_App_Bytes, Content_Length, and Source_App_Bytes.

Malicious Website: The websites have a higher number of special characters, in addition, much irrelevant information. For instance, a website that has the state California while the country is England. The content length and remote app bytes all have a higher number than others, while souce app bytes have the lower number.

(High) Suspect Website: The website that does not contains any senseless information, however, the number of special character still higher than average. Also the content length and remote app bytes still tend to be high while source app bytes remain low.

(Low) Suspect Website: The websites that contain reasonable geographical information and number of special characters. The content length and remote app bytes still tend to be high, but the source app bytes are about average.

Safe Website: The information from the website is up-to-date, in addition, the number of special characters, content length, remote, and sources app bytes are all in a reasonable range (average plus or minus ten).

Conclusion

Throughout the course of his project, we evaluated different types of models for the websites classification and prediction (mostly focus on decision tree and random forest since they gave our team the best result), we learned that malicious websites in this generation are classifiable and predictable. However, cyber attack is the field that will have continuous growth, and on the defensive side of it, it's cyber protection. Data analytics is one small part of cyber protection process. By studying data analytics, we will be able to apply the concept into cyber protection just like what our team performed in this project - classify potential threat and see the model accuracy. In addition, data analytics is the knowledge that can bring many contributions to different fields such as cyber security, healthcare, engineering, business, etc. What matters the most, is to be able to identify the opportunity and use data science practice to help make better-informed decisions and solve business problems of different complexity.