BDA Project: Malicious And Benign Website URL Detection

Nguyen Xuan Binh

1st February 2023

Contents

Introduction

1. Central problem

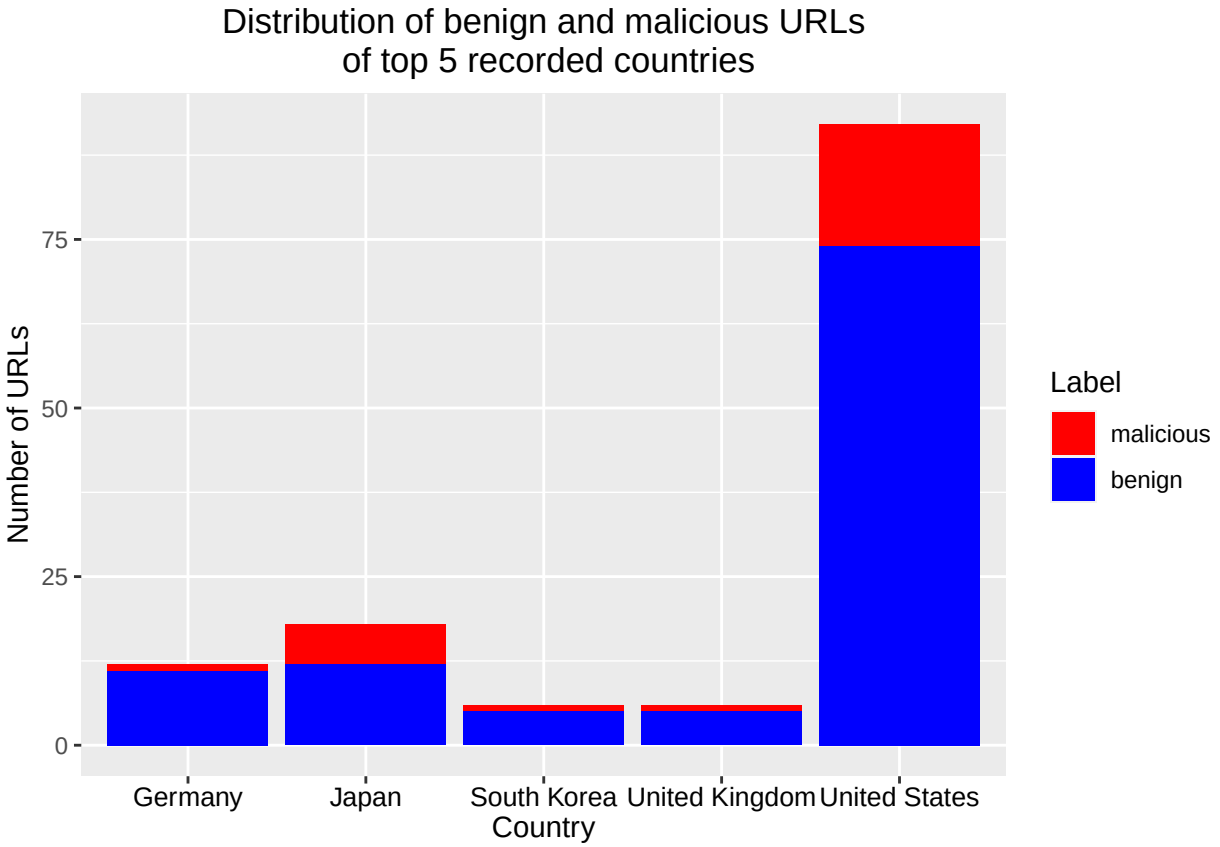
Detection of malicious URLs among the benign ones is a crucial goal of modern-day cybersecurity as it helps prevent individuals and organizations from falling victim to phishing, data breaching, malware infections, and other types of cyber threats. The most common type is phishing, where the URLs are disguised as valid sites to trick users into revealing their credentials. Some other types even install harmful softwares or redirect users to other malicious sites. With the rapid growth of the internet and the increasing dependence on technology, black-hat hackers and thieves have found innovative ways to spread their malicious content through fake URLs. A 2017 report from Cybersecurity Ventures predicted ransomware damages would cost the world $5 billion in 2017, up from $325 million in 2015 — a 15X increase in just two years. The damages for 2018 were predicted to reach $8 billion, and for 2019 the figure is $11.5 billion (Morgan (2019)). Therefore, it is an urgent task to automate the process of detecting and blocking malicious URLs floating on the net.

2. Motivation

In order to protect against these threats, it is essential to detect malicious URLs and prevent individuals and organizations from accessing them. This can be accomplished through various techniques, including URL reputation analysis, machine learning algorithms, and network security solutions. By detecting and blocking malicious URLs, individuals and organizations can better protect themselves and their sensitive information from cyber-attacks. In this report, I aim to detect malicious URLs among the benign or safe ones based on various features of the URLs and the websites associated with them. The analysis method will be based on the Bayesian inference approach to account for past data on the recorded URLs.

3. Main modeling idea

The problem is the detection of malicious URLs, which means it is a classification task. As a result, I will heavily use the Beta distribution to model the probabilities for each feature and the Bernoulli distribution for modeling the likelihood of both labels and features. Based on intuition, the rate of malicious URLs is expected to vary depending on the countries and regions. For example, reputable countries in cybersecurity, such as Finland and UK, will host much fewer malicious domains/URLs. In contrast, others, such as Russia, China, and Vietnam, are less regulated and will have more harmful network content. From this belief, I decided to split the recorded URLs depending on the countries and proceeded to perform two Bayesian models: the pooled model, where the rates of malicious URLs from each country are individually analyzed, and the pooled model, where all URLs are merged and treated as if they come from only one source. Both models are equally valid in that the pooledd model looks from the perspective of regional difference, while the pooled model looks from the common origin on the Domain Name System (DNS). Below is the illustration of malicious and benign URLs distribution among the recorded countries.



Dataset

1. Data description

The dataset in this report is collected by an author named A.K.Singh in his research paper for International Conference on Communication Systems & Networks (Singh and Goyal (2019)). This dataset specifically caters for machine learning-based classification analysis of malicious and benign webpages. According to the author, this dataset comprises of various extracted attributes and raw webpage content, which are:

- 'url' - (string) The URL of the webpage
- 'ip_add' - (string) IP address of the webpage.
- 'geo_loc' - (string - categorical) The geographic location where the webpage is hosted.
- 'url_len' - (float) The length of URL.
- 'js_len' - (float) Length of JavaScript code on the webpage.
- 'js_obf_len - (float) Length of obfuscated JavaScript code.
- 'tld' - (string - categorical) The top level domain of the webpage.
- 'who_is' - (binary) Whether the WHO IS domain information is complete or not.
- 'https' - (binary) Whether the site uses https or http.
- 'content' - (string) The raw webpage content including JavaScript code.
- 'label' - (binary) The class label for benign or malicious webpage.

Because his dataset is extremely heavy and extensive (1.3 million datapoints for training data and nearly 340000 datapoints for testing data), I only extract a tiny portion from it, which is 120 training and 250 testing datapoints to allow the Stan sampling to run adequately fast.

2. Data source and analysis difference

The source of the dataset can be found at:

Data source description: https://data.mendeley.com/datasets/gdx3pkwp47/2

Data source download website: https://www.researchgate.net/publication/347936136_Malicious_and_Benign_Webpages_Dataset

It is also available on Kaggle: https://www.kaggle.com/datasets/aksingh2411/dataset-of-malicious-and-benign-webpages

The difference between this report and the paper of A.K.Singh is that he focuses on comparing various machine learning strategies to tackle this problem, which are supervised and unsupervised learning, while this report solely focuses on the Bayesian inference technique, combined with supervised learning to predict the malicious URLs. His paper did mention a Bayesian technique, but it is Naive Bayes Classifier, while the Bayesian technique in this project is based on probabilistic sampling in separate and pooled models.

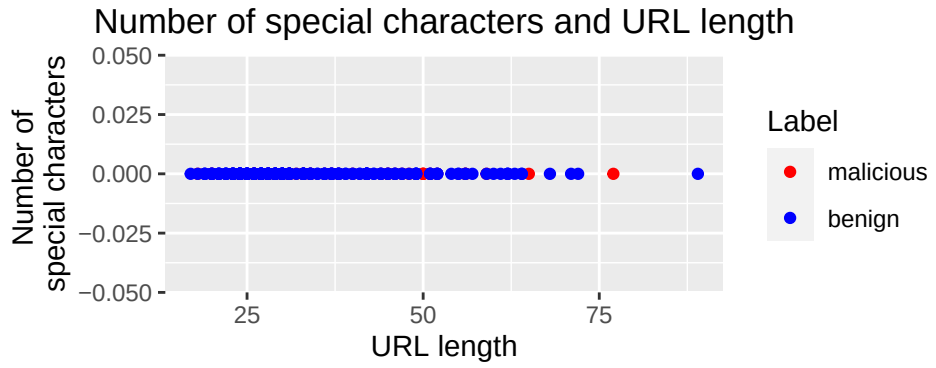
3. Feature selection and data cleaning

Feature selection is crucial to dimension reduction and model accuracy and runtime improvement. I have set two criteria: the number of features should be at most four, and the features should be highly correlated with the label (malicious/benign).

First is the URL itself. Based on the URL alone, it is hard to determine whether it is related to the underneath danger, so I decided to extract the number of special characters from the URL. This is the original dataset after I calculated the num_special column

```
head(test_websites)
```

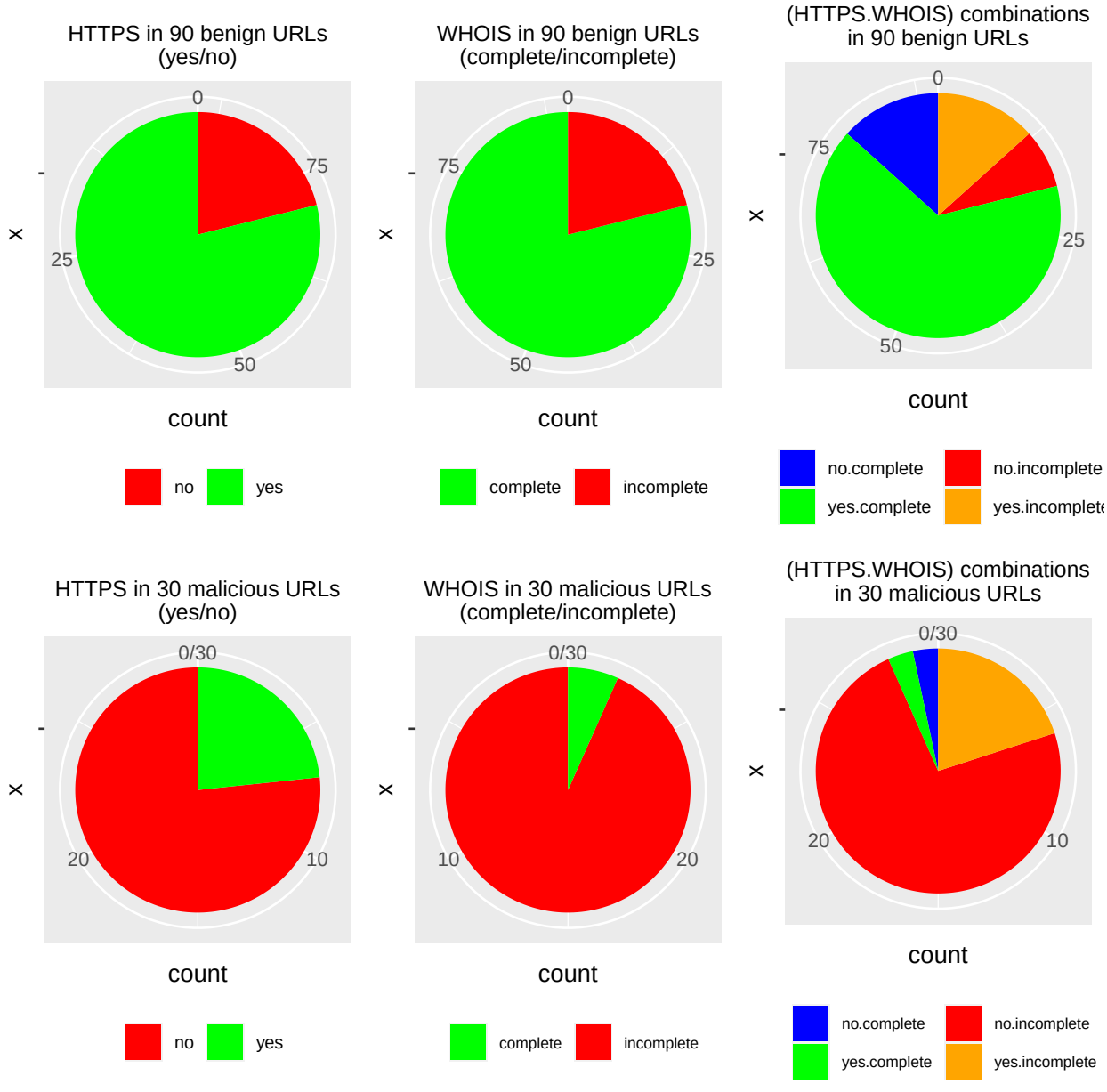
```
##   X label url_len       geo_loc https js_len js_obf_len     who_is num_special
## 1 1   bad      34 United States    no  324.0      0.000 incomplete           0
## 2 2   bad      50         China    no  449.1    273.951 incomplete           0
## 3 3   bad      29         India    no  484.2    329.256 incomplete           0
## 4 4   bad      46        Turkey    no  745.2    514.188 incomplete           0
## 5 5   bad      56       Germany    no  432.0    194.400 incomplete           0
## 6 6   bad      31     Argentina    no  393.3    275.310 incomplete           0
```



It appears that the URL name itself is not helpful for prediction, including its length. This can seen from the graph above as all malicious and benign URL lengths are randomly distributed, while the number of special characters are all 0s. Therefore, I omitted the features “url” and “url_len.”

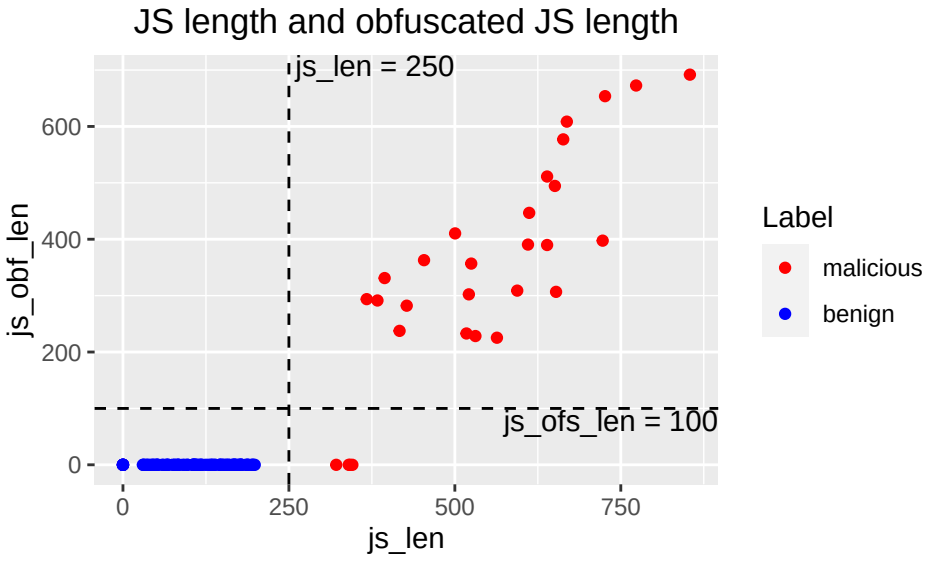
Next is the top level domain (tld) name. While some domains may be notoriously dangerous, such as .zip, .link and .review, most of the tld in the dataset are the most popular domains, such as .com, .org and .net. All of them are equally likely to be benign or malicious as they are prevalent on WWW. As a result, I omitted tld feature as it is not particularly helpful in prediction.

Next, I examine two features: https and whois. HTTPS (Hypertext Transfer Protocol Secure) is an extension of the HTTP. It uses encryption for secure communication over a computer network, and is widely used on the Internet. As a result, websites with https are much more likely to be secure and safe compared to http websites. On the other hand, WHOIS is a query and response protocol that is widely used for querying databases that store the registered users or assignees of an Internet resource, such as a domain name, an IP address block or an autonomous system. Websites with completed whois registration is much safer and transparent than unregistered websites. Therefore, I decided to keep these two features.



From the pie charts, it is evident that https and completed whois registration are strongly related to the label. Most benign websites have https and completed whois but otherwise for malicious ones.

Finally, two features left are the Javascript length and the obfuscated Javascript length of the web raw content. Javascript is the native language of web browsers and inherent to all running websites. Because Javascript code is open-source by Inspection, some organizations want to prevent others to copy their code. Therefore, JavaScript obfuscation is a series of code transformations that turn plain, easy-to-read JS code into a modified version that is extremely hard to understand and reverse-engineer.



When plotting them together, a clear pattern has arisen. It appears that all URLs having JS length longer than 250 are malicious and benign when smaller than 250. Regarding the obfuscated JS length, if it is larger than 100, the URL is almost certain to be malicious. If it is smaller than 100, the URL is likely to be benign, as the number of red points are much smaller than the blue points under the line js_obf_len = 100. Observing this distinction, I decided to transform these two float features into binary formats as follows:
- $js_{len} \geq 250 \Rightarrow js_{len\_binary} = 1$ and 0 otherwise
- $js_{obf\_len} \geq 100 \Rightarrow js_{len\_obf\_binary} = 1$ and 0 otherwise

Because the https, whois and label columns are in string formats, I also need to convert them to binary formats (0/1) so that it can be passed into the Stan models. The binary labels are:
- $label =$ "$good$" $\Rightarrow label\_bin = 0$ and $label =$ "bad" $\Rightarrow label\_bin = 1$
- $https =$ "yes" $\Rightarrow https\_bin = 0$ and $https =$ "no" $\Rightarrow https\_bin = 1$
- $whois =$ "$complete$" $\Rightarrow whois\_bin = 0$ and $whois =$ "$incomplete$" $\Rightarrow whois\_bin = 1$

From this binary format, it appears that js_len and js_obf_len have inverse proportion while https and whois have direct proportional to the label according to the analysis above. In total, there are four features in this report: https, whois, js_len and js_obf_len. Finally, the geo_loc column indicates which country the URL originates from. It is used to partition the URLs into different countries for the separate and pooled models. As a result, geo_loc is not a feature in this report. The cleaned dataframe now becomes:.

```
##   label_bin       geo_loc https_bin whois_bin js_len_bin js_obf_len_bin
## 1         1         China         0         1          1              1
## 2         1 United States         1         1          1              1
## 3         1       Germany         1         1          1              1
## 4         1 United States         1         1          1              1
## 5         1 United States         1         1          0              0
## 6         1         China         1         1          1              0
```

Separate model

1. Model description

In the separate model, the dataset is partitioned into different countries, each with its own sets of malicious and benign URLs. All of the features and labels in this report are binary, which means they are modeled with Beta distribution as the prior for their probabilities and Bernoulli distribution for their likelihood. This is because the Beta distribution is the conjugate prior of the Bernoulli distribution. Intuitively, the Beta distribution can be understood as representing a distribution of probabilities; that is, it represents all the possible values of a probability when we are unsure of what that probability is. After modeling the probabilities, the Bernoulli distribution models the probability of observing a malicious URL (1) in a single trial, given a prior probability from the Beta distribution.

In the separate model, $k \in 1 : K$ is the index of the country. For example, $k = 1$ corresponds to China, $k = 2$ is the USA, and $k = 3$ is Germany. There are four probability parameters to be modeled in total.

$$\theta_{https_k} \sim Beta(\alpha, \beta) \Rightarrow https_k \sim Bernoulli(\theta_{https_k})$$

$$\theta_{whois_k} \sim Beta(\alpha, \beta) \Rightarrow whois_k \sim Bernoulli(\theta_{whois_k})$$

$$\theta_{js\_len_k} \sim Beta(\alpha, \beta) \Rightarrow js\_len_k \sim Bernoulli(\theta_{js\_len_k})$$

$$\theta_{js\_obf\_len_k} \sim Beta(\alpha, \beta) \Rightarrow js\_obf\_len_k \sim Bernoulli(\theta_{js\_obf\_len_k})$$

Besides the features themselves, the coefficients for each feature are all modeled with normal distributions. This is because the coefficients are likely to indicate direct/inverse proportion of the features with respect to the label. Therefore, it is useful to know their expected value and standard deviation. In the separate model, they are formulated as:

$$intercept_k \sim Normal(\mu, \sigma)$$

$$c_{https_k} \sim Normal(\mu, \sigma), \quad c_{whois_k} \sim Normal(\mu, \sigma)$$

$$c_{js\_len_k} \sim Normal(\mu, \sigma), \quad c_{js\_obf\_len_k} \sim Normal(\mu, \sigma)$$

The final important task is the formulation of the relationship between the features and the label. In this report, I use multiple linear regression of the features with an intercept to predict the labels. Additionally, the label is also binary (malicious/benign URL), which means it also follows a Bernoulli distribution, and its probability θ_{label} must be inside the range $[0, 1]$. A problem arises in the range of multiple linear regression: the regression prediction can yield any arbitrary value. Therefore, I used the inverse logit function to achieve this task. It is a sigmoid function that maps arbitrary real values back to the range [0, 1]. The larger the value, the closer to 1 it becomes. This function is also supported in Stan and usually accompanies the Bernoulli distribution. The inverse logit function is of the form

$$inv\_logit(x) = \frac{1}{1 + exp(-x)}$$

The label y_k of malicious/benign URL now follows the Bernoulli distribution of the inverse logit function of the multiple linear regression.

$$regression_k = intercept_k + c_{https_k} https_k + c_{whois_k} whois_k$$

$$+c_{js\_len_k} js\_len_k + c_{js\_obf\_len_k} js\_obf\_len_k$$

$$\theta_{y_k} = inv\_logit(regression_k)$$

$$y_k \sim Bernoulli(\theta_{y_k})$$

2. Prior choice and justifications

Because the default protocol https now becomes the standard for secure websites, many domain registrar already provided https out of the box. This means https is much more popular than the insecure http version. By some statistics, https is used by 81.5% of all the websites (W3Techs (2023)).

Regarding the WHOIS registration, it is recorded that there are 1.24 billion websites with complete WHOIS registration (Chen, Guan, and Su (2014)). As a whole, there are currently around 1.7 billion websites hosted on WWW. Therefore, the ratio of complete WHOIS website is $1.24/1.7 \approx 0.73$

For js_len and js_obf_len, I cannot find out any information about their priors, so I decided to use the priors based on the training dataset.

We can determine the α and β parameters for the Beta distribution as

$$\alpha = k + 1, \quad \beta = n - k + 1$$

Where k is the number of successes and n is the number of trials. Because the ratio is between very large number, α and β parameters are simply reduced to their respective ratios. Therefore, the priors for the probabilities of the features become:

$$\theta_{https_k} \sim Beta(8,2), \quad \theta_{whois_k} \sim Beta(7,3), \quad \theta_{js\_len_k} \sim Beta(1,2), \quad \theta_{js\_obf\_len_k} \sim Beta(1,2)$$

For the coefficients, I set the mean of https and whois as -1 since they are inversely proportional to the label, while js_len and js_obf_len as 1 because they are directly proportional. The standard deviation is weakly informative. Regarding the intercept, I expect it to be around 0, and its deviation is also weakly informative. In brief, the priors for the coefficients are:

$$intercept_k \sim Normal(0,20), \quad c_{https_k} \sim Normal(-1,10), \quad c_{whois_k} \sim Normal(-1,10)$$

$$c_{js\_len_k} \sim Normal(1,10), \quad c_{js\_obf\_len_k} \sim Normal(1,10)$$

3. Stan code and running options

The Stan code for the separate model:

```
"
data {
  int<lower=1> Nmax; // Number of maximum URLs among all countries (training)
  int<lower=1> Mmax; // Number of maximum URLs among all countries (testing)
  int<lower=1> K; // Number of countries
  array[K] int<lower=1> N_list; // Number of URLs of each country (training)
  array[K] int<lower=1> M_list; // Number of URLs of each country (testing)
  // The training features
  array[K, Nmax] int<lower=0,upper=1> js_len_list;
  array[K, Nmax] int<lower=0,upper=1> js_obf_len_list;
  array[K, Nmax] int<lower=0,upper=1> https_list;
  array[K, Nmax] int<lower=0,upper=1> whois_list;
  // The testing predicting features
  array[K, Mmax] int<lower=0,upper=1> js_len_pred_list;
  array[K, Mmax] int<lower=0,upper=1> js_obf_len_pred_list;
  array[K, Mmax] int<lower=0,upper=1> https_pred_list;
  array[K, Mmax] int<lower=0,upper=1> whois_pred_list;
  // label for each URL: benign(0) or malicious(1)
  array[K, Nmax] int<lower=0,upper=1> label_list;
```

```
}

parameters {
  array[K] real<lower=0, upper=1> theta_js_len; // probability for js_len
  array[K] real<lower=0, upper=1> theta_js_obf_len; // probability for js_obf_len
  array[K] real<lower=0, upper=1> theta_https; // probability for https
  array[K] real<lower=0, upper=1> theta_whois; // probability for whois
  array[K] real js_len_coeff; // Slope coefficient for js_len
  array[K] real js_obf_len_coeff; // Slope coefficient for js_obf_len
  array[K] real https_coeff; // Slope coefficient for https_coeff
  array[K] real whois_coeff; // Slope coefficient for whois_coeff
  array[K] real intercept; // Intercept coefficient
}

model {
    // Prior probabilities of the features
    for (k in 1:K){
        theta_js_len[k] ~ beta(1,2);
        theta_js_obf_len[k] ~ beta(1,2);
        theta_https[k] ~ beta(8,2);
        theta_whois[k] ~ beta(7,3);
    }
    // likelihood for the features
    for (k in 1:K){
        js_len_list[k, 1:N_list[k]] ~ bernoulli(theta_js_len[K]);
        js_obf_len_list[k, 1:N_list[k]] ~ bernoulli(theta_js_obf_len[K]);
        https_list[k, 1:N_list[k]] ~ bernoulli(theta_https[K]);
        whois_list[k, 1:N_list[k]] ~ bernoulli(theta_whois[K]);
    }
    // priors of the coefficients
    for (k in 1:K){
       js_len_coeff[k] ~ normal(1,10);
       js_obf_len_coeff[k] ~ normal(1,10);
       https_coeff[k]  ~ normal(-1,10);
       whois_coeff[k] ~ normal(-1,10);
       intercept[k] ~ normal(0,20);
    }
    // Modelling of the label based on bernoulli logistic regression by
    // multiple variable linear regression
    for (k in 1:K){
      for (i in 1:N_list[k]){
        label_list[k, i] ~ bernoulli(inv_logit(intercept[k]
            + https_coeff[k] * https_list[k, i]
            + whois_coeff[k] * whois_list[k, i]
            + js_len_coeff[k] * js_len_list[k, i]
            + js_obf_len_coeff[k] * js_obf_len_list[k, i]));
      }
    }
}

generated quantities {
    array[Nmax] real log_likelihood;
    for (k in 1:K) {
```

```
      if (N_list[k] == Nmax){
        for (i in 1:Nmax){
          log_likelihood[i] = bernoulli_lpmf(label_list[k, i] | inv_logit(intercept[k]
            + https_coeff[k] * https_list[k, i]
            + whois_coeff[k] * whois_list[k, i]
            + js_len_coeff[k] * js_len_list[k, i]
            + js_obf_len_coeff[k] * js_obf_len_list[k, i]));
        }
      }
    }
}

"
```

- Next, I prepare the data to be passed to the Stan separate model

```
stan_data <- list(
  Nmax = Nmax,
  Mmax = Mmax,
  K = K,
  N_list = N_list,
  M_list = M_list,
  js_len_list = as.matrix(do.call(rbind, js_len_list)),
  js_obf_len_list = as.matrix(do.call(rbind, js_obf_len_list)),
  https_list = as.matrix(do.call(rbind, https_list)),
  whois_list = as.matrix(do.call(rbind, whois_list)),
  js_len_pred_list = as.matrix(do.call(rbind, js_len_test_list)),
  js_obf_len_pred_list = as.matrix(do.call(rbind, js_obf_len_test_list)),
  https_pred_list = as.matrix(do.call(rbind, https_test_list)),
  whois_pred_list = as.matrix(do.call(rbind, whois_test_list)),
  label_list = as.matrix(do.call(rbind, label_list))
)
```

- This is the sampling running options

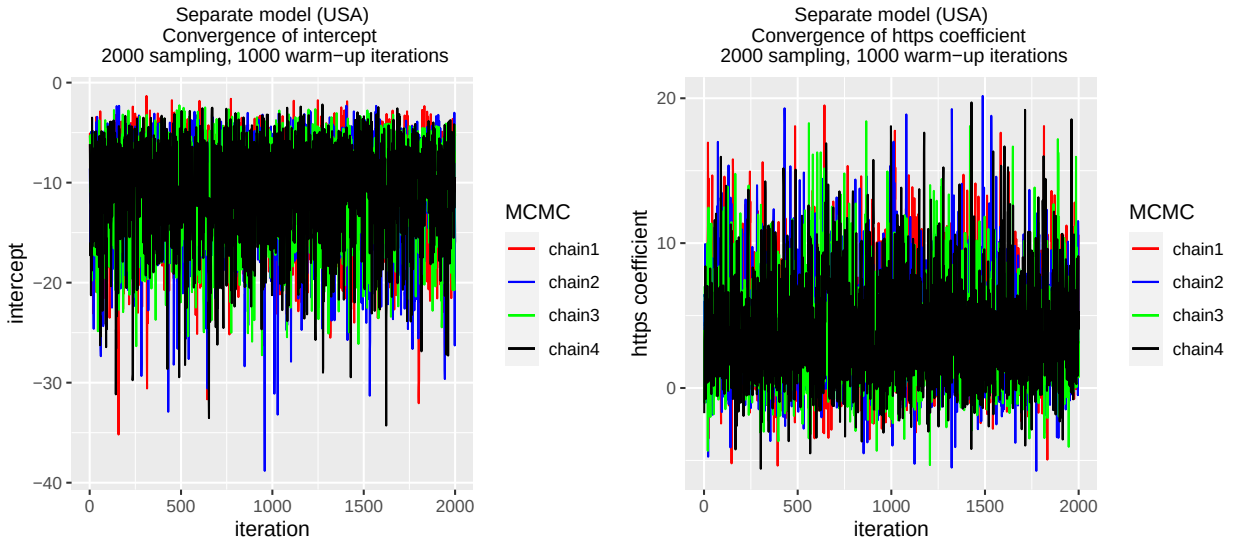
The sampling statement has 4 chains, which is the default number of chains that yield meaningful diagnostics such as $\hat{R}$ and ESS. The number of warm-up iterations is only 1000 to speed up the running time, since I see no significant improvement in the model performance when the warm-up iterations is 2000. Finally, the number of sampling iterations is chosen as the default number, which is 2000 iterations.

```
  separate_sampling <- model_separate$sample(data = stan_data, chains=4,
                    iter_warmup = 1000, iter_sampling = 2000, refresh=0)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 2.6 seconds.
## Chain 2 finished in 2.7 seconds.
## Chain 3 finished in 2.6 seconds.
## Chain 4 finished in 2.5 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 2.6 seconds.
## Total execution time: 10.9 seconds.
```

4. Convergence diagnostics

After running the sampling, I proceeded to plot the chains to verify the model convergence for some of the coefficients from a certain country. In this case, I choose USA as this country has the most number of URLs. The sampling output above does not have any errors, suggesting there are no serious errors in the model.



- From the figure above, it appears that all MCMC chains did not diverge. However, convergence verification via visualization is not enough. This is the HMC specific convergence diagnostics of the sampling process:

```
separate_sampling$diagnostic_summary()
```

```
## $num_divergent
## [1] 0 0 0 0
##
## $num_max_treedepth
## [1] 0 0 0 0
##
## $ebfmi
## [1] 0.9505421 0.9800215 0.9598423 0.9679725
```

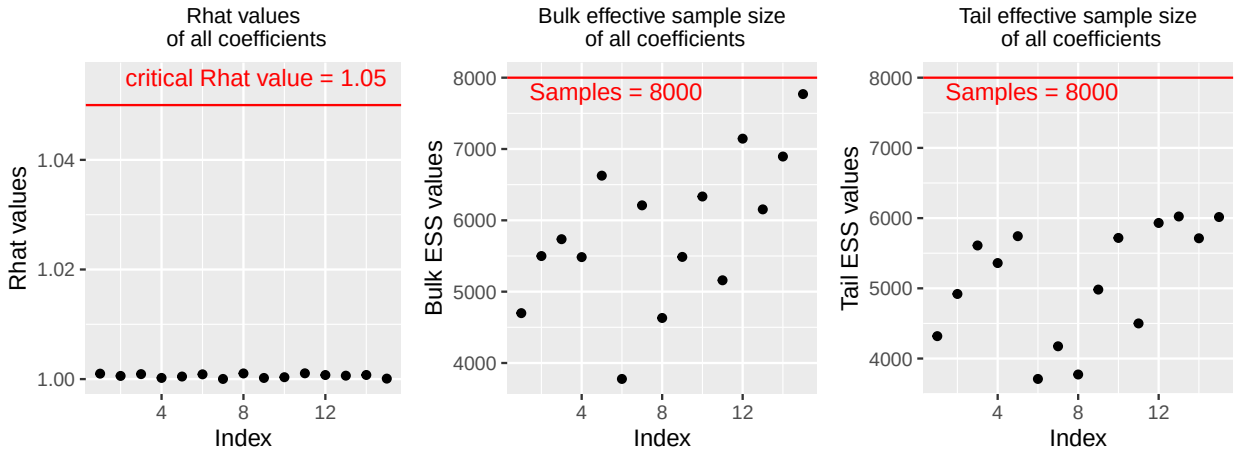
The number of divergences for all 4 chains are 0, suggesting that all of the chains have successfully converged. This means the separate model is not misconfigured or having wrong constrains on the variables.

Additionally, the number of iterations exceeding the maximum tree depth are also zero for all chains. Warnings about hitting the maximum treedepth are not as serious as other warnings. While divergent transitions, high R-hat and low ESS are a validity concern, hitting the maximum treedepth is an efficiency concern. Therefore, this separate model seems to be also efficient.

The diagnostic ebfmi stands for energy Bayesian fraction of missing information. It is particularly useful for diagnosing poorly chosen kinetic energies. Low values of ebfmi ($\leq$ 0.3) are considered problematic (Betancourt (2016)). As observed from the data, all four chains are all above 0.8, meaning that the chains have well chosen kinetic energies.

Documentation source at: https://mc-stan.org/misc/warnings.html

- Next, I analyze the $\hat{R}$-convergence diagnostics and the effective sample size ESS



$\hat{R}$-convergence diagnostics compares the between- and within-chain estimates for model parameters and other univariate quantities of interest. It is recommended to run at least four MCMC chains and only fully trust the sample if $\hat{R}$ values are less than 1.05. High $\hat{R}$ means that the chains have not mixed well and so there is not a good reason to think of them as them being fully representative of the posterior. Fortunately, from the figure above, all of the $\hat{R}$ values are very close to 1, suggesting that the samples in the separate model are representative of the posterior.

Roughly speaking, the effective sample size (ESS) of a quantity of interest captures how many independent draws contain the same amount of information as the dependent sample obtained by the MCMC algorithm. The higher the ESS the better. There are two types of ESS. The Bulk-ESS estimates the sampling efficiency for location summaries such as mean and median, while the Tail-ESS computes the minimum of the effective sample sizes (ESS) of the 5% and 95% quantiles. From the figure above, the Bulk-ESS has very high values, while Tail-ESS have moderate values. Overall, this separate model has a sampling that represents adequately effectiveness among all iterations.

Documentation source at: https://mc-stan.org/misc/warnings.html

5. Posterior predictive checks

After convergence analysis, it is time to test the model performance on the training data. First, I calculated the mean of each coefficients from the sampling. Then, I plugged them into the equation bernoulli_logit, where the multiple linear regression is calculated and its inverse logit is obtained according to the formula of the label model y_k mentioned above in part (1). Finally, if the probability is larger than 0.5, the URL is determined as malicious, while those smaller than 0.5 will be considered benign URLs.

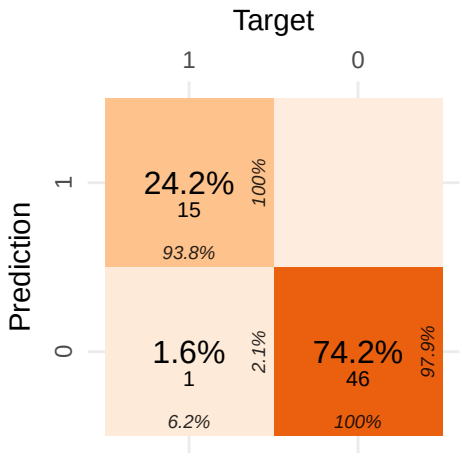
```
inv_logit <- function(vec){ return(1/(1+exp(-vec))) }

bernoulli_logit <- function (intercept, js_len_coeff, js_obf_len_coeff, https_coeff, whois_coeff,
                      js_len_truncated, js_obf_len_truncated, https_truncated, whois_truncated){
  probability <-  inv_logit(intercept + js_len_coeff * js_len_truncated +
                                js_obf_len_coeff * js_obf_len_truncated +
                                          https_coeff * https_truncated +
                                          whois_coeff * whois_truncated)
  classification <- ifelse(probability >= 0.5, 1, 0)
  return(classification)
}
```

```
##     Metrics China United States Germany All countries
## 1  Accuracy     1     0.9782609       1     0.9838710
## 2 Precision     1     0.9166667       1     0.9375000
## 3    Recall     1     1.0000000       1     1.0000000
## 4        F1     1     0.9565217       1     0.9677419
```

There are four main metrics in classification problem, which are accuracy, precision, recall and F1 score. Accuracy tells how often the model correctly classify benign and malicious URLs. On the other hand, precision is how good the model is at predicting a malicious URL, while recall tells how many times the model was able to detect a malicious URL. In this case, recall is more important than precision when the cost of blocking a malicious URL is low, but the cost of being attacked or phished by malicious URLs is seriously grave. Finally, F1 score can be interpreted as a measure of overall model performance from 0 to 1, where 1 is the best. Specifically, F1 score can be interpreted as the model's balanced ability to both capture positive cases (recall) and be accurate with the cases it does capture (precision).

From the dataframe result, it can be seen that all of the metrics are perfect for China and Germany since they have a relatively small number of training data. On the other hand, USA has a very high accuracy but a little lower precision. However, this does not matter much as the recall is perfect, since mistakenly blocking a benign URL is not as costly as allowing attacks from malicious URLs.

To visualize, this is the confusion matrix in prediction for all countries:

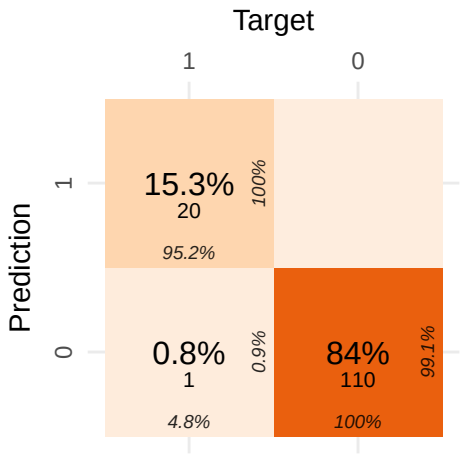


From the confusion matrix, most of the predictions lie on True Positive and False Negative, which is what expected from a classification model. There is only one wrong prediction. In summary, the separate model performs very well for all countries, especially for the perfect recall metric.

6. Predictive performance assessment

For the testing data, I apply the same procedure like part (5) above. Normally in machine learning field, the number of testing datapoints is more than the testing ones. However in this report, I experiment with my model to see if the model can generalize well over a larger testing data, which is 250 datapoints compared to only 120 training ones. This is the classification metric results

```
##     Metrics China United States Germany All countries
## 1  Accuracy     1     0.9891304       1     0.9923664
## 2 Precision     1     0.9444444       1     0.9523810
## 3    Recall     1     1.0000000       1     1.0000000
## 4        F1     1     0.9714286       1     0.9756098
```



It appears that the separate model also performs well in its prediction of malicious URLs, since all metrics are well above 0.9 and there is only one wrongly classified URL.
=> The separate model predicts malicious URL with great accuracy for both training and testing data

7. Prior sensitivity analysis

To analyze the prior sensitivity analysis in the separate model, I only change the prior for the probabilities, but keep the priors for the coefficients the same. In these new priors, all probabilities are modeled with uniform Beta distribution, which are:

$$\theta_{https_k} \sim Beta(1,1), \quad \theta_{whois_k} \sim Beta(1,1), \quad \theta_{js\_len_k} \sim Beta(1,1), \quad \theta_{js\_obf\_len_k} \sim Beta(1,1)$$

Flat uniform prior provides no information about the past data and is not recommended in Bayesian inference. If there is no need to account for the past data, the frequentist approaches such as supervised and unsupervised methods are much better than Bayesian approach.

To compare two models using LOO-CV, I need to calculate the log likelihood of the samplings.

```
loo_loglike_separate0 <- separate_sampling$loo(variables="log_likelihood",r_eff=TRUE)
loo_loglike_separate1 <- separate_sampling_prior1$loo(variables="log_likelihood",r_eff=TRUE)
```

Now I use LOO-CV method to compare the separate models with uniform and non-uniform priors:

```
loo_compare(x = list(loo_loglike_separate0=loo_loglike_separate0,
                     loo_loglike_separate1=loo_loglike_separate1))
```

```
##                        elpd_diff se_diff
## loo_loglike_separate1   0.0       0.0
## loo_loglike_separate0  -0.2       0.3
```

In LOO-CV model comparison, if the elpd difference is less than 4, the models have very similar predictive performance. If elpd difference is larger than 4, then we need to compare that difference to the standard error of elpd_diff (se_diff). When the difference (elpd_diff) is larger than 4, if the number of observations is larger than 100 and the model is not badly mis-specified, then the normal approximation and SE are quite reliable description of the uncertainty in the difference (Sivula et al. (2020)). Differences smaller than 4 are small, which means the models have very similar predictive performance. In this case, the elpd_diff is only -0.2, while the standard error difference is also 0.2. In other words, the separate model is quite insensitive to different priors because both elpd_diff and se_diff are smaller than 4.

Pooled model

1. Model description

In the pooled model, the URLs are treated as if they originate from the same source, which is the Domain Name Servers distributed around the world. This model thereby ignores the country in which the servers are established and thus, it is on the absolute opposite spectrum of separate model. In the pooled model, the probability and coefficients parameters are shared among all countries. This means all countries have the same regression model. However, aside from this difference, the pooled model is essentially following the same workflow as the separate model, which is already documented carefully above. Therefore, I will focus on reporting results of the pooled model without terminology and workflow explanations.

2. Prior choice and justifications

The priors of the probability and coefficient parameters in the pooled model are the same as the separate model, where explanations have been provided before. This time there is no longer the k index.

$$\theta_{https} \sim Beta(8,2), \quad \theta_{whois} \sim Beta(7,3), \quad \theta_{js\_len} \sim Beta(1,2), \quad \theta_{js\_obf\_len} \sim Beta(1,2)$$

$$intercept \sim Normal(0,20), \quad c_{https} \sim Normal(-1,10), \quad c_{whois} \sim Normal(-1,10)$$

$$c_{js\_len} \sim Normal(1,10), \quad c_{js\_obf\_len} \sim Normal(1,10)$$

3. Stan code and running options

The Stan code for the pooled model. Only the model code block differs. The rest code blocks are similar to the separate model

```
"
model {
    // Prior probabilities of the features
    theta_js_len ~ beta(1,2);
    theta_js_obf_len ~ beta(1,2);
    theta_https ~ beta(8,2);
    theta_whois ~ beta(7,3);
    // likelihood for the features
    for (k in 1:K){
        js_len_list[k, 1:N_list[k]] ~ bernoulli(theta_js_len);
        js_obf_len_list[k, 1:N_list[k]] ~ bernoulli(theta_js_obf_len);
        https_list[k, 1:N_list[k]] ~ bernoulli(theta_https);
        whois_list[k, 1:N_list[k]] ~ bernoulli(theta_whois);
    }
    // priors of the coefficients
    js_len_coeff ~ normal(1,10);
    js_obf_len_coeff ~ normal(1,10);
    https_coeff  ~ normal(-1,10);
    whois_coeff ~ normal(-1,10);
    intercept ~ normal(0,20);
    // Modelling of the label based on bernoulli logistic regression by
    // multiple variable linear regression
    for (k in 1:K){
      for (i in 1:N_list[k]){
        label_list[k, i] ~ bernoulli(inv_logit(intercept
```

```
            + https_coeff * https_list[k, i]
            + whois_coeff * whois_list[k, i]
            + js_len_coeff * js_len_list[k, i]
            + js_obf_len_coeff * js_obf_len_list[k, i]));
      }
    }
}
"
```

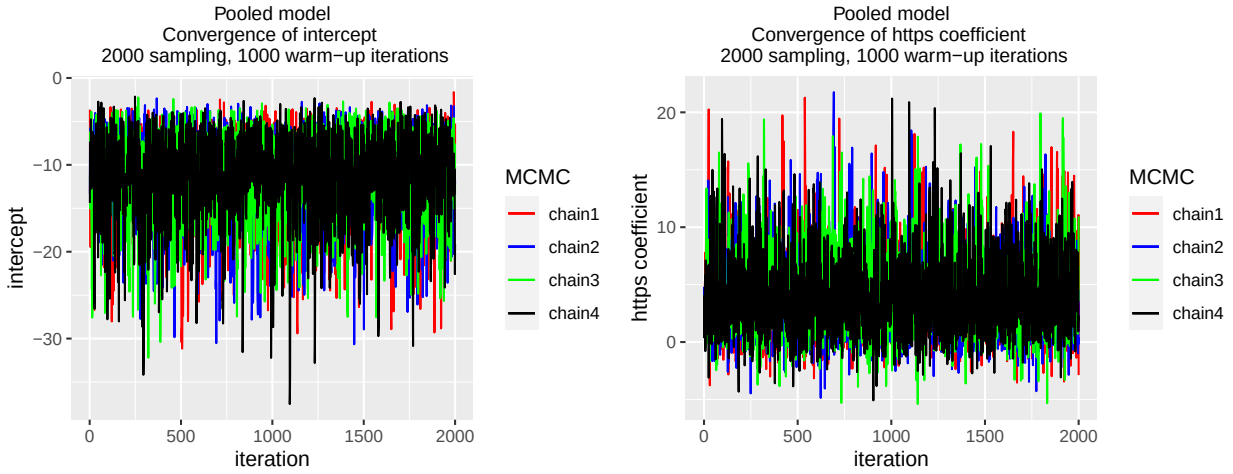
- This is the sampling running options, which is the same as the separate model.

```
pooled_sampling <- model_pooled$sample(data = stan_data, chains=4,
                    iter_warmup = 1000, iter_sampling = 2000, refresh=0)
```

```
## Running MCMC with 4 sequential chains...
##
## Chain 1 finished in 1.8 seconds.
## Chain 2 finished in 1.8 seconds.
## Chain 3 finished in 1.7 seconds.
## Chain 4 finished in 1.8 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 1.8 seconds.
## Total execution time: 7.5 seconds.
```

4. Convergence diagnostics

The sampling output above does not have any errors, suggesting there are no serious errors in the model. I proceeded to plot the chains to verify the model convergence for the pooled model



- It appears that all MCMC chains converge. Examining the HMC specific convergence diagnostics:

```
pooled_sampling$diagnostic_summary()
```

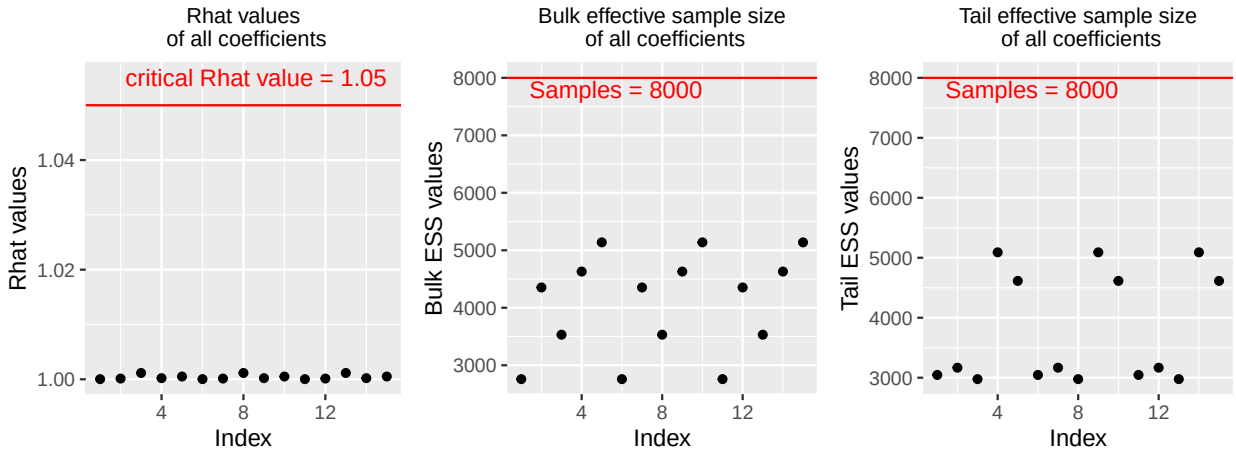
```
## $num_divergent
## [1] 0 0 0 0
##
## $num_max_treedepth
## [1] 0 0 0 0
##
## $ebfmi
## [1] 1.0075201 1.0500232 0.9708223 0.9780740
```

The number of divergences for all 4 chains are 0, suggesting that all of the chains have successfully converged. This means the pooled model is not misconfigured or having wrong constrains on the variables.

Additionally, the number of iterations exceeding the maximum tree depth are also zero for all chains. Therefore, this pooled model seems to be also efficient.

For ebfmi, all four chains are all above 0.8, meaning that the chains have well chosen kinetic energies.

- Next, I analyze the $\hat{R}$-convergence diagnostics and the effective sample size ESS



From the figure above, all of the $\hat{R}$ values are very close to 1, suggesting that the samples in the pooled model are representative of the posterior. However, the Bulk and Tail-ESS do not have very high values of effective sampling, suggesting that the pooled model needs some improvements, such as prior specification or dimension reduction. Furthermore, the scatterplots in the figure are cyclic since all countries share the same coefficients and diagnostics, even including $\hat{R}$ and ESS values.

5. Posterior predictive checks

The process of conducting predictive checks is the same as the separate model.

```
##     Metrics China United States Germany All countries
## 1  Accuracy     1     0.9782609       1     0.9838710
## 2 Precision     1     0.9166667       1     0.9375000
## 3    Recall     1     1.0000000       1     1.0000000
## 4        F1     1     0.9565217       1     0.9677419
```

Like the separate model, the pooled model performs very well for all countries.

6. Predictive performance assessment

For the testing data, I apply the same procedure like the separate model.

```
##     Metrics China United States Germany All countries
## 1  Accuracy     1     0.9891304       1     0.9923664
## 2 Precision     1     0.9444444       1     0.9523810
## 3    Recall     1     1.0000000       1     1.0000000
## 4        F1     1     0.9714286       1     0.9756098
```

It appears that the pooled model performs well in its prediction of malicious URLs, since all metrics are well above 0.9 => The pooled model predicts malicious URL like the separate model.

7. Prior sensitivity analysis

Like before, I choose a uniform prior to analyze the prior sensitivity analysis in the pooled model.

$$\theta_{https} \sim Beta(1,1), \quad \theta_{whois} \sim Beta(1,1), \quad \theta_{js\_len} \sim Beta(1,1), \quad \theta_{js\_obf\_len} \sim Beta(1,1)$$

Now I use LOO-CV method to compare the pooled models with uniform and non-uniform priors:

```
loo_compare(x = list(loo_loglike_pooled0=loo_loglike_pooled0,
                     loo_loglike_pooled1=loo_loglike_pooled1))
```

```
##                      elpd_diff se_diff
## loo_loglike_pooled0 0.0       0.0
## loo_loglike_pooled1 0.0       0.1
```

From LOO-CV model comparison, the elpd_diff is only -0.3, while the standard error difference is also 0.3. In other words, the pooled model is quite insensitive to different priors because both elpd_diff and se_diff are smaller than 4. However, it is still more sensitive than the separate model (-0.2 and 0.2)

Model comparison

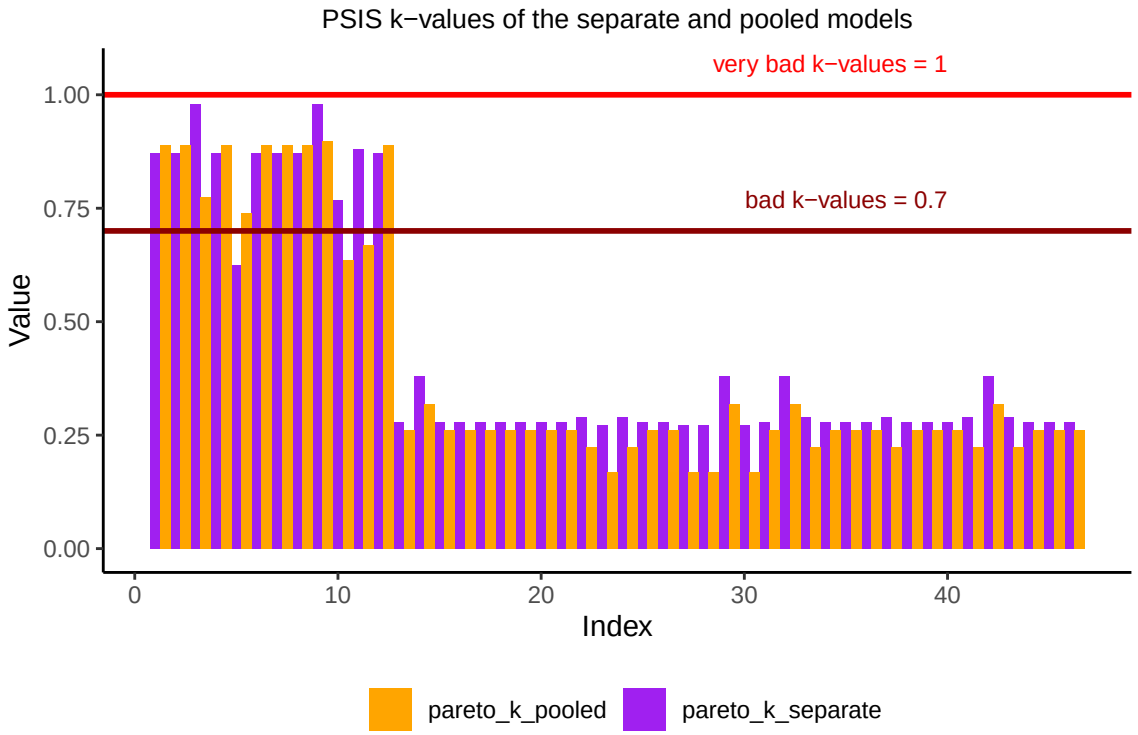
In order to compare the separate and pooled models, I need to LOO-CV diagnostics as well as the PSIS $\hat{k}$-values. These are the diagnostics of both models.
Separate model:

```
##           Estimate       SE
## elpd_loo -5.892196 4.041931
## p_loo     2.932652 2.286239
## looic    11.784393 8.083862
```

Pooled model:

```
##           Estimate       SE
## elpd_loo -5.182756 3.779026
## p_loo     2.274951 1.837612
## looic    10.365512 7.558052
```

Next, I plot the $\hat{k}$-values for both models



The Pareto k estimate is a diagnostic for Pareto smoothed importance sampling (PSIS), which is used to compute components of elpd_loo. If $k < 0.5$, then the corresponding component of elpd_loo is estimated with high accuracy. If $0.5 < k < 0.7$ the accuracy is lower, but it is still acceptable. If k > 0.7, then importance sampling is not able to provide useful estimate for that observation. From the figure, there are multiple k values larger than 0.7, suggesting that both separate and pooled models are overly optimistic. Therefore, I need to examine the p_loo values to verify whether this is a sign of a weak model. If k > 0.7 then we have the following interpretations:

If p_loo « p (the number of parameters), then the model is likely to be misspecified.
If p_loo $<$ p/5, it is likely that the model is so flexible or the population prior is very weak.
If p_loo $>$ p, then the model is likely to be badly misspecified.

In the separate model, p_loo is 2.597943 and 2.357919 in the pooled model. The number of parameters in my models are 4 probability parameters + 5 coefficients = 9. Since p_loo of both models are smaller than $9/5 = 1.8$, this does not fall into any of the bad cases mentioned previously. As a result, while both separate and pooled models are optimistic, the p_loo value interpretation otherwise suggests that these two models are still acceptable for future prediction of malicious URLs.

Finally, I compare two models with LOO-CV. It appears that the separate and pooled models are not so different from each other, as both of them have equally high classification metrics.

```
loo_compare <- loo_compare(x = list(loo_loglik_separate=loo_loglik_separate,
                                    loo_loglik_pooled=loo_loglik_pooled))
print(loo_compare)
```

```
##                       elpd_diff se_diff
## loo_loglik_pooled      0.0       0.0
## loo_loglik_separate -0.7       0.3
```

Discussion

Existing problems
- Some PSIS k-values are higher than 0.7 for both models.
- The ESS values are a little bit low for the pooled model.
Potential improvements
- I can try to implement the hierarchical model, which incorporates both elements of the separate and the pooled model. I believe the hierarchical model may deliver better results.
- Feature reduction: due to the high number of parameters (9), there is too much uncertainty. A solution is many binary features can be combined and modeled with the categorical distribution in Stan model.

Conclusion

This report has documented two Bayesian model: the separate and the pooled models, which are charged with the task of malicious URLs classification. The separate model classify the URLs based on individual countries, while the pooled model treats all URLs as if they come from a single source. While both models are not entirely reliable based on the PSIS-LOO diagnostics, they are nevertheless proven to deliver high accuracy, as witnessed from the classification metrics on the training and testing data.

In conclusion, the Bayesian approach in this report provides a systematic analysis for detecting malicious URLs on the internet in the cybersecurity field. Unlike traditional methods such as supervised learning, the Bayesian methods allow us to incorporate prior knowledge on the distribution of parameters, account for model uncertainty and update new beliefs as additional information becomes available.

Reflection

By doing this project, I have a chance to combine all of my previous learned knowledge in BDA into building a systematic study and report for an existing problem in cybersecurity. The most laborious step, in my opinion, is choosing the correct dataset and conducting data cleaning and feature selection.

The next challenge for me is writing the Stan model. During the course, I was only used to writing Stan models for continuous variables and regression for real-valued labels. However in this project, I have to deal with mostly binary/discrete variables and classification, which I have not done before in this course.

References

Betancourt, Michael. 2016. "Identifying the Optimal Integration Time in Hamiltonian Monte Carlo." arXiv. https://doi.org/10.48550/ARXIV.1601.00225.

Chen, Chia-Mei, D. J. Guan, and Qun-Kai Su. 2014. "Feature Set Identification for Detecting Suspicious URLs Using Bayesian Classification in Social Networks." *Information Sciences* 289: 133–47. https://doi.org/https://doi.org/10.1016/j.ins.2014.07.030.

Morgan, Steve. 2019. 2019. https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-20-billion-usd-by-2021/.

Singh, A K, and Navneet Goyal. 2019. "A Comparison of Machine Learning Attributes for Detecting Malicious Websites." In *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, 352–58. https://doi.org/10.1109/COMSNETS.2019.8711133.

Sivula, Tuomas, Måns Magnusson, Asael Alonzo Matamoros, and Aki Vehtari. 2020. "Uncertainty in Bayesian Leave-One-Out Cross-Validation Based Model Comparison." https://doi.org/10.48550/ARXIV.2008.10296.

W3Techs. 2023. "Usage Statistics of Default Protocol Https for Websites." 2023. https://w3techs.com/technologies/details/ce-httpsdefault.