# CYO Capstone

David Delfuoco

2024-10-24

## Contents

## 1 Introduction

This is the Choose Your Own Capstone report for the Harvard Data Science course on the edX platform. I chose to do this on a binary dataset of legitimate and phising URLs from the kaggle website[1], I chose this mostly because I wanted to try a binary dataset with some real life practicality. I believe it falls into this real life practicality because the dataset is not well documented, so it would be up to me to interpret the columns and do some data cleaning. It also doesn't seem that popular which was one of the prerequisites for choosing the dataset.

The goal is to develop a machine learning algorithm to catch whether or not an URL is phished based on it's whois data and characteristics of the URL itself. I believe these kind of detection systems are important to help people that aren't technological adept or careless when clicking links, and help them find out if the site they are going to is a dangerous one that could potentially get sensitive information that could do harm. In this day and age the amount of harm it can inflict can vary greatly from identity theft to exposing business secrets.

In this particular dataset an URL is labeled legitimate with a 1 and phished with a 0. For this reason I will be using the true negative rate(TNR) as the metric to judge how well my model is doing. This is because I believe catching a phished URL is more important than catching a legitimate URL. My reasoning is that at worst case the user gets a warning which will make them double check they clicked the right link or put in the correct URL, a minor inconvenience compared to losing money or your identity.

## 2 Dataset Exploration

Let's now explore the data, from the source site the only information given is that it contains data on 11,001 website URLs with 15 parameters, one of them being a label parameter that is 0 when an URL is phished and 1 when an URL is legitimate[1].

I named the original dataset phish, just to make sure you know what phish refers to in the following code. The quickest way to get as much as information about an unknown is using the str() function, let's take a look:

```
str(phish)
```

```
## 'data.frame':    11000 obs. of  15 variables:
##  $ whois_regDate    : int  8451 5741 8419 7695 9316 8158 8621 6407 6442 9422 ...
##  $ whois_expDate    : int  2870 102 345 2166 2455 241 2702 167 497 805 ...
##  $ whois_updatedDate: int  422 295 720 545 6 100 239 685 -1 322 ...
##  $ dot_count        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ url_len          : int  10 11 9 9 6 8 12 10 6 9 ...
##  $ digit_count      : int  0 0 0 0 0 0 0 0 3 0 ...
##  $ special_count    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hyphen_count     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ double_slash     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ single_slash     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ at_the_rate      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ protocol         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ protocol_count   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ web_traffic      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ label            : int  1 1 1 1 1 1 1 1 1 1 ...
```

It looks like there are actually 11,000 observations and 15 parameters. Majority of the column names look pretty straightforward. Some of them look like they can be up for interpretation, mostly the protocols, at_the_rate, and web_traffic column seem odd to me. The most troubling to me though is the first 3 columns referring to dates. There doesn't seem to be a clear pattern to me of how these dates are represented, and the -1 entry raises more questions. The data types are all integers which is a nice bonus but before we move on we need to answer some of the questions raised.

Before we try to answer the questions raised let's make sure there are no na values, even though its conveniently all integers, there could still be lurking na values deep in the data. Let's do a quick check for that:

```
any(is.na(phish))
```

```
## [1] FALSE
```

There are no official na values which is good, let's now try to feel out whats going on with those parameters that don't seem very clear. Since they are all integers I'm going to get the unique count and range of each column to see if things start making more sense.

```r
# Get unique count and range of columns
feelers <- lapply((1:ncol(phish)), function(n){
  uniq_count <- length(unique(phish[,n]))
  range <- range(phish[,n])
  return(c(uniq_count = uniq_count, range_min = range[1], range_max = range[2]))
})

# Get Column Names
Columns <- colnames(phish)

# Assign phish column names to feelers
names(feelers) <- Columns

# Check out the columns unsure about
feelers$at_the_rate[1:3]
```

```
## uniq_count  range_min  range_max
##          4          0          3
```

```r
feelers$protocol[1:3]
```

```
## uniq_count  range_min  range_max
##          2          0          1
```

```r
feelers$protocol_count[1:3]
```

```
## uniq_count  range_min  range_max
##          6          0          6
```

```r
feelers$web_traffic[1:3]
```

```
## uniq_count  range_min  range_max
##          2          0          1
```

So protocol_count seems to be an actual count of protocols, with there being up to 6 different protocols. I thought at_the_rate might refer to some rate, but the low max seems to suggest it's just a count of the @ symbol in the URL. On the other hand protocol and web_traffic column seem to be binary columns. Exactly what they mean is up for interpretation, but my guess is whether the URL starts with a protocol or not like http://, and the web_traffic is whether or not the URL gets a certain threshold of web traffic. It doesn't seem possible to know what that threshold is but at least we got a general ideal of its representation.

Now on to the dates:

```r
feelers$whois_regDate[1:3]
```

```
## uniq_count  range_min  range_max
##       4038         -1      12915
```

```
feelers$whois_expDate[1:3]
```

```
## uniq_count  range_min  range_max
##       1872       -327      31868
```

```
feelers$whois_updatedDate[1:3]
```

```
## uniq_count  range_min  range_max
##       1146         -1       3685
```

It seems each date column has some negative values, and expiration date goes all the way to -327. I'm going to check these -1s first since it seems so prevalent. Let's count these -1 values in the date columns.

```
sum(phish$whois_regDate == -1)
```

```
## [1] 2285
```

```
sum(phish$whois_expDate == -1)
```

```
## [1] 2298
```

```
sum(phish$whois_updatedDate == -1)
```

```
## [1] 2913
```

This is a surprisingly large amount, accounting for almost a 3rd of the data. Let's quickly see how many of these columns have all -1 values.

```
neg_ones <- phish %>% filter(whois_regDate == -1 & whois_expDate == -1 & whois_updatedDate) %>%
  mutate(count = n())

neg_ones$count[1]
```

```
## [1] 2252
```

This shows us that most of the time when one of the date columns are -1 the other two columns are also -1. This seems to point to the fact that -1 is representing na values. Let's check if the rest of the parameters are okay on these full -1 date columns.

```
head(neg_ones)
```

```
##   whois_regDate whois_expDate whois_updatedDate dot_count url_len digit_count
## 1            -1            -1                -1         2      11           0
## 2            -1            -1                -1         2      12           0
## 3            -1            -1                -1         2      12           0
## 4            -1            -1                -1         1       9           0
## 5            -1            -1                -1         2      13           0
## 6            -1            -1                 -1         1       9           0
##   special_count hyphen_count double_slash single_slash at_the_rate protocol
## 1             0            0            0            0           0        0
## 2             0            0            0            0           0        0
## 3             0            0            0            0           0        0
## 4             0            0            0            0           0        0
## 5             0            0            0            0           0        0
## 6             0            0            0            0           0        0
##   protocol_count web_traffic label count
## 1              0           0     1  2252
## 2              0           0     1  2252
## 3              0           0     1  2252
## 4              0           0     1  2252
## 5              0           0     1  2252
## 6              0           0     1  2252
```

The varying url_len numbers seem to indicate that the rest of the parameters are fine.
I'm going to remove the negative value since they seem to be nas or errors anyways.

```
phish_clean <- phish %>%
  filter(whois_regDate > -1 & whois_expDate > -1 & whois_updatedDate > -1)
```

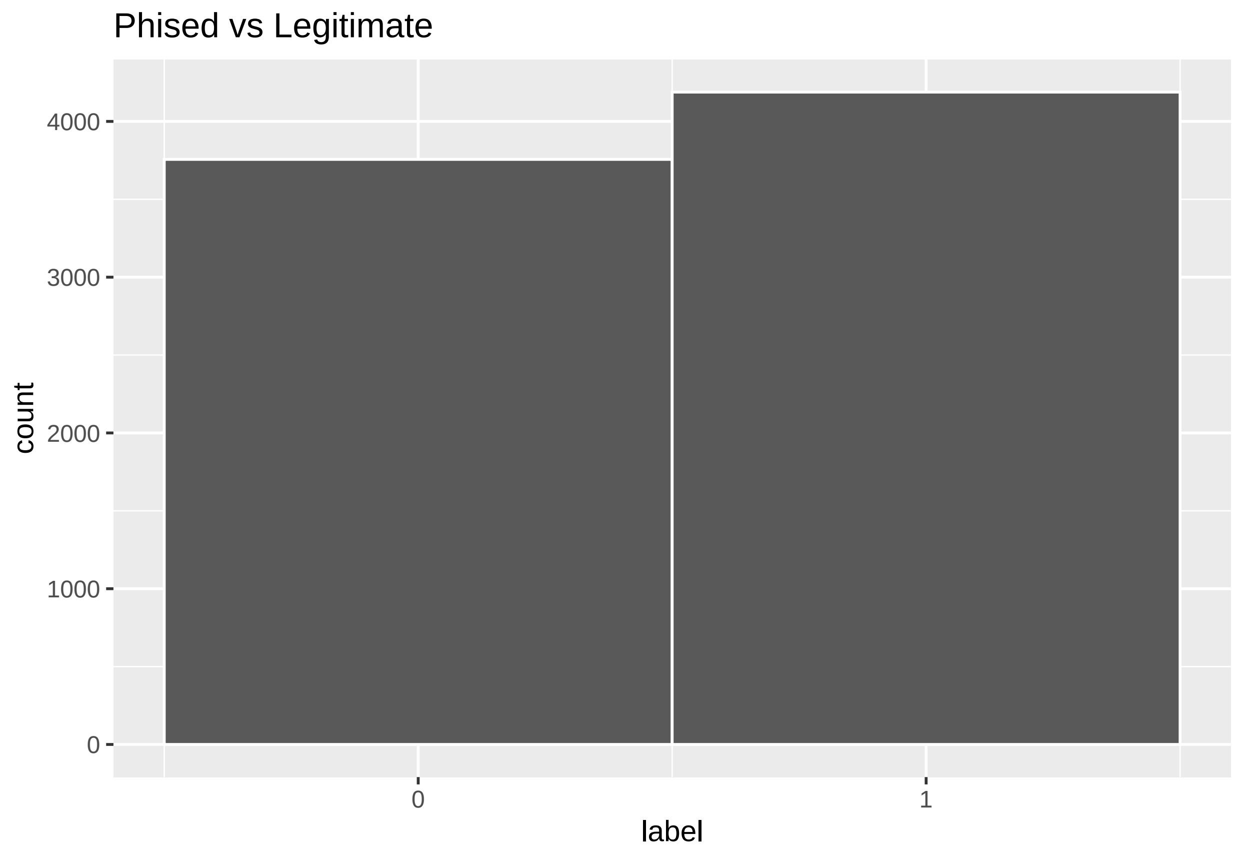
Now let's check the non-negative values to see if we can make sense of them. From the ranges it doesn't seem like its in POSIX(seconds from epoch) format, and the there doesn't seem to be a pattern of some kind of particular combination of year, day, month. My last guess would be a day count, hopefully its the epoch. A quick google search shows the whois was created in 1970s and established in the 1980s[2]. Let's check the minimum registration date that's not negative and see if it lines up with either of these years.
This shows it most likely not days from the epoch but an arbitrary date, with out knowing the URLs or the correct date this doesn't seem like it will be possible to figure out this date.

Let's finally give a table of the columns and their definitions:

| Columns | Definitions |
|---|---|
| whois_regDate | Registration Date |
| whois_expDate | Expiration Date |
| whois_updatedDate | Last Updated Date |
| dot_count | Count of dots |
| url_len | Length of URL |
| digit_count | Count of digits |
| special_count | Count of special characters |
| hyphen_count | Count of hyphens |
| double_slash | Count of double slashes |
| single_slash | Count of single slashes |
| at_the_rate | Count of @ symbol |
| protocol | Begins with a protocol |
| protocol_count | Count of protocols |
| web_traffic | A threshold of web traffic |
| label | If it is a legitimate URL |

## 3 Visual Exploration

Since this is a binary dataset, and I care about the specificity, let's check the ratio of the label column:



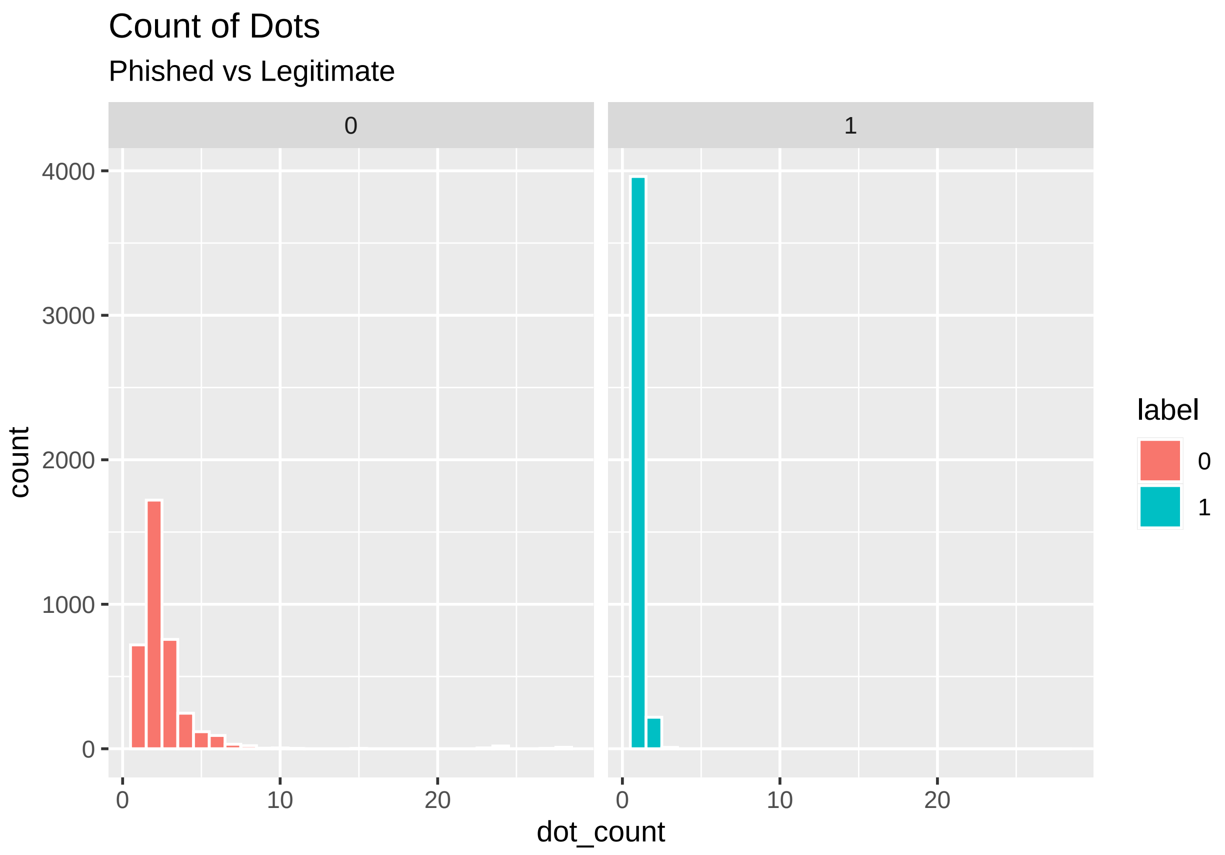
Well that looks close, let's check how close:

```r
sum(phish_clean$label == 0) / length(phish_clean$label)
```

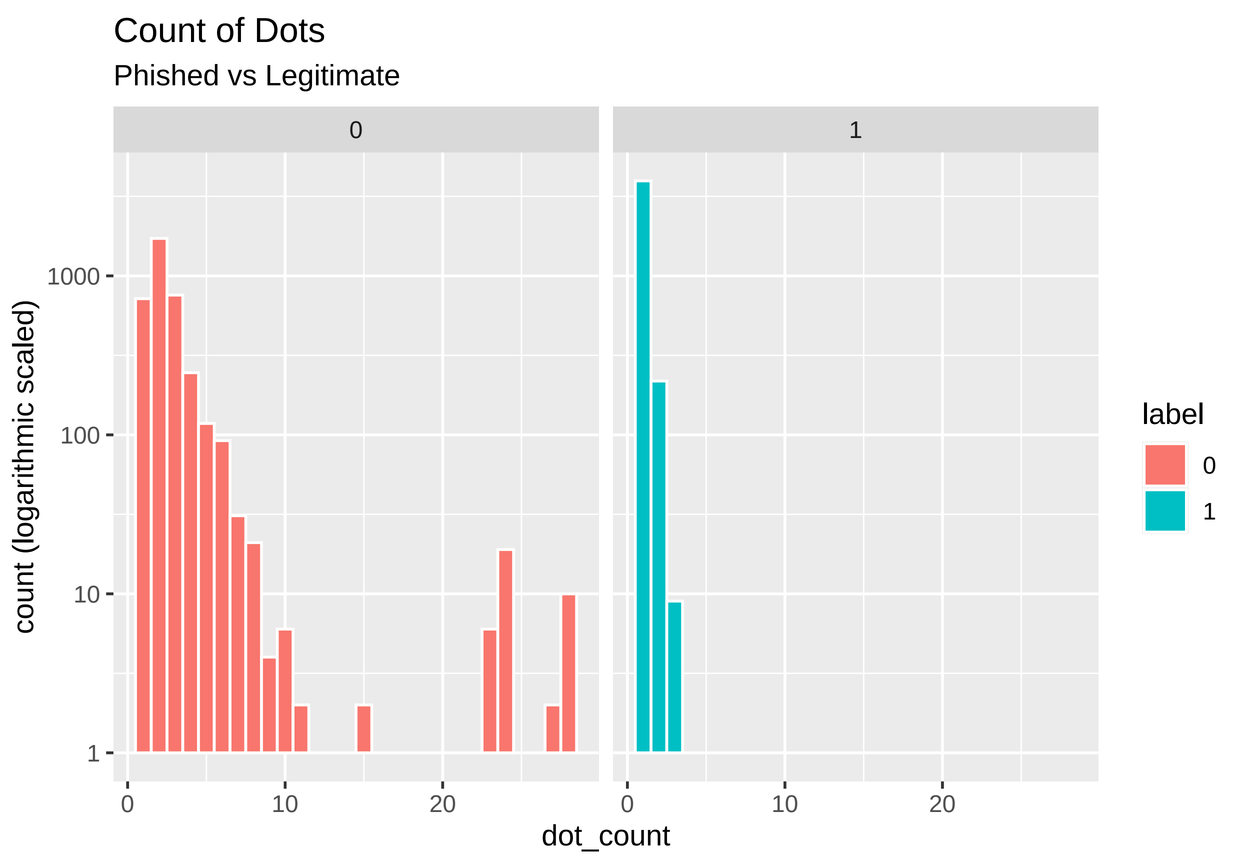
```
## [1] 0.4728097
```

Looks like the data is a very closely split of legitimate and phished URLs. This means I don't have to adjust the weights of the label to make them balanced for the base models. I most likely will try different weights though since I want to prioritize catching phished URLs.

Let's start exploring how much each parameter is present in each of the label column:

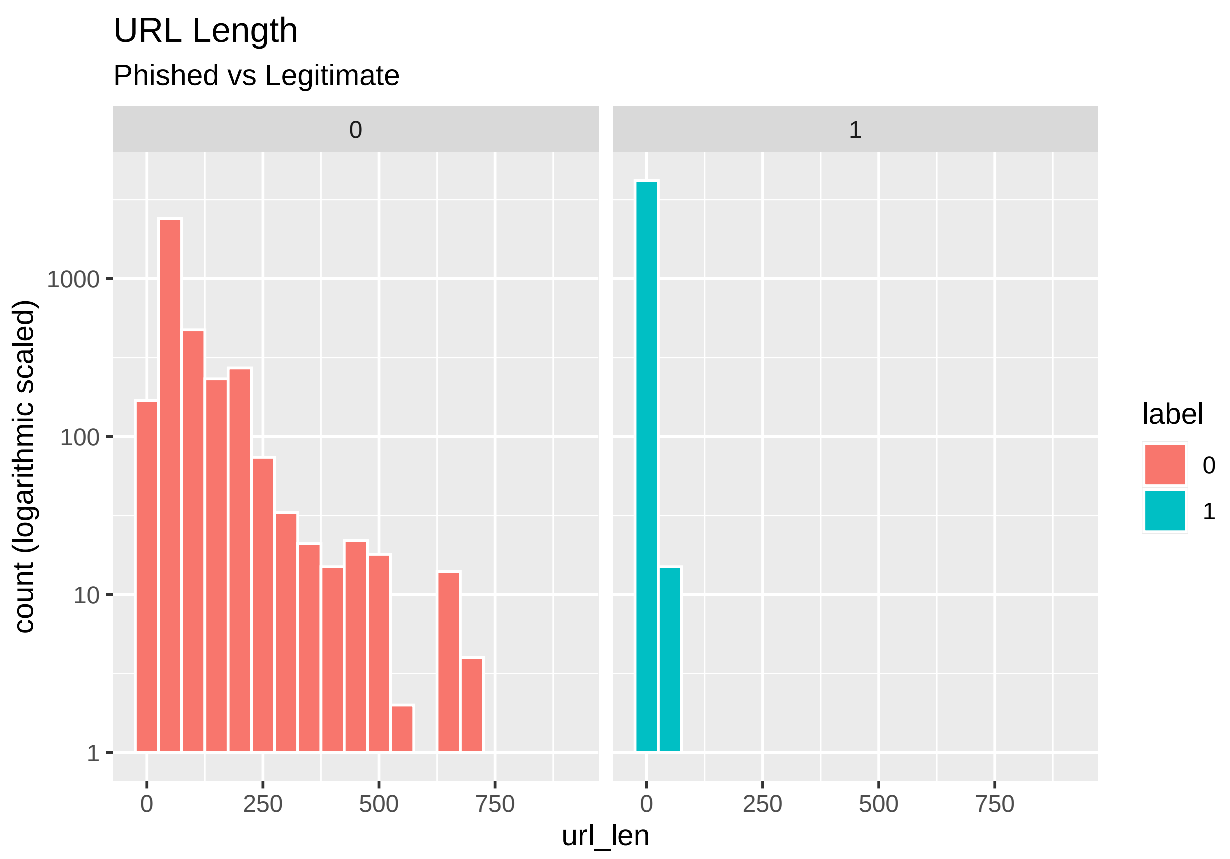


Looks like having 0 dots makes it far more likely that it will be a legitimate site, since it's so heavy and effecting the other values lets take another look while scaling the y axis.



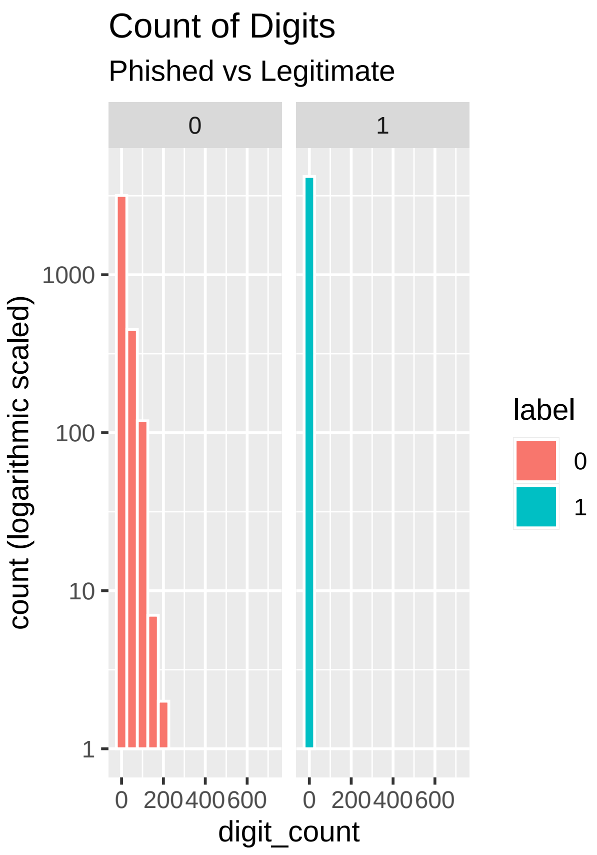
With this better view it pretty much shows any dot count greater than 3 should be a phished URL.

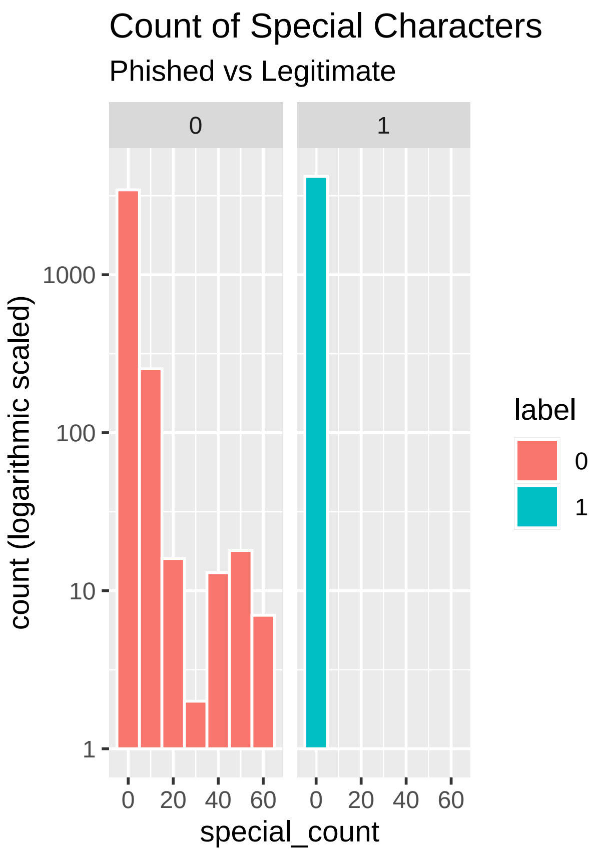
Now for the url_length:



A similar scenario where after a certain threshold its most likely a phished URL, the threshold looks to be around 100.

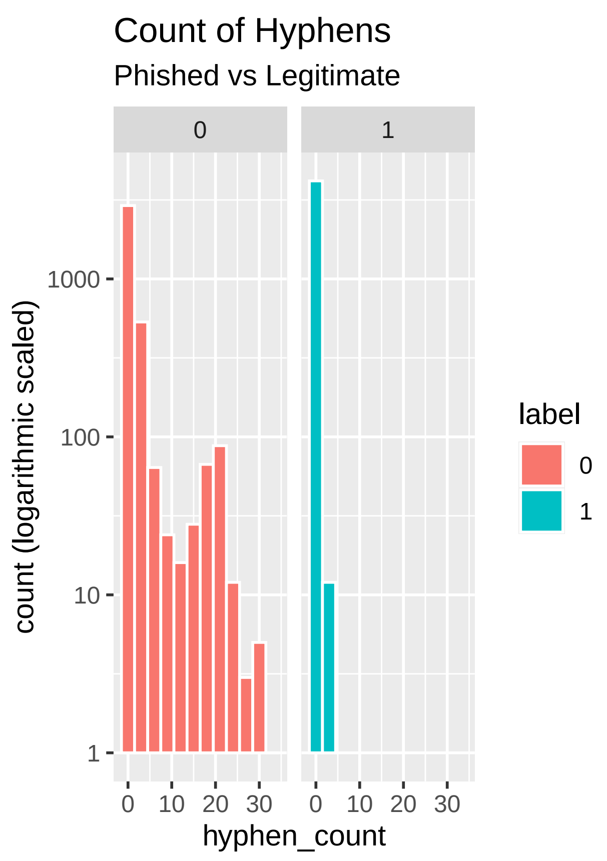
digit_count and special_count:



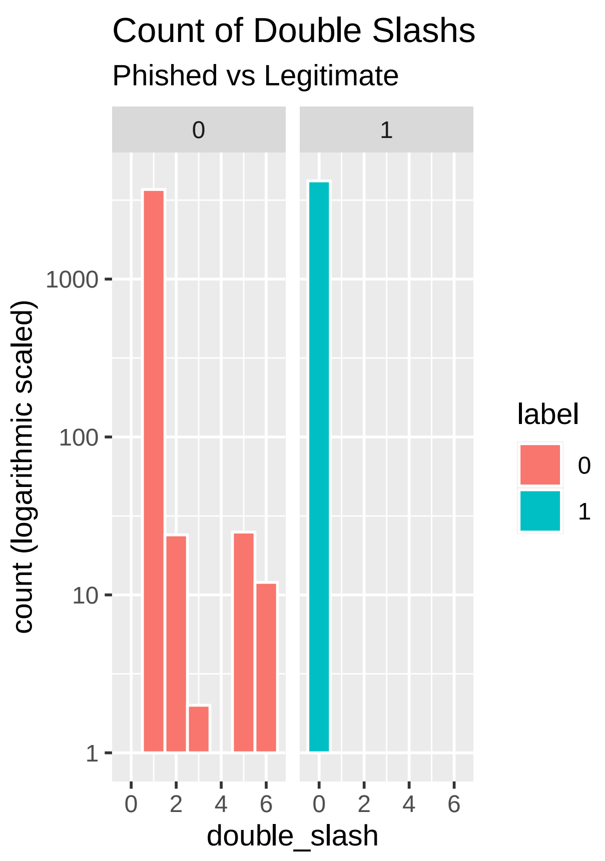


The trends continues of having any extreme values makes it most likely a phished URL.

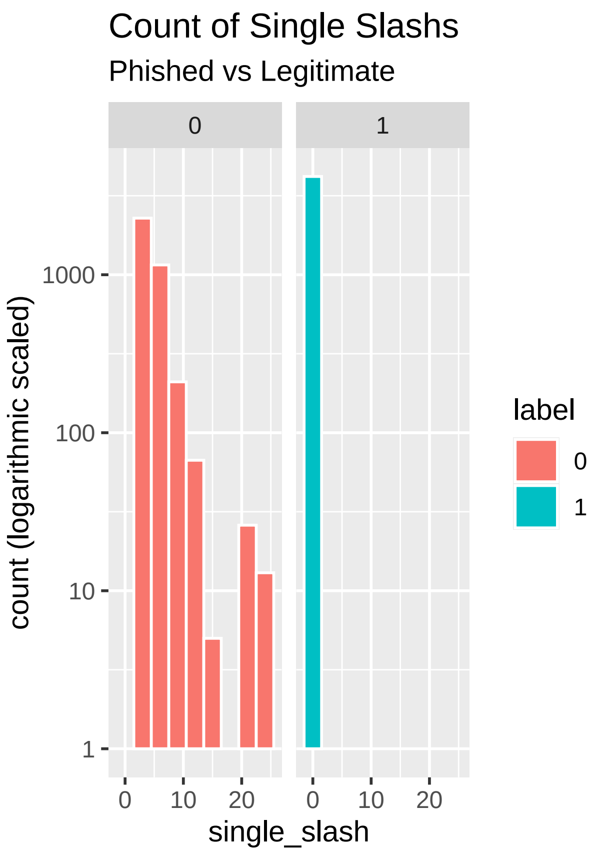
Let's do the rest of the count columns as this trend most likely will continue.

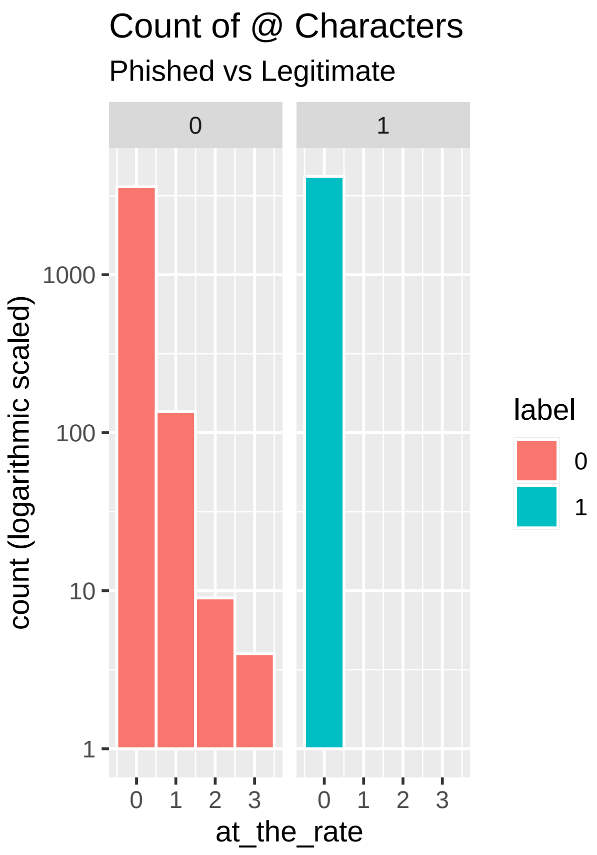


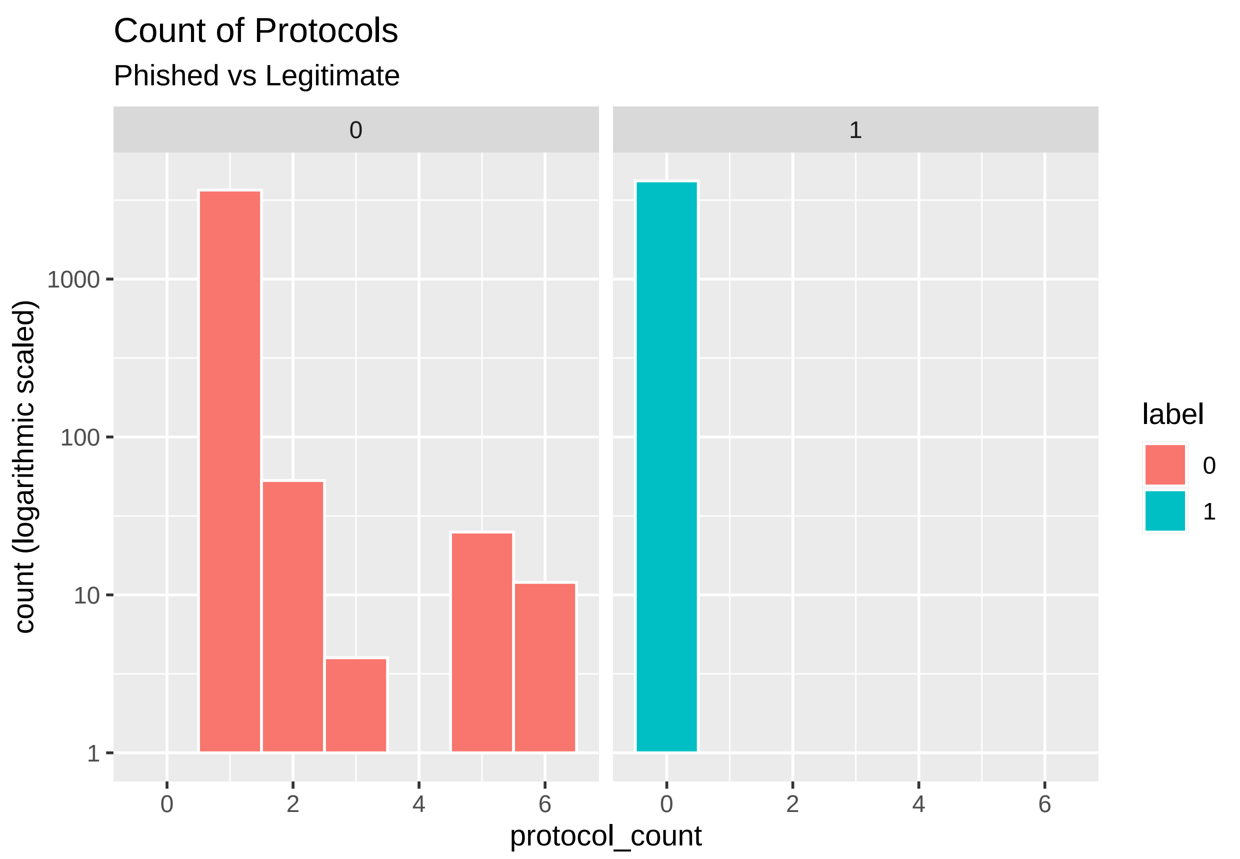
Count of Hyphens
Phished vs Legitimate
0
1
count (logarithmic scaled)
1000
100
10
1
0 10 20 30
0 10 20 30
hyphen_count
label
0
1



Count of Double Slashs
Phished vs Legitimate
0
1
count (logarithmic scaled)
1000
100
10
1
0 2 4 6
0 2 4 6
double_slash
label
0
1

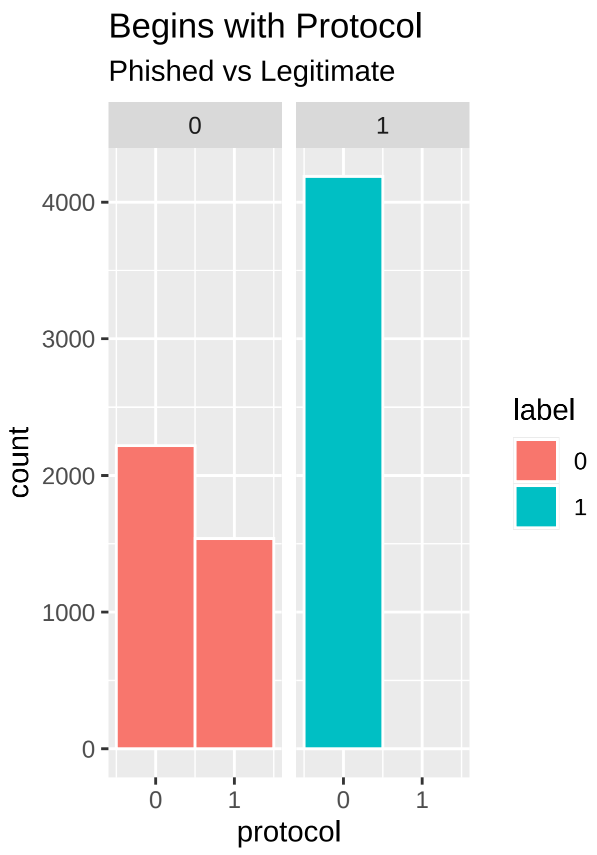

Count of Single Slashs
Phished vs Legitimate
0
1
count (logarithmic scaled)
1000
100
10
1
0
10
20
0
10
20
single_slash
label
0
1

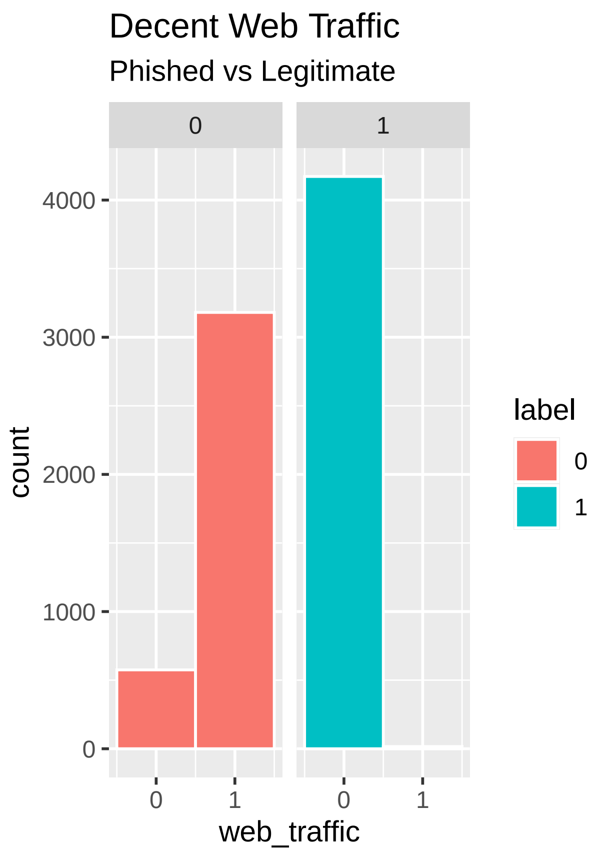

Count of @ Characters
Phished vs Legitimate
0
1
count (logarithmic scaled)
1000
100
10
1
0
1
2
3
0
1
2
3
at_the_rate
label
0
1



The trend does indeed continue with having over a specific threshold will automatically put you in phished URL category.

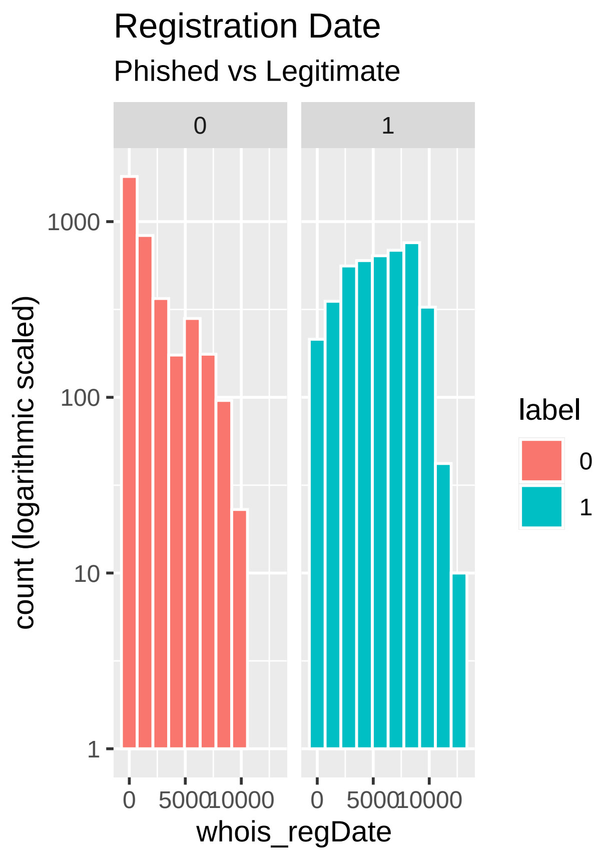
Let's move on to the binary predictors.

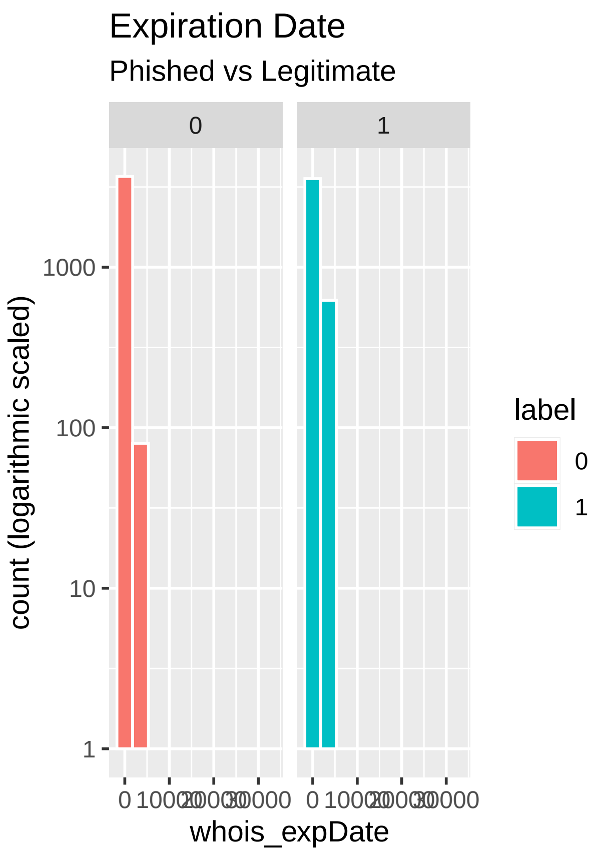


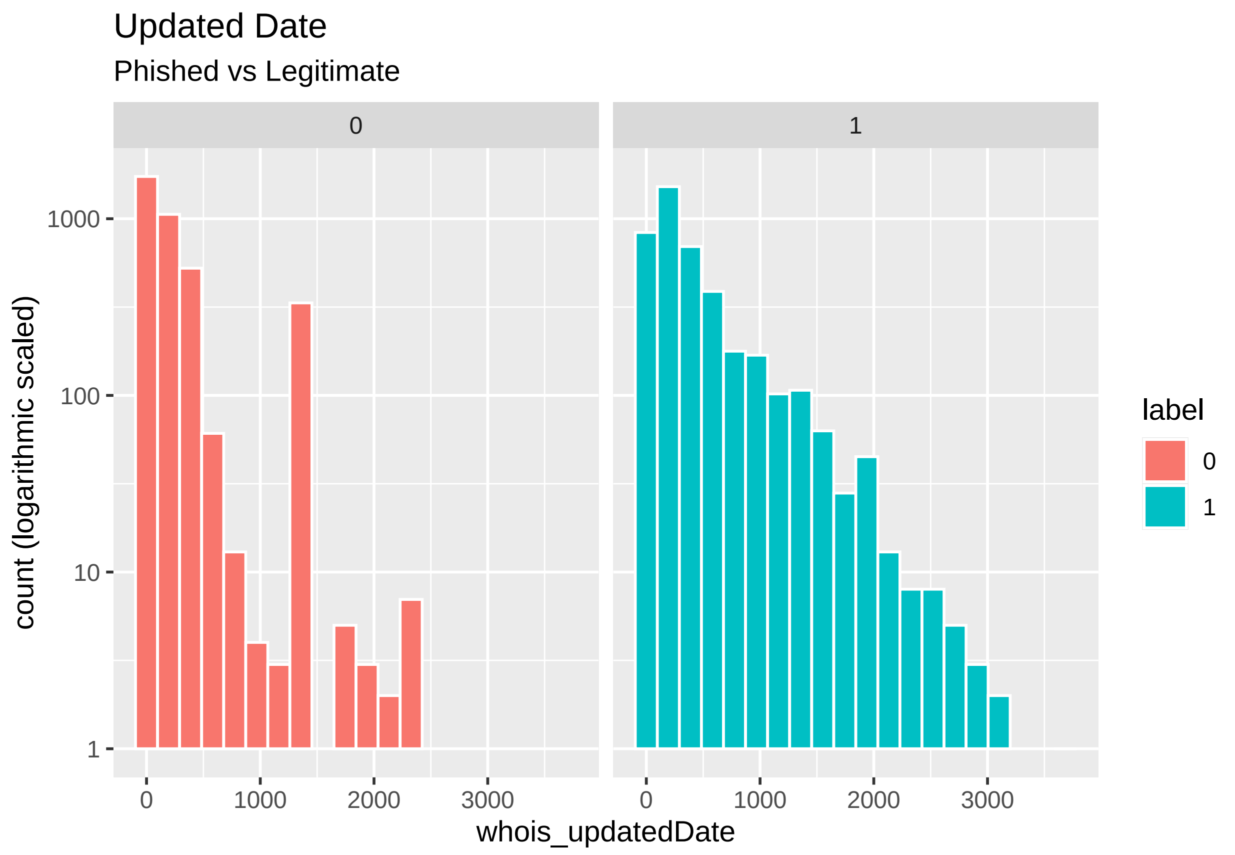


These are interesting because they make me second guess my earlier definitions, perhaps web traffic column is binary of whether it has suspicious web traffic, and I'll admit my lack of knowledge on internet protocols, but I would of guessed the the protocol graph would be the opposite. Regardless without more information I'm still just guessing, and the new information I obtained is that having a protocol and having web traffic make it most likely a phished URL.

Let's do the same with the odd date columns:

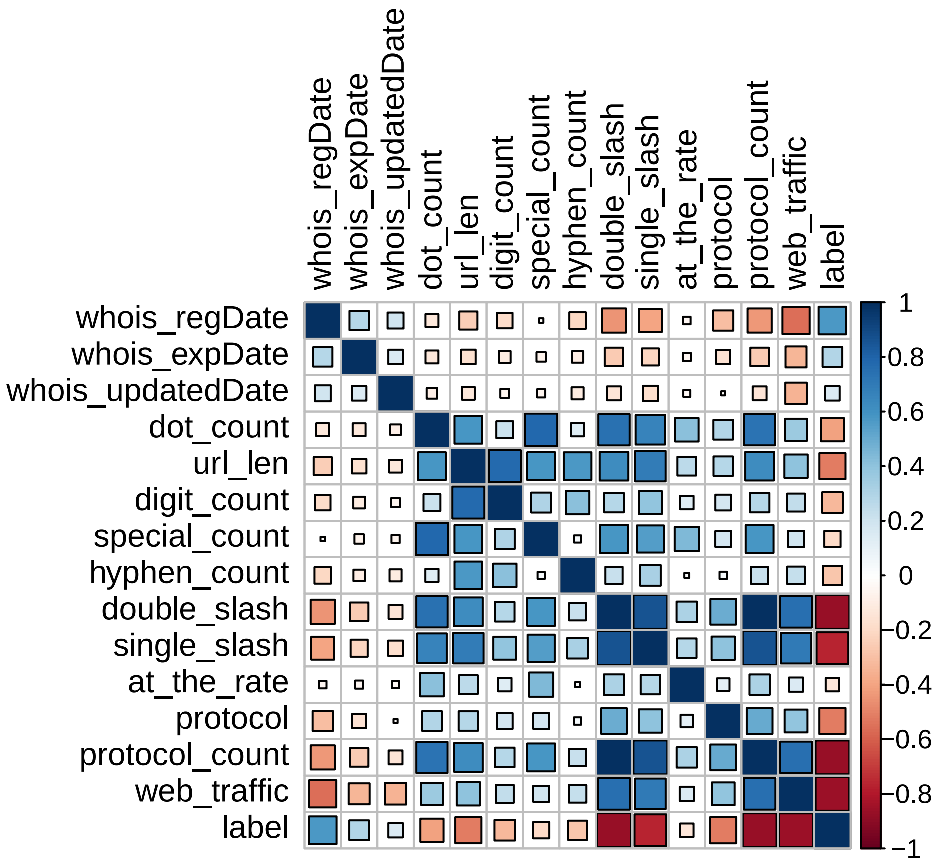

Registration Date
Phished vs Legitimate
0
1
1000
100
10
1
count (logarithmic scaled)
0
5000
10000
whois_regDate
label
0
1


Expiration Date
Phished vs Legitimate
0
1
1000
100
10
1
count (logarithmic scaled)
0
10000
20000
30000
whois_expDate
label
0
1



These seem pretty close, with the best predictor being updated date.

Let's move onto the correlation to see if we should remove any predictors if they are too correlated.



It looks like a lot of the predictors are highly correlated with each other, to account for this we need to use PCA or drop one of the highly correlated pairs and keeps the one that is more correlated with the label. I'll leave it for now and leave it as a step to improve the base model.

## 4 Constructing The Train and Test set

Creating a separate train and test set even when you only have one dataset to work with is good practice because you treat your test set like outside data, and never use your test set to help train the data. This helps avoid overtraining and gives your model a better chance to perform well on outside data. I will create a train and test with a 80% to 20% split.

```
set.seed(5)
phish_test_index <- createDataPartition(y = phish_clean$label, times = 1, p = .2, list = F)

phish_test <- phish_clean[phish_test_index,]
phish_train <- phish_clean[-phish_test_index,]

propor_train <- sum(phish_train$label == 0) / length(phish_train$label)
propor_test <- sum(phish_test$label == 0) / length(phish_test$label)
```

After cleaning and splitting the data, the proportion of negative label in the train set are 0.4700236 and the in the test set is 0.4839522. These are still very balanced, so no need to change the weights for our base models.

Lastly before making the base models, we need to make the label a factor instead of an integer.

```
phish_train <- phish_train %>% mutate(label = factor(label))
phish_test <- phish_test %>% mutate(label = factor(label))
```

## 5 Constructing the Base Models

From the earlier histograms I believe the best models would be support vector machines(SVM) and decision trees(DT). The reason for picking them are the same, the data seems very split between the parameters to make a clear cut choice for one of the binary outcomes. Decision trees very nature of finding the split threshold and going down the path that is most likely in the right direction would make it a good fit. For SVM from what I understand of its definition is that it attempts to find the plane that separates the data at the maximum margin making it pick the best threshold. I was convinced from this blog[3] that this is the right direction to go in.

The base models will be the default settings of the train function from the caret package using the methods that build the model with the DT and SVM. Based on the outcome we can fine tune the settings to help it.

```
svm_train <- train(label ~ ., data = phish_train, method = "svmLinear")
svm_predict <- predict(svm_train, newdata = phish_test)

dt_train <- train(label ~ ., data = phish_train, method = "rpart")
dt_predict <- predict(svm_train, newdata = phish_test)

svm_confu_matr <- confusionMatrix(svm_predict, phish_test$label)
svm_confu_matr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 769   0
##          1   0 820
##
##                Accuracy : 1
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.516
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.484
##          Detection Rate : 0.484
##    Detection Prevalence : 0.484
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

```
dt_confu_matr <- confusionMatrix(dt_predict, phish_test$label)
dt_confu_matr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 769   0
##          1   0 820
##
##                Accuracy : 1
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.516
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.484
##          Detection Rate : 0.484
##    Detection Prevalence : 0.484
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

Well this is unexpected, both models performed perfectly, something is obviously up with the data, I'll into this but a quick test to see if the data is bad. I'll grab
outside data from another random URL dataset[4], and check how poor this model
will perform because the data must be skewed.

```
# Get new data2
url <- "https://raw.githubusercontent.com/David-DD-Del/edX-Harvard-Data-Science/refs/heads/main/CYO_caps

malicio_csv <- "data2.csv"

if(!file.exists(malicio_csv))
  download.file(url, malicio_csv)

malicio <- read.csv(malicio_csv)

# Match malicio with phish columns that I have access to
```

```
malicio <- malicio %>% mutate(dot_count = str_count(url, "\\."), url_len = str_count(url, "."),
                  digit_count = str_count(url, "[0-9]"), special_count = str_count(url, "[^[:alnum:]]")
                  hyphen_count = str_count(url, "-"), double_slash = str_count(url, "//"),
                  single_slash = str_count(url, "/"), at_the_rate = str_count(url, "@"),
                  protocol = ifelse(str_starts(url, "http"), 1, 0), label = ifelse(label == "bad", 0, 1
  mutate(label = factor(label))

malicio <- malicio %>% select(-url)
```

Now make the phish_train and test match malicio.

```
phish_train <- phish_train %>% select(-c(whois_regDate, whois_expDate, whois_updatedDate,
                                         protocol_count, web_traffic))
phish_test <- phish_test %>% select(-c(whois_regDate, whois_expDate, whois_updatedDate,
                                       protocol_count, web_traffic))
```

Retest phish train and test with the missing columns, then the outside data.

```
phish_short <- train(label ~ ., data = phish_train, method = "svmLinear")
p_short_pred <- predict(phish_short, newdata = phish_test)
svm_short <- confusionMatrix(p_short_pred, phish_test$label)
svm_short
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0   1
##            0 769   0
##            1   0 820
##
##                Accuracy : 1
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.516
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.484
##          Detection Rate : 0.484
##    Detection Prevalence : 0.484
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

```
pred_outside <- predict(phish_short, newdata = malicio)
svm_outside <- confusionMatrix(pred_outside, malicio$label)
svm_outside
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0   2666   1122
##          1  72977 343699
##
##                Accuracy : 0.8238
##                  95% CI : (0.8226, 0.8249)
##     No Information Rate : 0.8201
##     P-Value [Acc > NIR] : 2.594e-10
##
##                   Kappa : 0.0508
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.035245
##             Specificity : 0.996746
##          Pos Pred Value : 0.703801
##          Neg Pred Value : 0.824859
##              Prevalence : 0.179904
##          Detection Rate : 0.006341
##    Detection Prevalence : 0.009009
##       Balanced Accuracy : 0.515995
##
##        'Positive' Class : 0
##
```

```
dt_short_train <- train(label ~ ., data = phish_train, method = "rpart")
dt_short_pred <- predict(phish_short, newdata = phish_test)
dt_short <- confusionMatrix(p_short_pred, phish_test$label)
dt_short
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 769   0
##          1   0 820
##
##                Accuracy : 1
##                  95% CI : (0.9977, 1)
##     No Information Rate : 0.516
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
```

```
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.484
##          Detection Rate : 0.484
##    Detection Prevalence : 0.484
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

```
dt_pred_outside <- predict(phish_short, newdata = malicio)
dt_outside <- confusionMatrix(pred_outside, malicio$label)
dt_outside
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0   2666   1122
##          1  72977 343699
##
##                Accuracy : 0.8238
##                  95% CI : (0.8226, 0.8249)
##     No Information Rate : 0.8201
##     P-Value [Acc > NIR] : 2.594e-10
##
##                   Kappa : 0.0508
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.035245
##             Specificity : 0.996746
##          Pos Pred Value : 0.703801
##          Neg Pred Value : 0.824859
##              Prevalence : 0.179904
##          Detection Rate : 0.006341
##    Detection Prevalence : 0.009009
##       Balanced Accuracy : 0.515995
##
##        'Positive' Class : 0
##
```

Oddly DT and SVM performed exactly the same, maybe they end up picking the same threshold to split between the outcomes. The phish train and test still performed perfectly.

## 6 Results

Lets officially compute the best results and make a table before going over it.
A quick rundown of the models:

| Methods | Accuracy |
|---|---|
| Default train() with rpart method(DT) | 1.0000000 |
| Default train() with svmLinear method(SVM) | 1.0000000 |
| Default DT on Short Phis | 1.0000000 |
| Default SVM on Short Phis | 1.0000000 |
| Default DT on Outside | 0.8237685 |
| Default on SVM on Outside | 0.8237685 |

Even though I want to only care about the TNR at first, that no longer mattered, when a perfect everything was obtainable. Though I'm quite positive it has nothing to do with the models and more has to do with the data. I assumed the data was going to do poorly on the outside data, but it did okay. The only problem is there is no way to improve the model on the phish data because it achieved a perfect score right away. Lesson learned the hard way when picking your own dataset, do a quick base model real quick to get a feel for the data. I fell the only thing left to do is use that new data and start over the whole process, but I don't have the time.

## 7 Conclusion

Unfortunately I feel like I didn't produce any outstanding models, and the only outcome I can conclude is that the data is either manufactured or cherry picked to get a great model for results on itself but that leaves no room for improvement to try on outside data. If I had more time I would pull multiple URL datasets to test against to further improve this model, I assume thats the general approach anyways, start with some data improve it until you can't then use new data that fits the model to make its accuracy more robust on outside data.

## 8 References

1. dataset kaggle source:
   https://www.kaggle.com/datasets/sagarbanik/phishing-url-binary-datatset

2. Wikimedia Foundation. (2024, September 20). Whois. Wikipedia. https://en.wikipedia.org/wiki/WHOIS

3. Ruslan Magana Vsevolodovna. (2022, April 10). The Best Machine Learning Model for Binary Classification. https://ruslanmv.com/blog/The-best-binary-Machine-Learning-Model

4. data2 kaggle source: https://www.kaggle.com/datasets/antonyj453/urldataset