HarvardX PH125 Capstone: Detection of phishing using machine learning algorthms

Joseph Esensten

9/30/2019

Introduction

Phishing is a type of attack used in computing in which malicious emails masquerade as legitimate to fool users into taking an action which compromises security - in most cases, clicking a link. Phishing attacks are the number one information security threat to businesses and government costing billions of dollars in damages each year. The complexity of the attacks has risen over time and successful phishing attacks will often be targeted to individual persons within a company.

Emails are the method used for phishing attacks, but what makes phishing emails dangerous are the embedded hyperlinks and URLs. A hyperlink is the text which is displayed to the user while a URL is the actual pointer to a web resource. The links/URLs point to malicious websites which attempt steal information or compromise systems. These hyperlinks can be analyzed through automated means to make intelligent decisions on whether the link is safe or malicious. These decisions can be used to protect users and systems before they visit the malicious sites. This is often the purpose of web content filters both on workstations and the network. Identifying a highly accurate and efficient method for classifying hyperlinks and URLs as either phishing or non-phishing would give companies, government and users a fighting change against this threat.

The Data

I chose the "Phishing Dataset for Machine Learning: Feature Evaluation" dataset. This dataset was discovered through a search at Mendeley, an index of open data sets. It can be downloaded from: https://data.mendeley.com/datasets/h3cgnj8hft/1. From the author: This dataset contains comprehensive and up-to-date features of phishing and legitimate webpages. It contains 48 features extracted from 5000 phishing webpages and 5000 legitimate webpages. The webpages were downloaded from January to May 2015 and from May to June 2017. Phishing webpages were sourced from PhishTank and OpenPhish, while legitimate webpages were sourced from Alexa and Common Crawl.

This dataset was chosen due to several factors:

- Recency
- Applicability
- Size

Recency - The data was one of the most recent data sets I could find dealing with this subject. Most phishing-related sets were much older and did not reflect current threat conditions. Additionally, there were several research papers utilizing this data set and that gave me confidence in it.

Applicability - The data contained columns which were reflective of true hyperlinks and the metadata which can be gathered to make threat decisions from. This means that the data is relevant and at first glace, be used to make predictions from my experience in the field of information security. Additionally, the phishing and legitimate hyperlinke data came from reputable sources.

Size - The data was one of the largest phishing-related data sets. Throughout my search I found smaller (<5000 rows) data sets but these were discarded as candidates.

The Data is described below:

Attribute	Type	Description
NumDots	double	Number of dots in url
SubdomainLevel	double	# Levels of subdomains in URL
PathLevel	double	# Levels of path in URL
UrlLength	double	URL Total Length
NumDash	double	# of - symbols
NumDashInHostname	double	# of - symbols in hostname
AtSymbol	double	# of @ symbols
TildeSymbol	double	# of ~ symbols
NumUnderscore	double	# of _ symbols
NumPercent	double	# of % symbols
NumQueryComponents	double	Number of Queries =?
NumAmpersand	double	# of & Symbols
NumHash	double	# of # symbols
NumNumericChars	double	# of Numbers
NoHttps	double	Non HTTPS Page
RandomString	double	Contains Random String
IpAddress	double	URL is an IP Address
DomainInSubdomains	double	Domain Name contained in Subdomains (google.google.com)
DomainInPaths	double	Domain Name contained in paths (www.google.com/google)
HttpsInHostname	double	HTTPS in name (https.google.com)
HostnameLength	double	Length of hostname
PathLength	double	Length of path
QueryLength	double	Length of query
DoubleSlashInPath	double	Path contains double slash
NumSensitiveWords	double	# of sensitive words
EmbeddedBrandName	double	Inc Brand Name (Bank of America)
PctExtHyperlinks	double	% External Hyperlinks
PctExtResourceUrls	double	% External resource URLs
ExtFavicon	double	External icon
InsecureForms	double	Insecure forms submission
RelativeFormAction	double	Form action (submit) is relative to the page (../action)
ExtFormAction	double	External form action
AbnormalFormAction	double	Abnormal form action
PctNullSelfRedirectHyperlinks	double	_target can be a trick to get a new blank page to open yet still have some control over the original page. % of links that do this.
FrequentDomainNameMismatch	double	Common domain name does not match certificate (google.com)
FakeLinkInStatusBar	double	onMouseOver: Fake URL shown in status bar to trick users
RightClickDisabled	double	Code disables right click
PopUpWindow	double	Code contains a popup
SubmitInfoToEmail	double	Code contains a mailto:
IframeOrFrame	double	Frames used
MissingTitle	double	Title of page missing
ImagesOnlyInForm	double	Not sure here. Guessing images in forms are not good.
SubdomainLevelRT	double	

Attribute	Type	Description
UrlLengthRT	double	
PctExtResourceUrlsRT	double	
AbnormalExtFormActionR	double	
ExtMetaScriptLinkRT	double	
PctExtNullSelfRedirectHyperlinksRT	double	Null _target redirect attack method to an external site
Phish	double	Phishing = 1, Not = 0

The data set is comprised of 49 columns and 10000 rows. (5000) 50% of the rows are verified phishing sites (5000) 50% of the rows are valid safe sites

The raw data was in ARFF format. This format works well with Weka software, but not R. It was converted using the farff::readARFF function.

Column headers were descriptive except for several cases. Fields with the suffix RT could not be determined though they contained data. There were no clues in the data source metadata or on the website from the data. These fields lack a description. The "Description" field was added to this table during data preparation to aid in reading this report. This field would only be used for this report and was not used for any data analysis. The "Description" field is a concise explaination of each columns data.

The field "Phish" is the only non-numeric field. This field was originally named "CLASS_LABEL", but was renamed for readability. All other fields are numeric and of type "double". This field is the target field for prediction.

The data was checked for NA values.

Methods/Analysis

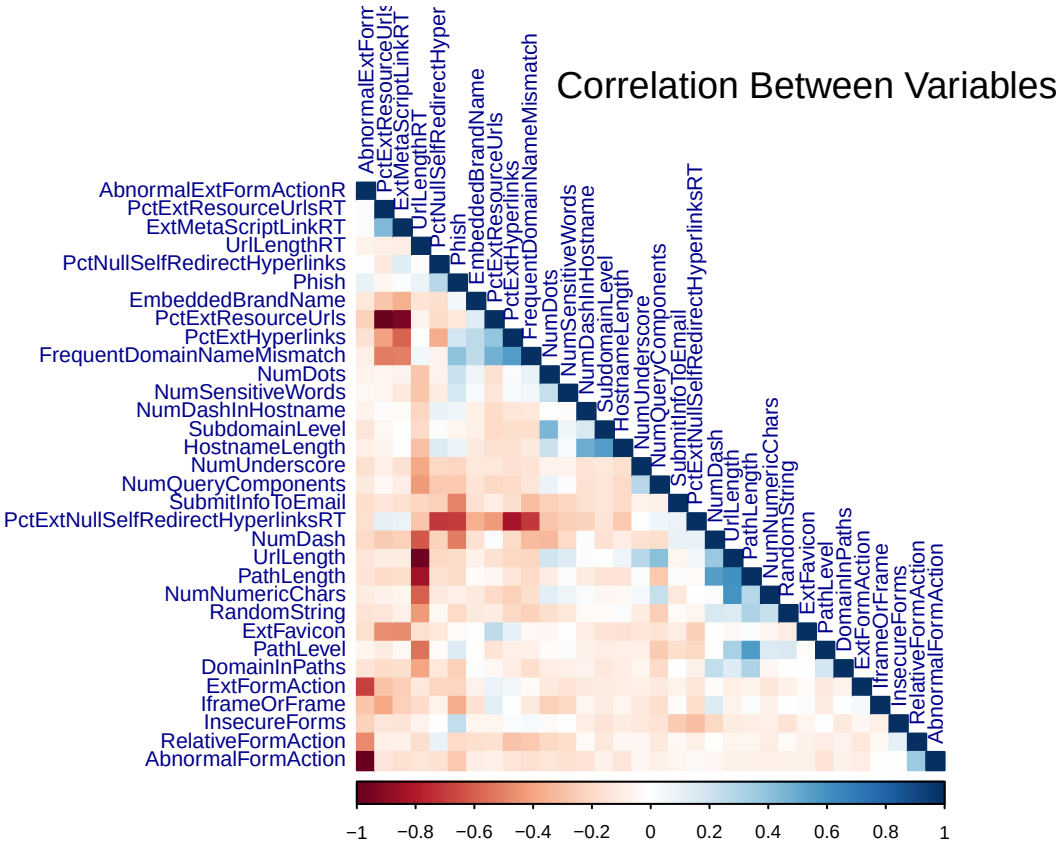
Preprocessing

The first step taken was to prepare the data for prediction. Several methods were used to do this:

1. Check for and remove data with Near-Zero Variance (NZV).

```
NZV Variables Removed:
AtSymbol TildeSymbol NumPercent NumAmpersand NumHash NoHttps IpAddress
DomainInSubdomains HttpsInHostname QueryLength DoubleSlashInPath FakeLinkInStatusBar
RightClickDisabled PopUpWindow MissingTitle ImagesOnlyInForm SubdomainLevelRT
```

2. Check for and analyze correlations between the variables.



The diagram below shows the correlation between variables. The strongest correlations are represented by the darkness of the squares, both blue and red. There were three sets of fields that were moderately correlated, but did not cross the 90% threshold I set for exclusion:

- [4] UrlLength, [27] UrlLengthRT: 80% correlation
- [17] PctExtResourceUrls, [28] PctExtResourceUrlsRT: 80% correlation
- [22] AbnormalFormAction, [29] AbnormalExtFormActionR: 81% correlation

Clearly RT and non-RT variables are correlated for similar names. I left them in the data set as the correlation was not above 90% showing that there is some variance which could be helpful in prediction.

3. Do the Shuffle.

When the dataset was created, a benign set and a phishing set were appended together. This created a dataset with the first 50% and the last 50% similar in terms of the Phish class. To remove any bias due to this, the dataset was shuffled using a random seed.

4. Split the data into Training and Test Sets.

The phishData data set was split utilizing a random seed into an 80/20 split.

- Training Set (phishTrain): 80%, 8000 rows

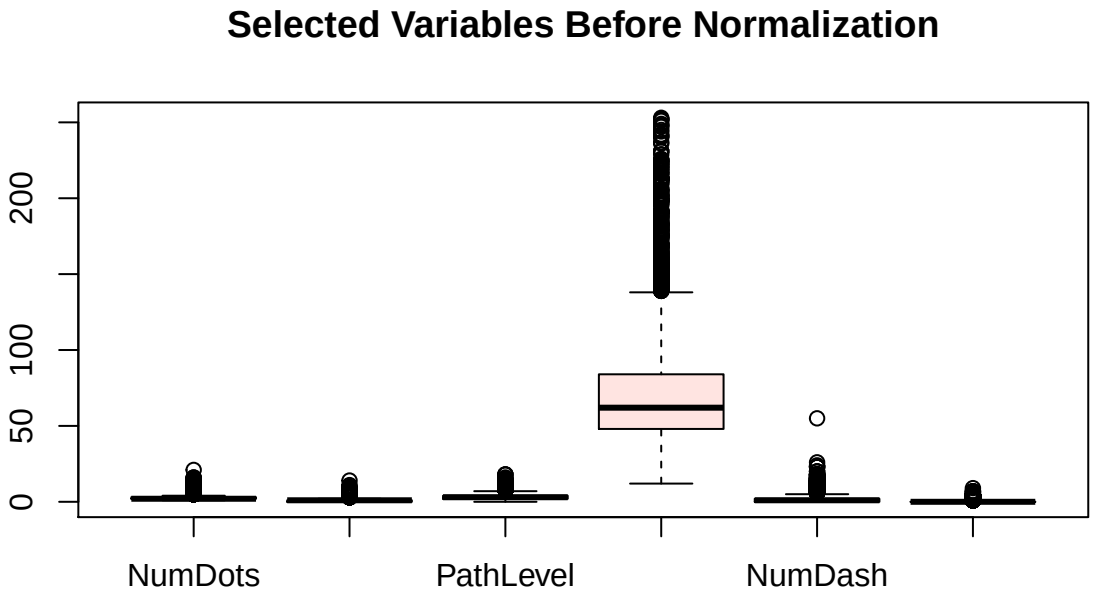
- Test Set (phishTest): 20%, 2000 rows

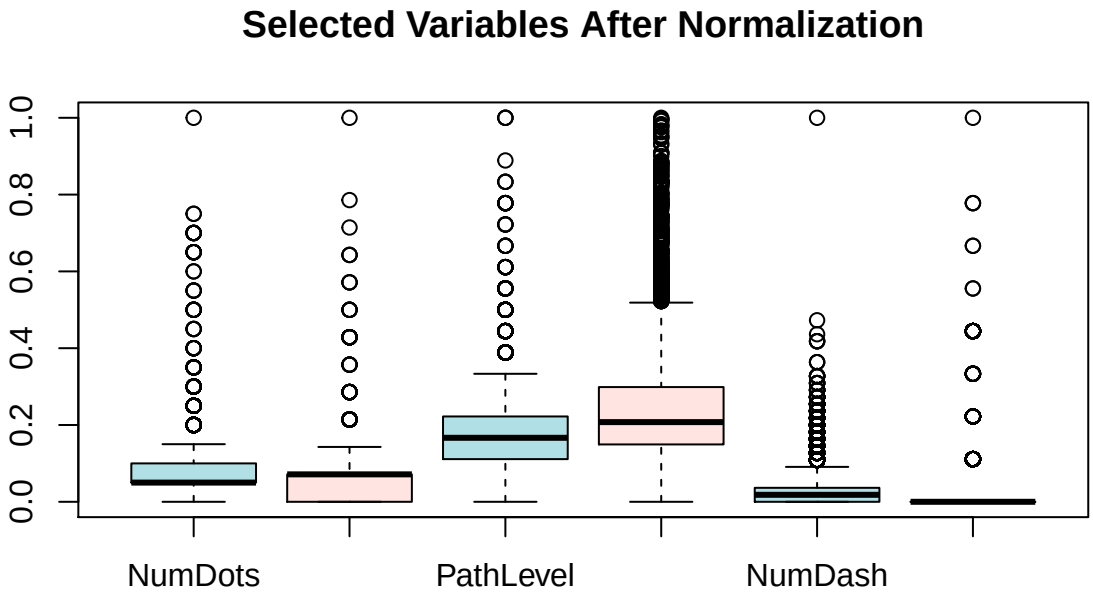
Additional derivatives were created for the random forests and KNN algorythms which requires the "Phish" (target) field to be separate from the training and test data sets. Those were:

- phishTrainTarget <- phishTrain$Phish - Target set including only the "Phish" field from the training set
- phishTestTarget <- phishTest$Phish - Test set including only the "Phish" field from the test set
- phishTrain2 <- phishTrain[,-32] - Training set without the "Phish" field
- phishTest2 <- phishTest[,-32] - Test set without the "Phish" field.

5. Normalize data for comparison.

Fields will be ranged from 0 to 1 so they can be easily measured for distance such as in the K-Nearest Neighbors algorythm. In order to compare training and test data, the test data will also be normalized, but by the training metrics. As an example, The following graphs show the first 6 columns of data on a boxplot. The first shows the data before normalization and the second shows after. Clearly the second, normalized data is easier to compare. A stark contrast is the variability of the ranges of each data point in the second diagram compared to the first.

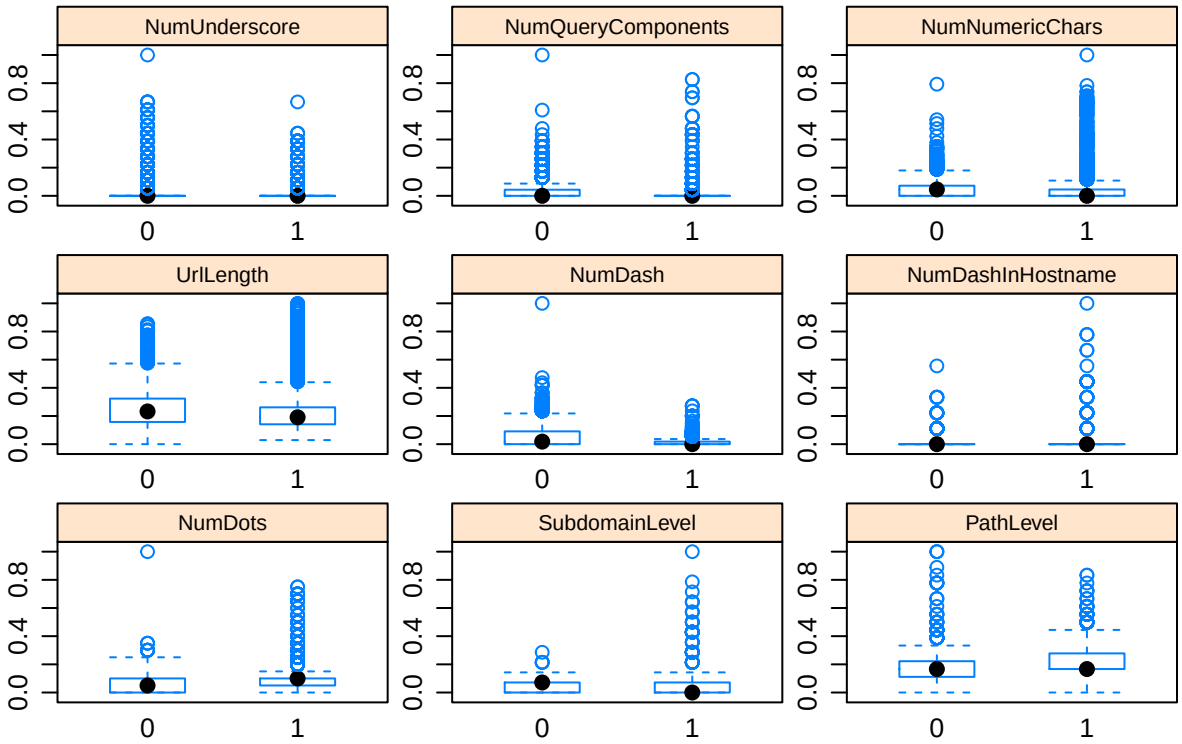




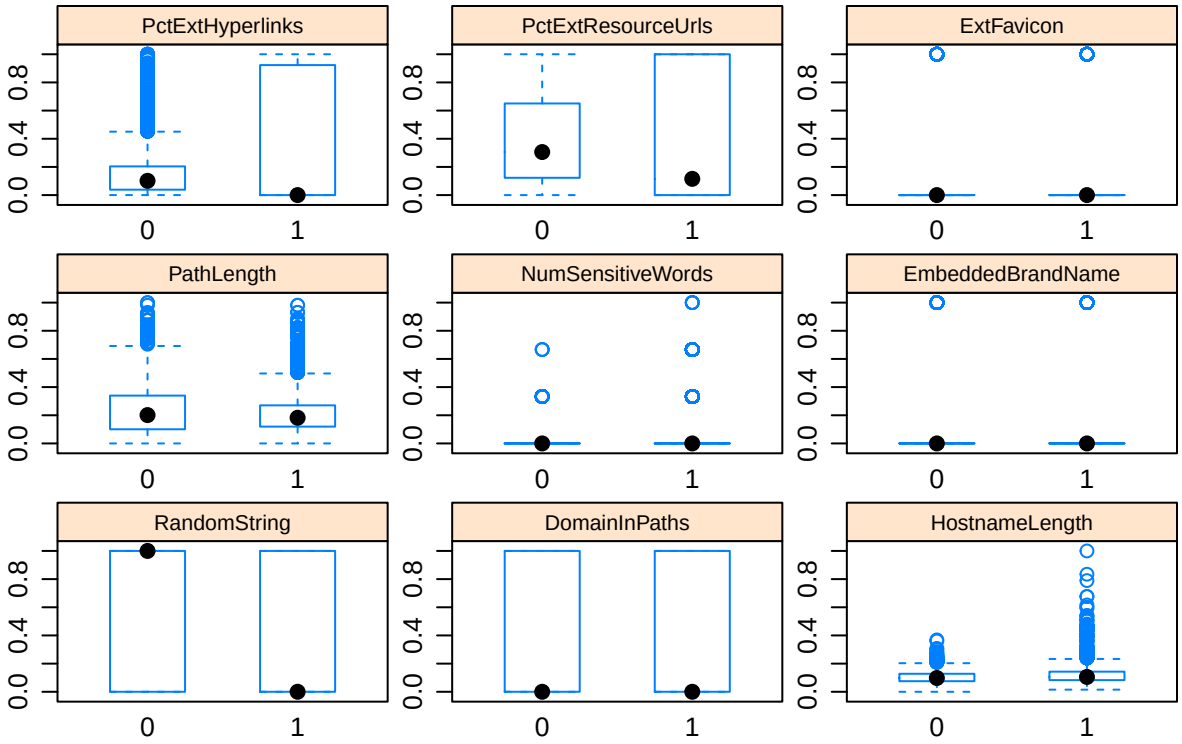
Variable Importance

Nonwithstanding specific algorythm variable importance, such as that from Random Forest, I have created a general view idea of variable importance below. It will allow us to see if a given predictor is important

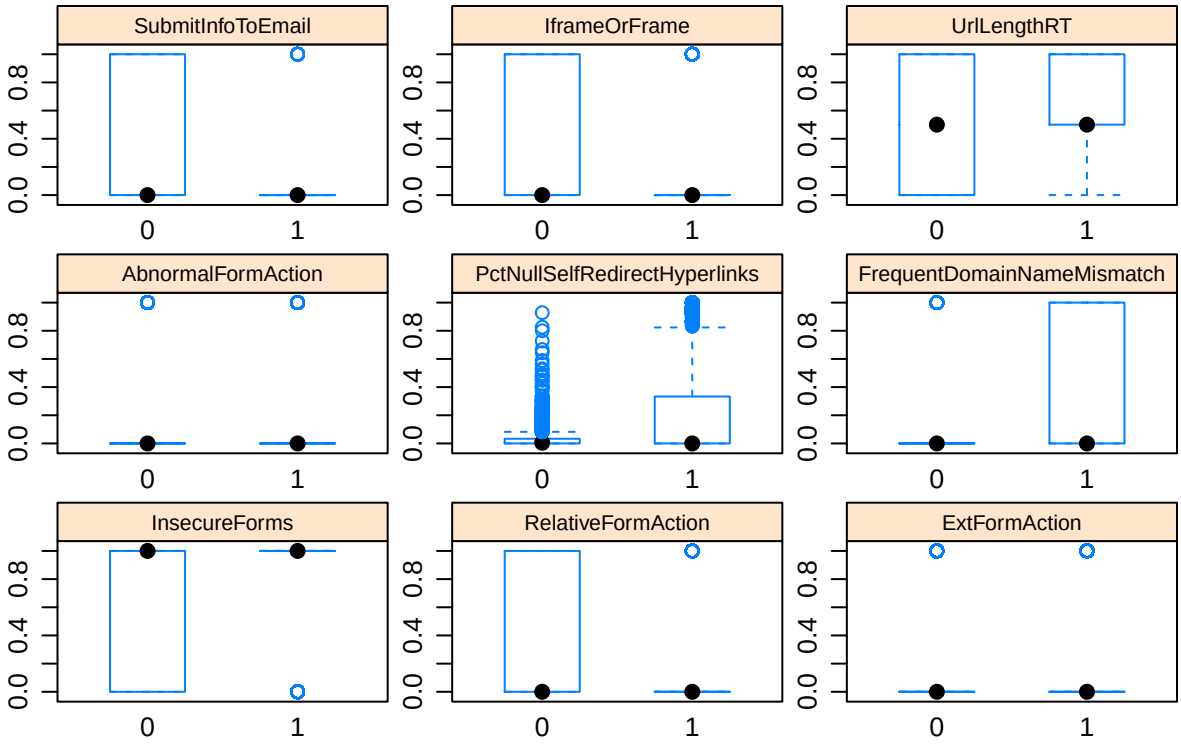
to the class of Phish. Each box represents a variable and displays in the X Axis: 0 (Not a Phish) and 1 (A Phish) to represent the categories. Two boxplots reside inside each box and represents data for that variable as it related to the Phish class. The black dot represents the mean and most values will fall into the blue boxes. The top and bottom of the box represent the 25th and 75th percentile respectively. The blue blue dots represent outliers. Significant differences between the boxplots within each varaible's plot should represent a high level of importance to the Phish class.



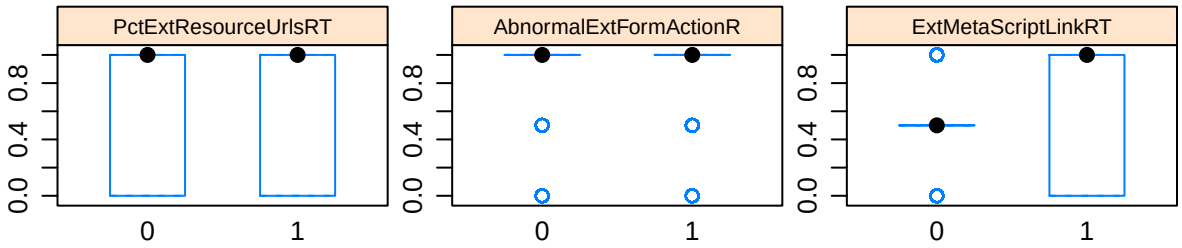
Variable Importance to Target



Variable Importance to Target



Variable Importance to Target



Variable Importance to Target

Several variables have significant differences in their boxplots. It will be likely these will have a high importance for prediction:

- PctExtHyperlinks
- PctExtResourceUrls
- RandomString
- SubmitInfoToEmail
- IframeOrFrame
- FrequentDomainNameMismatch
- InsecureForms
- RelativeFormAction
- ExtMetaScriptLinkRT

Inversely, the following variables do NOT appear to have a likely impact on the target:

- ExtFavicon
- PathLength
- EmbeddedBrandName
- DomainInPaths
- ExtFormAction
- UrlLengthRT
- PctExtResourceUrlsRT
- AbnormalExtFormActionRT

Algorythms

In order to make a prediction off of the training data machine learning is used. In supervised machine learning, a training set is used to build logic which represents patterns in the data which can be used to make predictions. Predictions are then made on the learned logic utilizing the variables from a test set. The test data result (phish/no phish) is then compared to the predicted output (phish/no phish). The accuracy of the algorythm is the number of predictions made correctly out of the total number of predictions. We will use accuracy as the differentiator between how the algorythms performed. Each algorythm will perform differently given the data set composition and purpose of the prediction. Therefore it is important to try several algorythms which appear appropriate and measure the outcomes. It was important for this study to select algorythms which could predict a two-factor or binary outcome based upon multiple numeric input variables. The algorythms chosen were:

- K-Nearest Neighbors
- Random Forests
- Logistic Regression
- Penalized Logistic Regression
  - Lasso
  - ElasticNet
  - Ridge

In addition to Accuracy, the Root Mean Squared Error (RMSE) was tracked with each algorythm. This is a measure of the prediction errors. A lower RMSE is an indication of a better fit should two algorythms perform similarly. Tuning was also performed on each algorythm to ensure that performance was optimized. Specific tunings for each algorythm will be discussed in the results section. Additionally, ensemble methods were tried using the above algorythms in an attempt to achieve greater results. This will be discussed in the Results section.

Results

The results are displayed below. Accuracy and RMSE measures were used to determine the best algorythm for the dataset and purpose.

	Model	Accuracy	RMSE
1	RF	0.9825	0.1322876
8	Stack	0.9820	0.1341641
7	Ensemble	0.9800	0.1414214
2	KNN	0.9535	0.2156386
3	LogR	0.9430	0.2387467
4	Lasso-PenalizedLogR	0.9405	0.2439262
5	ElasticNet-PenalizedLogR	0.9385	0.2479919
6	Ridge-PenalizedLogR	0.9310	0.2626785

Algorythm Performance

Random Forest

Random forest performed the best against the data set with a 98.25% accuracy rate. The random forest algorythm is made up of many decision trees. Each decision tree generates an outcome (Phish/No Phish) and the predominant outcome is selected. The power of this algoryhtm lies in the community of decisions

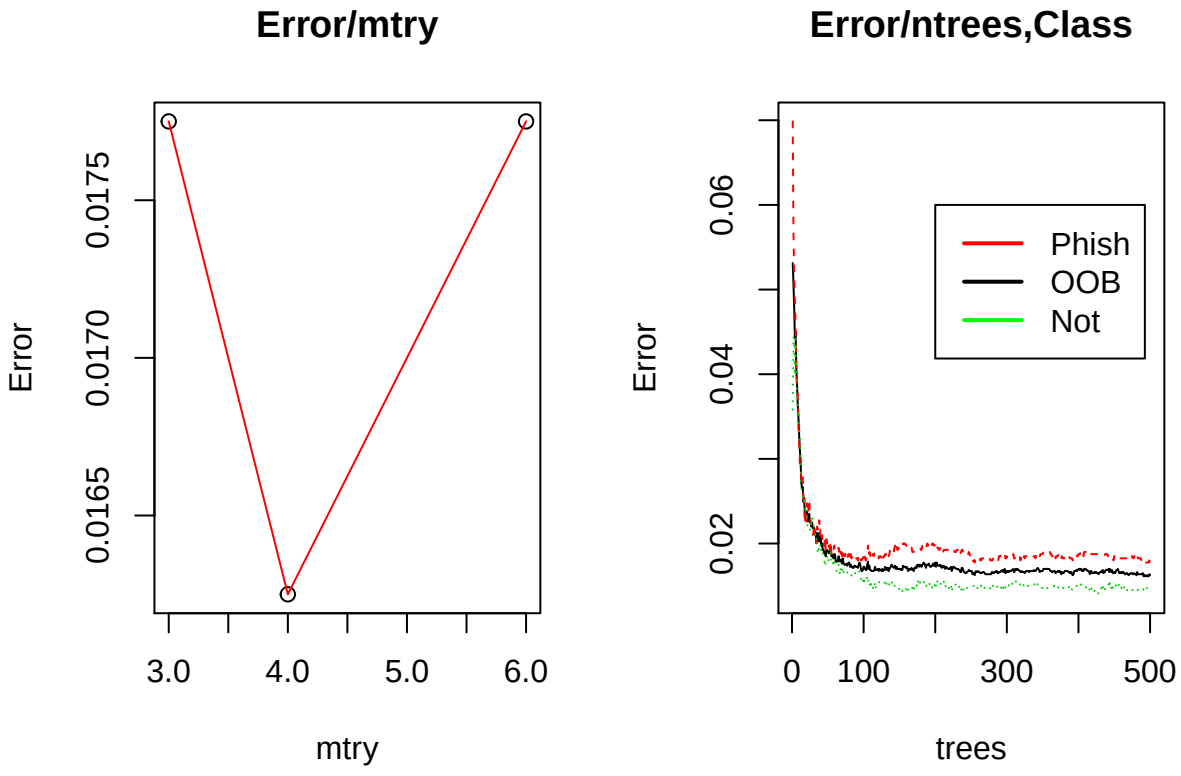
it creates which is more powerful than a single node. This method improves upon the deep learning of trees and averages a collection of them in order to reduce variance. The error rate is controlled as the trees in the forest tend to correct for one another.

The data is posed well for this algorythm as we desire a single classification prediction from multiple input variables. Bootstrap samples are used to grow a forest of trees. Nodes are created using a number (mtry) of random variables from the set. The best split is found for the mtry variables. The trees are grown to max depth and the average of the trees is used for prediction data.

In order to find the lowest error rate and therefore highest accuracy, a number of factors will be tuned for the random forest:

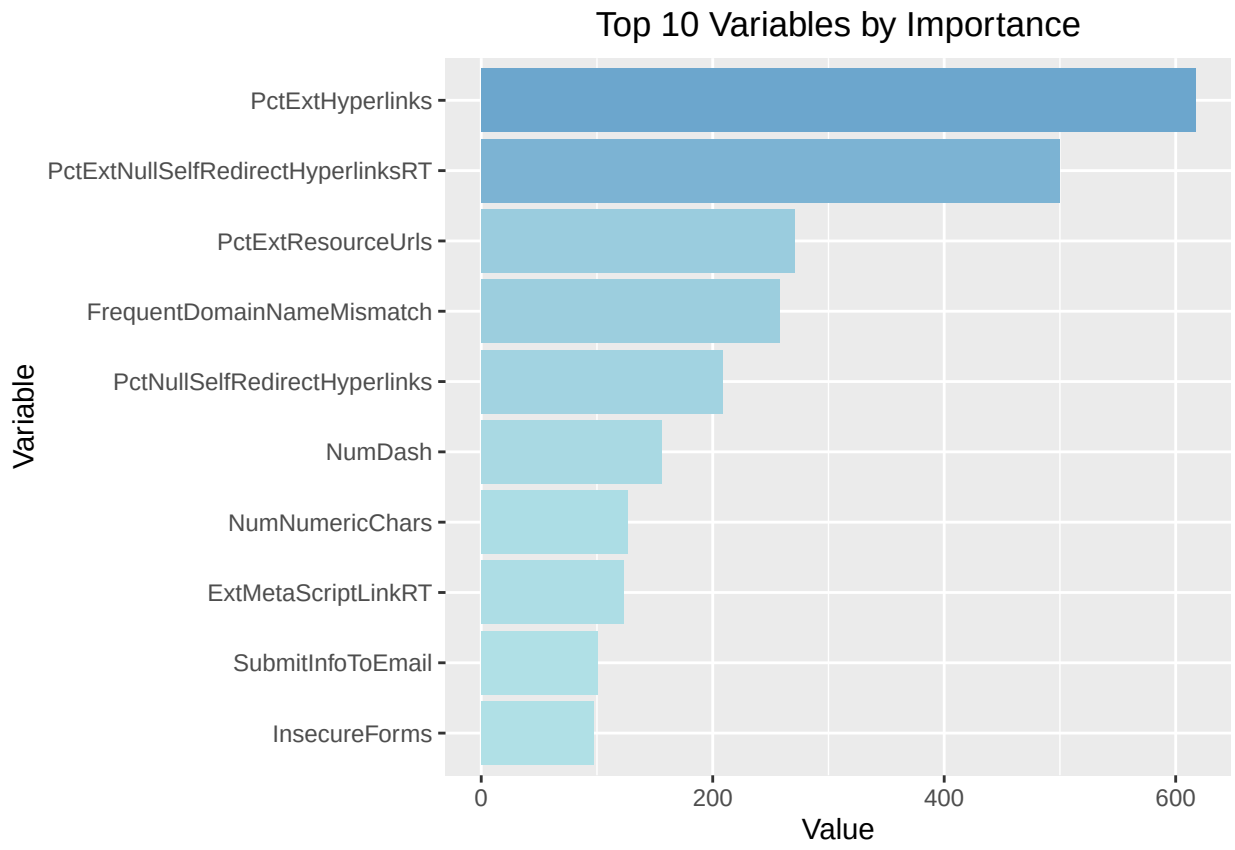
- number of trees (ntree)
- number of variables tried at each split (mtry)
- size of nodes (node_size)
- sample size (samp_size)
- split rule: gini or extratrees

To begin, TuneRF and RandomForest were used to get in the ballpark for mtry and ntrees respectively. The following two plots represent the best tuning values. The plot on the left shows that an mtry value of 4 minimizes error. The plot on the right shows the error rate stabilized with 500 trees. The OOB error is out-of-bag error and is the mean of the error for each prediction. The error for each class (Phish/No Phish) is also displayed.



Further tuning was performed utilizing the ranger library due to its speed over the other implemenations of random forest. Ranger was used to test ranges of mtry, samp_size, and node_size. From the tuning results, 500 was selected for the number of trees and mtry selected was 4. The output below details the importance

of the variables to the model. This is the contribution the variable played in the model. Only the top 10 are shown for brevity.



Variable importance is an indicator of model effectiveness. Were the important variables the ones which make sense for the data? Were variables that were thought to be important rated lower than expected? Can any variables be removed to improve training time?

The four most important variables to this model are:

- PctExtHyperlinks (percentage of hyperlinks that were external [internet])
  - This makes sense as phishing emails will almost always point to external sites. Though phishing sites can be created inside a trusted network, this is almost always not the case as it assumes compromise has already occurred.
  - This variable was identified as a possible important predictor in the Methods/Analysis section.
- PctExtNullSelfRedirectHyperlinksRT (percentage of null target hyperlinks)
  - This variable makes sense as this method of _target should only be seen in an attack.
  - This variable was identified as a possible important predictor in the Methods/Analysis section.
- FrequentDomainNameMismatch (Domain name does not match security certificate)
  - This makes sense as this is either from a gross misconfiguration or more commonly a phishing attack
  - This variable was identified as a possible important predictor in the Methods/Analysis section.
- PctExtResourceURLs (URLs which point to external sites [internet])
  - Much like the top variable, this is expected as phishing emails will always point to external resources, but the inverse is not true. Legitimate emails can also point to external resources, and do so often.

– This variable was identified as a possible important predictor in the Methods/Analysis section.

Interesting as well is what did not come up in the top 10, which I would have expected due to the earlier predictor variable analysis found in the Methods/Analysis section:

- IframeOrFrame (18th)
- Random String (27th)

This algorythm was highly accurate and fast utilizing the Ranger library. The results were outstanding and are listed below. I will highlight some of the data. Sensitivity is the percentage of time a positive result is not overlooked. Specificity is the percentage of time a negative result was actually negative. This algorythm performed highly in both of these categories. The difference between the two is neglegible and fits with overall accuracy. Relevance measures of precision and recall are also very high. Precision is the percentage of correct positive predictions to the total positive predictions. Recall is the percentage of correct positive predictions to the total positives. F1 is a function of precision and recall which measures the balance between the two.

Random Forest (Ranger) Performance:

Measure	Value
Accuracy	0.9825
Sensitivity	0.9820
Specificity	0.9830
Precision	0.9830
Recall	0.9820
F1	0.9825

Of note was the initial use of the caret::RandomForest function to make the model. This resulted in extremely slow computation times. Ranger, another implementation of random forest, was chosen to improve the speed of computation and it delivered on that expectation.

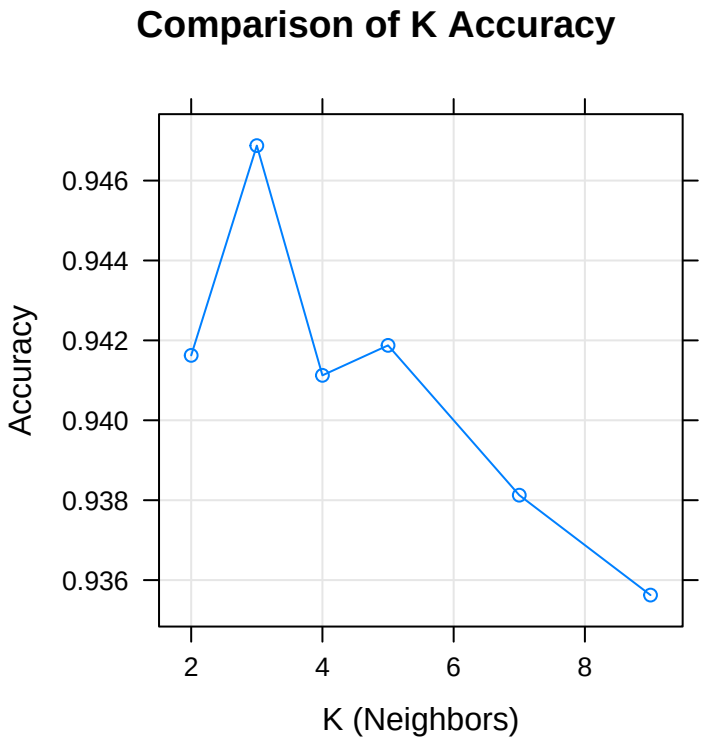
K-Nearest Neighbors

Coming in second place in terms of accuracy was KNN or K-Nearest Neighbors. This algorythm groups data by similarity and uses Euclidean distance from those groups to predict the outcome. Like Random Forest, KNN also is able to predict a categorical outcome based upon multiple input variables. KNN relies on a value of K, which is the number of nearest neighbors to consider when making a prediction.

The data is poised well for KNN because of the desired output classification, which KNN can handle. The variables easily form groups which can be used by the algorythm to measure new data against and make a prediction. In order for KNN to work well, some data cleaning was necessary as out of the box Training and Test sets would not provide accurate results. For instance, much of the numeric data variables from the training set was not easily comparable due to having different scales. The data was normalized specifically to ensure KNN was able to make unbiased predictions. This normalization was discussed earlier in the "Data" section.

The first step was to determine the best value of K (Neighbors) given the training data. The caret package was used to set parameters for the model. There is a risk of overtraining with low values of K. This is because of noise having a higher influence at these low values. At high values, the algorythm becomes computationally more expensive. Often a K value of K=sqrt(n) is recommended as a starting point. This value was tested and found to provide significantly less accuracy. values of 2,3,5,7 and 9 were tested.

The resulting output is below:



K = 3 was the value which provided the highest accuracy.

KNN Performance:

Measure	Value
Accuracy	0.9535
Sensitivity	0.9440
Specificity	0.9630
Precision	0.9623
Recall	0.9440
F1	0.9531

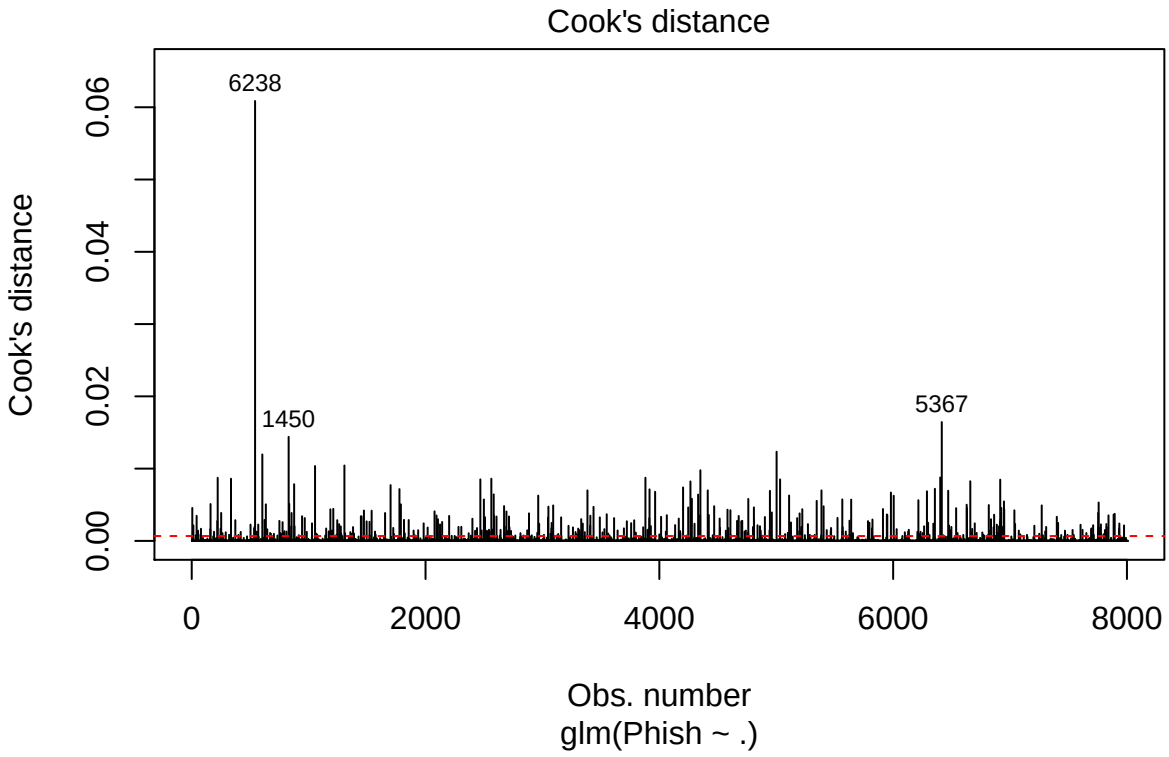
Logistic Regression

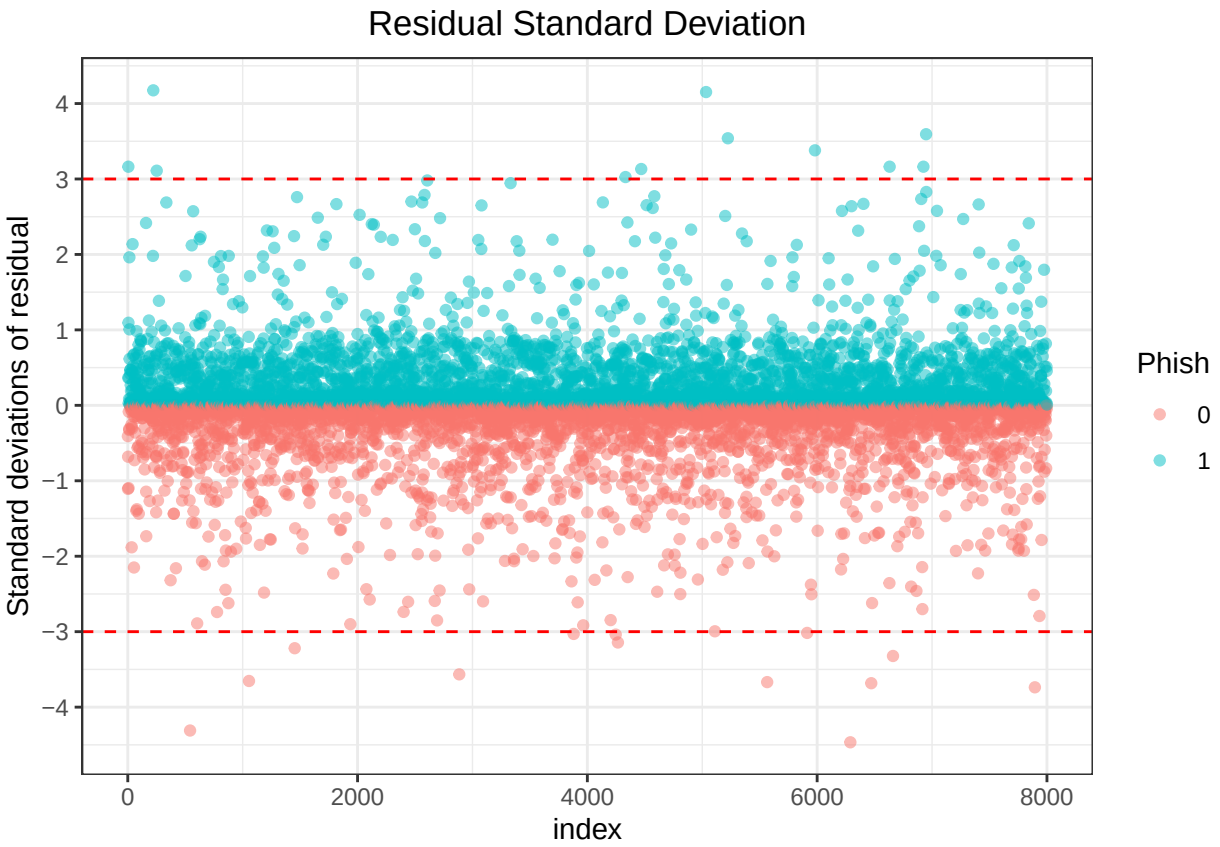
Logistic regression is a method which predicts a target binary variable based upon one or more independent variables. It is similar to linear regression, but takes advantage of a sigmoid (logit) function to predict a categorical outcome. This is calculated using the odds that a prediction belongs to a category.

A logistic regression model was created using the generalized linear models (GLM) function. This model was able to predict outcomes with 94% accuracy. Below are some interesting performance metrics:

- Cooks distance depicts the influence of observations on the fitted response values in the model. The data below shows that at least three observations are worth investigating as outliers. The red dashed line depicts values which are outside 4 times the mean of cooks distance. In this case, over 300 observations.

- The standardized residuals plot shows the deviation of the residuals. Residuals over 3 standard deviations can skew the results and should be investigated. These are depicted above and below by the red dashed lines respectively.





The outliers identified by cooks distance and standardized residuals plots above are shown below. These were calculated by finding observations which met both cooks distance and standardized residuals outliers classification. These observations were investigated in detail.

.rownames	.std.resid	.cooksd
3792	3.163080	0.0045585
3012	4.175108	0.0087271
4278	3.109937	0.0038680
2394	3.025808	0.0064121
3917	3.132315	0.0047955
1973	4.152352	0.0085084
3083	3.539741	0.0043868
4719	3.380031	0.0066918
3788	3.163080	0.0045585
3791	3.163080	0.0045585
4721	3.593484	0.0054580

Upon investigation, it was found that the observations did not contain data which was was out of the ordinary, out of range, or in error. All observations were kept in the dataset.

Logistic Regression Performance:

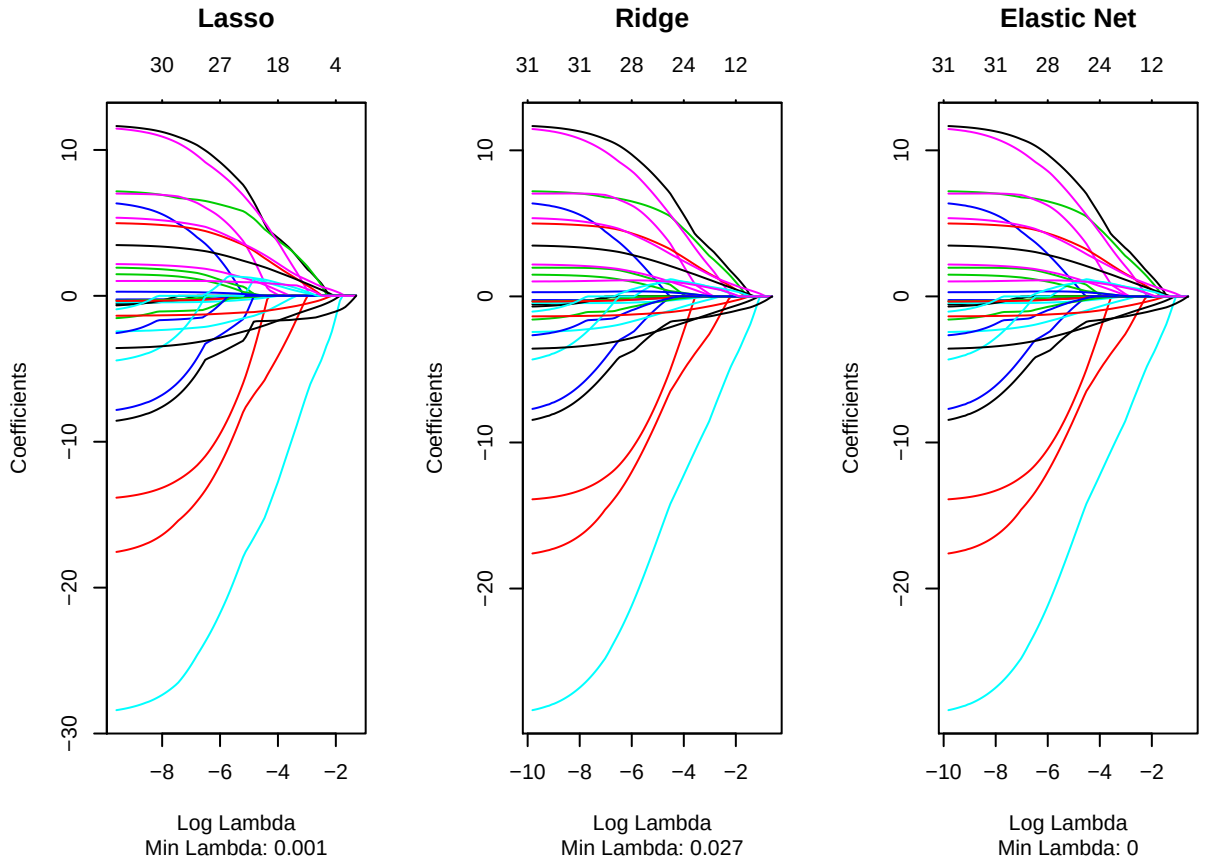
Measure	Value
Accuracy	0.9430
Sensitivity	0.9390
Specificity	0.9470
Precision	0.9466
Recall	0.9390
F1	0.9428

Penalized Logistic Regression

Three penalized regularization methods were also utilized to attempt to improve the Logistic regression accuracy. This was achieved through the addition of a penalization term to the log likelihood function within the prediction. In practice, this was achieved by modifying the value of alpha of our prediction by the glmnet function. Lambda is a parameter in logistic regression which is selected which minimizes error. It is tuned using cross-validation.

- Lasso (Least Absolute Shinkage and Selection Operator) - alpha = 1
- Ridge - alpha = 0
- Elastic Net - alpha = .5

Below are the plots of each method showing the log of lambda and the coefficients. When the coefficients reach zero, the optimum value of lambda is achieved. These optimum values are listed below each method. The plots were created to validate the min lambda calculation from cross-validation.



The plots confirm the cross-validation lambda value. Predictions were made off this value of lambda for each method of penalized logistic regression. These predictions were then compared to the test data and overall

accuracy was calculated. Of the penalized logistic regression models, ridge regression was the most accurate. The results are shown below.

Algorythm	Accuracy
Lasso	0.9405
Ridge	0.9385
ElasticNet	0.9310

Penalized Logistic Regression (Lasso) Performance:

Measure	Value
Accuracy	0.9405
Sensitivity	0.9380
Specificity	0.9430
Precision	0.9427
Recall	0.9380
F1	0.9404

Penalized Logistic Regression (Ridge) Performance:

Measure	Value
Accuracy	0.9385
Sensitivity	0.9330
Specificity	0.9440
Precision	0.9434
Recall	0.9330
F1	0.9382

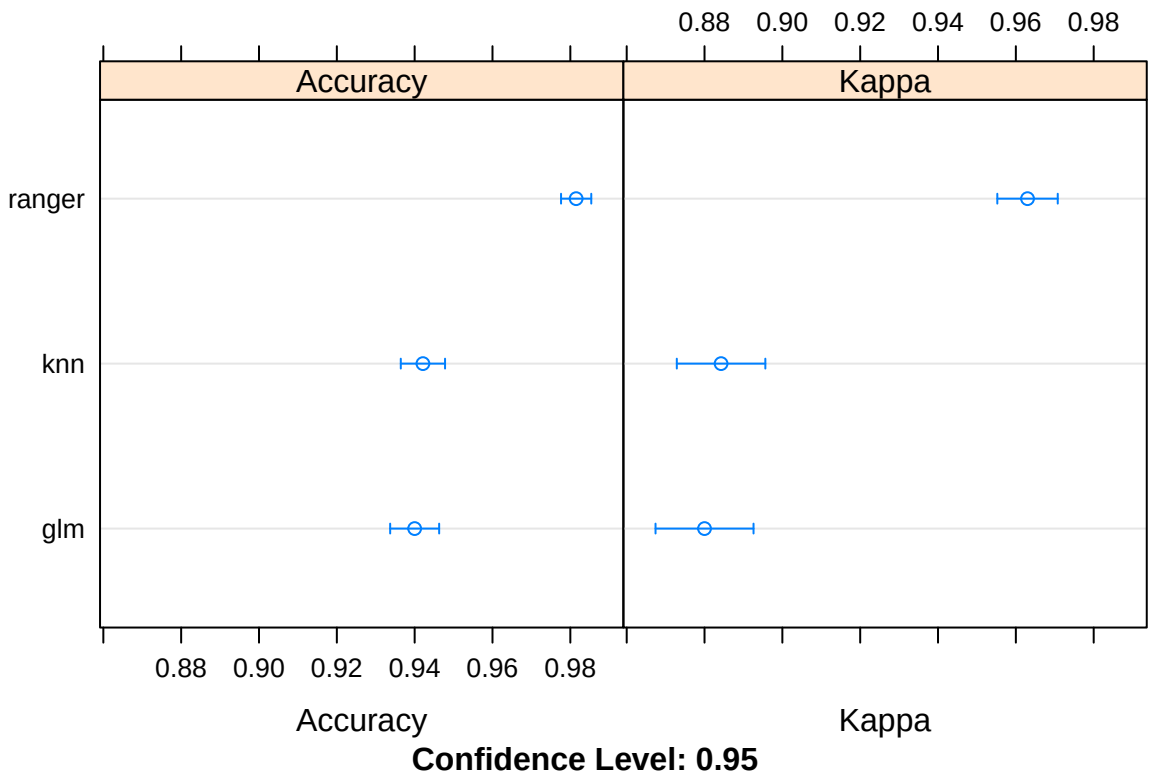
Penalized Logistic Regression (ElasticNet) Performance:

Measure	Value
Accuracy	0.9310
Sensitivity	0.9350
Specificity	0.9270
Precision	0.9276
Recall	0.9350
F1	0.9313

Ensemble and Stacking

Combining methods can often lead to higher accuracy. This is where the ensemble and stacking methods come in. KNN, Ranger (Random Forest), and Generalized Linear Model (GLM) were combined using the caretEnsemble package.

10 fold cross validation was used to calculate accuracy and kappa values for KNN, ranger and glm. This was necessary to perform as all algorythms needed to have the same resampling parameters. The output is visualized below:



Before proceeding, the models were checked for correlation. Combining models which are highly correlated is unnecessary and inefficient. Data below shows no strong correlation between models.

	knn	ranger	glm
knn	1.0000000	0.3243484	0.6079786
ranger	0.3243484	1.0000000	0.2908772
glm	0.6079786	0.2908772	1.0000000

Ensembling and Stacking were the two methods attempted. Each method takes a different approach.Ensembling utilized a generalized linear model (GLM) to blend KNN and Random Forest. Stacking utilized the svmPoly algorythm to combine the outputs of KNN and Random Forest. The svmPoly algorythm was chosen after manual testing of various options. These are detailed below:

Name	Accuracy
svmPoly	0.982
multinom	0.981
extratrees	0.981
pda	0.981
parRF	0.981
rFerns	0.981
lda	0.981
pls	0.981
penalized	0.981
LogitBoost	0.980
svmRadial	0.950

Clearly there was not much of a differentiator between most of the algorythms and exhaustive testing of more options was not attempted. Adding the additional complexity of the ensemble methods did not result in a measurable gain over the most accurate single method: Random Forest.

The results are below:

	Accuracy
Ensemble	0.980
Stack	0.982

Ensemble Performance:

Measure	Value
Accuracy	0.9800
Sensitivity	0.9830
Specificity	0.9770
Precision	0.9771
Recall	0.9830
F1	0.9801

Stack Performance:

Measure	Value
Accuracy	0.982
Sensitivity	0.983
Specificity	0.981
Precision	0.981
Recall	0.983
F1	0.982

Conclusion

Predicting phishing emails with high confidence is possible using machine learning. With additional time, tuning and experimentation, it is probable that the accuracy can be pushed even higher. As part of this exercise I was able to achieve a 98.25% accuracy rate. This was achieved utilizing the Random Forests algorythm on preprocessed data. Additional preprocessing may also result in additional accuracy gains. With any additional processing there are diminishing returns as the time it takes to calculate predictions increases. In the case of URL and email analysis, time is a factor as delaying URL filtering and email delivery for a long period of time is not a desired side effect for recipients despite security protections implemented. This is exacerbated by the volume of URLs and emails which require analysis received by an organization's cyber security systems.

Limiting this study were several factors which can be easily addressed in further study. First and foremost, my experience level was a limiting factor. This having been my first self-directed data science and machine learning project: This meant that I spent many nights frustrated and searching for solutions. As I discovered solutions to problems, I would inevitably uncover a step that I had missed earlier in the analysis and have to return to that portion. Secondly, the size of the dataset was limited to 10,000 observations. I was unable to find additional datasets which were able to augment this dataset or find larger URL datasets which contained this level of detail. Sponsorship of a larger dataset by an organization would be a next step to further refine the methods. This would also enable testing against live data.

Compared to similar studies I was able to achieve a respectable accuracy level. Each of these studies cannot be compared exactly to this study and used varied datasets. This list is also not exhaustive.

Citation	**Accuracy**
Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, Banu Diri, Machine learning based phishing detection from URLs, Expert Systems with Applications, Volume 117, 2019, Pages 345-357, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2018.09.029.	**97.98**
Bergholz, André & Chang, Jeong Ho & Paass, Gerhard & Reichartz, Frank & Strobel, Siehyun. (2008). Improved Phishing Detection using Model-Based Features. https://www.researchgate.net/profile/Gerhard_Paass/publication/220271860_Improved_Phishing_Detection_using_Model-Based_Features	**99.85**
Information Sciences, Volume 484, May 2019, Pages 153-166, A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. https://www.sciencedirect.com/science/article/pii/S0020025519300763	**94.60**
A. Hamid I.R., Abawajy J. (2011) Hybrid Feature Selection for Phishing Email Detection. In: Xiang Y., Cuzzocrea A., Hobbs M., Zhou W. (eds) Algorithms and Architectures for Parallel Processing. ICA3PP 2011. Lecture Notes in Computer Science, vol 7017. Springer, Berlin, Heidelberg, https://link.springer.com/chapter/10.1007%2F978-3-642-24669-2_26	**96.00**

Future work should focus on testing to assess the requirements necessary to use the algorythm with live data. Performance and accuracy would both have to be weighed and the most efficient and accurate values found. Future work could also focus on attempting more advanced algorythms such as neural networks which could result in higher accuracy. I attempted to implement Multi-Layer Perceptron (MLP) after discovering another study analyzing machine learning for phishing detection: https://ieeexplore.ieee.org/document/6510259. After many days attempts, this was abandoned due mostly to the author's inexperience. Overall, I was able to apply machine learning to a real world problem which is faced every day and achieve results which could be applied to protect an organization. This experience was outstanding in applying the concepts of data science, r, and machine learning which could be applied directly to future data science problems.

References

Brownlee, Jason. 2019. Machine Learning Mastery. R Machine Learning. Tune Machine Learning Algorithms in R (random forest case study). https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/

Choon, Lin Tan. 2018. Phishing Dataset for Machine Learning: Feature Evaluation. https://data.mendeley.com/datasets/h3cgnj8hft/1

Irizarry, Rafael. 2019. Introduction to Data Science. Data Analysis and Prediction Algorithms with R. https://rafalab.github.io/dsbook/

Jawaharlal, Vijayakumar. 2014. kNN Using caret R package. http://rstudio-pubs-static.s3.amazonaws.com/16444_caf85a306d564eb490eebdbaf0072df2.html

Kassambara, Alboukadel. 2018. Statistical tools for high-throughput data analysis. Model Selection Essentials in R. Penalized Regression Essentials: Ridge, Lasso & Elastic Net http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/153-penalized-regression-essentials-ridge-lasso-elastic-net/#ridge-regression

Kassambara, Alboukadel. 2018. Statistical tools for high-throughput data analysis. Classification Methods Essentials. Logistic Regression Assumptions and Diagnostics in R. http://www.sthda.com/english/articles/36-classification-methods-essentials/148-logistic-regression-assumptions-and-diagnostics-in-r/

Le, James. 2019. Data Camp: Logistic Regression in R Tutorial. https://www.datacamp.com/community/tutorials/logistic-regression-R

Mayer, Zach. 2018. caretEnsemble. Class predictions from ensemble classifiers are reversed. https://github.com/zachmayer/caretEnsemble/issues/189

Mayer, Zach. 2016. r-project.org. A Brief Introduction to caretEnsemble https://cran.r-project.org/web/packages/caretEnsemble/vignettes/caretEnsemble-intro.html

Pham Dinh Khanh. 2018. Caret Practice. https://rpubs.com/phamdinhkhanh/389752

Pierobon, Gabriel. 2018. Towards Data Science. A comprehensive Machine Learning workflow with multiple modelling using caret and caretEnsemble in R https://towardsdatascience.com/a-comprehensive-machine-learning-workflow-with-multiple-modelling-using-caret-and-caretensemble-in-fcbf6d80b5f2

Post, Justin. 2019. North Carolina State University. LASSO_Ridge_Elastic_net_-_Examples. https://www4.stat.ncsu.edu/~post/josh/LASSO_Ridge_Elastic_net_-_Examples.html

Prabhakaran, Selva. 2018. Machine Learning Plus. Caret Package – A Practical Guide to Machine Learning in R. https://www.machinelearningplus.com/machine-learning/caret-package/

The University of Cincinatti. 2019. UC Business Analytics R Programming Guide. Random Forests. https://uc-r.github.io/random_forests

Unknown. 2019. Model Selection. https://rstudio-pubs-static.s3.amazonaws.com/133416_8bc14091dac24831a1ad08c1b1d0e
html

Yihui Xie. 2014. Github. knitr-examples/077-wrap-output.Rmd https://github.com/yihui/knitr-examples/blob/master/077-wrap-output.Rmd