Classifying Phishing Websites Using Learned Predictive Models

Joshua Ying (jmy48), Jamie Wong (jmw488), and Arshi Bhatnagar (ab2248)

***Abstract*— Phishing is the act of acquiring private information by means of appearing as a legitimate service and has become rampant in the digital age, costing Internet users billions of dollars per year. Using a dataset from UCI, we generated models which classify a website as phishing or legitimate using various machine learning techniques. We used models including logistic regressions, decision trees, and random forests with a focus for generalizing well to testing data and minimizing false positive error. We penalize false positive (Type 1) errors more because it is more detrimental for a user to click on a phishing link that is classified as safe (false positive) than for users to click on a legitimate website that is classified as phishing (false negative). In the following, we will be exploring different classification techniques and explain how they perform on validation sets.**

I. INTRODUCTION

A phishing attack is a form of Internet fraud in which a hacker steals sensitive information by supplying a malicious link disguised as a reliable source. In recent years, hackers have expanded from Business Email Compromise (BEC) to social media phishing scams. According to Forbes, phishing scams cost American businesses half a billion dollars annually. Not only do these scams wreak financial havoc, but also they threaten sensitive data of everyday people. Hackers often exploit the vulnerability of the human factor using social engineering techniques. Our goal is thus to create a model that classifies a website as either phishing or non-phishing to prevent users from haphazardly entering seemingly innocuous, compromised pages.

II. DATASET: DESCRIPTION & CLEANING

Phishing URL data from UCI Machine Learning Repo[1]

[1] Mohammad, Rami, Thabtah, Fadi Abdeljaber and McCluskey, T.L. (2015) Phishing Websites Dataset. [Dataset] http://eprints.hud.ac.uk/id/eprint/24330/

A. 32 Features

Our dataset is comprised of features derived from the address bar, HTML and JavaScript, domain based features and other abnormalities of the website. -1 is phishing, 1 is non-phishing, 0 is suspicious. Most features are either ordinal $\{-1, 0, 1\}$ or binary $\{-1, 1\}$.

B. Feature Descriptions

Address Bar Features

1) Having_an_IP_address: Binary feature where 1 means the URL contains IP address -1 means does not. i.e. http://150.23.9.143/index.html.
2) URL_Length: Categorical feature, If char count $<$ 54 characters long then 1, if between 54 and 75 then 0, if $>$ 75 then -1.
3) Shortening_Service: Binary feature where -1 implies a service like TinyURL, bit.ly, etc was used and 1 is legitimate.
4) Double_slash_redirecting: Binary feature. If there exists a // within the URL path which redirects a user to another website then -1 else 1. For example, http://badwebsite//http://no
5) Prefix_Suffix: Binary feature. If exists $-$ in url then -1 else 1.
6) having_Sub_Domain: Categorical feature. If $>=$ one dot in the domain after www. then 1, else if there are 2 dots it is suspicious, else if there are more than 2 then -1.
7) Domain_registration_length: Binary feature. If domain expires in $<$1 yr then -1, since phishing websites have short lives. Else 1.
8) Favicon: Binary feature denoting the existence of a favicon, which is an icon linked to a particular page, for example Facebooks blue f icon.. If the favicon is loaded from an external page then -1. Else 1.

9) port: Binary feature. checks if ports are open. If more ports open, then -1 because hackers can run any service, else 1.
10) HTTPS_token: Binary feature. https is the more secure version of http. If https is in the domain name ex http://httpspotato.com then -1 since hackers try to disguise http as https, else 1.

HTML and Javascript Features

1) Website forwarding: Binary feature. . If page redirects less than one time then 1, else 0 meaning suspicious.
2) on_Mouseover: Binary feature. Use javascript to show fake URL status bar. If onMouseover changes status, then -1. Else 1.
3) rightclick: Binary feature. Javascript event is called event.button = 2. If right click disabled then -1 else 1.
4) popUpWindow: Binary. If PopUpWindow contains text fields -1, otherwise 1.
5) iFrame: Binary. iFrame is used to display an additional webpage. If using iframe,1, else -1.
6) DNS: Binary. No DNS (Domain Name Server) record: -1. Else: 1
7) Web_traffic: Categorical. Using the Alexa Database. If website rank $<$ 10,000 then 1, if $>$ 10,000 then 0 otherwise (no traffic) -1.
8) Page_Rank Binary. Greater page rank $\rightarrow$ more valuable webpage. -1 if page rank $<$ 0.2 else 1.
9) Google_Index: Binary. Indexed by google means shows up in search results. If indexed by Google: 1, else -1.
10) Link_pointing_to_page: Categorical. If no pages pointing to it then -1. If between 2 and 4 pages, then 0 (suspicious. If greater than 2 then 1.
11) Statistical_report: Using database like Phishtank to see if site is on it. If belongs to database then -1. Else:1

Other Abnormalities

1) Request_url: Categorical. If $>$22 $<$66 external objects within the page, then 0, if $>$66 external objects within page then -1, else 1.
2) SSLfinal_state: Categorical. If $\%$ URL of Anchor tag <a> $>$31$\%$ $\&$ $<$67$\%$, then 0, If $>$67$\%$ then -1, else 1.
3) Links_in_tags: Categorical. If $\%$ links in <Meta>, <Script>, <Link>, $>$17$\%$ $<$81$\%$ then 0, if $>$81$\%$ then -1, else 1.
4) SFH: Binary. Server form handler. If about:blank or empty or refers to different domain then -1, else 1.
5) Submitting_to_email: Binary. Server side script that sends inputted information to email. If using mail() or mailto then -1, else 1.
6) Abnormal_url: Binary. Data from WHOIS database. If hostname not part of URL then -1, else 1.

C. Cleaning the Data

For missing NA values, we imputed NAs with the mode value of each column. We chose this discretized method to maintain the binary/ordinal nature of our dataset and to ensure that our predicted $\hat{y}$ returns are binary. The mode of the column is the most likely value of any missing value and is hence a good guess.

Something of note about the dataset is that it is relatively small; this is a consequence for the amount of sophistication in the data itself. All features are already "classified" somewhat into ordinal values by hand, and given this kind of input our models can produce very accurate results (upwards of 90$\%$ accuracy).

D. Splitting Methodology for 5142 Observation

We shuffled the dataset to randomly separate 80$\%$ of the observations for training and 20$\%$ for testing. By convention the training/testing split should be 60-80$\%$/40-20$\%$, but our dataset is relatively small and therefore the sheer volume of data for the training portion needs to be adequate to reduce bias. Our test set has 1025 data points. We believe this splitting percentage worked well, since the out-of-sample error was roughly $\leq \pm$ 0.01 every shuffle.

All accuracy results provided in this report are the results of running the different models on this test set, which has not at all been touched by the models.

III. Methods and Models

Our goal was to perform binary classification on a website as phishing or non-phishing. Initially, we tried various classification methods with some default parameters like K-Nearest Neighbors, SVC, Decision Trees, Random Forest, ADA Boost, Stochastic Gradient Descent Classification, and Gaussian NB. These classification methods mostly performed decently with > 0.9 accuracy on the test set, except SVC and Stochastic Gradient Descent with hinge loss and L2 regularizer, which were the worst at 0.7 and 0.4 accuracy respectively. We decided to focus on tuning Logistic Regression, Decision Trees, and Random Forest since these provided an initial > 0.93 accuracy without parameter tuning. In addition, we felt that some of these techniques might have performed lower than they could because of their initial parameters. We also thus wanted to explore the effect of the parameters on the accuracy.

Initial Accuracy Scores:
note: We ran the classifiers on our training data with the default parameters to see which classification methods would be promising. We then followed up by fine tuning the parameters and finding the optimal results.

1) K-Nearest Neighbors: 0.932682
2) SVC(linear): 0.93268
3) SVC(gamma): 0.73560
4) Decision Tree: 0.945365
5) Random Forest: 0.95317
6) ADA Boost: 0.91512
7) Naive Bayes: 0.92097
8) Linear Discriminant Analysis: 0.926829
9) Quadratic Discriminant Analysis: 0.919024
10) Stochastic Gradient Descent (hinge loss, L2 penalty) : 0.4526829

A. *K Nearest Neighbors*

Reasoning for trying this model:
K Nearest Neighbors seemed to do well in our initial run, so we wanted to explore its parameters further to see if it can do better. It also makes sense that the answer to whether a website is phishy would be similar to other websites or neighbors that have very similar features.

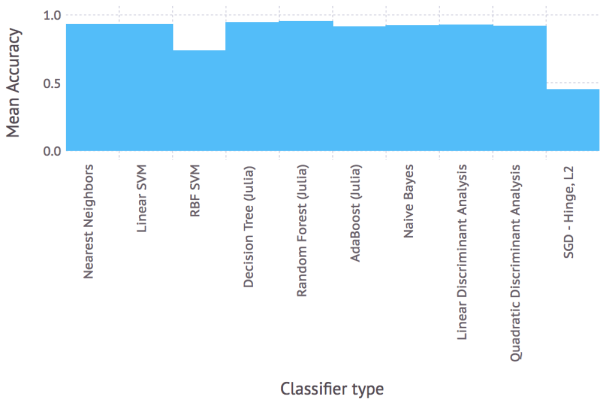


Fig. 1. initial mean accuracies on some default parameters

Parameter Exploration:
The KNN classifier allows you to specify the number of nearest neighbors you want to average over. Initially, this was set to 3. Let see how other values perform:

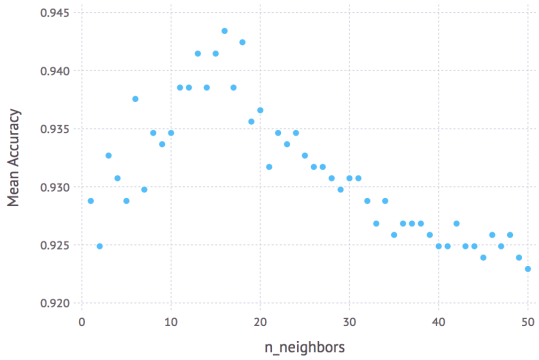


Fig. 2. neighbors vs mean accuracy

As seen by the above figure, the peak of the graph occurs at 16 neighbors with a value of 0.943 accuracy. This is fairly decent but is still below how well decision trees and random forest performed.

Verdict on KNN:
K Nearest Neighbors Classification still did not do better than decision trees and random forests, so it was not used for our final model.

B. *Linear SVM*

Reasoning for trying this model:
Linear SVM gave a 0.93 accuracy in the initial run. Its success made us want to explore its parameters further. The initial parameter used a C (penalty) factor of 0.025. Lets see how other C values change the results.

Parameter Exploration:
We tried out many different penalty values to get the following chart:

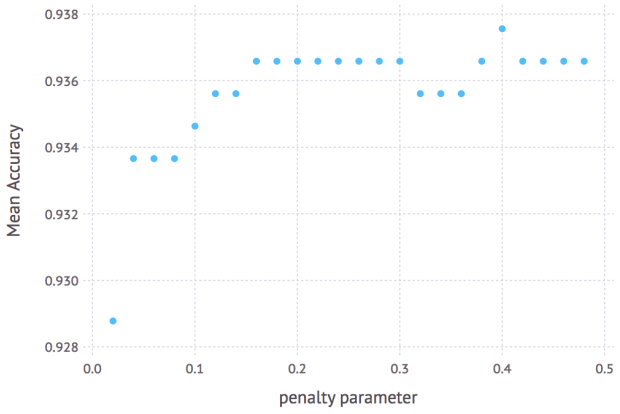


Fig. 3. penalty vs mean accuracy

As seen by the above figure, the peak of the graph occurs at a penalty parameter of 0.4 with a value of 0.9375 accuracy. This is fairly decent but is still below how well decision trees and random forest performed.

Verdict on Linear SVM:
Linear SVM still did not do better than decision trees and random forests, so it was not used for our final model.

C. *Logistic Regression*

With the goal of minimizing overfitting, we first ran Logistic Regression on the training data with the $l1$ regularizer and the $l2$ regularizer. Logistic Regression seemed promising since it finds values for the coefficients that minimize the error in the probabilities predicted by the model to those in the data

Reasoning for trying this model:

1) Logistic Regression finds the best fitting relationship between dependent (y $\in \{-1, 1\}$ and independent variables ($X \in R^d$ our features)
2) Wanted to test if sparsity was important for the model by comparing $l1$ and $l2$ regularization performance.
3) Wanted to gauge how well and how sure the model is by looking at the probabilities associated with each prediction.
4) Easily implemented using ScikitLearn package.

Cost function with $l2$ regularization:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

Cost function with $l1$ regularization:

$$\min_{w,c} ||w||_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

Determining C:
To find the value of C that maximizes cross validation performance for $l1/l2$ regularization, we ran the Grid Search Algorithm that runs cross validation using values of C from 0.1 to 2.0. For $l2$ regularization, the most reliable C value with the best cross validation score was C=0.1, for $l1$ it is C = 0.5.

Accuracy:
We measured accuracy by comparing each 1142 predicted $\hat{y}$ obs with the corresponding y_{test}. The accuracy is the percentage of the predicted $\hat{y}$ that match with y_{test}

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Fig. 4. TP = true pos, TN = true neg, FP = false pos, FN = false neg.

- Logistic Regression with $l1$ regularizer = 0.93505
- Logistic Regression with $l2$ regularizer = 0.93416

False Positive and False Negatives:
False positive occurs when y_{test} is negative but $\hat{y}$ is misclassified as positive. In other words, the website is a phishing website but the model did not classify it as one. False negative is when y_{test} is positive but $\hat{y}$ is misclassified as negative. In other words, the website is not a phishing link but the model classified it as phishing.

	l1	l2
false positive	0.037366	0.0418149
false negative	0.02758	0.02402

Fig. 5. False positive/negative chart for l1 and l2 regularization

Cross-Validation:
To evaluate the robustness of this model, i.e. whether it would generalize, we performed k-fold cross validation (ten times) for each Logistic Regression $l1$/$l2$ regularizer combination on our dataset and gathered summary statistics. The

	l1	l2
mean	0.935	0.9318
min	0.927	0.9259
1st quartile	0.932	0.9283
median	0.933	0.9306
3nd quartile	0.939	0.9359
max	0.943	0.9394

Fig. 6. Cross validation results

statistics show that $l1$ regularization performs slightly better than 12 in terms of generalizing to other datasets.

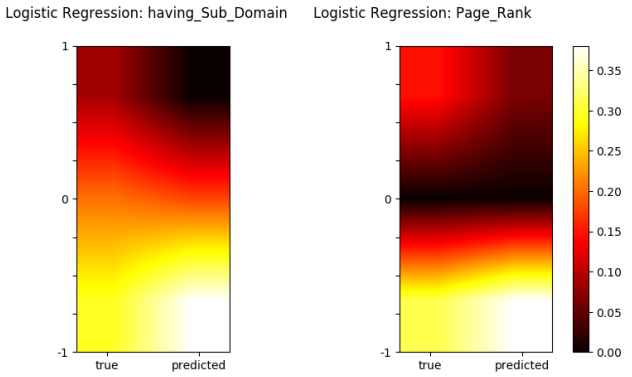
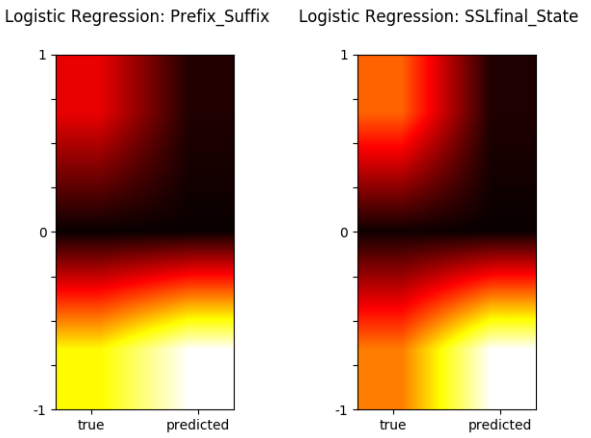


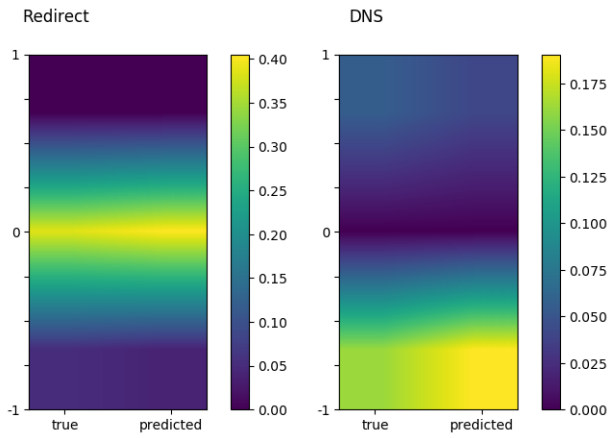
Fig. 7. Linear regression comparison: the left column of each plot corresponds to true classification and the right pertains to predicted. A brighter hue indicates more classified as phishing for that particular feature value (-1, 0, or 1).

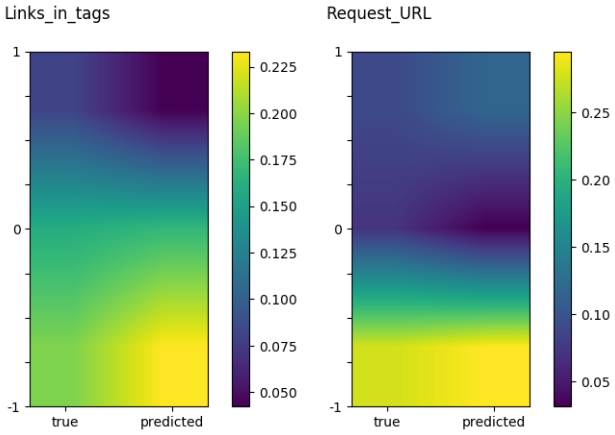
Verdict for Logistic Regression:
Running Logistic Regression with the respective optimal C values for $l1$ and $l2$ lead to $l1$ accuracy of 0.93505 and $l2$ accuracy of 0.93416. Even though the two accuracies are similar, $l1$ regularization appears to work slightly better. Our priority is to minimize false positive errors, which $l1$ outperforms $l2$ in at 3.74% false positive errors versus 4.18% false positive errors respectively. The cross validation score for $l1$ also outperformed that of $l2$ at a mean of 93.5% accuracy versus 93.18% respectively, meaning that using Logistic Regression with $l1$ regularizer is better for generalization.

Such a seemingly small difference between the $l1$ and $l2$ accuracy percentages (within $\pm$ 0.1 % accuracy of each other) is actually crucial since 0.1 percent difference in a huge dataset (5000 in our case) could lead to tens of hundreds of additional errors (50 in our case).

Feature Importance Evaluation:
As we previously mentioned, Logistic Regression is good for finding the relationship between the independent and dependent variables. We used this model to understand important and not-as-important features to understand what features should be prioritized in our attempt at other classification models. The good performance of $l1$ regularization suggests that a sparse model is possibly better, meaning that some features are less important to the model than others. We thus tried to remove features when trying other models







to see if it could improve the performance.
The $w_{logistic}$ was difficult to find because Julia's ScikitLearn's .coef_ function that exists in Python did not exist, so we used LowRankModels.jl and ran a proximal gradient on a logistic loss and l1 regularizer. We also checked that the accuracy was the same (both 0.93). The $w_{logistic}$ returned after convergence is as follows: [-0.0586168, 0.00572274, 0.028818, -0.480006, -0.176463, 0.99219, 0.708417, 1.48303, -0.16776, 0.16691, -0.140699, -0.158266, -0.255536, 1.44747, 0.240192, -0.0905416, 0.0952734, -0.250008, -0.402402, -0.40801, -0.525907, 0.174016, -0.12011, 0.278611, 0.218597, 0.453741, 0.225497, -0.183281, -0.367895, 0.0315023, 0.295577].

Possible Non-important Features

The coefficient closest to 0 is the second feature URL_Length, w[2] = 0.0057. Other possibly not as important features are the third feature Shortening_Service, w[3] = 0.028, and the thirtieth, Statistical_report w[30] = 0.031.

Important Features The most important feature is SSL_FinalState w[8] = 1.48302 and URL_of_anchor w[14] = 1.44747.

D. Decision Trees

Reasoning for trying this model:

The next model we tuned was decision trees, since in the initial classifier score comparison it was our second highest performer, receiving an accuracy score of 0.9453. We started by performing cross fold validation to compare whether the model could be better for generalizing than logistic regression. The mean score received was 0.953363, median = 0.956140, and min = 0.935. Compared to Logistic Regression with both $l1$ and $l2$ regularizers, the mean cross validation score for the Decision Trees model is at least 2% higher. Overall, decision trees did an excellent job in the initial run itself, which led us to believe that our data set is shaped as boxes. We thus wanted to spend more time exploring this model and its parameters.

Parameter Exploration:

To avoid overfitting, we are pruning the tree. Tree pruning takes in a parameter x and merges leaves having $= x$ percent combined purity. Hence, the lower this parameter, the less leaves the tree has and the more generalized it is. Lets see how the accuracy varies by this parameter:

As seen by the above figure, the algorithm performs its best when its parameter is above 0.55, which gives a 0.945 accuracy on the test set.

Cross-Validation:

To ensure that we are not having too much faith in

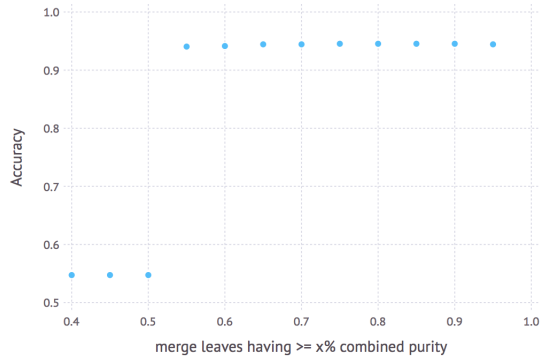


Fig. 8. parameter for merging vs mean accuracy

the accuracy result on the test set, we also decided to check the accuracy using cross validation with k=10. Here are the results:

1) 0.949318
2) 0.94347
3) 0.945419
4) 0.918129
5) 0.955166
6) 0.94347
7) 0.9375
8) 0.941406
9) 0.935421
10) 0.95499

Verdict on Decision Trees:
0.945 is one of the best accuracies we have seen so far. The cross validation scores also seem very good. It follows that we should now explore random forests more, since it combines several decision trees and can improve results of decision trees.

E. Random Forest

Reasoning for trying this model:
Random Forest is an improvement on decision trees by combining multiple trees. Since decision trees is our best model so far, we want to try to further improve its results. Thus, combining the results of several trees is the next logical step.

Parameter Exploration:
Random Forest takes in a parameter for the number of trees we want. In our initial run, we used 30 trees. Lets see if this can be improved.

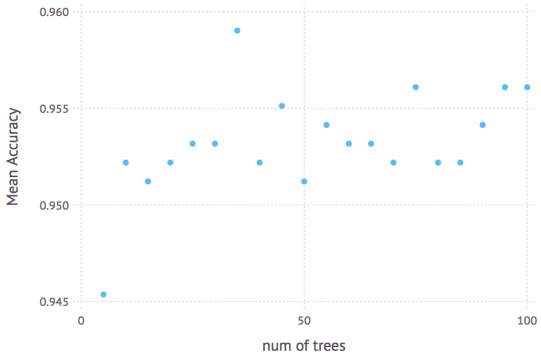


Fig. 9. number of trees vs mean accuracy on test set

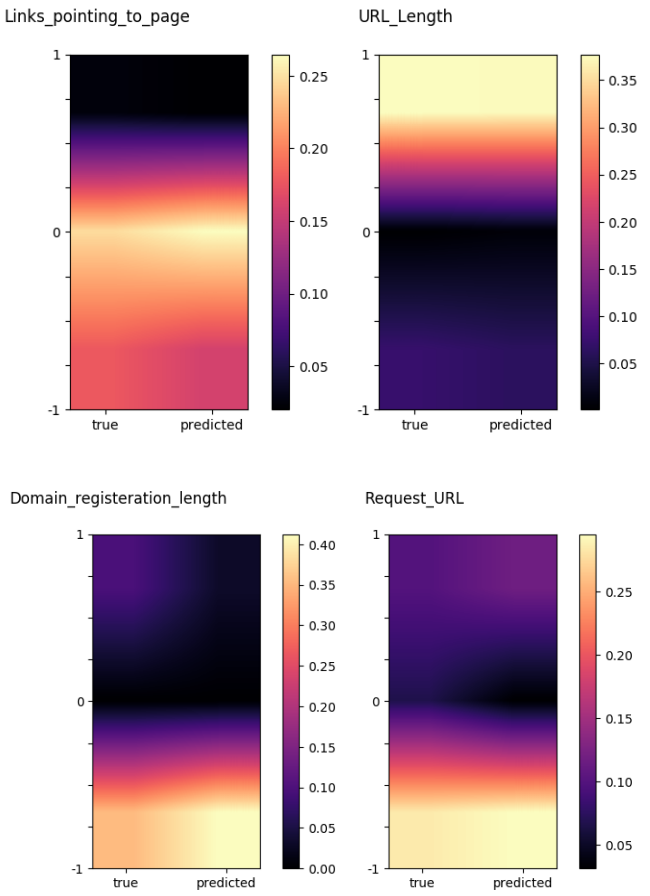
As seen by the above figure, the algorithm performs its best when its parameter is set to 35 trees, which gives a 0.9590 accuracy on the test set. This is slightly better than the initial run and is our best model.

We are using this graph to get a better balance on the bias variance tradeoff. We do not want too little leaves where we have too much bias, yet have enough leaves to have enough variance and not overly generalize. Pruning with different parameters allows us to find this balance.

Results on reduced matrix:
We also ran random forest on the reduced matrix, with the not as important columns taken out. We knew what columns to take out by the logistic regression with L1 regularizer. We took out columns with weights close to zero. By taking out these columns, we were able to slightly boost our accuracy to 0.9609756.

Cross-Validation:
To evaluate the robustness of our best model, i.e. whether it would generalize, we performed cross validation ten times using Random Forests using 35 trees, which we found to be the best. The mean is .953.

Verdict on Random Forests:
Overall, the cross validation gives a pretty good accuracy for every fold. This made us happy with the model. 0.9609756 accuracy on the test set was the best we were able to do out of all the models



we tried. This is the final model we used.

However, it must also be kept in mind that even though this gave the best accuracy, it penalizes false positives and false negatives the same. In fact, the number of false positive and false negatives were almost equal on the test set. Logistic regression allows us to penalize marking a unsafe website as safe more than marking a safe website as unsafe. In a lot of cases, this is better because it is better to be overly safe than to accidentally go on a fake website. We are choosing random forests due to its high accuracy, but the logistic regression also provides a very good model that is also safer.

IV. CONCLUSIONS

After exploring a variety of different classifiers, we discovered that Decision Trees and Random Forests performed the best. This result makes it likely that the data is best fit by rectangles. Random Forest results were improved slightly by reducing the data matrix based on the sparsity results of the L1 regularizer.

However, as stated before, logistic regression gave a very good model as well, which also allows us to penalize false positives and false negatives differently. Thus, this safer model can also be used as a final model.

A. *Confidence in our Model*

Our final model (random forest) gave an accuracy of 0.9609756 on the test set using the reduced data matrix. Since the test set was never touched by the model, this accuracy gives us a fair idea of how the model would perform out of sample. This gives us a good amount of confidence in our model. The cross validation scores also look good, which gives us even more confidence.

Our training set consisted of 4099 data points. This was a good amount of data to draw a model from. Our test set consisted of 1025 points, which once again was a good amount of data to represent the outside world. Once again, this gave us confidence in our model.

However, we do need to consider that technology is always changing and thus people are constantly developing new ways to create fake websites. The dataset that the model has been trained on is a couple of years old. This means that any new techniques that fraudsters are using today might not even be incorporated into the dataset. This fact makes us cautious of completely trusting the 0.9609 test set accuracy of the model and using the results in production.

REFERENCES

[1] Mohammad, Rami, Thabtah, Fadi Abdeljaber and McCluskey, T.L. (2015) Phishing Websites Dataset. http://eprints.hud.ac.uk/id/eprint/24330/.