Phishing Web Page Detection Using Optimised Machine Learning

Jordan Stobbs and Biju Issac
Computing and Information Sciences
Northumbria University
Newcastle-upon-Tyne, UK
{jordan.stobbs, biju.issac}@northumbria.ac.uk

Seibu Mary Jacob
Computing, Engineering and Digital Technology
Teesside University
Middlesbrough, UK
s.jacob@tees.ac.uk

***Abstract* - Phishing is a type of social engineering attack that can affect any company or anyone. This paper explores the effect that different features and optimisation techniques have on the accuracy of intelligent phishing detection using machine learning algorithms. This paper explores both hyperparameter optimisation as well as feature selection optimisation. For hyperparameter tuning, both TPE (Tree-structured Parzen Estimator) and GA (Genetic Algorithm) were tested, with the best option being model dependent. For feature selection, GA, MFO (Moth Flame Optimisation) and PSO (Particle Swarm Optimisation) were used with PSO working best with a Random Forest model. This work used URL (Uniform Resource Locator), DOM (Document Object Model) structure, page rank and page information related features. This research found that the best combination was Random Forest using PSO for feature selection and TPE for hyperparameter optimisation, giving an accuracy of 99.33%.**

Keywords: Phishing detection; Bio-inspired optimisation; Anti-Phishing; Optimisation

I. INTRODUCTION

Phishing is a social engineering attack which attempts to gather personal information or sensitive data such as usernames and passwords. using unsolicited e-mails with questionable web links that could direct users to fake websites. It is important for users to be aware of lookalike pages that are harvesting their information. Reference [1] found that 1 in 25 branded emails are phishing emails which shows how prevalent phishing can be. 76% of large enterprises experienced some kind of phishing attack in 2017 [2]. Universities have also been targeted within the last year with phishing campaigns specifically targeting students [3].

This paper explores meta data, URL structure and DOM structure features with both feature and hyper parameter bio-inspired optimisation. This is tested using two datasets taken ten months apart to check for robustness and the effects time has on the model. This paper also attempts to create a database with a large and adaptable feature set to ensure comparisons can be done in the future. This is an important area of research that could be implemented to recognise phishing pages and alert users or companies about them.

II. RELATED WORKS

There has been a lot of research work on phishing detection including content-based approaches to phishing detection [6], this is things such as looking at all the URLs on the page and determining if they are suspicious and checking if there are forms on the page and if they are looking for personal information. There are URL feature-based approaches [5] which look at items that are suspicious in the URL such as potential emails in a URL, number of sub domains, or the length of the URL. Page rank features [4] are another set of features that are used, this will be features which look at how search engines view the page, and if it appears on the front page or not. Alexa page rank is another popular feature in this area. Finally, data about the page such as certificate information [5], or the ASN (Autonomous System Number) of the page.

This paper combines features that have worked out well for others with the aim of getting higher accuracy rates. There are currently few that combines page information with URL, DOM structure and page rank features. There are no phishing detection algorithms that also optimise their model, at the time of writing this paper. Also, to the best of our knowledge there is no phishing URL database that stores data, and that allows for tests on different features.

III. ANALYSIS

A. Features

The right phishing page features are to be used for the phishing detection to work well. Table I gives a quick overview of the features that are used and analysed in this paper. In order to work out which features are likely to be significant; Z tests are run on numerical items using data from the dataset this paper uses. The null hypothesis will be that the feature has no impact on phishing detection and the alternate hypothesis will be that the feature does have an impact. This will return a Z score which is the number of standard deviations from the mean. This can be equated to a p-value which is the probability of the result assuming the null hypothesis is true. Typically, a null hypothesis is rejected if the p value < 0.05 as this shows the result is statistically significant. Table II shows the results of this analysis. Genuine mean is the average value of the feature for genuine URLs and phishing mean is the average value of the feature for phishing URLs

For the results to be statistically significant (and so worth being a feature) the P value should be less than 0.05. From the

results in table II, it is shown that five features can be dropped. Four of which are due to having too high P values, the other due to none of the URLs in the dataset containing the word dispatch.

TABLE I. LIST OF FEATURES EXPLORED

Feature type	Feature
URL	Domain length [5]
	Number of dashes (-) in the URL [4]
	Number of underscores (_) in the URL
	Number of at symbols (@) in the URL [4]
	Number of equals (=) in the URL
	Number of forward slashes (/) in the URL
	Number of dots (.) in the domain [4]
	Number of digits (0-9) in the domain)
	Uses HTTPS (true/false) [5]
	URL contains "http" after the protocol (true/false)
	URL contains an IP address (true/false) [4]
	Domain contains "www" (true/false)
	Domain contains common file extensions (css, php, html) (true/false for each)
	URL contains certain key words (e.g. log, pay) (true/false for each) [7]
	URL contains a certain TLD (top level domain) (true/false for each)
DOM structure	Percentage of links that do not lead to another page (link leads to "" or "#") [8]
	Percentage of links that lead to an external page. [4]
Page rank	Domain of page appears on the front page of Google (true/false) [4]
Page information	Country of origin of the page
	Number of requests
	Percentage of external requests
	Percentage of internal requests (same domain)
	Percentage of dead requests (4XX status code)
	ASN of the page
	Certificate issuer [9]
	Certificate duration [4]
	Hash of page appears on googles safe browsing blacklist

B. Machine Learning Techniques

This sub-section will look at 5 different machine learning techniques: linear regression, SVM (support vector machine), decision trees, random forest and neural networks. This paper looks at binary classification, either phishing or genuine.

Linear regression works by attempting to find a straight line that separates the two classifications using all the variables available. Everything above the line would be given one classification, everything below the line would be given the other. This sort of model works well where there is a linear correlation between the dependent variables and the independent variable. This means this would not work well for certificate issuers, TLDs and ASNs without adding a true or false condition for each ASN. This is something that can be automated.

TABLE II. ANALYSIS OF FEATURE SIGNIFICANCE

Feature	Genuine mean (2 s. f.)	Phishing mean (2 s. f.)	P value (2 s. f.) [a]
Domain length	8.2	11	~0
Dashes	0.087	0.88	~0
Underscores	0.021	0.36	~0
At symbols	0.039	0.00010	~0
Equals	0.034	0.57	~0
Slashes	0.99	3.1	~0
Dots	1.1	1.4	0.0000000010
Digits	0.13	0.37	~0
Http in domain (1=true, 0=false)	0.0021	0.023	~0
IP in URL (1=true, 0=false)	0.000050	0.020	~0
Www in domain (1=true, 0=false)	0.0035	0.18	~0
Css in domain (1=true, 0=false)	0.00040	0.00061	0.43
Php in domain (1=true, 0=false)	0.00030	0.00020	0.63
HTML in domain (1=true, 0=false)	0.00035	0.00071	0.17
Log in URL (1=true, 0=false)	0.0080	0.032	~0
Pay in URL (1=true, 0=false)	0.0034	0.0088	0. 0000000085
Web in URL (1=true, 0=false)	0.0067	0.065	~0
Cmd in URL (1=true, 0=false)	0.000050	0.00071	0.001
Contains account (1=true, 0=false)	0.00015	0.0023	0.0000000017
Contains dispatch (1=true, 0=false)	0	0	N/A
Contains free (1=true, 0=false)	0.0061	0.0077	0.093
Percentage of links to same page	0.030	0.19	~0
Percentage of links to external sites	0.49	0.36	~0
Certificate duration	200	9.2	~0

[a] ~0 means the result is less than 1e-10

SVM is similar in that a line is used to separate the data into both classifications. However, with SVM if a single line can be drawn to separate all the data, the target is to use the line with the maximum distance from the closest data point for each classification rather than using a line of best fit. SVM uses hinge loss to try and optimise the line drawn which essentially sums up the distance of incorrect classification data points to the drawn line. The lower the sum the better. Whether linear regression is better than SVM or not will

depend on the distribution of the data points as well as if any loss algorithms are used with linear regression or not. Both can also use polynomial lines rather than straight lines which may improve accuracy. A decision tree is simply a large amount of true or false statements. The machine learning comes in with how the true or false are organised as well as the value to split on. The nodes are calculated using information gain. The advantage of decision trees over the statistical models is that categorical data can be used meaning all ASN, certificate issuers and TLDs could be used as features without altering them to be all true or false answers during pre-processing, as this is done automatically by the decision tree algorithm. Less relevant features are also not used as much, so optimisation in this area is less relevant for accuracy reasons. The disadvantage is that decision trees are more susceptible to overfitting.

Random forests try and resolve the overfitting of decision trees. They do this by creating multiple decision trees that have random subsets of features. So, decision tree 1 could have number of dots and number of dashes, decision tree 2 could have number of digits in the domain and length of the URL, and decision tree 3 could have number of dots and length of the URL. The data used for prediction is passed to these trees, and a simple vote is done based on the output. So, if decision tree 1 and 2 classified the URL as genuine, whilst decision tree classifies it as phishing, the resulting prediction would be genuine. This lowers the chance of overfitting but does takes longer to classify than a single decision tree. The number of features supplied to each decision tree as well as the number of decision trees created are variables that can be changed and so could be optimised.

Neural networks are made up of many neurons. These are neurons are split into layers. There are three types of layers: the input layer, the hidden layer and the output layer. A neural network will have one input layer, one output layer, and one or more hidden layers. Generally, each neuron in the hidden and output layer is connected to every neuron in the layer next to it. This is called a fully connected network. Each connection has a weight, which can be positive if it has a positive impact on the result, or negative, meaning the impact is detrimental to the result.

Information flows through the network starting from the input layer, flowing through the hidden layer(s) and ending at the output layer. When an input is supplied to the network, it is multiplied by the weight of the connection it followed. This value is compared to the activation function. If the result is 0, the neuron is not activated. This can be common when using the rectified linear unit (ReLU) activation function, resulting in many neurons that are never activated. If at the end it is deemed to be a positive result (determined by the guess at the end) then each weight that contributed to the result is improved, relative to how confident of the correct result the network was, and how much the weight contributed. If it is deemed to be a negative result, then the opposite occurs. This effect is called backpropagation. This process is repeated multiple times, with weights adjusted as needed. This can be done with multiple epochs, which means running the data through the network again, but with the newly adjusted weights.

Neural networks are very different to the previously mentioned algorithms. The complexity can make it more intelligent; they are able to model non-linear and more complex relationships, but also make optimisation very difficult due to the large amounts of parameters that can be changed. When compared to random forests and decision trees, the process is very different. Whilst the trees calculate the best features at the beginning, the neural network gradually adjusts weights to try and find the optimal solution. The easiest way to compare them is through experimentation. The discussion in paper [10] shows that a neural network had the lowest error rate with a 7.50% error rate, with random forest and SVM being level at an 8.50% error rate. The random forest classifier resulted in a higher accuracy over a decision tree and a logistic regression classifier on their dataset [11]. RF was also found to have a higher accuracy than a decision tree by 3.05% using URL and hyperlink-based features [12]. RF works best when compared to linear regression, decision trees and SVMs using URL based features, beating decision trees by 2.14%, linear regression by 4.02% and SVMs by 5.18% [13].

This paper will investigate how well a neural network performs using a combination of the features defined in section II. This paper will also investigate SVMs and random forests, as well as optimised linear regression to see if it can achieve similar results to the other algorithms. An optimised approach will also be used for the neural network, SVM and random forest to see how they compare to their non-optimised versions as well as each other.

C. Optimisation Techniques

There are two forms of optimisation that will be useful within this project namely, feature optimisation and hyper parameter tuning. Feature optimisation simply removes the weaker features, leaving behind the strongest and as a result hopefully improving accuracy. Feature optimisation will be more useful for the non-tree algorithms. This is because, by design, trees focus on feature importance when they are created. It will, however, have an impact on random forests as there is less of a chance that any sub tree created is made up mostly/fully of weaker features. This will have more of an impact on SVM and linear regression as they both look at all available features which can sometimes be to their detriment if a feature is particularly weak.

Hyper parameter tuning makes changes to the machine learning algorithms themselves. Each algorithm has parameters passed to it that changes slightly how it works. For Random forests, this could be the number of features each tree has, and how many trees to create. For SVM and linear regression this could be the choice between a linear or polynomial function. This is mostly useful for getting minor increases in accuracy towards the end of the development cycle. There have been a few papers comparing algorithms for feature optimisation. From these papers three algorithms stand out. Genetic algorithm (GA), Particle swarm optimisation (PSO) and Moth flame optimisation (MFO).

GA works by initially creating a random population. The second step is to create a new population. This is done by computing a fitness score for each member of the population.

These scores are scaled into expectation values. Next is the breeding step. A number of children are created equal to the initial population, which is done by randomly selecting two parents from the population. The higher a member's expectation value, the more likely they are to be a parent. This means that members with higher expectation values will likely have more children than those with low expectation values or scores. The child is created by taking some values from parent A and some values from parent B. This is called a crossover. Next a slight random variation is applied called a mutation, which ensures that a variety of combinations are tested, rather than just a combination of values that were available in the first population. This process is repeated until there are a number of children and then the process starts again. This continues usually until a certain stopping condition is met which could be factors such as time or number of generations.

PSO is influenced by the flocking and schooling of birds and fish rather than the passing on of genetic material. Similar to the Genetic algorithm the initial particles are randomised, and this is PSOs equivalent of a random population. Unlike the Genetic algorithm, each particle (member) also holds a velocity value and a pBest score. The pBest is the particle's best accuracy. The particle with the highest pBest is also set as the gBest, which is the best accuracy across all particles. Under each epoch (generation), particles move closer to the particle with the current gBest. based on by how far from the gBest a particle is. This process is repeated until a number of epochs are met, or a certain threshold of movement is met, where movement is small enough that minimal further impact will be made.

MFO again has different names that will also be compared to similar items from the GA. Flames (members) are randomly placed on the search space, and a moth is assigned to each flame. On the first iteration, each moth is in the same position as the flame. Each moth's fitness score is calculated using a fitness algorithm. The flames are given the same fitness score as they are in the same position as the moth. Each moth will always follow the flame with the same fitness ranking. So, the moth that currently follows the best performing flame will always follow the best performing flame, even if that becomes a different flame. After the fitness score is calculated, the moth moves around the flame. On the second and subsequent iterations, the max number of flames is calculated using the following equation (1), where N represents the number of moths or the starting number of flames, i is the current iteration number and T is the maximum number of iterations.

$$\text{Max flames} = \text{round}\ (N - i*(N-1)/T) \quad (1)$$

Whilst GA combines the best features of good fitness results, MFO has a movement approach. Unlike PSO, MFO focusses less on convergence, at least in the beginning and more on exploring a larger area of the search space. MFO focusses on converging near a solution later on. This will make MFO more robust than PSO as there is a greater focus on exploration at the beginning. But if PSO has a good start then PSO is likely to have a higher accuracy. GA and MFO to be best across 18 datasets as in [16]. But PSO outperformed GA in some datasets [17]. Finally, PSO was found to have slightly higher accuracy over GA [18]. This paper will compare GA, PSO and MFO for feature selection.

For hyper parameter optimization, the paper [19] compared TPE (Tree-structured Parzen Estimator), MCMC (Markov chain Monte Carlo) and SMAC (Sequential Model-based Algorithm Configuration) optimisation techniques and found TPE to be around 4% more accurate than the other optimisers. There are few papers comparing different hyper parameter optimisation algorithms. As such, an investigation will be done on the algorithms that appear to be the most common, TPE and GA. TPE is a simpler model where the first defined number of iterations uses a random search. Then from the initial searches, the top 25% of combinations that give the best score are assigned to be the good parameter set. Whilst the bottom 75% is assigned to be a bad parameter set. Parzen-window density estimation is used to create a probability distribution for both of these sets. The 2 probability distributions are stacked on top and a search is done for a hyper parameter combination that is most likely to be in the good parameter set, and least likely to be in the bad parameter set. The hyper parameter set that is found to be the best is tested and the probabilities are recalculated.

IV. Methodology

A. Data collection

The data for this project was collected rather than taken from existing datasets since phishing pages can go down quickly. This means additional data is not able to be collected from the website and would limit the features that are able to be used. A larger database has been created alongside this research work to hopefully resolve this issue and provide all the information required for any potential phishing APIs developed in the future.

URLs were collected from two sources: PhishTank and Alexa. PhishTank gives the ability to fetch all active phishing pages via an API, whilst Alexa allows you to download the top 1 million pages in a CSV format. These URLs are all stored in the database. Overall, the dataset consisted of 20,000 URLs from Alexa and 10, 000 URLs from PhishTank. Extra information was obtained using Urlscan's API. The information was extracted from the JSON file and separated for storage in the database. The information retrieved from here was used for the majority of the non-URL features. The only exceptions were the two Google features (i.e. checking if the domain appears on the front page or checking if the hash of the page appears on Googles safe browsing blacklist). Also stored in the database is the ID of the scan which can be used to view the results of the scan and get all the information fetched including data not in the database such as the DOM structure of the page. In order to get the feature which checks if a URL is on Google's safe browsing blacklist or not, the Google safe browsing API is used. Figure 1 shows the ER diagram of the full database. In order to get the feature which checks if the URL's domain is on the front page of Google, first search terms are needed. The ten words used on the visible page (words within HTML tags, excluding script or stylesheet tags) with the highest count are used. This search term can then be used on Google and if the domain of the page

appears on the front page, this feature is marked as true for the URL, otherwise, it is marked as false.

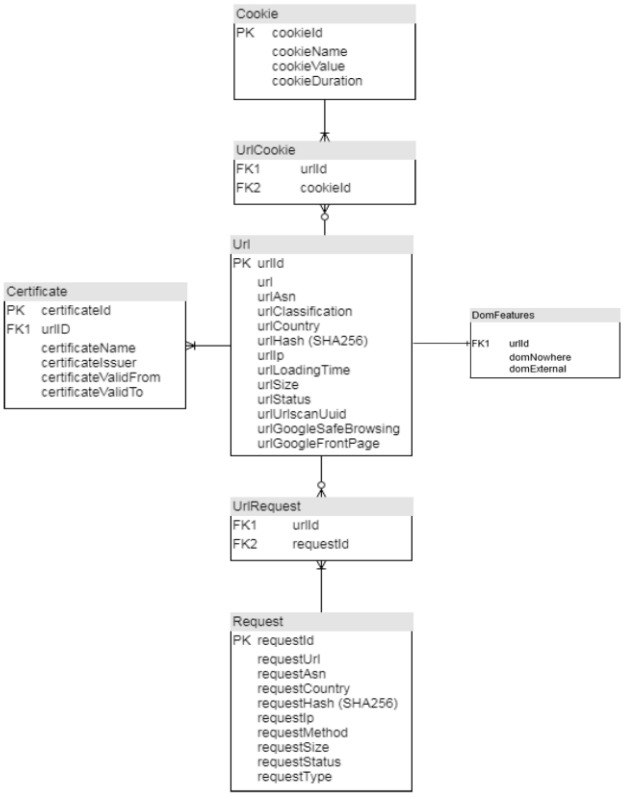

Fig 1. ER diagram of the full database

B. *Data manipulation*

Some of the features are made during run time which includes all the URL features, the number of requests and the certificate duration. The URL features are all obtained by either using a 'count' function or a 'contains' function, to count the number of times a certain character or string appears, or to check if the string or character does appear. A 'starts with' function is also used to check if the page uses HTTPS. Another function is also used to extract the TLD of the URL so it can be used in training. The number of requests is retrieved using a count function on the database, and the certificate duration is created by finding the difference between the start and end times of the certificates. All other features can be retrieved directly from the database.

C. *Machine learning and Optimisation*

Firstly, base models are created to see which performs the best with no optimisation and to ensure they work as expected. With no optimisation, the expected result is for Random forest to work best simply by observation of other experiments. As stated in section III, SVMs, Random forests, Linear regression and NNs, are all to be tested. Each is tested using 10-fold cross-validation and the average accuracies are recorded. After this hyper parameter optimisation will be used on all models to try and get slightly improved accuracy. Finally, feature selection optimisation will be used on the best performing models to check if certain models perform better after the removal of certain features.

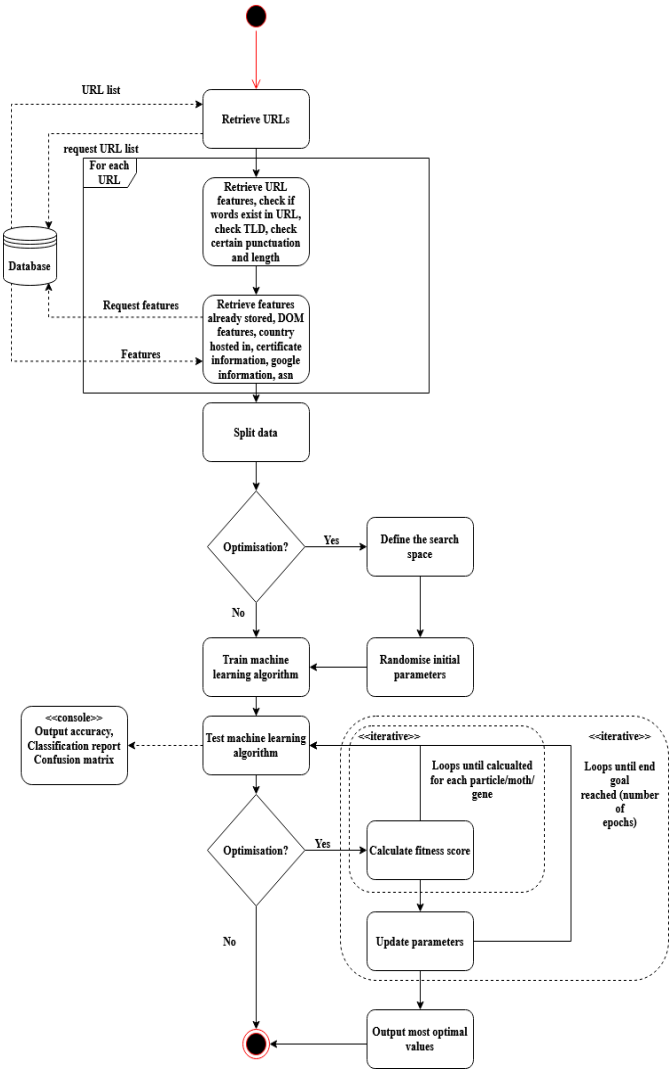

Fig 2. Activity diagram of data manipulation and machine learning process

This is expected to make more of a difference for the non-tree models because the tree models do this during their creation due to their important value scores. The accuracy after feature optimisation will be compared to the base model to see if feature optimisation is worth keeping for that model. Each particle/moth/member is given a feature mask, which holds information on the number of features used as well as the right features. Fitness for each algorithm is decided by accuracy. As stated in section III, GA and TPE will be used in hyper parameter optimisation and GA, MFO and PSO will be used in feature selection optimisation. Figure 2 shows a simplified activity diagram of this process.

V. Evaluation

This section shows and discusses the results obtained using the different machine learning algorithms. This includes unoptimised, optimised with hyper parameter optimisation and optimised with both hyper parameter optimisation and feature selection optimisation. It is seen from table III, Random forest is by far the most accurate model for both true positives and true negatives. Surprisingly NN is the worst for all accuracies.

This may simply because there are a lot more parameters that can be changed such as the number of neurons in each layer, or the percentage of neurons to drop out.

SVM as expected performed better than linear regression but was not better than Random forest. It was expected that both SVM and linear regression would underperform whilst all features are included. This is because features such as ASN, is categorical and so may confuse the algorithms. It is expected that both these algorithms will improve with feature optimisation, whilst a lesser impact will be had on random forest due to the innate importance values used to create the various trees.

TABLE III. ACCURACY OF UNOPTIMISED MACHINE LEARNING MODELS

Model	Accuracy (% 4 s. f.)
Linear regression	0.6789
SVM	0.8004
Random forest	**0.9603**
1 layer NN	0.5491
2 layer NN	0.6753
3 layer NN	0.6760

TABLE IV. ACCURACY OF MACHINE LEARNING MODELS WITH OPTIMISED HYPERPARAMETERS

Model	Accuracy (% 4 s. f.)
Linear regression TPE	0.9128
Linear regression GA	0.8065
SVM TPE	0.8063
SVM GA	0.8058
Random forest TPE	**0.9618**
Random forest GA	0.9582
3 layer NN TPE	0.6867
3 layer NN GA	0.6909

All the algorithms were optimised with both feature selection optimisation and hyper parameter optimisation, excluding the 1- and 2-layer neural networks (NNs) as it appeared that the extra complexity was helping this problem. It is seen from table IV that hyperparameter optimisation caused improvements all round, the exception being the Random forest combined with the GA resulted in a drop of accuracy. This could be because the GA had a poor start or because the population size was lowered in order for the model to be completed in a reasonable amount of time. Linear regression TPE had the greatest improvement, with an almost 0.24 accuracy increase. This may suggest that the initial starting parameters were a poor choice and should have been researched more before implementation. However, when compared with the GA optimisation for linear regression, it is still a 0.11 accuracy increase. So, it could simply be that the hyper parameters chosen, have a large effect on linear regression accuracy. On the overall TPE performed better than GA for hyper parameter optimisation. The exception to this was Neural networks which may be more suited to GA hyper parameter optimisation. Furthermore, TPE was much faster and so it would be recommended to use TPE over GA.

TABLE V. ACCURACY OF MACHINE LEARNING MODELS WITH OPTIMISED HYPERPARAMETERS AND FEATURE SELECTION

Model	Accuracy (% 4 s. f.)
Linear regression GA	0.9078
Linear regression PSO	0.9063
Linear regression MFO	0.9129
SVM GA	0.9026
SVM PSO	0.9056
SVM MFO	0.9093
Random forest GA	0.9882
Random forest PSO	**0.9933**
Random forest MFO	0.9928
3 layer NN GA	0.9118
3 layer NN PSO	0.9101
3 layer NN MFO	0.9187

It is seen from table V that with feature selection optimisation, accuracy was improved significantly across the board. Linear regression was the only exception to this with a minimal increase of 0.001% in accuracy meaning the features that were poor had minimal impact on the accuracy. Random forest was the only model where MFO was not the best optimiser. This was the case even with a second dataset made ten months later (10152 genuine sites, 13296 phishing sites). PSO outperformed the hyper parameter optimised model by 0.0315 and MFO by 0.0005. GA performed the worst of the three with a 0.9882 accuracy, this being behind PSOs by 0.9933. For Neural networks, there has been a large improvement over just having hyper parameters optimised. MFO performed with 0.9187 accuracy and it outperformed PSO by 0.0086 and GA 0.0069. On the overall, they did all improve the results quite significantly, with the worst scoring over the hyper parameter optimised model by 0.2195, and the best improving the results by 0.2281. So overall, on the dataset created in this project, Random forest with TPE hyper parameter optimisation and PSO feature selection optimisation worked the best. All features mentioned in section II were used and the main use of feature selection was to remove some of the features that were more split out, i.e. TLD for example.

For hyper parameters, the following were found to be optimal values: Criterion = ‘entropy’; N_estimators = 100; Max_depth = 100; Max_features = ”log2”; Min_samples_leaf = 1; Min_samples_leaf=4. This gives a maximum operational complexity of (10000N), the exact number would depend on the actual depth of the trees. The same results should be able to be obtained on the created dataset using the above hyper parameters and the feature set. Figures 3 to 7 shows the comparison of LR, SVM, NN and RF with different optimisation techniques for feature selection. Accuracy is the overall accuracy of the model, accuracy genuine is the accuracy of the model at detecting genuine sites, accuracy

phishing is the accuracy of the model at detecting phishing sites.

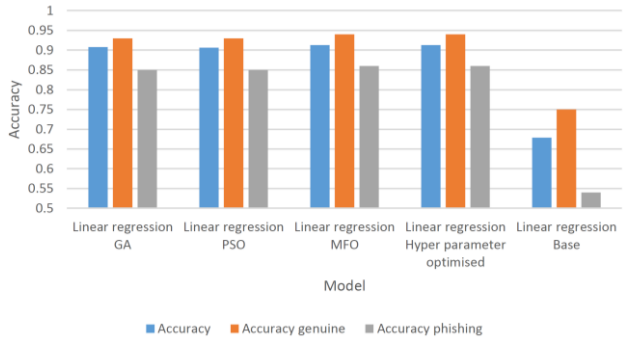

Fig 3. Comparing the accuracy of Linear regression with different optimisation techniques for feature selection.

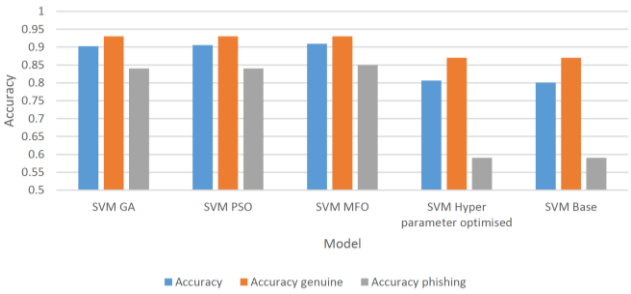

Fig 4. Comparing the accuracy of SVM with different optimisation techniques for feature selection.

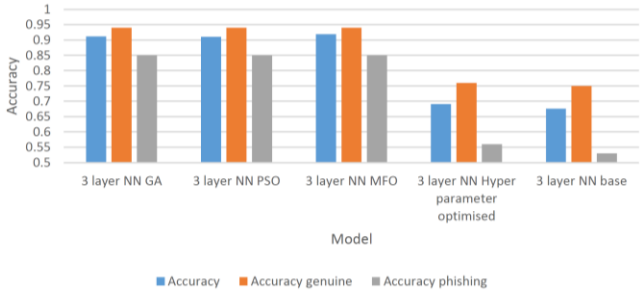

Fig 5. Comparing the accuracy of Neural Network with different optimisation techniques for feature selection

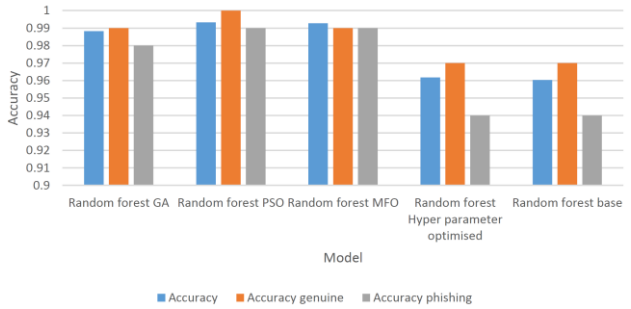

Fig 6. Comparing the accuracy of Random forest with different optimisation techniques for feature selection.

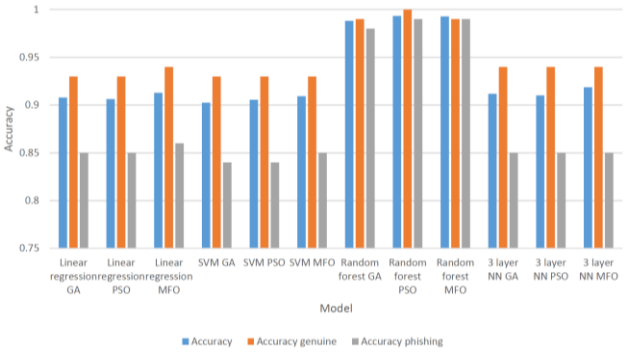

Fig 7. Overall comparison of the accuracy of four machine learning algorithms with different optimisation techniques for feature selection

As can be seen from the comparison table VI, the model presented in this paper is the best in terms of recall and accuracy and second in terms of precision.

VI. Conclusion and Future Works

Phishing is one of the most effective threats we face in our day. This research explored both hyper parameter optimisation as well as feature selection optimisation. For hyper parameter tuning, both TPE (Tree-structured Parzen Estimator) and GA (Genetic Algorithm) were tested, with the best option being model dependent. For feature selection, GA, MFO and PSO were used with PSO working best with a Random forest model. This work used URL, DOM structure, page rank and page information related features. We found that the best combination was a random forest using PSO for feature selection and TPE for hyper parameter optimisation, giving an accuracy of 99.33%. After multiple tests it was found that both hyperparameter and feature selection generally improve results. One option that was found in most successful models was the Alexa page rank which was not used in this work. The accuracy could be improved by implementing this and by doing more iterations of optimisations. Although it is likely that the increase would be rather small, any increase can make a large difference when the model is already at a high accuracy. It may also be worth investigating an optimised word focussed CNN combined with this feature set, as from other research, CNN seems to be promising. This could also be combined with an existing whitelist/blacklist database. The hash of the URL could be compared to the database, and if there is a match then the URL can be automatically predicted without having to go through a model. Finally, with the second dataset (collected 10 months later) it can be seen that phishing sites try to adapt to avoid detection.

TABLE VI. Comparison of Results with Other Works

Paper	Technique	Precision (% 4 s. f.)	Recall (% 4 s. f.)	Accuracy (% 4 s. f.)
[11]	RF using limited features	None given	None given	0.9658
[12]	Rule based learning system	0.9839	0.9819	0.9839
[6]	Fuzzy rough set feature selection	None given	None given	~0.9500

Paper	Technique	Precision (% 4 s. f.)	Recall (% 4 s. f.)	Accuracy (% 4 s. f.)
[20]	Fuzzy logic using mostly DOM and URL features	None given	None given	0.9146
[21]	Word focussed feature set, as well as search engine results	0.9844	None given	0.9491
[13]	RF with natural language features	None given	None given	0.9799
[22]	RF using principal component analysis and broad feature set.	0.9770	0.9859	0.9840
[14]	CNN combined with XGBoost using mostly URL features.	**0.9941**	0.9857	0.9899
[15]	CNN + RNN using word based features.	0.9791	None given	0.9791
This paper (first dataset)	PSO Optimised RF with broad feature set	0.9934	**0.9932**	0.9933
This paper (second dataset - 10 months later)	PSO Optimised RF with broad feature set	0.9869	0.9790	**0.9939**

REFERENCES

[1] Avanan, “How email became the weakest link,” [Online], 2019, Available at: https://www.avanan.com/how-email-became-the-weakest-link [Accessed 29 October 2019].

[2] Wombat, “State of the phish 2018,” [Online], 2019, Available at: https://info.wombatsecurity.com/hubfs/2018%20State%20of%20the%20Phish/Wombat-StateofPhish2018.pdf.

[3] C. Turner, “Universities are failing to protect themselves from cyber-attacks, report warns,” [Online], 2019, Available at: https://www.telegraph.co.uk/education/2019/04/03/universitiesfailing-protect-cyber-attacks-report-warns.

[4] M. Zabihimayvan and D. Doran, "Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection," 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), New Orleans, LA, USA, 2019, pp. 1-6, doi: 10.1109/FUZZ-IEEE.2019.8858884.

[5] A. K. Jain and B. B. Gupta, "Comparative analysis of features-based machine learning approaches for phishing detection," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 2125-2130.

[6] Y. Zhang, J. Hong, & L. Cranor, “Cantina: A Content-Based Approach to Detecting Phishing Web Sites,” Proceedings of the 16th international conference on World Wide Web - WWW '07, 2007, pp. 639-648.

[7] R. B. Basnet and T. Doleck, "Towards Developing a Tool to Detect Phishing URLs: A Machine Learning Approach," 2015 IEEE International Conference on Computational Intelligence & Communication Technology, Ghaziabad, 2015, pp. 220-223, doi: 10.1109/CICT.2015.63.

[8] Rao, R. & Ali, S., 2015. PhishShield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach. Procedia Computer Science, pp. 147-156.Z.

[9] Dong, A. Kapadia, J. Blythe and L. J. Camp, "Beyond the lock icon: real-time detection of phishing websites using public key certificates," 2015 APWG Symposium on Electronic Crime Research (eCrime), Barcelona, 2015, pp. 1-12, doi: 10.1109/ECRIME.2015.7120795.

[10] N. Sanglerdsinlapachai and A. Rungsawang, "Using Domain Top-page Similarity Feature in Machine Learning-Based Web Phishing Detection," 2010 Third International Conference on Knowledge Discovery and Data Mining, Phuket, 2010, pp. 187-190, doi: 10.1109/WKDD.2010.108.

[11] V. Patil, P. Thakkar, C. Shah, T. Bhat and S. P. Godse, "Detection and Prevention of Phishing Websites Using Machine Learning Approach," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697412.

[12] M. M. Yadollahi, F. Shoeleh, E. Serkani, A. Madani and H. Gharaee, "An Adaptive Machine Learning Based Approach for Phishing Detection Using Hybrid Features," 2019 5th International Conference on Web Research (ICWR), Tehran, Iran, 2019, pp. 281-286, doi: 10.1109/ICWR.2019.8765265.

[13] V. M. Yazhmozhi and B. Janet, "Natural language processing and Machine learning based phishing website detection system," 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2019, pp. 336-340, doi: 10.1109/I-SMAC47947.2019.9032492.

[14] P. Yang, G. Zhao and P. Zeng, "Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning," in IEEE Access, vol. 7, pp. 15196-15209, 2019, doi: 10.1109/ACCESS.2019.2892066.

[15] Y. Huang, Q. Yang, J. Qin and W. Wen, "Phishing URL Detection via CNN and Attention-Based Hierarchical RNN," 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 2019, pp. 112-119, doi: 10.1109/TrustCom/BigDataSE.2019.00024.

[16] H. M. Zawbaa, E. Emary, B. Parv and M. Sharawi, "Feature selection approach based on moth-flame optimization algorithm," 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, 2016, pp. 4612-4617, doi: 10.1109/CEC.2016.7744378.

[17] B. S. Saini, N. Kaur and K. S. Bhatia, "Performance Comparison of Particle Swarm Optimization and Genetic Algorithm for Feature Subset Selection in Keystroke Dynamics," 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 2019, pp. 684-689, doi: 10.1109/ICCIKE47802.2019.9004246.

[18] J. Huang, H. Li and J. Guo, "Application of PSO in the improved real-time evolution," 2011 11th International Conference on Hybrid Intelligent Systems (HIS), Melacca, 2011, pp. 318-323, doi: 10.1109/HIS.2011.6122125.

[19] C. Maurice, F. Madrigal and F. Lerasle, "Hyper-Optimization tools comparison for parameter tuning applications," 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, 2017, pp. 1-6, doi: 10.1109/AVSS.2017.8078499.

[20] H. Chapla, R. Kotak and M. Joiser, "A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier," 2019 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2019, pp. 383-388, doi: 10.1109/ICCES45898.2019.9002145.

[21] S. Marchal, J. François, R. State and T. Engel, "PhishStorm: Detecting Phishing With Streaming Analytics," in IEEE Transactions on Network and Service Management, vol. 11, no. 4, pp. 458-471, Dec. 2014, doi: 10.1109/TNSM.2014.2377295.

[22] I. Tyagi, J. Shad, S. Sharma, S. Gaur and G. Kaur, "A Novel Machine Learning Approach to Detect Phishing Websites," 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, 2018, pp. 425-430, doi: 10.1109/SPIN.2018.8474040.