Estimating the Phishing/Non-phishing Probability of a URL using a Tree-based Ensemble Model

Megh Agarwal, Ibrahim Khalil

Abstract— Phishing attacks are the oldest kind of cyber assault, yet continue to be the most common and a rising menace. A minuscule effort of social engineering on the attacker's part can result in a credible, tailored email that is deceitful enough for victims to fall for. With the skyrocketing number of phishing attempts, campaigns, and websites, this is a hugely escalating threat to cybersecurity. This study lexically examines the dangerous, deceptive links used in phishing attacks and focuses on building a model that provides the probability of a link being malicious based on common phishing link observations. Detection is solely based on the estimation of the probability (returned by the random forest classifier) and not classification thereby removing bias in classifying URLs.

Index Terms— Phishing, Random Forest, Decision Trees, Estimating probabilities, Phishing URLs, Machine Learning, Analysis

1 Introduction

Phishing is an insidious, fraudulent attempt, a deception that involves impersonating reputable companies, organizations, and/or persons to entice targets to give out sensitive information to non-legitimate sources. These tactics are a prevalent attack vector that has been employed for many years, with masses being aware of the potential ramifications and risks associated. It, however, still is a prevalent attack out there, infecting a large number of people worldwide. The brand trust is also, sometimes, leveraged to disseminate the phishing campaign by incorporating steps that force the target to share the content via the target's social network, thereby enhancing the effect and dissemination of the campaign. The repercussions of falling for this attack might be severe, ranging from identity theft to personal data breach to fund stealing to social consequences implicating individual targets and gaining privileged access to encrypted data or trade secrets, and corporate espionage at the corporate level, resulting in large financial losses, a drop in market share, and loss of customer/consumer faith. The emphasis of this research paper is on lexically analysing URLs. The URLs are split down into parts in order to examine them more deeply and find similarities based on specific categories. A Uniform Resource Locator(URL) comprises several components namely, orderly, protocol, subdomain(s), SLD, TLD, path, quer(y/ies) and fragment(s). Following typical structure is ubiquitous in any URL- (note all these components are used as features in the detection process)

protocol://subdomain(s).second-level-domain.top-level-domain/path?quer(y/ies)
\#fragment(s)
(The subdomain(s), quer(y/ies) and fragment(s) parameters are optional.)

The above components are used as features in the model.

2 Protocol

The first root protocol is (being) used to determine the probability of a URL performing phishing. The URLs are not being labeled as 'phishing' or 'non-phishing' since doing so could lead to prejudice. Outliers exist in every real-world circumstance, which could create a bias in the judgment. A URL might have a 90% probability of being a phishing URL—as predicted by the model—but it also has a 10% chance of not being a phishing URL. A carefully tailored URL could also be deceitful to the model. Hence, it is rather practical to offer the URL a 'chance' of being a phishing URL than to designate it as phishing or non-phishing. This eliminates the classification bias.

The 1st protocol decided for this research was simple but cumbersome. The protocol is based on decision trees (was used because of its advantages of considering both continuous and categorical variables, and forces the consideration of all possible outcomes of a decision and traces each path to a conclusion [1]) but does not rely completely on the same. The main idea of decision trees is creating a tree on which decisions are taken based on the input features. Now, the issue is one decision tree would return a binary output: 0, or 1. We need probability, as proposed in the first protocol, and for this, we need to create different decision trees with different samples, since decision trees are based on information gain and entropy, which is based on the input sample. Imagine a bunch of trees created with different samples and features. Once input is given, each tree will give a different output (mixture of true's and false's as the problem is a binary classification). For example, let's imagine 500 trees made from different samples and features. If we input "www.ulb.ac.be/scmero/", we get the label false i.e non-phishing from 400 trees. The rest 100 trees return that the URL is phishing. Getting the proportion, 100 / 500, we get the probability as 0.2 or 20%. This was based on 500 trees, and we got the proportion of trees that said 'True'. Note, this example considers the proportion of trees, whereas we will be finding the estimated probability given by the trees. The above example is just for explanation.

The random forest classifier was well in sync with the above discussed protocol thought for this research. It is a classification and regression issue that is an extension of bootstrap aggregation (bagging) of decision trees [2]. Hence, the Random Forest Classifier is used in this research.

The model is structured in two ways. Firstly, the model is created and tuned with good accuracy. After the creation of the basic model, further analysis, tuning, changes are made to the model concerning the estimation of probabilies to achieve the final protocol of the paper.

3 Features

There are many ubiquitously found aspects and parameters in URLs that could be indicative of phish suspicion. Eg. Length of the hostname, number of special characters, etc. However, it was important to determine the effective continuous and categorical-based ones that could be used in the model. Some chosen features had to be rejected due to difficulties in data collection. Even though many other features could be included, it could also lead to a flawed model. Looking at the URLs, an easily noticeable feature is the length of the hostname. Consider a URL like this: "www.ulb.ac.be/scmero/". The URL's host name's character count is small ("www.ulb.ac.be") compared to the character count of the hostname ("www.paypal.com.3.kaseg-spe.org") of a URL like "www.paypal.com.3.kaseg-spe.org/webscr/1/webscr.php?cmd= login-run" which is common in malicious links. Hence, continuous variables like the length of the hostname, the number of digits in the hostname, the number of special characters in the hostname, etc were included as features. Similarly, comparing the first URL's subdomain -[www]- to the second URL's -[www, paypal, com. 3]-, the second one appears to be more suspicious. Therefore, characteristics such as the number of subdomains, length of the subdomains, the length of the domain name, etc. could also be suggestive. Furthermore, we see that some URLs contain query parameters. Doing a quantitative analysis of the dataset, it was noticed that phishing URLs have numerous queries. Therefore, features like the number of parameters, the length of each query parameter, the number of special characters in the parameters, etc were also included. Similarly, it was noticed that phishing URLs consisted of multiple directories, which again is a crucial aspect of URLs; hence, features like the number of directories, special characters in those directories, etc, were also included. Subsequently, features like the number of fragments, the length of the fragments, etc were considered as well. A full list of these URL-based features is provided at the end of this section. All of these URL-based features are continuous and a component of the URL.

Considering the same URL- www.paypal.com.3.kaseg-spe.org, we can see that the 'paypal' brand is being impersonated. Phishers most pervasively disguise themselves as brands to persuade the victim to fall for the clickbait. Detecting these brand imbibed URLs can help the model judge phishing. Hence, detecting brand-based features like the presence of a brand name, the number of brands present in the URL, etc could be helpful. For this, a dataset of common brands was made. Along the way of this brand detection, a commonly used technique called typosquatting or URL hijacking was noticed. 'google.com' and 'goggle.com' have a minuscule difference; 'o' is replaced by 'g'. A user is very commonly in a state of placidity which persuades them to believe that they are in the right place. It was hence significant to detect if the URL is using the typosquatting technique which helps us judge phishing. For detecting this, we need to know which brand is the URL most similar to. This typosquatting detection was done in compliance with the same brands' dataset mentioned before. Now, to detect whether a URL is typosquatting, we need to measure the similarity index between the brand and the given URL. For this purpose, a similarity index metric called the Levenshtein distance was used. *The Levenshtein distance is a string metric that is used to compare two sequences. The Levenshtein distance between two words is the number of single-character edits (i.e. insertions, deletions, or substitutions) needed to change one word into the other* [3].

Using this metric, the similarity index of the two domain names is found. Mathematical analysis was done to determine a threshold (x), wherein if the similarity index is greater than x, we term it as typosquatting. The similarity index was calculated for the entire dataset. This calculation was done for both phishing and non-phishing URLs so that it can mathematically be checked whether it can be included in the final model as this would affect the final class probabilities. Once the similarity index was calculated, the data was split for phishing and non-phishing URLs and then was analyzed. The only problem considering the similarity index as a feature is that the index for both phishing and non-phishing classes do not differ much. In fact, the actual indices of the similarity index of both phishing and non-phishing overlap. This overlap can create a confusion for the model utliamately leading to poor results. This is clearly visual in Figure 1.

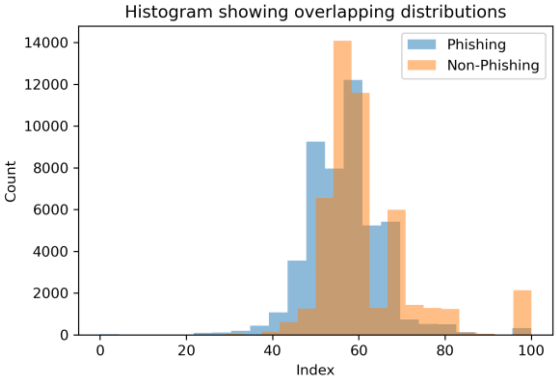

Fig. 1. Histogram showing overlapping distributions of phishing and non-phishing URLs

An overlap is seen to be existent in the two distributions using some visual analysis, but a proper test would explain whether the distributions are actually similar or not. Before completing the test, we need to first go back and realize why we are finding the similarity index. This similarity index could prove to be a

- *Megh Agarwal is an undergraduate student, class of 2026, at the University of Toronoto, Mathematical and Physical Sciences, Faculty of Arts and Sciense, St George Campus, Toronto, Canada, PH-8291851671. E-mail: agarwalmegh2004@gmail.com*
- *Ibrahim Khalil is a Second Year Junior College Science student at St. Xavier's College, Mumbai, India, PH-9820891974. E-mail: ibrahimkhalil.xaviers@gmail.com*
(*This information is optional; change it according to your need.)*

very important feature that could instantly provide a good way for the model to describe whether the URL is phishing or not. Therefore, we need to model this similarity index for phishing and non-phishing URLs. We need to check whether the indices are similar or not because if they are similar, it would only confuse the model. For checking this similarity between distributions, we completed the two-sample Kolmogorov-Smirnov test. *The Kolmogorov-Smirnov test tests whether two samples come from the same distribution. It can be used to compare two empirical data distributions, or to compare one empirical data distribution to any reference distribution* [4] The test gives us a statistic called D-statistic which would help us get the distance between the two samples. Let's imagine the sample s_1 belonging to the distribution having c.d.f $F(x)$. Sample s_1 represents the phishing URLs. Similarly, sample s_2 belongs to the distribution having c.d.f $G(x)$ and represents the non-phishing URLs. The null hypothesis is that both c.d.fs are equal. The alternate hypothesis is that both c.d.fs are not equal. If the p-value is less than 0.05, we can successfully reject the null hypothesis and consider both samples coming from different distributions. Else both come from the same distribution. If the alternate hypothesis comes true, it means that both the phishing URLs and non-phishing URLs' similarity indices come from different distributions which would make it almost impossible for us to include as a core feature. If both the samples were from the same distribution, it would have been easier for us to calculate the probability for that similarity index using the appropriate probability distribution. If both are from different distributions, it is almost impossible for us to differentiate between the two distributions as both have very similar similarity indices. The null hypothesis is $H_0: F = G$ and the alternate hypothesis is $H_a: F \neq G$ [5]. The D-statistic was 0.213. The p-value was 0.00. Since 0.00 < 0.05 we successfully reject that both samples are from the same distribution. The empirical cumulative density function is also plotted. (Figure 2)

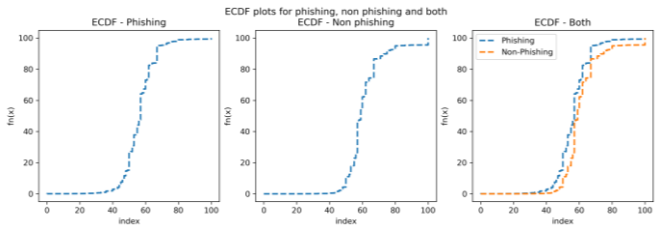

Fig. 2. ECDF plots for phishing, non phishing and both

The distance between the orange and the blue line is what the D-statistic is. Looking at the graph, we can visualize that the samples have different distributions. We can conclude now that we cannot include the similarity index as a core feature as it would in fact confuse the model. Due to significant similarity between the indices of phishing URLs and non- phishing URLs, we cannot differentiate between these continuous numbers for a new URL given to predict. Hence, the similarity index was not considered. Although the similarity index was not considered, it was necessary to include whether the URL is using the typosquatting technique or not. The typosquatting detection was built on this similarity index of URLs in compliance with brands, but the biggest difference between typosquatting and the similarity index was that typosquatting had higher numbers (85 - 95 index), while the actual similarity index model ranged from 0 - 100. Therefore, it was easier to model and detect whether the URL is typosquatting or not. The typosquatting histogram is shown in Figure 3.

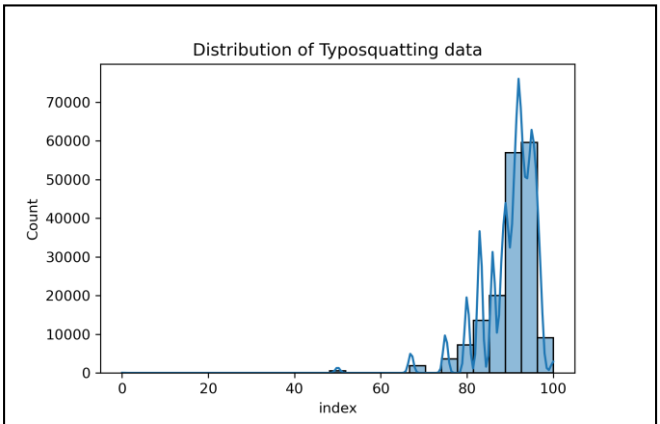

Fig. 3. Distribution of Typosquatting data

Notice that the distribution is left-skewed, therefore the Gaussian distribution cannot be considered. Moving forward, we need to determine a threshold value using which we can actually term a URL as typosquatting.

Let's say a URL's similarity index was found to be x. What threshold value k needs to be used to determine whether the URL with similarity index x can be termed as a typosquatting URL, if $x > k$.

To determine this optimum threshold, the mathematical technique or statistic used was the confidence interval of the population mean. This was chosen as the optimum threshold because if we would have chosen the confidence interval of the sample mean, there would have been biasing, affecting the model. Consequently, the estimation of the population mean was considered the best optimum threshold. Looking at the histogram of typosquatting similarity index, we see that most of the values lie in the 85 - 95 bin range. This means that the mean will also lie around 90, and our threshold will also lie around 85 - 95. To estimate the population mean, these requirements are to be fulfilled:

1. The sample is a simple random sample.
2. The sample size > 30 or the population is normally distributed.

In our case, both conditions are met. Therefore, we can use the following formula to estimate the population mean [6]

$$CI = \bar{x} \pm E \quad (1)$$

$$E = t_{\alpha/2}\left(\frac{s}{\sqrt{n}}\right) \quad (2)$$

$$\bar{x} - t_{\alpha/2}\left(\frac{5}{\sqrt{n}}\right) < \mu < \bar{x} + t_{\alpha/2}\left(\frac{5}{\sqrt{n}}\right) \quad (3)$$

Using this, the confidence interval of the estimated population mean was found to be 90.17 < 90.20 < 90.23.

Rounding off, we can conclude that the threshold value for terming the URL as typosquatting is 90. Although this is not a perfect value, it acts as a good estimation in the current situation.

Examining a URL, alongside URL-based features, we see that there are some web-based features as well. For example, the number of visits to a specific URL, the page's ranking, a WHOIS query, and so on. Unfortunately, due to the cumbersome data gathering, these external aspects are not included in this paper. Some URLs have page readers, whereas others do not. As a result, there would be a lot of missing data, prompting us to apply imputation techniques to eliminate the rows with missing values. This would have had a significant impact on the model's performance and, as a result, these features were rejected and excluded from the paper. The final chosen features are to be seen in table 1. Table 2 provides the purpose of each feature in the respective order of features given in Table 1.

TABLE 1
SELECTED FEATURES WITH THEIR TYPES

Features	Type
Hostname length	Continuous
Number of digits (Hostname)	Continuous
Number of special characters (Hostname)	Continuous
Subdomain length	Continuous
Domain length	Continuous
Number of subdomains	Continuous
Number of parameters	Continuous
Length of parameters	Continuous
Length of values of parameters	Continuous
Number of special characters in query	Continuous
Number of directories	Continuous
Length of directories	Continuous
Number of special characters in directories	Continuous
Number of fragments	Continuous
Length of fragments	Continuous
Length of values of fragments	Continuous
Brand found	Binary
Number of brands found	Continuous
Brand in domain	Binary
Brand in domain exactly	Binary
Is typosquatting	Binary

TABLE 2
PURPOSE OF EACH FEATURE

Purpose (Stores)
The length of the hostname of the URL
The number of digits in the hostname of the URL
The number of special characters in the hostname of the URL
The length of all subdomains of the URL
The domain length of the URL
The number of subdomains in the URL
The number of query parameters in the URL
The length of all query parameters in the URL
The length of all the respective values of the query parameters in the URL
The number of special characters in the query parameters in the URL
The number of directories in the URL
The length of all the directories in the URL
The number of special characters in the URL
The number of fragments in the URL
The length of all the fragments in the URL
The length of all the respective values of the fragments in the URL
Whether a URL consists of a brand in any of its component
The total number of brands found in the URL
Whether a URL consists of a brand in its domain
Whether the URL's domain is exactly same as the brand
Whether the URL is typosquatting or not

4 CREATION OF THE MODEL

In this section, a very basic model is created that could return required basic probabilities and/or binary outputs based on the provided input URL. This section completed the tuning of parameters to create a model that returns fine output.

Before tuning the parameters, it is important to consider the feature engineering aspect of the ML pipeline. For the current problem statement, much feature engineering was not needed as the data needs to be inputed into the model in its original form without any scaling, transformations, normizaltions, etc because the inputted feature numbers would be itself indiciative of the difference between phishing and non-phishing urls. Although these techniques were excluded, the check for imbalanced dataset was done.

The count of '0' or 'non-phishing' in the dataset is 48008, and the count of '1' or 'phishing' in the dataset is 47897. As there is not much difference between the count of the two, there is no need to fix the imbalanced dataset. Moving on, parameters like number of estimators, max depth, min samples split, min samples leaf, max features, bootstrapping are to be tuned so as to make the random forest model more reliable and accurate. The following process using cross-validation was followed to tune the parameters:

1. List value/values of each parameter (they are random, from the basic values to large values corresponding to the actual parameter range.).
2. Tune the model using RandomizedGridCV to get the values of the tuned parameters. These values are returned on the basis of a random check, wherein a random value from the list of values for each parameter is choosen. The model is created from this randomly chosen value of each parameter, following the calculation of the model's accuracy. This accuracy is compared to the previously created model by RandomizedGridCV, and stores the values of the parameter with the best accuracy. On each iteration, the RandomizedGridCV will choose a different combination of the features. Altogether, there are $17 * 14 * 16 * 15 * 3 * 2 = 342720$ settings. Note, the number of K-folds used were 10.
3. The final values returned from RandomizedGridCV is further split into values/ranges which is then fit into GridSearchCV. GridSearchCV unlike RandomizedGridCV would choose all provided values for each parameter and compare the accuracy to return the model with the best values of the tuned parameters.

The above process was followed because through RandomizedGridCV, the range for each parameter was narrowed down. Now that it's known about where to concentrate our search, every combination of settings to be tried can be explicity specified.

The following tables list the parameters and its values created originally, the values retrieved from RandomizedGridCV, splitting the values retrieved from RandomizedGridCV, final values of each parameter returned by GridSearchCV.

TABLE 3
PARAMETERS TO BE TUNED WITH THEIR VALUES

Parameter	Values
n_estimators	[1, 2, 4, 8, 16, 64, 100, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]
max_depths	[1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None]
min_samples_splits	[2, 5, 10, 20, 30, 50, 70, 100, 130, 150, 200, 300, 500, 700, 900, 1000]
min_samples_leafs	[1, 2, 4, 8, 16, 32, 64, 100, 200, 300, 500, 700, 800, 900, 1000]
max_features	['auto', 'sqrt', 'log2']
bootstraps	[True, False]

TABLE 4
TUNED PARAMETERS RETURNED BY RANDOMIZEDGRIDCV

Parameter	Values
n_estimators	200
max_depths	30
min_samples_splits	5
min_samples_leafs	1
max_features	'sqrt'
bootstraps	True

TABLE 5
PARAMETERS TO BE TUNED WITH NEW VALUES

Parameter	Values
n_estimators	[150, 200, 250]
max_depths	[25, 30, 35]
min_samples_splits	[3, 4, 5, 6, 7, 8]
min_samples_leafs	1
max_features	'sqrt'
bootstraps	True

TABLE 6
FINAL VALUES OF PARAMETERS

Parameter	Values
n_estimators	200
max_depths	30
min_samples_splits	7
min_samples_leafs	1
max_features	'sqrt'
bootstraps	True
random_state	42
oob_score	True

5 ANALYSIS AND EVALUATION

The accuracy of the model is 0.913. The out-of-bag error of the model is 0.09. Let us look at the confusion matrix (Figure 4).

Referring to the confusion matrix, we can point out that the model predicts 90% of the URLs as phishing correctly. Similarly, it predicts 93% of the URLs as non-phishing correctly. Let us calculate the sensitivity and specificity. Sensitivity is just the recall that was calculated before.

The specificity is 0.923, and sensitivity is 0.903. Evaluating a model based on both sensitivity and specificity is appropriate for most datasets because these measures consider all entries in the confusion matrix [7].

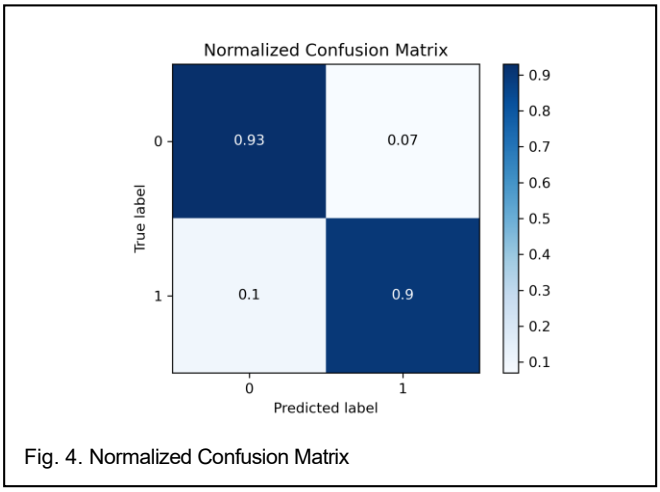

Fig. 4. Normalized Confusion Matrix

In this situation, since specificity is 92.3%, it implies that if a URL is classified as non-phishing, there is a 92.3% chance that the model will also classify the URL as non-phishing. Whereas, our sensitivity is 90.3%, which implies that given a URL as phishing, there is a 90.3% chance that the model classifies it as a phishing URL. Hence, our classifier does a great job at picking out the negatives, as well as the positives. While sensitivity deals with true positives and false negatives, specificity deals with false positives and true negatives. This means that the combination of sensitivity and specificity is a holistic measure when both true positives and true negatives should be considered. Sensitivity and specificity can be summarized by a single quantity, the balanced accuracy, which is defined as the mean of both measures [7].

The balanced accuracy is 0.913. This shows that the model is pretty good at predicting both true positives, and true negatives. Now let us analyze the ROC curve of the model. The ROC curve is shown in Figure 5.

In ROC curves, the true positive rate (TPR, y-axis) is plotted against the false positive rate (FPR, x-axis). Each point in a ROC curve arises from the values in the confusion matrix associated with the application of a specific cutoff on the predictions (scores) of the classifier [8]. The ROC curve is beneficial as it takes into consideration the entire confusion matrix. The closer the ROC to the top left corner, the better the classifier. To evaluate the ROC curve, the single metric AUC (area under the curve) can be calculated. The AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance [9]. The AUC is 0.97 for both classes. This shows that our classifier's performance is very good.

We will not be describing the model's performance now, as the different evaluation metrics like ROC, AUC, etc, gave us a good idea about the model's performance. According to the original protocol, we need to output the probability and not the classification. Hence, metrics like ROC won't prove much helpful as they should be used when you ultimately care about ranking predictions and not necessarily about outputting well-calibrated probabilities [10]. But these evaluation metrics gave us an overall idea of the model's performance for classifying the URL as phishing or not. Therefore, in

a situation wherein classification is the root protocol, this model can essentially be used after some minute further modifications.

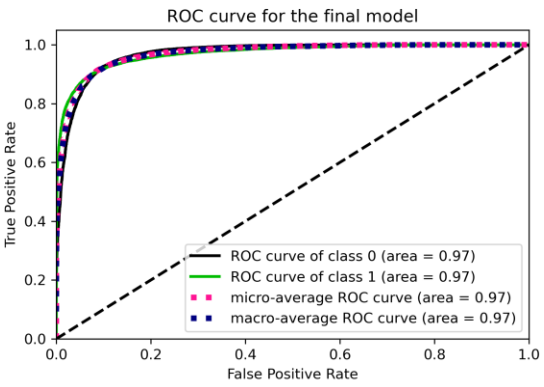

Fig. 5. ROC curve for the final model

6 ESTIMATING PROBABILITIES

As stated earlier our main protocol included the model giving us the probability of the URL trying to phish, rather than the actual classification. Even though the classification is based on the predicted probability, it is biased as discussed before. There are multiple ways using which we can predict the probability i.e *Average voting, Relative Class Frequency, Laplace estimate, The m-estimate* [11]. One important concept to understand is that these methods are better for trees known as probability estimation trees. *Probability estimation trees are nothing but the standard decision tree algorithm, and running a bunch of data through it and counting what portion of the time the predicted label was correct in each leaf* [12]. Fortunately, the way we implemented random forest in scikit-learn had us create ensemble trees that have a function called predict_proba which follows the concept of probability estimation trees. Now, we need to understand the methods in our context and finalize the way of estimating the probabilities.

The first method, average voting, defines a class probability distribution by averaging the unweighted class votes by the members of the ensemble, where each member votes for a single (most probable) class [11]. More detailed information on this method is provided here: [11]. *The second way, Relative Class Frequency, which is a class probability distribution by averaging the relative class frequencies of the members of the ensemble* [11]. The other two ways are very similar to relative class frequency. Based on the conclusion of [11], we will be focusing on the two methods: average voting, and relative class frequency. The reason is that the last two estimates are based on the relative class frequency, and relative class frequency outperforms those two estimates. Therefore, we will be analyzing, and evaluating the first two methods.

For average voting, we need to determine an optimum threshold that could classify the results and then can be used to estimate the probability. This optimum threshold determination is calculated in Section 7. The optimum threshold is 0.506451. Based on this optimum threshold, we can classify the results in '0' and '1'. Note: relative class frequency is the same implementation which is done in scikit-learn's predict_proba for random forest classifier. predict_proba for random forest takes the mean of the class predicted probabilities given by each decision tree in the forest. The class predicted probabilities given by each decision tree are the fraction of samples of the same class in a leaf [13]. We will try to evaluate these two ways and decide which one would be the best in our case. We will do a statistical significance test, as well as an analysis of other metrics.

We'll undertake a visual examination first, including a basic study of some assessment metrics linked to the estimation of probability, before going on to a more statistical decision. After we've completed the visual analysis, we'll look at the statistical significance of the observed difference in error caused by average voting and relative class frequency. We'll look at the area under the curve, the brier score, and the reliability curve for now.

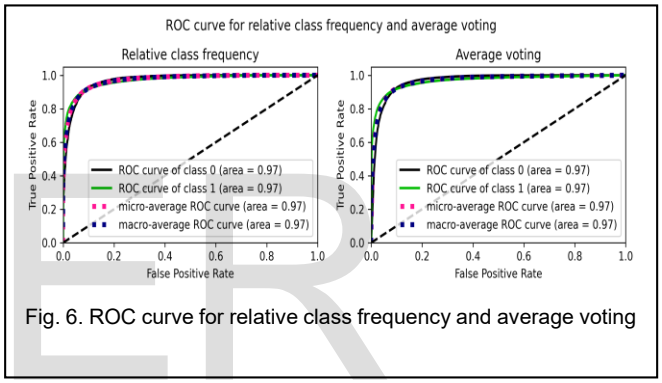

Fig. 6. ROC curve for relative class frequency and average voting

In figure 6, the first graph shows the ROC curve for relative frequency, while the second shows average voting. The area under the roc curve for relative frequency (0.97) is the same as the average voting area under the curve. Even when the trade-off between the false positive rate and the true positive rate of both methods is considered, it can be concluded that both strategies perform wonderfully. When the false positive rate is 0.0, the true positive rate is roughly 0.5 for both approaches. When the false positive rate rises, the true positive rate rises as well, reaching around 0.9, the largest rate of change in the entire function. This suggests that the true positive rate would be 0.9 at a false positive rate of 0.1. This would be balanced since, in order to reduce the false positive rate, the true positive rate must also be increased sufficiently. As a result, the final tradeoff would be 0.1 false positive rate and 0.9 true positive rate. Both methods perform admirably, according to this analysis. As a result, various methods of evaluation are required to establish which method is superior. Moving on to the Brier score, it is the mean squared error between predicted probabilities and the expected values. The score summarizes the magnitude of the error in the probability forecasts [14]. For average voting, the brier score loss is 0.067, and for relative class frequency, it is 0.066. It's clear that the error isn't significant, and both error values are fairly close to one another. In general, the brier score makes an essential point concerning probability. Between the anticipated

probability and the expected values, the mean squared error is modest. This is the error in probability predictions in general, and both methods have extremely low errors. As a result, the brier score reflects the accuracy of both systems' probability estimates. The reliability curve is the last visualisation we'll look at. The reliability curve for both methods is shown in figure 7.

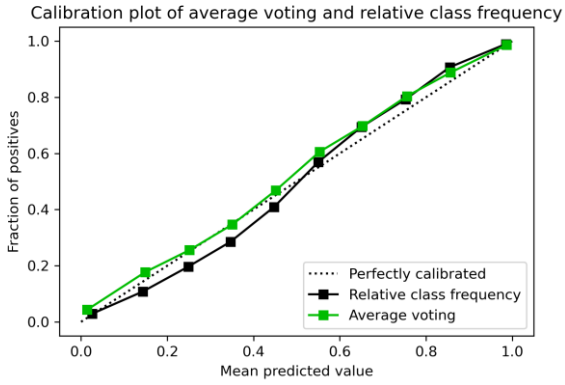

Fig. 7. Calibration plot of average voting and relative class frequency

Both methods are near to properly calibrated predictions, according to the calibration plot or reliability curve shown above. In both directions, there are occasional overestimations and underestimations. When compared to the calibration plot for relative class frequency, the average voting calibration plot is better. This behaviour is intriguing because relative class frequency underestimation is higher than average voting. Because the probabilities pertaining to that particular bin range may have an average that is closer to the fraction of positives, average voting may have smaller underestimations than relative class frequency.

Because the calibration plots show the mean predicted probability of a bin range in relation to its fraction of positives (the proportion of samples in a bin range that belong to that positive class), it can be concluded that average voting has a mean predicted probability that is closer to the fraction of positives for the bin range 0.3 - 0.5 when the total number of bins is 10. This refers to the predicted probability provided by average voting in general vs relative class frequency for bins ranging from around 0.3 to 0.5. From the 0.3 to 0.5 bin range, average voting has superior probability estimates, however higher bin ranges have estimates close to that of relative class frequency. The model would be completely calibrated if the fraction of positives and the mean predicted probability were equal. A general grasp of both approaches' probability estimations can be gained by analysing the calibration plot. In reality, as compared to relative class frequency, average voting performs slightly better, according to the plot.

Despite the fact that this is a visual analysis, a statistical test will help us determine which method is superior. Calibration errors are used to compare the two methods in a statistical test. The statistical test is done for the following errors:

1. Estimated calibration error (ECE) - The expected calibration error measures the expected difference between accuracy and confidence by grouping all samples [15].
2. Maximum calibration error (MCE) - The maximum calibration error is the maximum difference between accuracy and confidence overall bins by grouping all samples into K bins [16].
3. The average calibration error (ACE) - The average calibration error measures the average difference between accuracy and confidence [17].

The table with the errors, and the statistical test results is shown in Table 7.

TABLE 7
STATISTICAL TEST RESULTS FOR RELATIVE CLASS FREQUENCY AND AVERAGE VOTING

Error	t-statistic	p-value	result
ECE	-1.85643	0.122532	H_0
ACE	0.249772	0.812701	H_0
MCE	1.32088	0.243764	H_0

Note, the statistical test was completed using the 5x2cv paired t test. *"The 5x2cv paired t test is a procedure for comparing the performance of two models"* [18]. This is how the statistical test works:

Let the two methods be A and B. In this statistical test, the dataset is split into training and testing data, each having 50% of the data from the original dataset. This splitting is done 5 times, hence there are 5 iterations. In each of the iterations, the A and B are fit to the training data and the error is calculated based on the testing data predicted by the trained model. After the error is calculated, the training and testing data are reversed (previous training data becomes new testing data and vice versa). Therefore, there are two errors:

$$e^{(1)} = e_a^{(1)} - e_b^{(1)} \quad (4)$$

$$e^{(2)} = e_a^{(2)} - e_b^{(2)} \quad (5)$$

After the error estimates are calculated, the estimated mean and variance is calculated. The estimated mean is

$$\bar{e} = \frac{e^{(1)} + e^{(2)}}{2} \quad (6)$$

and the estimated variance is

$$s^2 = \left(e^{(1)} - \bar{e}\right)^2 + \left(e^{(2)} - \bar{e}\right)^2 \quad (7)$$

This variance is calculated for 5 iterations and then the t statistic is computed:

$$t = \frac{e_1^{(1)}}{\sqrt{\left(\frac{1}{5}\right) * \Sigma_{i=1}^{5} s_i^2}}$$

(8)

Here $e_1^{(1)}$ is $e^{(1)}$from the first iteration. The degrees of freedom is 5 and considering this value with the t-statistic, the p-value can be calculated. Once the p-value is calculated, it can be compared to the significance level which is 0.05. If the p-value is greater than 0.05, the null hypothesis cannot be rejected, else it can be rejected. Therefore, $H_0 =$ A and B are not statistically different. $H_q =$ A and B are statistically different.

The two probability estimating methods are not statistically different when considering the three error estimates, according to the findings of the statistical test shown in table 7. Let's look at the system's binary cross-entropy for more information. It will be possible to see how the goal probability distribution and the estimation probability distribution differ by examining the cross entropy (indirectly giving an idea about the estimations). The greater the difference in the distribution, the higher the cross entropy. The cross entropy of the model that predicts random probability will be considered when comparing cross entropy. In the case of binary classification, this random probability distribution will show how effectively the estimated probabilities of the stated methods defer to this random probability distribution. Our dataset is balanced, and hence the probability mass function of the labels of our dataset is

$$a_i = \frac{1}{2} \quad (9)$$

Logically, any random prediction won't have any discriminative power on average. Therefore, on average the predicted probability assigned to any observation would be

$$b_i = \frac{1}{2} \quad (10)$$

Where n is the number of classes. Therefore, the average cross entropy for a random prediction would become:

$$H[a,b] = -\sum_{i=1} a_i \, log(b_i) = -2 \cdot \frac{1}{2} \cdot log\left(\frac{1}{2}\right) = log(2) = 0.69$$

(11)

A random probability distribution's average cross-entropy is 0.69. The average voting average cross-entropy is 0.295. When the average cross entropy of average voting is compared to the average cross entropy of the random distribution, it is evident that average voting's estimated probability are superior. The important thing to note here is that, while average cross entropy for average voting is better than random probabilities, the discrepancy between the target and estimated probabilities distributions is significant. This illustrates that some of the average voting probabilities deviate from the desired distribution, causing the average cross entropy to rise. Note, since it is a binary classification problem, our target distribution is a Bernoulli random variable where $X \sim$ Bernoulli(p). A plot of cross entropy and the probability distribution for a specific event can be used to visualise the divergence from the target distribution. The target distribution is [0.0, 1.0], whereas the estimated probability distribution is the probability calculated using any of the methods for a random event X. The cross entropy between this target distribution and the estimated probability distribution will be determined. For all estimated probabilities obtained by average voting, this will be plotted. Only the target variable 'phishing' is shown for illustration purposes.

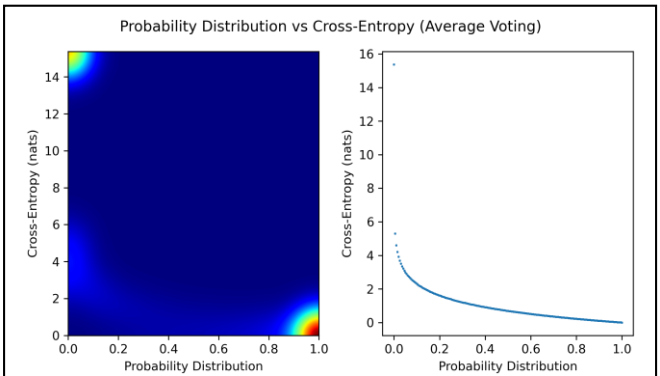

Fig. 8. Probability distribution vs cross entropy (average voting)

Figure 8 consists of two plots. The first plot shows the heatmap of its respective scatter plot which is shown in the 2nd plot. The scatter plot depicts the mentioned relationship between the probability distribution and its cross entropy. The relationship is negative. The cross entropy decreases as the probability distribution approaches the target distribution. It also achieves a cross entropy of 0 when the target distribution and the calculated probability distribution are identical. This assertion is logical since the cross entropy reduces or becomes zero as the predicted probability for the phishing class approaches or equals 1.0. The scatter plot depicts the relationship between calculated probabilities and average voting's cross entropy. It also demonstrates that some probabilities are diametrically opposed to the intended distribution. The cross entropy in this scenario is greater than 15. The heatmap represents the scatter plot but shows how many points lie in which range. The more the area is red, the more data points lie in that area and vice versa. The heatmap shows that most of the data points lie in the 0.9 to 1.0 region which corresponds to the target distribution, [0.0, 1.0]. Note, 0.9 - 1.0 region represents the 1.0 of the target distribution, and if we subtract it from 1.0, the 0.0 aspect of the target distribution can be retrieved. Logically, the graph, if plotted for non-phishing, should be a reflection of the x-axis of phishing. The heatmap also shows that many data points lie in the 0.0 - 0.1 range, which is just opposite of the target distribution, leading to an extremely high cross entropy. This is why the average cross entropy of average voting is high, since many probabilities are just opposite to the target distribution.

Now let us compare relative class frequency's average cross entropy to random probability and average voting. The average relative class frequency cross entropy is 0.221. The method has very good probabilities when compared to random probabilities. The probabilities provided by relative class frequency are superior to those provided by average voting. A cross entropy

vs probability distribution map can be used to see why relative class frequency has a lower average cross entropy.

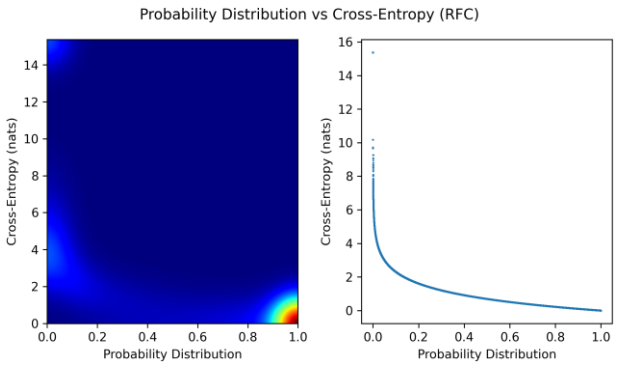

Fig. 9. Probability distribution vs cross entropy (RFC)

According to the scatter plot in figure 9, the probability distribution and cross entropy have a negative relationship once again. The trend is pretty similar to average voting, but the most notable difference is that the heatmap shows very few data points in the 0.0 - 0.1 range. Because there are few points that stray from the desired distribution, the average cross entropy for relative class frequency decreases.

Overall, based on cross entropy and the previous study, it is obvious that relative class frequency has better probabilities. Moving forward, the average cross entropy of relative class frequency is still high. This indicates that the probabilities are not adequately calibrated, and that more work needs to be done to produce accurate probability estimates. Probability calibration on the random forest classifier can be done in this scenario. As a result, probability calibration is used to make even more improvements. Two methods are used to calibrate the probabilities: Sigmoid and Isotonic Regression. In the case of probability calibration, both methods are dominant [19]. Because relative class frequency provided superior estimates, it was used for further calibration, and average voting was not taken into account.

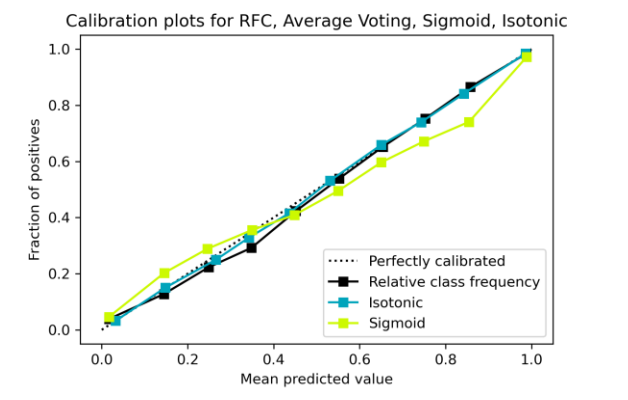

Fig. 10. Calibration plots for RFC, Average Voting, Sigmoid, Isotonic

According to the calibration plot in the figure 10, probabilities after calibration with an isotonic regressor are clearly superior to relative class frequency. In comparison to relative class frequency, there are extremely few under and over estimations for isotonic, implying solid probability estimates.

Calibration plots, as previously mentioned, display the mean predicted probability vs. the fractions of positives, demonstrating that the isotonic probabilities are close to the real predicting class. Due to the large amount of samples, Sigmoid could not perform better (in fact, it performs far worse than relative class frequency). "Isotonic' will perform as well as or better than 'sigmoid' when there is enough data (greater than ~ 1000 samples) to avoid overfitting" [19]. The logistic model (sigmoid regressor applies a logistic model) also assumes the output of the classifier to be normally distributed with the same variance. The probabilities are bimodally distributed when the relative class frequency is used to determine the probability (shown in figure 11).

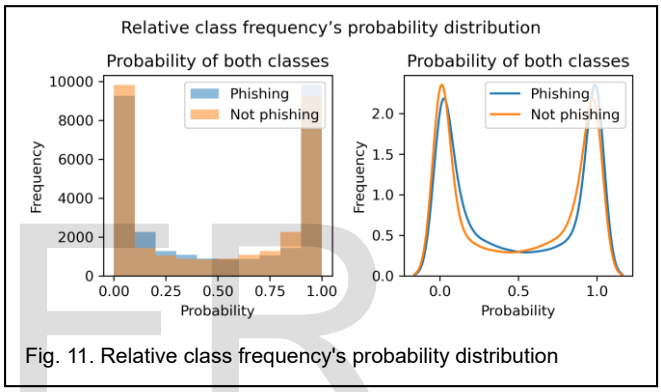

Fig. 11. Relative class frequency's probability distribution

The isotonic regressor's average cross entropy of calibrated probabilities is 0.218. The average cross entropy informs about very good probability when compared to random probabilities.

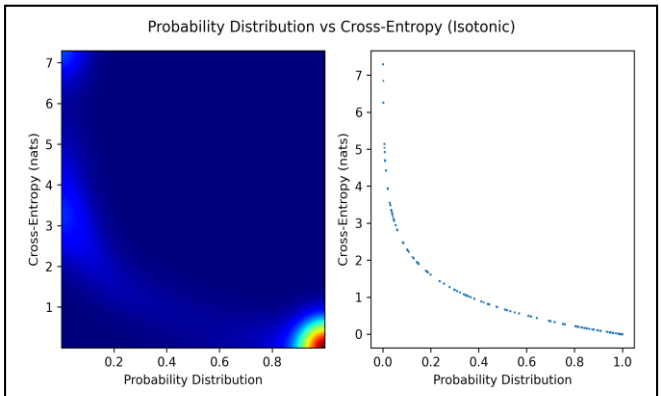

Fig. 12. Probability distribution vs cross entropy (Isotonic)

The probability distribution and the cross entropy have a negative relationship again, as seen in figure no 12. The pattern resembles that of average voting and relative class frequency. The heatmap displays a small number of data points around 0.0 - 0.1, indicating that the divergence from the target distribution is low, resulting in a low average cross entropy. In compared to relative class frequency, the most significant difference to note is that when considering the opposite of the target distribution

being estimated, the largest cross entropy achieved is roughly 7 nats.

The largest cross entropy was 16 nats when the identical case was considered for relative class frequency. This means that the probability calculated by isotonic are more accurate. Another thing to note in the plot is that the cross entropy is missing for some probability distributions, and the negative trend is not as smooth as relative class frequency. This is because there are no probability estimates for those distributions.

The average cross entropy for calibrated probabilities is only 0.03 nats, which is quite low when compared to relative class frequency. This demonstrates that relative class frequency has a high ability to predict probabilities. If this is the case, the calibration had little impact on the predicted probabilities, despite the calibration plot indicating otherwise. This is an intriguing behaviour that will be further investigated and evaluated. The behaviour of the error function: mean squared error and expected calibration error (root mean square error) can be analysed by looking at the calibration plot in more detail in order to understand how the error behaves internally and then the difference between relative class frequency and calibrated probabilities given by isotonic regressor. The calibration plot can be viewed as a regression problem wherein the perfectly calibrated line is a linear line where $y = x$. Therefore, logically, when the mean predicted probability is equal to the fractions of positive, the model will be perfectly calibrated leading to $y = x$ for every point in the plot. In response to the linear line, it can also be perceived as a linear regression problem but with a twist. The twist is, rather than predicting data points, the x is considered as the truth value and y is considered as the predicted value. By subtracting $x - y$, the residual can be retrieved, following the mean squared error can be retrieved, which can be analysed. The number of bins in a calibration plot is an important consideration. The x and y variables, which are the mean predicted probability and fraction of positives, are affected by the number of bins. Let's have a look at an example. If the bins are 10, the probabilities are 0.0 to 0.1, 0.1 - 0.2, 0.2 - 0.3, 0.9 - 1.0, and so on. Let's say the number of samples in the first bin range is 2000. These samples are based on probability estimates. When the number of bins is increased to around 100, each bin becomes 0.01 - 0.02, 0.02 - 0.03, and so on. As a result, depending on the real value of the probability, the number of samples will likewise drop. Now, if we increase the number of bins by a substantial amount, say 20000, the mean predicted probability and the actual bin value will be quite near in case the probability falls inside that bin range. Given only one probability occurring within such a limited range, the fractions of positive may be 0 or 1. As a result, if the fractions of positive are 0, both the mean predicted probability and the fractions of positive will be very close to each other, resulting in a very low residual and a very close match to the properly calibrated straight line. As a result, this behaviour will be investigated further to see how this miscalibration error behaves when the number of bins meets the total number of samples, as well as when the number of bins exceeds the total number of samples for which the model was evaluated.

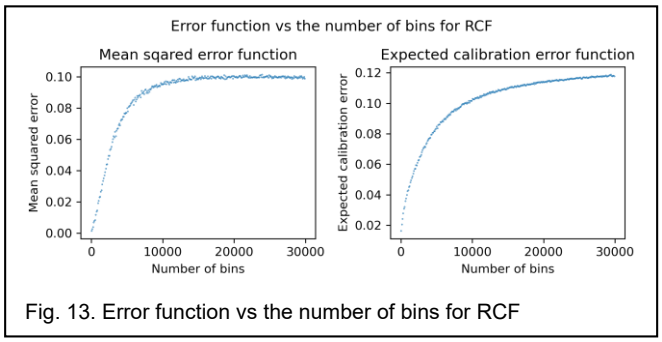

Fig. 13. Error function vs the number of bins for RCF

Figure 13 depicts the error function vs the number of bins for relative class frequency. The first plot depicts how the mean squared error function error changes as the number of bins increases from ten to thirty thousand. Initially, as the number of bins grows, so does the mean squared error, until the number of bins hits around 10000. When the number of bins exceeds 10,000, the error stabilises and remains within 0.09 - 0.10 until 30000 bins. Note, this error solely applies to the testing dataset. Based on the plot, the greatest deviation is around 0.10. This pattern makes sense from a logical standpoint. To better comprehend and assess it, let us split it down into sections. Because the residual (x - y) is minimal, the mean squared error is small as well. Furthermore, as previously stated, when the number of bins increases, the actual bin range shrinks, and the number of samples lowers, resulting in a low mean predicted probability that is closer to the actual bin range. This is shown in the figure no 14.

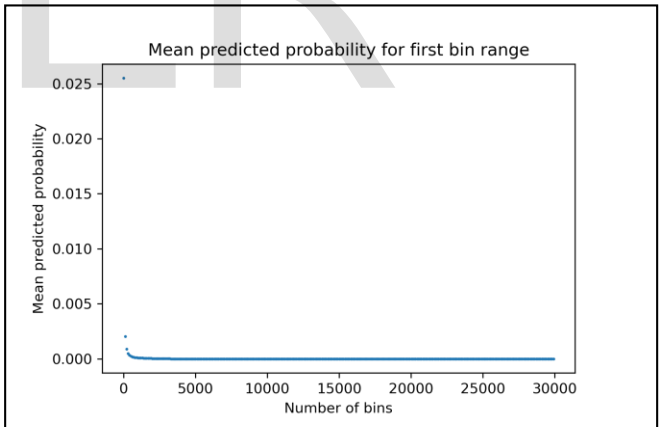

Fig. 14. Mean predicted probability for the first bin range

The mean predicted probability goes from 0.025 to nearly 0.0 as the number of bins increases. If this is the case, the mean squared error should decrease because the residual for high bin values is smaller. The fact that the fractions of positive change become opposites of the mean predicted probability reveals that this is not the case. This is depicted in the scatterplot and heatmap in figure 15.

The calibration curve data points for two absolute numbers of bins, 1000 and 10000, are depicted in the figure 15. A weak linear relationship can be seen in the scatterplot of 1000 bins. It is weak because the data points are dispersed and not particularly

close to one another, resulting in outliers and, as a result, a rise in the mean squared error when x = y is used as the line of best fit.

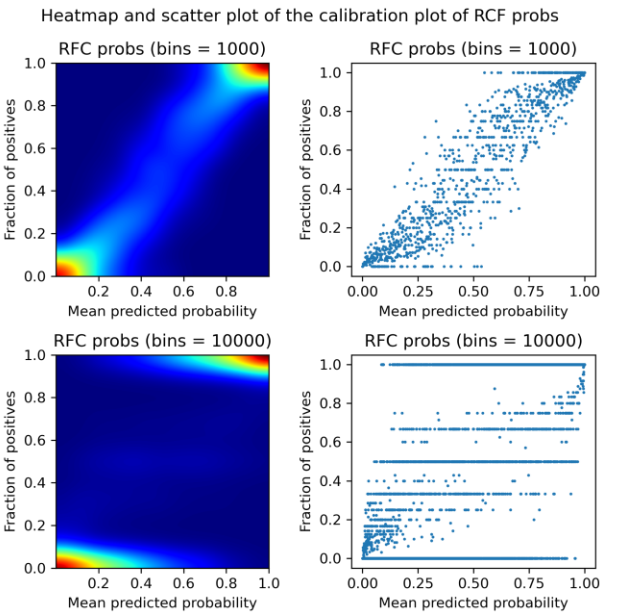

Fig. 15. Heatmap and scatter plot of the calibration plot of RCF probs

The most essential thing to take away from the plot is where the most data points are spread out. The majority of the data points on the heatmap are between 0.0 and 0.1 and 0.9 and 1.0. Because of the positive fractions, this is the case. As previously stated, as the number of bins increases, the mean predicted probability approaches the actual bin value, resulting in a small number of data points falling inside that bin range, maybe only a single sample causing only one or no positive classes. If there is only one positive class sample, the fraction becomes the number itself, resulting in a data point with a y variable value of 1 or 0.

This is still in favour of the line of best fit for the particular bins 1000, because the fractions of positive are related to the mean predicted probability, i.e., when the mean predicted probability is 0.0 or close to that number, the fraction of positive corresponds to the mean predicted probability. They are not opposite to each other. In the plot with the bins set to 10000, this opposing behaviour is plainly visible. There is no relationship between the variables because the bins are so large. In reality, the trend resembles a mirrored version of the letter 'Z.' The plot shows the reverse behaviour because the fractions of positive are 0 for the mean predicted probability of around 0.8, leading the mean squared error to rise. This can also happen in a different way. The mean predicted probability is near to 0.0, while the fractions of positive are 1, resulting in a large mean squared error. The heatmap also reveals that the majority of the points are in the 0.0 - 0.1 and 0.9 - 1.0 ranges, but the map is skewed near the opposite end of the bin ranges, indicating that some points are pointing in the opposite direction, as previously discussed. This explains why the mean squared error increases as the number of bins increases, which is an intriguing behaviour. Returning to figure 13, the mean squared error function and its trend have been investigated. The expected calibration error follows the same pattern as the mean squared error. Because of the behaviour explained before, the error grows as the number of bins increases. The expected calibration error does not settle even when the bins approach a large amount, which is the gap between the trend of mean squared error and the expected calibration error. This is yet another intriguing characteristic of expected calibration error that we can investigate by raising the number of bins to a large amount.

In order to understand what happens to the expected calibration error, the number of bins is increased to 100,000. The plot shows the error vs the number of bins is shown in the figure no 16.

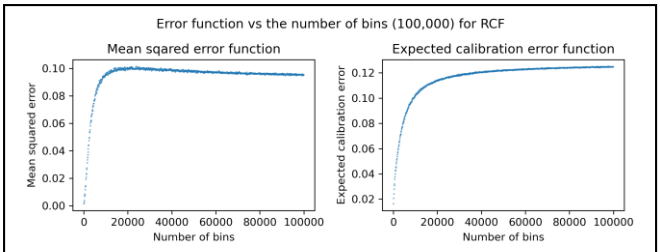

Fig. 16. Error function vs the number of bins (100,000) for RCF

Once the number of bins reaches 100,000, the MSE error function begins to reduce. This is a novel behaviour since, previously, the error rose as the number of bins increased, but if the error exceeds 30,000, the opposite occurs. This could be attributed to the fact that there were 28,771 total samples in the testing dataset. As a result, the probabilities within a certain bin range may be empty, lowering the total MSE. In the case of the ECE error function, the error is still rising rather than stabilising. Although the rate of change is low, it's possible that extending the bins more would stabilize the error. To check this, the number of bins has been increased to 500,000. Logically, the MSE should still decrease and for ECE it should stabilize based on the rate of change at approximately 90,000 bins.

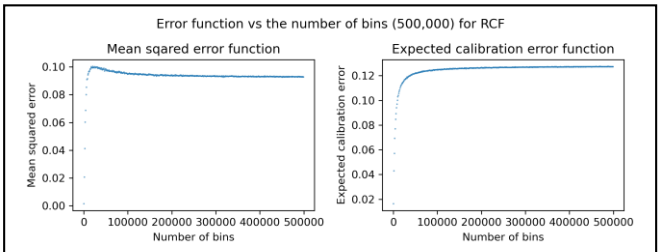

Fig. 17. Error function vs the number of bins (500,000) for RCF

The error vs 500,000 bins is shown in figure no 17. Logically, the MSE should be dropping, and it is, but at a relatively slow rate when compared to 100,000 bins. After 400,000 bins, the error for ECE starts to settle. As a result, both errors stabilize at different bins, ECE does not diminish, MSE does decline, but at a slower rate after a certain point. The error function has a trend

for relative class frequency in general, but this pattern changes as the number of bins increases sufficiently. The data points show a trend and are not widely scattered; thus, the probabilities are not really random. If the probabilities were random, the error function's data points would be scattered. Overall, increasing the number of bins to investigate both errors reveal how the error behaves when the mean predicted probability is nearly equal to the actual bin value. It also demonstrates how the positive fractions differ from the mean expected probability. Let's look at the same error function for calibrated probability and see how it compares to relative class frequency.

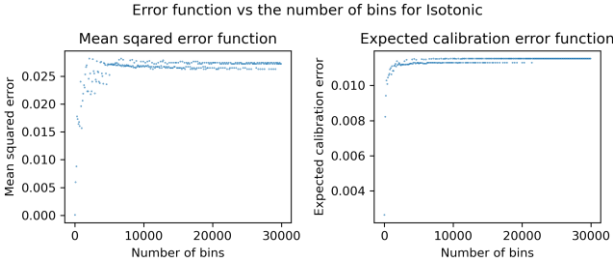

Fig. 18. Error function vs the number of bins for Isotonic

Figure 18 depicts the error function vs number of bins for the calibrated probabilities. The first graph depicts how the mean squared error function error changes as the number of bins increases from ten to thirty thousand. When the number of bins is increased, the mean squared error increases as well. When the number of bins reaches around 8000, the error stabilises and stays within 0.025 - 0.030 until the total number of bins reaches 30000. Based on the plot, the greatest error is around 0.030. The MSE is minimal because the residual for each point is small, as previously stated. In terms of the trend, as previously noted, as the number of bins increases, the actual bin range shrinks, and the number of samples shrinks as well, resulting in a low mean predicted probability that is closer to the actual bin range. It was previously addressed why the mean squared error is high despite the fact that the increment in bins should logically do the reverse. The trend of isotonic calibrated probabilities for MSE differs dramatically from the relative class frequency trend. The trend is different because, for instance, the data points are dispersed and lack the smooth flow of relative class frequency. This is a crucial point to keep in mind while comparing the two. Because the data points for isotonic are a little more dispersed, the probabilities are a little more random than those for relative class frequency. By random, it is meant that the probabilities don't have a continuation and would differ unexpectedly. The figure 19 helps to visualise this.

The calibration curve for bins 10 raised to increasing power from 1 to 4 is shown in figure 19. The probabilities provided by isotonic are precisely calibrated, shown in the first plot, whereas relative class frequency contains some over and under estimations. The most essential conclusion to draw from the plot is that the fluctuation of the data points for relative class frequency persists. As the number of bins grows, so does the variation, until the mean predicted probability and the proportions of positives are equal.

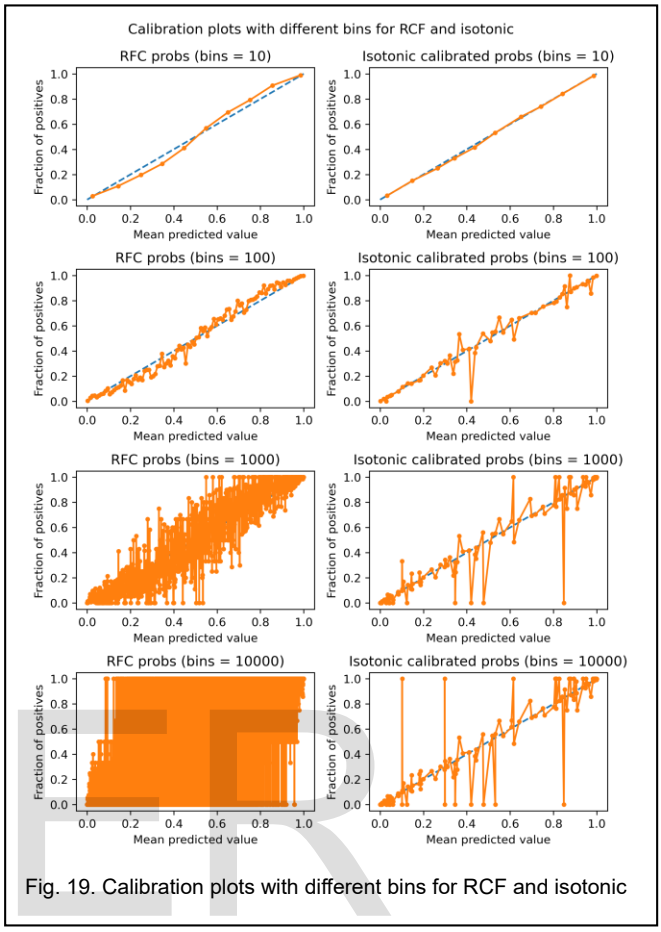

Fig. 19. Calibration plots with different bins for RCF and isotonic

For isotonic-based calibrated probabilities, the situation is different. There is no notable fluctuation in the calibrated probabilities; in fact, at some places, the fractions of positive are 0 and 1. This indicates that there is no apparent fluctuation trend. Furthermore, because there is no variation and certain places have a high residual even though the number of bins is clearly minimal, this suggests that very few probabilities are in particular bin ranges analytically. As a result, the data points in the MSE error function plot for isotonic based probabilities are a little scattered. The expected calibration error function stabilises after around 5000 bins, which is an intriguing fact to investigate in the case of calibrated probabilities. Figure 20 illustrates this point well. Because the calibrated probabilities do not have a large fluctuation, the ECE does not continue to rise as the number of bins grows, which is why the ECE stabilises. Because of the nature of the estimated probability, there is no fluctuation in the probabilities. Many bins are left empty when isotonic probabilities are disseminated further by significant bins because that particular bin range has no estimated probabilities. Additionally, some bins may include a few samples. If there are few samples, the probabilities may be incorrectly assessed, resulting in an opposing fraction of positive, resulting in more fluctuation. This is the case for relative class frequency, which is why when the bins are increased, there is a huge fluctuation. This is not the case with isotonic. Histograms for different

amounts of bins can be used to visualise this rationale.

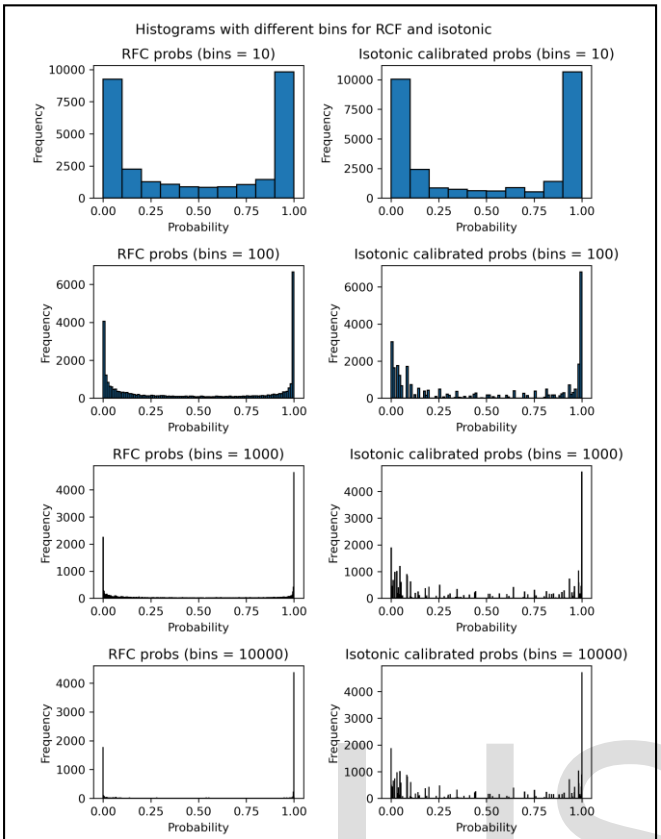

Fig. 20. Histograms with different bins for RCF and isotonic

As the number of bins increases to 100,000, the sample size for relative class frequency decreases. Finally, bins with a small number of samples have a high fluctuation, whereas bins with a large number of samples have a lesser fluctuation. There are a few samples in certain bins, which could produce a sudden swing. This is why, when compared to relative class frequency, the ECE and MSE for isotonic stabilise at a lower bin value. The entire analysis of cross entropy and calibration plots have given us the following results:

1. Probabilities are kind of random (the probabilities don't have a continuation and would differ unexpectedly) for isotonic, whereas for relative class frequency, there is a proper trend.
2. There is a very low fluctuation for isotonic when the number of bins increases, whereas relative class frequency has a high fluctuation.
3. Comparing the cross-entropy plot (figure no 12) and histograms (figure no 20) for isotonic, it is clear that some probability ranges are not estimated by isotonic.

The behaviour of MSE and ECE when increasing the number of bins to a considerable amount was known through the analysis of the calibration curve. It's also clear what the main difference is between relative class frequency probabilities and isotonic probabilities. The differences between the estimates of probabilities of both methods could also be understood via cross entropy analysis. Based on the results of the analysis, it can be concluded that the probabilities provided by isotonic are superior to those provided by relative class frequency. This is because isotonic has lower fluctuations as the number of bins increases, implying that the probabilities provided by isotonic are clustered towards certain bin ranges. Only a few probability estimates differ from the desired distribution. The cross entropy as well as the fluctuation analysis demonstrates this. The only issue that can be derived from isotonic probabilities is that they do not have a continuation and can differ unexpectedly. This is depicted in figure 19, which shows the MSE with some scattered data points. This indicates that the data point was spread due to an abrupt fluctuation in the calibration plot (fluctuation being an outlier causes the data point to be more scattered).

A statistical test can also be used to support the completed analysis by comparing calibrated probability and relative class frequency. The results of the statistical tests are reported in the table 8.

TABLE 8
STATISTICAL TEST RESULTS FOR ISOTONIC AND RCF

Error	t-statistic	p-value	result
ECE	7.48546	0.000672292	H_a
ACE	2.67625	0.0440176	H_a
MCE	0.261262	0.804311	H_0

Because the p-value is less than 0.05, the null hypothesis may be rejected and the alternate hypothesis, which claims that both procedures are statistically distinct, can be accepted. This confirms the findings of the earlier investigation.

Therefore, considering the method, isotonic probabilities can be used to estimate probabilities for phishing detection. The probability estimates can also be made better by training the model on the following criterias:

1. Cross validated average cross entropy - Train the model with parameters that could best fit the situation of predicting as close as possible to the target distribution.
2. Cross validated MSE - Train the model with parameters that could best fit the situation of predicting as close to the perfectly calibrated line (MSE being as close to 0).
3. Cross validated ECE - Train the model with parameters that could best fit the situation of predicting as close to 0 for ECE.

The random forest was retrained using cross validated average cross entropy to find the best model. Although any of the three techniques can be used to retrain the model and find the optimal parameters, the average cross entropy was chosen since it calculates cross entropy for each prediction in relation to its target distribution.

Because the computational time required by traditional GridSearchCV is so high, the random forest was retrained using Bayesian Optimization. In the table 9, the parameters returned

TABLE 9
FINAL MODEL WITH TUNED PARAMETERS VIA BAYESIAN OPTIMIZATION

Parameter	Values
n_estimators	198
max_depths	23
min_samples_splits	1
min_samples_leafs	2
max_features	'auto'
bootstraps	True

by Bayesian Optimization are listed.

The parameter values are quite identical to those of the previously trained model, with the exception that the final model is trained via cross validated average cross entropy. The model has a cross validated average cross entropy of 0.2125. It is lower than the previously trained calibrated model, as cross validation has been used. For isotonic based calibrated probabilities, the cross validated average cross entropy is 0.212.

7 THRESHOLD DETERMINATION

For estimating probabilities using average voting, we need to determine an optimum threshold based on which the classifier can classify the estimated probability to a particular class. Normally, the threshold of 0.5 can be used, but this threshold is not always optimum. We will be utilizing two different metrics that will lead us to the final threshold: the ROC plot and the precision-recall curve. The ROC Curve is a useful diagnostic tool for understanding the trade-off for different thresholds and the ROC AUC provides a useful number for comparing models based on their general capabilities. If crisp class labels are required from a model under such an analysis, then an optimal threshold is required. [20]. Note, we need to find the threshold wherein there is a balance between false positive and true positive rates. The ROC curve is shown in Figure 21. The curve has the point wherein the false positive and true positive rates are balanced.

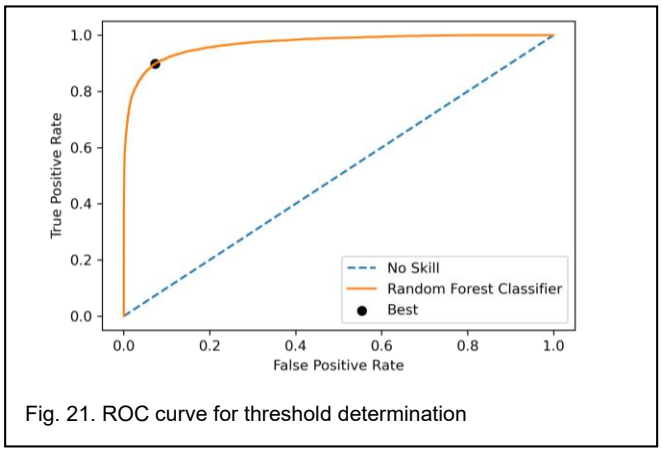

Fig. 21. ROC curve for threshold determination

Since, the black dot is the closest point to the top-left corner, this would definitely maintain a balance between false positive and true positive rate.

This threshold point is calculated using Youden's J statistic. Youden's J statistic = True positive rate - False positive rate. This statistic is calculated, and then the maximum value is chosen. The threshold respective to the maximum Younden's J statistic is then used for classification. The maximum Younden's J statistic is 0.826. The threshold respective to the maximum Younden's J statistic is 0.506451. Similarly, we will analyze the precision-recall curve and then select the threshold. The precision-recall curve with the threshold is shown in Figure 22.

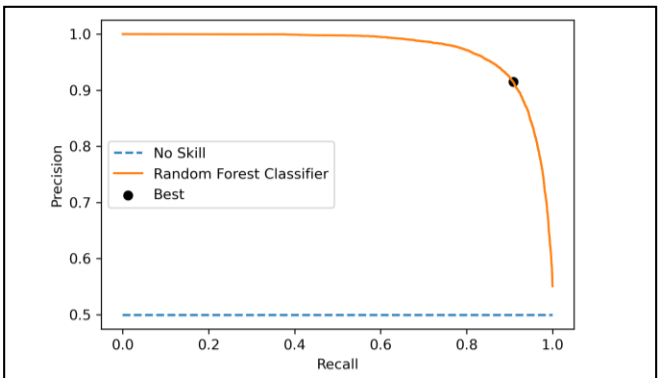

Fig. 22. Precision-recall curve for threshold determination

In a precision-recall curve, a no-skill model is represented by a horizontal line with a precision that is the ratio of positive examples in the dataset, or 0.01 on our synthetic dataset. A perfect skill classifier has full precision and recall with a dot in the top-right corner. Again, as we are interested in the balance between precision and recall, this is the same as optimizing the F-measure that summarizes the harmonic mean of both measures [20].

Similar to ROC, we will find the best F-score and the threshold concerning the best F-score. Since the black dot in the precision-recall curve is close to the top right corner, it represents the best threshold. The F-score is 0.912, and the respective threshold is 0.466890. Through both metrics, we see we have got different thresholds. Now which to choose? It is important to realize that precision-recall focuses on the performance of a classifier on the positive (minority class) only [20]. If this is the case, we need to choose the threshold respective to the ROC curve, as the negative class is also very important to us. Therefore, the final threshold we will choose is 0.506451.

8 CONCLUSION

Based on various features, we developed a model that estimates the probability of a URL attempting to phish. Based on the predicted probability, the same algorithm may also be used to classify the URL as phishing or non-phishing. We successfully evaluated and discovered a method for estimating probabilities that corresponded to our original methodology. The research was conducted in accordance with the root protocol. We were also able to successfully identify and examine the model's performance in both classification and probability predictions.

ACKNOWLEDGMENT

The authors sincerely express their gratitude towards Dr. Soumya Bhattacharya, who supported us throughout the research term with his feedback and suggestions.

REFERENCES

[1] Woodruff", J. (2019). What Are the Advantages of Decision Trees? *What Are the Advantages of Decision Trees?* Retrieved from https://smallbusiness.chron.com/advantages-decision-trees-75226.html

[2] Brownlee", J. (2020). How to Develop a Random Forest Ensemble in Python. *How to Develop a Random Forest Ensemble in Python*. Retrieved from https://machinelearningmastery.com/random-forest-ensemble-in-python/

[3] cuelogic. (n.d.). The Levenshtein Algorithm. *The Levenshtein Algorithm*. Retrieved from https://www.cuelogic.com/blog/the-levenshtein-algorithm

[4] Nonparametric statistics and model selection. (n.d.). *Nonparametric statistics and model selection*. Retrieved from http://www.mit.edu/ 6.s085/notes/lecture5.pdf

[5] Kolmogorov-Smirnov test. (n.d.). *Kolmogorov-Smirnov test*. Retrieved from https://ocw.mit.edu/courses/mathematics/18-443-statistics-for-applications-fall-2006/lecture-notes/lecture14.pdf

[6] College, L. B. (2019). Estimating a Population Mean. *Estimating a Population Mean*. Retrieved from https://www.slideshare.net/mmirfattah/estimating-a-population-mean-163426261

[7] The Case Against Precision as a Model Selection Criterion. (2018). *The Case Against Precision as a Model Selection Criterion*. Retrieved from https://www.datascienceblog.net/post/machine-learning/specificity-vs-precision/

[8] Interpreting ROC Curves, Precision-Recall Curves, and AUCs. (2020). *Interpreting ROC Curves, Precision-Recall Curves, and AUCs*. Retrieved from https://www.datascienceblog.net/post/machine-learning/interpreting-roc-curves-auc/

[9] Fawcett", T. (2004). ROC Graphs: Notes and Practical Considerations for Researchers. *ROC Graphs: Notes and Practical Considerations for Researchers*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi101110977 7&repr̄ep1&typep̄df

[10] ROC Graphs Which Evaluation Metric Should You Choose? *F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose?* Retrieved from https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc

[11] Bostrom", H. (n.d.). Estimating Class Probabilities in Random Forests. *Estimating Class Probabilities in Random Forests*. Retrieved from http://www.diva-portal.org/smash/get/diva2:305369/FULLTEXT01.pdf

[12] Danica". (2016). Is decision tree output a prediction or class probabilities? *Is decision tree output a prediction or class probabilities?* Retrieved from https://stats.stackexchange.com/q/193430

[13] sklearn.ensemble.RandomForestClassifier — scikit-learn 0.24.2 documentation. (n.d.). *sklearn.ensemble.RandomForestClassifier — scikit-learn 0.24.2 documentation*. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html #sklearn.ensemble.RandomForestClassifier.predict_{p}{r}{o}ba

[14] Brownlee", J. (2020). A Gentle Introduction to Probability Scoring Methods in Python. *A Gentle Introduction to Probability Scoring Methods in Python*. Retrieved from https://machinelearningmastery.com/how-to-score-probability-predictions-in-python/

[15] netcal.metrics.ECE — calibration-framework 1.1.3 documentation. (n.d.). *netcal.metrics.ECE — calibration-framework 1.1.3 documentation*. Retrieved from https://fabiankueppers.github.io/calibration-framework/build/html/_{a}{u}{t}osummary/_{a}{u}{t}osummary_{m}{e}{t}ric/netcal.metrics.ECE.html#id3

[16] netcal.metrics.MCE — calibration-framework 1.1.3 documentation. (n.d.). *netcal.metrics.MCE — calibration-framework 1.1.3 documentation*. Retrieved from https://fabiankueppers.github.io/calibration-framework/build/html/_{a}{u}{t}osummary/_{a}{u}{t}osummary_{m}{e}{t}ric/netcal.metrics.MCE.html#netcal.metrics.MCE

[17] netcal.metrics.ACE — calibration-framework 1.1.3 documentation. (n.d.). *netcal.metrics.ACE — calibration-framework 1.1.3 documentation*. Retrieved from https://fabiankueppers.github.io/calibration-framework/build/html/_{a}{u}{t}osummary/_{a}{u}{t}osummary_{m}{e}{t}ric/netcal.metrics.ACE.html#netcal.metrics.ACE

[18] G. Dietterich, T. (1997). Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms*. Retrieved from https://sci2s.ugr.es/keel/pdf/algorithm/articulo/dietterich1998.pdf

[19] 1.16. Probability calibration. (2021). *1.16. Probability calibration*. Retrieved from https://scikit-learn.org/stable/modules/calibration.html

[20] Brownlee, J. (2020). A Gentle Introduction to Threshold-Moving for Imbalanced Classification. *A Gentle Introduction to Threshold-Moving for Imbalanced Classification*. Retrieved from https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/