Phishing Detection: L2 Norm Minimization and Support Vector Machines

Marialena Sfyraki, Joseph Del Val, and Keye Li
{msfyraki,jdelval,kel039}@ucsd.edu

March 22, 2023

1 Task Assignment

In this project we have collectively discussed, analyzed and worked towards the following tasks as a group:

(a) Preprocessing the dataset, separating the dataset into training, validation and testing set

(b) Performing descriptive analysis on the dataset

(c) Statement of primal, dual problem and KKT conditions of our two approaches

(d) Implementation of two approaches

(e) Analysis and discussion of the results

(f) Report writing

2 Introduction

2.1 Motivation

Phishing attacks continue to be a pervasive and significant problem, with a 2021 report claiming that phishing attascks have reached an all-time high [1]. As a result, computationally efficient and efficacious methods for detecting phishing websites are necessary for any who are interested in maintaining the security not only websites but individuals as well. We attempt to explore such detection methods by investigating this as a convex optimization problem.

In transforming this topic into an L2-Norm minimization, we are able to leverage the speed of solving convex optimization problems in order to find a computationally efficient method of detecting phishing websites, which could thereby be easily ran as an additional security verification method.

2.2 Previous Works

Perhaps as a result of the ubiquity of phishing attacks, a wide variety of strategies for phishing detection have been explored and evaluated. However, it appears that Support Vector Machines (SVM) are most often used as a comparative baseline for investigating new approaches, with further research on their efficacy continuing into recent years.

For instance, Anupam et. al (2021) investigated using Support Vector Machines in order to carry out this classification task, achieving 90.21% accuracy when using the Grey Wolf optimization algorithm. [2] Babagoli et. al (2019) evaluated using SVM as well, achieving 91.83% test accuracy, and compared it to using the metaheuristic algorithm Harmony Search, which obtained 92.81% test accuracy. [3] Additionally, in their 2014 paper, Mohammad et. al investigated using a self-structuring neural network in order to detect such websites, achieving 90.35% accuracy after 50 epochs, and a peak of 92.48% accuracy after 500 epochs[4]. However, one of the highest accuracies recorded in the literature was in a 2012 conference paper by Ravi et. al, where a multi-layer perceptron model achieved 98.12% accuracy. [5]

Overall, these various strategies are dependent on the training set and the features therein– so given new data, evaluations of multiple methods are needed to properly contextualize the data.

2.3 Intended Contributions

In this paper, we seek to evaluate different approaches towards phishing detection. The first of these approaches is one of our own, and centers around L2 Norms minimization, using the *cvxpy* library to detect phishing websites by treating it as a convex minimization problem.

Following this approach, the second approach we will evaluate uses support vector machines on the same training set, which also allows us to contextualize these approaches with other methods seen in previous works.

2.4 Organization of Paper

Our paper first gives an overview of the dataset and its features. Following this, it explores the first approach, the mathematics behind it (its primal/dual form, KKT conditions, etc.) and its results. Subsequently, a similar exploration of SVM takes place, after which we conclude by comparing the efficacy of these two approaches.

3 The Dataset

This dataset, discovered via Kaggle, was initially created by Choon Lin Tan in 2018 as part of their paper *A new hybrid ensemble feature selection framework for machine learning-based phishing detection system.* [6] [7] It comprises 5,000 phishing websites collected from PhishTank and OpenPhish, as well as 5,000 legitimate websites collected from Alexa and Common Crawl.

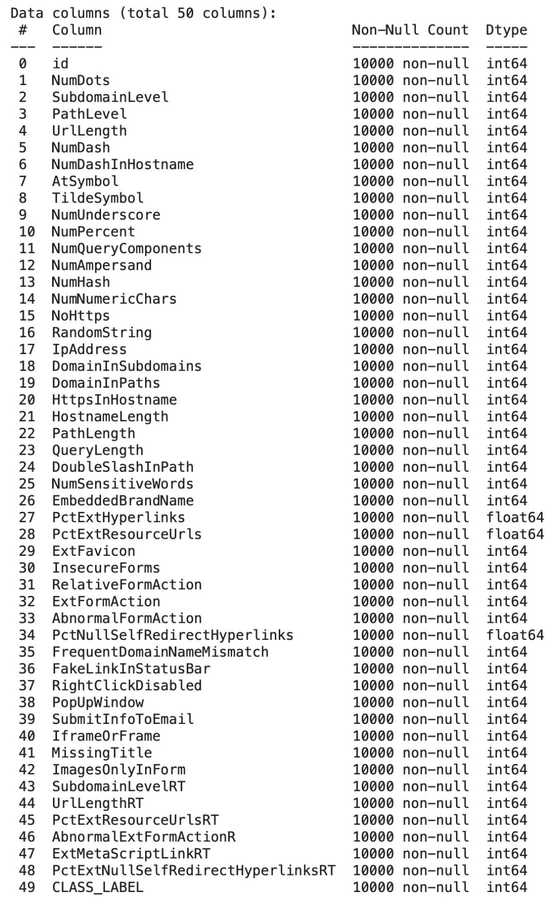
For the 10,000 rows in the dataset, each of them contains 50 columns. Among these, there is one unique numerical identifier, one binary class label, and 48 descriptive features.

3.1 Features

The features explore a variety of information about the websites, particularly about the URL.

Semantically, they encode information concerning the path levels and lengths of the urls, the amount of dots or dashes in the url, whether an IP address is in the url, whether it follows the http protocol, etc. Additionally, the dataset contains information about how the website functions, such as if right click is disabled, whether it contains a pop-up window, or if the most frequent domain name in the HTML source does not match the URL's domain name.

Figure 1 shows some information about the dataset, including the column names, index dtype, and non-null values. A full list of the meanings of each of the



```
Data columns (total 50 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   id                                  10000 non-null  int64
 1   NumDots                             10000 non-null  int64
 2   SubdomainLevel                      10000 non-null  int64
 3   PathLevel                           10000 non-null  int64
 4   UrlLength                           10000 non-null  int64
 5   NumDash                             10000 non-null  int64
 6   NumDashInHostname                   10000 non-null  int64
 7   AtSymbol                            10000 non-null  int64
 8   TildeSymbol                         10000 non-null  int64
 9   NumUnderscore                       10000 non-null  int64
 10  NumPercent                          10000 non-null  int64
 11  NumQueryComponents                  10000 non-null  int64
 12  NumAmpersand                        10000 non-null  int64
 13  NumHash                             10000 non-null  int64
 14  NumNumericChars                     10000 non-null  int64
 15  NoHttps                             10000 non-null  int64
 16  RandomString                        10000 non-null  int64
 17  IpAddress                           10000 non-null  int64
 18  DomainInSubdomains                  10000 non-null  int64
 19  DomainInPaths                       10000 non-null  int64
 20  HttpsInHostname                     10000 non-null  int64
 21  HostnameLength                      10000 non-null  int64
 22  PathLength                          10000 non-null  int64
 23  QueryLength                         10000 non-null  int64
 24  DoubleSlashInPath                   10000 non-null  int64
 25  NumSensitiveWords                   10000 non-null  int64
 26  EmbeddedBrandName                   10000 non-null  int64
 27  PctExtHyperlinks                    10000 non-null  float64
 28  PctExtResourceUrls                  10000 non-null  float64
 29  ExtFavicon                          10000 non-null  int64
 30  InsecureForms                       10000 non-null  int64
 31  RelativeFormAction                  10000 non-null  int64
 32  ExtFormAction                       10000 non-null  int64
 33  AbnormalFormAction                  10000 non-null  int64
 34  PctNullSelfRedirectHyperlinks       10000 non-null  float64
 35  FrequentDomainNameMismatch          10000 non-null  int64
 36  FakeLinkInStatusBar                 10000 non-null  int64
 37  RightClickDisabled                  10000 non-null  int64
 38  PopUpWindow                         10000 non-null  int64
 39  SubmitInfoToEmail                   10000 non-null  int64
 40  IframeOrFrame                       10000 non-null  int64
 41  MissingTitle                        10000 non-null  int64
 42  ImagesOnlyInForm                    10000 non-null  int64
 43  SubdomainLevelRT                    10000 non-null  int64
 44  UrlLengthRT                         10000 non-null  int64
 45  PctExtResourceUrlsRT                10000 non-null  int64
 46  AbnormalExtFormActionR              10000 non-null  int64
 47  ExtMetaScriptLinkRT                 10000 non-null  int64
 48  PctExtNullSelfRedirectHyperlinksRT  10000 non-null  int64
 49  CLASS_LABEL                         10000 non-null  int64
```

Figure 1: Feature information

48 features can be found in their 2018 paper. [7]

Syntactically, all of the features are numerical, and most are binary. For example, some of the few non-binary features include the length of the path, the percentage of external resource urls, or the total depth of the path in the website url.

Additionally, some features have three possible values (such as UrlLengthRT being either -1, 0, or 1)– these were generated by rules and/or thresholding, detailed in the paper.

As for the distribution of some of these features, there's quite a bit of variety. For example, the "IpAddress" variable which denotes whether the url contains an IP Address, has the value of 0 for the overwhelming majority of entries. However, RandomString (whether there is a random string in the url) is split relatively evenly, as is UrlLengthRT (whose thresholding-based value creates an even distribution by design). Among other distributions, UrlLength and PathLength are right-skewed, and features such as the percentage of external hyperlinks are heavy at both extremes. Figures 2, 3 illustrate the distribution of data for each of the 48 features in the dataset.

Overall, this dataset contains a large amount of information about the urls, yet is at the same time extremely convenient and usable due to the numeric nature of all of its entries and lack of missingness in any of the features.

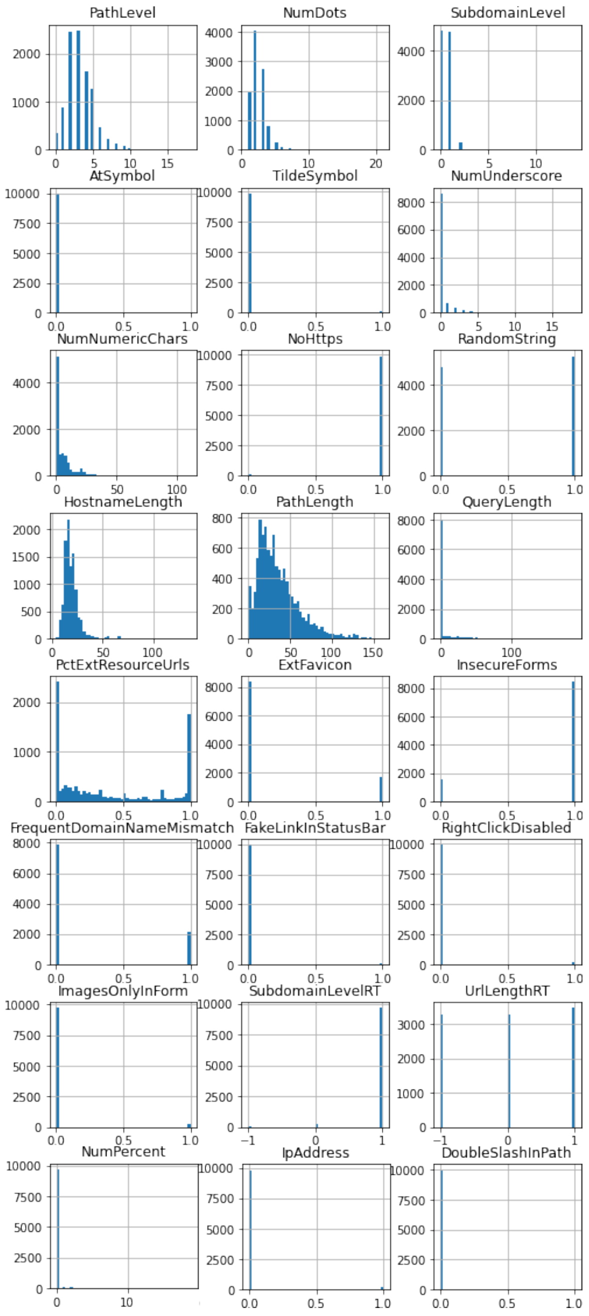


Figure 2: Distribution of first half of features

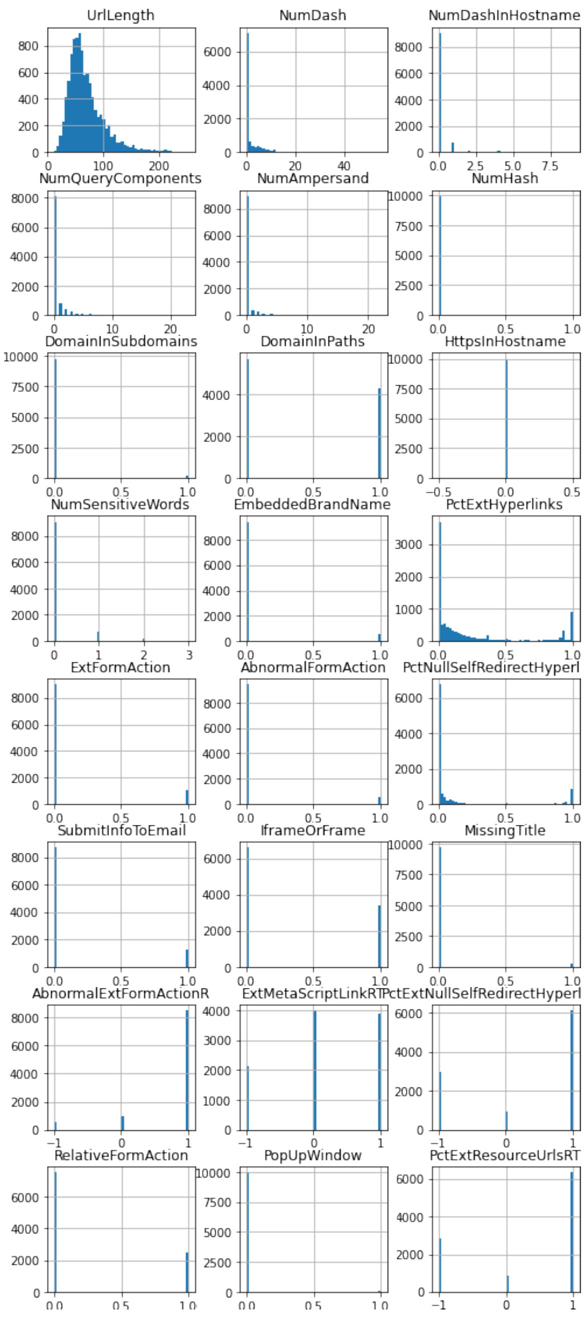


Figure 3: Distribution of second half of features

4 Approach 1

4.1 L2 Norm Minimization

Let the vector $\boldsymbol{x} \in \mathbb{R}^m$ represent the feature vector for some given website (i.e., each entry is the numerical value of some feature pertaining to said website), where m denotes the number of features. With this, we can define a weight vector $\boldsymbol{w} \in \mathbb{R}^m$ with which $\boldsymbol{w}^T\boldsymbol{x} = 1$ if this website is a phishing website, and $\boldsymbol{w}^T\boldsymbol{x} = 0$ if this website is not a phishing website.

Given this definition, we can transform our training set of website urls into the matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ and vector $\boldsymbol{y} \in \mathbb{R}^n$, where each row $\boldsymbol{x}$ of $\boldsymbol{X}$ is as described above, and each element y_i of $\boldsymbol{y}$ is an element of the set $\{0,1\}$ and determines whether the instance $\boldsymbol{x}$ is a phishing website. Here, n denotes the number of examples in the training set. In order to solve this minimization problem, we'll relax the problem such that each component of $\boldsymbol{Xw}$ takes any continuous value in the range $[0,1]$, and afterwards round it to the nearest integer in order to retrieve the label for the respective instance in the dataset. As a measure of distance from the prediction $\boldsymbol{Xw}$ to the target $\boldsymbol{y}$ we will use the squared L2-norm. We chose the squared L2-norm as an evaluation metric in our problem because (a) it is a convex function, (b) it eliminates negative values from the differences, (c) it ensures that the error is always positive and and (d) it penalizes the model more for making larger errors as compared to smaller ones.

With this in mind, we'll formulate the primal problem as follows:

4.2 Statement of the Problem: Primal Formulation

Given our problem:

$$\begin{aligned} &min_{\boldsymbol{w}}||\boldsymbol{Xw}-\boldsymbol{y}||_2^2 \\ &s.t.\ \ 0 \le v_i \le 1,\ \ \forall\ v_i \in \boldsymbol{Xw}, \end{aligned} \tag{1}$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, $\boldsymbol{w} \in \mathbb{R}^m$, $\boldsymbol{y} \in \{0,1\}^n$ and $v_i \in \mathbb{R}$.

We can rewrite it as follows:

$$\begin{aligned} &min_{w}||\boldsymbol{Xw}-\boldsymbol{y}||_2^2 \\ &s.t. -\boldsymbol{Xw} \le \boldsymbol{0},\ \ \boldsymbol{Xw}-\boldsymbol{1} \le \boldsymbol{0}, \end{aligned} \tag{2}$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, $\boldsymbol{w} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \{0,1\}^n$.

4.3 Dual Formulation

Given the above problem, we can derive its Lagrangian form as follows:

$$L(\boldsymbol{w},\lambda_1,\lambda_2) = \|\boldsymbol{Xw}-\boldsymbol{y}\|_2^2 + \lambda_1^T(-\boldsymbol{Xw}) + \lambda_2^T(\boldsymbol{Xw}-\boldsymbol{1}), \tag{3}$$

where $\lambda_1, \lambda_2 \in \mathbb{R}_+^n$.

From this, we can derive the Lagrangian dual function:

$$g(\lambda_1,\lambda_2) = inf_{\boldsymbol{w}\in D} L(\boldsymbol{w},\lambda_1,\lambda_2) \tag{4}$$

From which we can derive the dual problem:

$$max_{\lambda_1,\lambda_2} g(\lambda_1,\lambda_2),\ \ s.t.\ \ \lambda_1,\lambda_2 \in \mathbb{R}_+^n \tag{5}$$

4.4 KKT Conditions

In this context, our KKT conditions are:

Our primal constraints:

$$-\boldsymbol{Xw} \le \boldsymbol{0},\ \ \boldsymbol{Xw}-\boldsymbol{1} \le \boldsymbol{0} \tag{6}$$

Our dual constraints:

$$\lambda_1,\lambda_2 \in \mathbb{R}_+^n \tag{7}$$

Complementary slackness:

$$\lambda_1^T(-\boldsymbol{Xw}) = 0,\ \ \lambda_2^T(\boldsymbol{Xw}-\boldsymbol{1}) = 0 \tag{8}$$

And gradient of the Lagrangian with respect to $\boldsymbol{w}$:

$$\begin{aligned} &\nabla_{\boldsymbol{w}} L(\boldsymbol{w},\lambda_1,\lambda_2) = 0 \\ &\Leftrightarrow 2\boldsymbol{X}^T\boldsymbol{Xw} - 2\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{X}^T\lambda_1 + \boldsymbol{X}^T\lambda_2 = 0 \\ &\Leftrightarrow \boldsymbol{X}^T\boldsymbol{Xw} = \boldsymbol{X}^T\boldsymbol{y} + \frac{1}{2}\boldsymbol{X}^T(\lambda_1-\lambda_2) \\ &\Leftrightarrow \boldsymbol{w} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\left[\boldsymbol{X}^T\boldsymbol{y} + \frac{1}{2}\boldsymbol{X}^T(\lambda_1-\lambda_2)\right] \end{aligned} \tag{9}$$

Hence, we can write the dual problem as:

$$\begin{aligned} &max_{\lambda_1,\lambda_2}\left\|\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\left[\boldsymbol{X}^T\boldsymbol{y}+\frac{1}{2}\boldsymbol{X}^T(\lambda_1-\lambda_2)\right]-\boldsymbol{y}\right\|_2^2 \\ &+\lambda_1^T\left(-\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\left[\boldsymbol{X}^T\boldsymbol{y}+\frac{1}{2}\boldsymbol{X}^T(\lambda_1-\lambda_2)\right]\right) \\ &+\lambda_2^T\left(\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\left[\boldsymbol{X}^T\boldsymbol{y}+\frac{1}{2}\boldsymbol{X}^T(\lambda_1-\lambda_2)\right]-\boldsymbol{1}\right) \\ &s.t.\ \ \lambda_1,\lambda_2 \in \mathbb{R}_+^n \end{aligned} \tag{10}$$

4.5 Results

For our first approach, we solved the L2 Norm minimization problem using the *cvxpy* library.

After shuffling the dataset and creating a 90/10 train-test split, we removed the "CLASS_LABEL" column from the training set and assigned it to y, and created X from the remaining columns, sans the "ID" column. As for the test set, we created X_{test} and y_{test} in the same fashion.

Following this, we wrote the primal form of the problem in cvxpy, and solved it with our aforementioned constraints in order to find w.

To evaluate accuracy, we rounded each element of Xw and $X_{test}w$ to the nearest integer, and compared it with the ground truth y and y_{test}, revealing the accuracies reported in Table 1.

Dataset	Accuracy
Training Set	91.80%
Test Set	92.40%

Table 1: Training and test set accuracy using the L2-Norm minimization method

Precision, Recall and BER computed using the test set are reported in Table 2.

Precision	Recall	BER
0.9368	0.9175	0.0762

Table 2: Precision, Recall and BER reported on the test set using the L2-Norm minimization method

Fortunately, the similarity between the accuracy of both sets suggests that this approach suffers little from overfitting.

5 Approach 2

5.1 Support Vector Machines

Support Vector Machines (SVM) are a supervised learning model, which can be used for binary data classification. As described in Babagoli et. al, [3], Support Vector Machines plots the data onto a graph and attempts to construct a hyperplane to separate two distinct classes – in this case, separating the phishing websites from the legitimate websites. The SVM objective function is formulated in a way that ensures that the hyperplane maximizes the distance between the hyperplane and the observations closest to it (defined by support vectors), while correctly classifying the data points. To do this, a vector $\boldsymbol{w} \in \mathbb{R}^m$ and a scalar $b \in \mathbb{R}$ are searched for such that $||\boldsymbol{w}||_2^2$ is minimized, and $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1, \quad \forall i$. Note that here, y_i is formulated as -1 or 1 instead of 0 or 1. The constraint ensures that no data points lie between the support vectors. In order to address the problem of noisy data and prevent the hyperplane from overfitting on the dataset, we introduce additional slack variables $\xi_i \in \mathbb{R}, \ \forall i$. Each of the ξ_i represents the amount by which the i-th data point violates the margin - the distance between the decision boundary (hyperplane) and the closest data points from each class - and we want to minimize it. We also introduce a parameter $C \in \mathbb{R}$, which determines how much each misclassification is penalized. Parameter C controls the trade-off between maximizing the margin and minimizing the extent of violations.

For our problem, each $\boldsymbol{x}_i$ is a row from our dataset, comprising numerical features about the website. The target y_i is then 1 if the website is legitimate, and -1 if it is not. The problem is then defined as follows:

5.2 Statement of the Problem: Primal form

As described above (and in Babagoli et. al) [3], support vector machines give us the following minimization problem:

$$\begin{aligned} &min_{\boldsymbol{w},b,\boldsymbol{\xi}}(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}) + C\sum_{i=1}^{n}\xi_i \\ &s.t. \ \ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i, \ \ \xi_i \geq 0, \ \ \forall i, \end{aligned} \tag{11}$$

where $\boldsymbol{w} \in \mathbb{R}^m$, $\boldsymbol{\xi} \in \mathbb{R}^n_+$, $C \in \mathbb{R}_+$, $y_i \in \{-1, 1\}, \forall i$, $\boldsymbol{x}_i \in \mathbb{R}^m$ and $b \in \mathbb{R}$.

We can transform the first constraint as follows:

$$\begin{aligned} &y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) + \xi_i - 1 \geq 0, \ \ \forall i \\ &\Leftrightarrow 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - \xi_i \leq 0, \ \ \forall i \\ &\Leftrightarrow \mathbf{1} - \boldsymbol{y} \odot (\boldsymbol{X}\boldsymbol{w} + b\mathbf{1}) - \boldsymbol{\xi} \leq \mathbf{0}, \end{aligned} \tag{12}$$

where $\boldsymbol{y} \in \{-1, 1\}^n$, $\boldsymbol{w} \in \mathbb{R}^m$, $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}$, $\boldsymbol{\xi} \in \mathbb{R}^n_+$ and $\odot$ denotes the elementwise multiplication.

5.3 Dual Formulation

Given the above problem, we can derive its Lagrangian form as follows:

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \lambda, \mu) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}\xi_i + \lambda^T(\mathbf{1} - \boldsymbol{y} \odot (\boldsymbol{X}\boldsymbol{w} + b\mathbf{1}) - \boldsymbol{\xi}) - \mu^T\boldsymbol{\xi}, \tag{13}$$

where $\lambda \in \mathbb{R}^n_+$, $\mu \in \mathbb{R}^n_+$.

From which we can derive the Lagrangian dual function:

$$g(\lambda, \mu) = inf_{\boldsymbol{w},b,\boldsymbol{\xi}} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \lambda, \mu) \tag{14}$$

From which we can derive the dual problem:

$$max_{\lambda,\mu} g(\lambda, \mu), \quad s.t. \;\; \lambda \in \mathbb{R}^n_+, \mu \in \mathbb{R}^n_+ \tag{15}$$

5.4 KKT Conditions

In this approach, our KKT conditions are:

Our primal constraints:

$$\mathbf{1} - \boldsymbol{y}(\boldsymbol{X}\boldsymbol{w} + b) \leq \mathbf{0} \tag{16}$$

$$-\boldsymbol{\xi} \leq \mathbf{0} \tag{17}$$

Our dual constraints:

$$\lambda \in \mathbb{R}^n_+, \quad \mu \in \mathbb{R}^n_+ \tag{18}$$

Complementary Slackness:

$$\lambda^T \left[\mathbf{1} - \boldsymbol{y} \odot (\boldsymbol{X}\boldsymbol{w} + b\mathbf{1}) - \boldsymbol{\xi}\right] = 0 \tag{19}$$

$$-\mu^T\boldsymbol{\xi} = 0 \tag{20}$$

The gradient of the Lagrangian with respect to $\boldsymbol{w}$:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \lambda, \mu) = 0 \tag{21}$$

$$\Leftrightarrow \boldsymbol{w} - \sum_{i=1}^{n}\lambda_i y_i \boldsymbol{x}_i = 0 \tag{22}$$

$$\Leftrightarrow \boldsymbol{w} = \sum_{i=1}^{n}\lambda_i y_i \boldsymbol{x}_i \tag{23}$$

The gradient of the Lagrangian with respect to b:

$$\nabla_b L(\boldsymbol{w}, b, \boldsymbol{\xi}, \lambda, \mu) = 0 \tag{24}$$

$$\Leftrightarrow -\lambda^T\boldsymbol{y} = 0 \tag{25}$$

And the gradient of the Lagrangian with respect to $\boldsymbol{\xi}$:

$$\nabla_{\boldsymbol{\xi}} L(\boldsymbol{w}, b, \boldsymbol{\xi}, \lambda, \mu) = 0 \tag{26}$$

$$\Leftrightarrow \lambda = C\mathbf{1} - \mu \tag{27}$$

Hence, the dual problem can be written as:

$$max_\lambda \lambda^T\mathbf{1} - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \lambda_i \lambda_j \boldsymbol{x}_i^T \boldsymbol{x}_j \quad s.t. \;\; \lambda^T\boldsymbol{y} = 0, \quad \mathbf{0} \leq \lambda \leq C\mathbf{1} \tag{28}$$

5.5 Results

For this approach, we used the support vector machine implementation within scikit-learn. After shuffling the dataset, we created an 80/10/10 train-validation-test, so we could optimize our C value. We once again created y from the class label column, and X from the remaining columns sans the ID column. Similarly, this applies for X_{valid}, y_{valid}, X_{test}, and y_{test}.

Following this, we ran sklearn SVM implementation, fitting it to X and y, with our regularization parameter C. To locate our optimal C, we first performed a grid search over $C = 2^{-7}, 2^{-6}, 2^{-5}, ..., 2^6, 2^7$. For each C, we fit our model to X and y, retrieved the train data by evaluated its predictions from X against the ground truth labels y, and then obtained the validation accuracies by comparing its predictions of X_{valid} against the ground truth labels y_{valid} The results of this grid search are detailed in Figure 4.

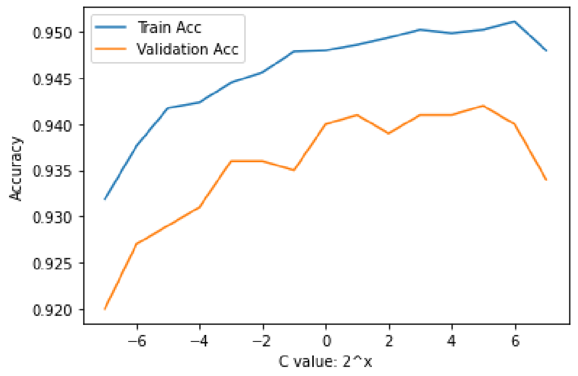


Figure 4: Initial Grid Search

Identifying the peak as around 2^5, we began a second grid search over values $20, 24, 28, ...40, 44$. Results of this grid search are included in Figure 5.

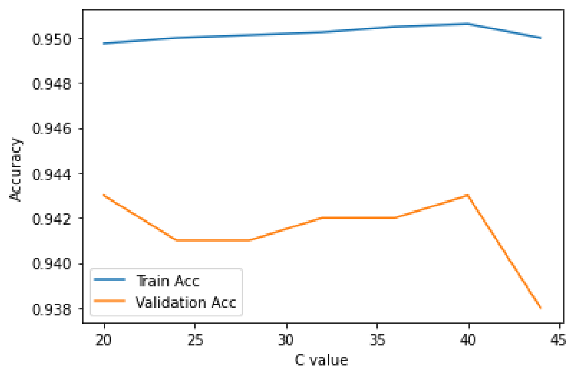


Figure 5: Second Grid Search

As a result of this search, we decided on a final value for our regularization parameter C as 40. Following this, we evaluated it against our final holdout set X_{test} and y_{test}, resulting in the accuracies reported in Table 3.

Dataset	Accuracy
Training Set	95.06%
Validation Set	94.30%
Test Set	94.50%

Table 3: Training and test set accuracy using the SVM method

Precision, Recall and BER computed using the test set are reported in Table 4.

Precision	Recall	BER
0.9383	0.9480	0.0549

Table 4: Precision, Recall and BER reported on the test set using the SVM method

Once again, we are happy to see that there is similar accuracy on both the training and test set, which suggests that the model has not overfitted to a significant degree.

6 Final Results

The results in terms of training and testing accuracy using each of the methods mentioned in Sections 4 and 5 are reported in Table 5.

Approach	Training Acc.	Test Acc.
L2 Norm	91.80%	92.40%
SVM	95.06%	94.50%

Table 5: Training and test set accuracy of the two methods

7 Conclusion

In the paper, we investigated two approaches: First, we defined phishing detection as a L2-norm minimization problem, and we also used Support Vector Machine to solve the problem. The dataset that we are using contains 10,000 data points. Each data point contains 48 features which are mostly descriptive information regarding the URL of the websites.

As it is shown in the results, while neither approach led to overfitting to the training set, it is clear that the SVM approach performs better in terms of accuracy than the L2-Norm approach. It is not a surprising result, as the SVM approach has been investigated by multiple articles before and has been proven to be able to achieve good performance.

One reason why SVM performs much better than the L2-Norm approach for our dataset is because most of the features of the data points are binary data. SVM is well-suited for binary data by nature because it constructs a hyperplane that separates data into 2 classes, and it is optimized to find the hyperplane that has the largest distance to both classes.

8 Future Work

One thing we want to keep in mind is that the performance of different approaches can vary a lot with different selections of datasets. It is noteworthy that although SVM performs well with our dataset, when SVM was used in Babagoli et. al (2019) [3] the test accuracy was 91.83%, which was not particularly good and was not able to outperform the meta-heuristic algorithm approach in the paper. Hence, the first thing we would like to do in the future is to try both the L2-Norm approach and the SVM approach on different datasets related to phishing-website detection, and see how their performance varies and if there is a chance that the L2-Norm approach can outperform the SVM approach.

We would also like to explore different approaches that could be used in phishing-website detection. We believe that the recent advancements in deep learning show promise in providing highly efficient alternative methods for URL phishing detection compared to other current approaches. In the context of identifying phishing URLs, CNNs can be trained on the sequence of characters or words that make up a URL to learn a representation that captures the inherent structural and semantic properties of the URLs. CNNs can also be used to extract features from URLs, such as character n-grams, word n-grams, and structural features, which can be used as input to a classification model. As demonstrated in [8] convolutional neural networks (CNN) have shown remarkable performance in the task of phishing URL detection. Additionally, as argued in [9], the power of transformer architecture using multiple stacked self-attention layers shows promising results. Transformers are particularly well-suited for processing sequential data and have been shown to be very effective in capturing complex dependencies between different parts of the input. While more research is needed to fully evaluate the effectiveness of advanced transformer-based models for malicious URL detection, the initial results are promising and suggest that these models have the potential to improve the performance of existing detection systems.

References

[1] S. Cook, "Phishing statistics and facts for 2019–2023," *Comparitech*, 2023.

[2] S. Anupam and A. K. Kar, "Phishing website detection using support vector machines and nature-inspired optimization algorithms," *Telecommunication Systems*, 2020.

[3] M. Babagoli, M. P. Aghababa, and V. Solouk, "Heuristic nonlinear regression strategy for detecting phishing websites," *Soft Computing*, 2019.

[4] R. M. Mohammad, F. Thabtah, and L. McCluskey, "Predicting phishing websites based on self-structuring neural network," *Neural Computing and Applications*, 2014.

[5] M. Pandey and V. Ravi, "Detecting phishing e-mails using text and data mining," *2012 IEEE International Conference on Computational Intelligence and Computing Research*, 2012.

[6] C. L. Tan, "Phishing dataset for machine learning: Feature evaluation," *Mendeley Data*, 2018.

[7] K. L. Chiew, C. L. Tan, K. Wong, K. S. Yong, and W. K. Tiong, "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system," *Information Sciences*, 2019.

[8] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "Urlnet: Learning a URL representation with deep learning for malicious URL detection," *CoRR*, vol. abs/1802.03162, 2018.

[9] P. Xu, "A transformer-based model to detect phishing urls," *CoRR*, vol. abs/2109.02138, 2021.