Phishing URL Classification

CS 4262
Spring 2023

Andrew Cascio [*1] **Scott Kang** [*1]

Abstract

Many people fall victim to social engineering regularly, namely phishing attacks. A phishing URL is disguised as a familiar URL, often accompanied by urgent or threatening messages with the intent of extracting personal information from its victim. This project aims to compare the accuracies and training times of three main classification methods to determine the best way to identify these scams.

1. Introduction

Millions of scam emails are sent daily, and within them, phishing URLs. These links are often sent with malicious intent. Such emails are the cause of most security breaches in companies. Most people can identify them, but those that are less tech savvy fall into the crossfire. A phishing URL is often accompanied by an urgent message. Often times, it's asking the recipient to verify their banking information, or to log into their social media account due to 'suspicious' activity. While most of these emails are easy to spot, and often filtered by email sites such as Gmail, many still seep through the cracks.

1.1. Problem

The problem we want to solve can be framed as: given a URL, is it a legitimate URL or a phishing URL? We aim to find the best method of determining this by comparing models in both accuracy and training time over a range of different feature subsets extracted by various feature selection techniques.

[*]Equal contribution [1]School of Engineering, Vanderbilt University, Nashville, Tennessee. Correspondence to: Andrew Cascio <andrew.c.cascio@vanderbilt.edu>, Scott Kang <kyung.ho.kang@vanderbilt.edu>.

1.2. Dataset

The dataset is taken from Kaggle. It contains 11430 URLs with 87 features after pre-processing. In terms of these features, 56 are extracted from the structure and syntax of the URLs, 24 are extracted from the content of their corresponding pages, and 7 are extracted by querying external services. The dataset is balanced, as it contains exactly half phishing and half legitimate URLs.

1.3. Related Work

The detection and classification of phishing URLs remains a critical challenge in the field of information security. A range of approaches have been proposed to address this problem, including those that rely on analyzing various features associated with the URLs. For instance, Garera proposed a machine learning-based approach that utilizes features such as page ranking and word analysis to classify malicious URLs. Similarly, Blum developed a model that only uses URL-based features such as domain name and special characters to detect phishing URLs. These studies highlight the potential of feature-based approaches for accurately identifying malicious URLs. We will use some combination of both methods in this study.

2. Methods

We chose to delve into three different supervised learning techniques. Namely, these are K-Nearest Neighbors, Support Vector Machines, and a feedforward Neural Network. The two primary metrics of success are:

1. accuracy on a held-out test set, and

2. time to train the model.

All three classification techniques will be evaluated according to these primary metric across multiple training feature sets. The sets of features used to train the models are:

1. all 87 features,

2. features that explain 90% of the variance as selected by Principal Component Analysis, and
3. the top 40 features corresponding to the highest mutual information with the output.

2.1. Pre-processing

Of the dataset's 89 features, there are two non-numeric features that must be translated. The first is the URL itself. We removed this feature from the dataset entirely as it is not a feature. The second is the classification. Each URL corresponds to a string feature that is either 'legitimate' or 'phishing', which serves as its classification. In order for our algorithms to function properly, we mapped 'legitimate' to 1 and 'phishing' to 0. The remaining features are discrete and numeric.

All features are discrete, however, some features take values up to several hundreds while others are can be represented by a single bit—simply a 1 or 0. Taking this into account—and in order to perform Principal Component Analysis later—we standardized each feature to a mean of 0 and a standard deviation of 1. We shuffled and split the dataset into train and test sets using a standard 80/20 split.

2.2. K-Nearest Neighbors

To begin our analysis of phishing URLs, we decided to explore a classification algorithm with essentially zero training time, K-Nearest Neighbors (KNN). Each input corresponds to an 87-dimensional vector, and Euclidean distance was chosen to be the distance metric. The main hyperparameter of the KNN algorithm is k, which we chose by running 10-fold cross validation within our training set for each $k \in \{1, 2, 3, ..., 15\}$. Our primary performance metric was accuracy on our held-out test set. The best performing hyperparameter was chosen to be $k = 3$, with declining performance for higher k, as seen in Figure 1.

2.3. Support Vector Machine

The second classification algorithm we leveraged was the Support Vector Machine (SVM). We opted for a radial basis function (RBF) kernel. The RBF kernel is a popular choice in SVMs because it can model complex nonlinear decision boundaries by mapping the original input data into a higher-dimensional feature space. This mapping is accomplished by applying a Gaussian function to the pairwise Euclidean distances between each pair of data points. To classify a new vector, the SVM calculates the distance between the new vector and the decision boundary in the feature space. In binary classification, if the distance is positive, the point is classified as belonging to one class, and if it is negative the point is classified as belonging to the other class. The RBF kernel has two hyperparameters: the γ (gamma) parameter controls the width of the Gaussian function, and the regularization parameter C controls the trade-off between achieving a low training error and a large margin. A smaller value of γ results in a wider bell-shaped curve and a smoother decision boundary, while a larger value of γ results in a narrower curve and a more complex decision boundary, but lends to possible overfitting. A smaller value of C allows for a larger margin and more misclassifications, while a larger value of C results in a smaller margin and fewer misclassifications, but possible overfitting. To determine the optimal hyperparameter pair, we selected C from $\{0.01, 1, 100\}$ and γ from $\{0.001, 0.5, 1\}$. For each possible combination, we ran 4-fold cross validation. We found that $C = 100$ and $\gamma = 0.001$ gave the best accuracy score.

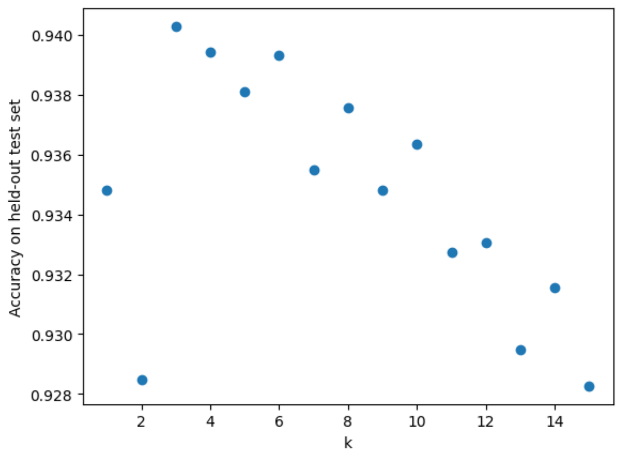

Figure 1. A plot of accuracies for each value of k. Accuracies were measured using 10-fold cross validation with all 87 features for each potential k. The optimal value of $k = 3$ was chosen for the model.

2.4. Neural Network

The third classification model we used in this study was a feedforward neural network, namely a multilayer perceptron. For our purposes, this vanilla neural network of a single hidden layer is sufficient to obtain high test accuracies. Our network consists of a single hidden layer with 16 nodes, each with a ReLU activation function. The output layer consists of a single node with a sigmoid activation function for classification. We decided to use Adam as our optimization algorithm with a learning rate of 0.001. The training and validation accuracies are shown in Figure 2.

2.5. Principal Component Analysis

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction in machine learning and

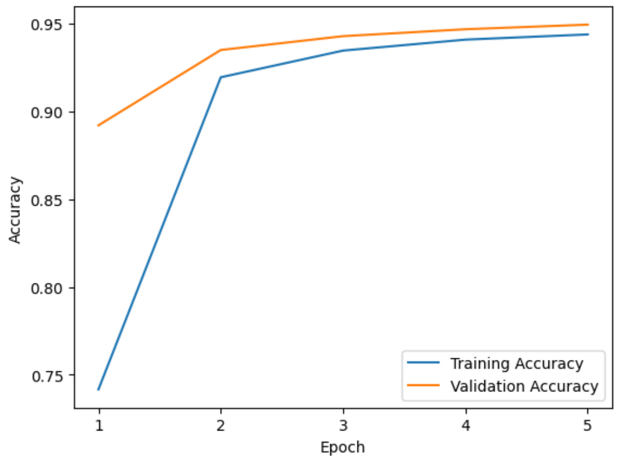

Figure 2. The training and validation accuracies of the neural network as it is trained over 5 epochs.

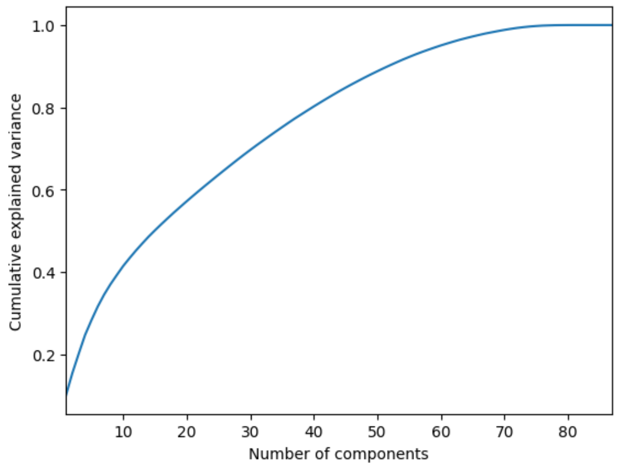

Figure 3. A function of the cumulative explained variance over the principal components. The first 51 principal components explain 90% of the variance.

data analysis. In this study, we performed PCA on the dataset of size 9144 x 87, where 9144 is the number of samples and 87 is the number of features. We identified a set of eigenvectors, called principal components, that capture a maximum amount of variance in the original dataset. We computed the covariance matrix of the training data set to derive eigenvalues and eigenvectors. Then, we sorted the principal components in order of largest corresponding eigenvalue to lowest corresponding eigenvalue. Figure 3 shows a cumulative function of the explained variance of the principal components used to identify the number of eigenvectors that capture 90% of variance. The result showed that the first 51 principal components explain approximately 89.9% of the cumulative variance in the training set. We performed a transformation of the dataset using the first 51 principal components, reducing the dimension of the dataset from 87 to 51. After the transformation, using the 51 principal components we retrained the previous models to compare the performance again over our primary metrics of accuracy on the test set and training time. The hyperparameters chosen stayed constant for each model.

2.6. Mutual information

Main approaches for feature selection fall into two categories: wrapper methods and filter methods. The feature selection method that we chose for this study is the filter method of mutual information. Mutual information (MI) is a measure of dependence between two variables. The value indicates the amount of information that knowing one variable provides about the other variable. Figure 5 shows mutual information values between top features and the output. From the initial set of 87 features, we selected the top 40 features that had the highest mutual information with the output variable. We then used this reduced feature set to retrain our models and compare the performance once again to the original models.

The hyperparameters chosen using cross validation stayed constant for the SVM model, but changed for the KNN model. The optimal value of $k = 5$ was selected, as opposed to the optimal value of $k = 3$ selected by using all 87 features.

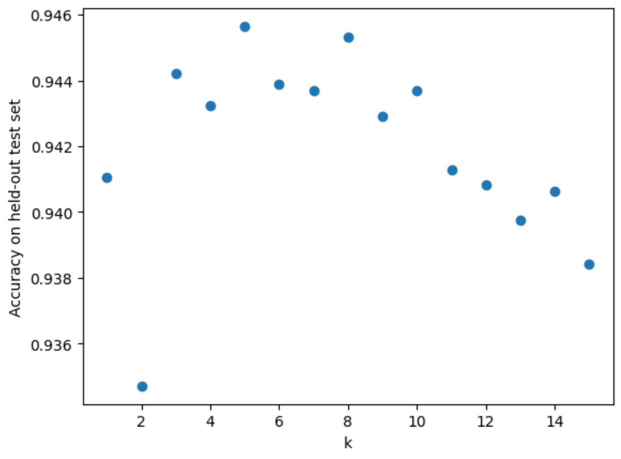

Figure 4. A plot of accuracies for each value of k. Accuracies were measured using 10-fold cross validation using the 40 features with the highest MI with the output for each potential k. The optimal value of $k = 5$ was chosen for the model.

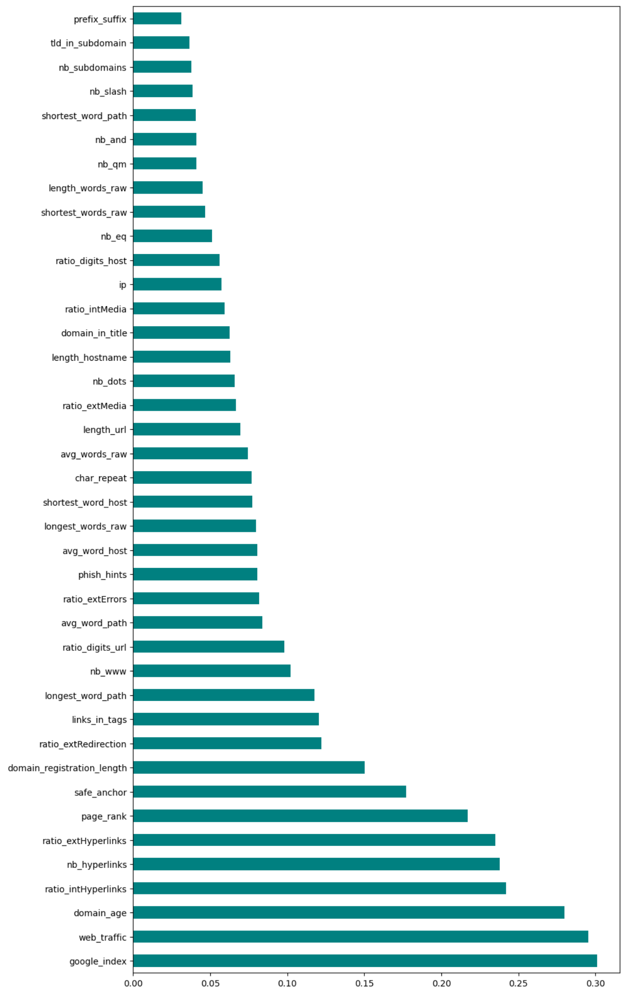

Figure 5. Mutual information between the 40 features with the highest MI and the output.

3. Results

As previously stated, the primary metrics we used to measure success are accuracy on a held-out test set, and time to train. We trained each optimized model 10 times for each training feature set and took the average time to train. These sets are: all 87 features, the 51 eigenvectors that explain 90% of the variance as selected by PCA, and the top 40 features with the highest MI with the output. The accuracy results for each model and each feature set are displayed in Figure 6, the training times are displayed in Figure 7, and the KNN prediction times are displayed in Figure 8.

3.1. All features

When training each model using all 87 features, SVM had the highest accuracy on the held-out test set—at the expense of the greatest training time—with an accuracy of 95.9%. The neural network (MLP) had the lowest average training time of 0.463 seconds. Note that KNN does not have a training time; its prediction time is graphed separately in Figure 8.

3.2. Principal Component Analysis

When training each model using features selected by PCA, the training time for our SVM decreased by 17.7%, and the training time for our MLP decreased by 2.8%. For the KNN model, the prediction time decreased by 14.6%. Models after PCA decreased the accuracy by 0.6% for our SVM and 0.3% for our MLP. For the KNN model, the accuracy on the test set increased by 0.6%.

3.3. Mutual information

When training each model after feature selection, we compare to the original feature set of all 87 features. The training time for our SVM decreased by 39.7% and the training time for our MLP increased by 1.5%. For KNN, the prediction time decreased by 16.5%. Models after feature selection decreased the accuracy on the held-out test set by 0.1% for our SVM. For KNN, the accuracy increased by 1.4%, and for our MLP it increased by 0.3%.

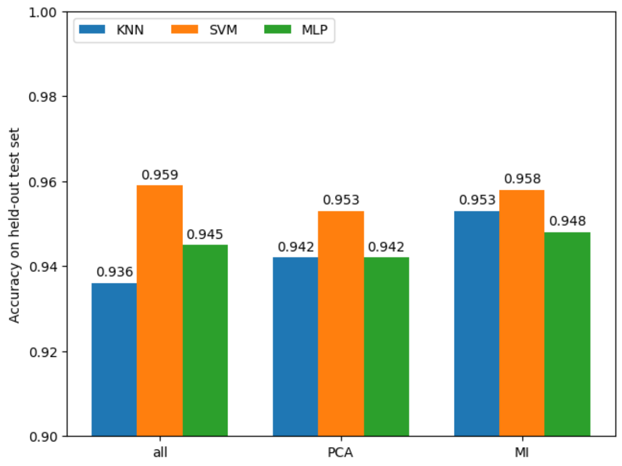

Figure 6. A multi-bar graph of the accuracies of each model for each feature set on a held-out test set.

4. Discussion and Conclusions

In summary, the SVM model outperformed the rest with test accuracies upwards of 96%. Part of this can be attributed to the fact that it is great when learning in a high dimensional feature space. It maintained a similar score for each feature set, performing the best with all 87 features. The KNN model improved as the number of dimensions decreased, which can be attributed to what is commonly referred to

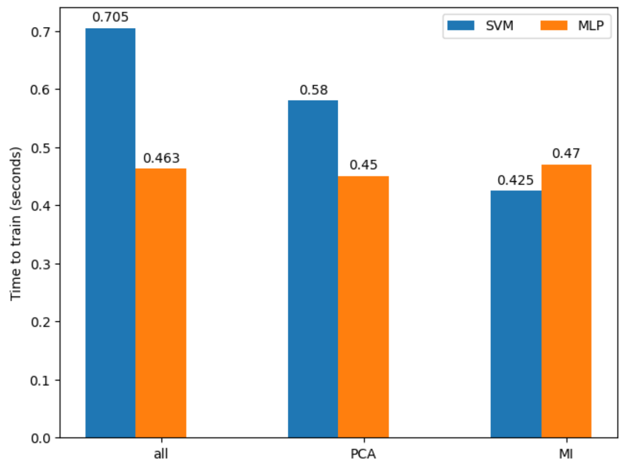

Figure 7. A multi-bar graph of the average training time of each model for each feature set taken over 10 samples.

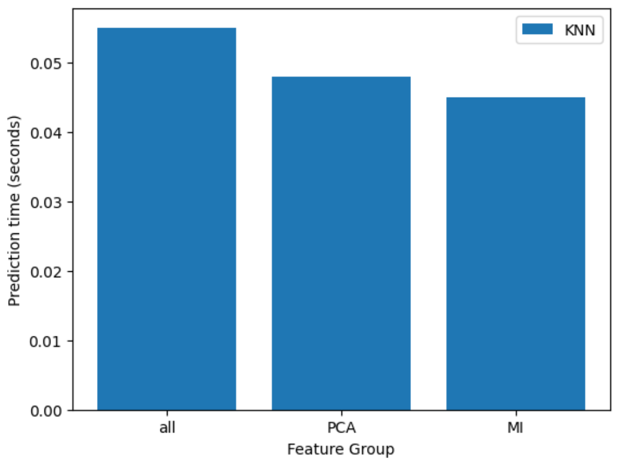

Figure 8. A multi-bar graph of the prediction time of the K-Nearest neighbors classifier for each feature set.

as 'the curse of dimensionality'. That is, as the number of dimensions increase, the distance to the nearest vector approaches the distance to that of the average vector. This hurts the assumption that the closest vectors are the most similar. Since the number of dimensions are decreasing with each feature group, the prediction time of KNN also decreases, as seen in Figure 8. Similarly, the training time for the SVM model decreased with decreasing number of dimensions, as seen in Figure 7. This is likely because calculating the inner product is linear in n. In contrast, the training time of the neural network stayed relatively constant, eventually surpassing that of the SVM after MI feature selection.

Visualizing our findings posed a challenge in this study due to the comparison of three different models with distinct data transformations. Effectively comparing and contrasting these models was difficult.

In the future, we want to consider optimizing hyperparameters for our neural network, and comparing logistic regression with the models we have already studied. In addition, we may test if performing Principal Component Analysis after feature selection produces different outcomes. Lastly, we may implement other popular machine learning models, such as convolutional neural networks and decision trees, which are widely used to classify phishing URLs.

Acknowledgements

Andrew Cascio worked on training the KNN and SVM models with each feature set, and finding the optimal hyperparameters for each model.

Scott Kang worked on training the neural network and implementing dimensionality reduction and feature selection.

References

Garera, S., Provos, N., Chew, M., Rubin, A.D. (2007, November). A framework for detection and measurement of phishing attacks. In Proceedings of the 2007 ACM workshop on Recurring malcode (pp. 1-8).

A. Blum, B. Wardman, T. Solorio, G. Warner Lexical feature based phishing URL detection using online learning Proceedings of the 3rd ACM workshop on security and artificial intelligence, AISEC, Chicago,Illinois,USA (2010), 10.1145/1866423.1866434

Ahmed, N. (2022, October 6). A python implementation of PCA with NumPy. Medium. Retrieved April 13, 2023, from https://medium.com/@nahmed3536/a-python-implementation-of-pca-with-numpy-1bbd3b21de2e

Bas, L. (2020, April 24).K-nearest neighbors classification from scratch with NumPy. Medium. Retrieved April 13, 2023, from https://towardsdatascience.com/k-nearest-neighbors-classification-from-scratch-with-numpy-cb222ecfeac1