CS235 Fall'22 Project Midterm Report: Detection of Phishing Websites

YASH AGGARWAL #1, NetID: yagga004
NITYASH GAUTAM #2, NetID: ngaut006
HRITVIK GUPTA #3, NetID: hgupt010
SIDDHANT POOJARY #4, NetID: spooj003
SHUBHAM SHARMA #5, NetID: sshar180

Additional Key Words and Phrases: data, mining, phishing websites, computer security, supervised learning, machine learning, Perceptron, Multi-layer Perceptron

ACM Reference Format:
Yash Aggarwal #1, Nityash Gautam #2, Hritvik gupta #3, Siddhant Poojary #4, and Shubham Sharma #5. 2022. CS235 Fall'22 Project Midterm Report: Detection of Phishing Websites. 1, 1 (November 2022), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 BRIEF SUMMARY OF THE STATE OF THE PROJECT

1.1 Current state

We had to change our original dataset so we have been able to get a different dataset, visualize it, and convert it from byte-stream to .csv format. We have implemented all the assigned off-the-shelf implementations on the dataset using scikit-learn. We have been able to set a baseline for each algorithm after multiple rounds of cross-validation and comparison of the model with itself and with the implementation of the team members.

We have also started working on our individual implementations (from scratch) of the models. We are using the same dataset for it. For Perceptron, we have created a class with the required functionality of fit, predict score methods. It is currently not generalized for all type of data sets(only works with the current dataset with limited input), need to work on that and generalize it. Also, need to fine-tune the tolerance and look into early stopping.

The problem definition is still unchanged. In this project, we will aim to classify if a website is phishing or benign using ML algorithms. However, we have had to change the dataset we are using. A few members have updated their algorithms as well.

1.2 Updates to the proposal

We had some issues with the original dataset. As it was unprocessed, we had to process it and extract features. During that extraction process, we found that many of the links provided in the dataset were of pornographic and gambling nature, and we could not include them in our project. We are now using a processed dataset with abstracted website

Authors' addresses: Yash Aggarwal #1NetID: yagga004; Nityash Gautam #2NetID: ngaut006; Hritvik gupta #3NetID: hgupt010; Siddhant Poojary #4NetID: spooj003; Shubham Sharma #5NetID: sshar180.

names to bypass these issues. Also, team member Nityash Gautam and Hritvik Gupta have changed their choice of algorithm from SVM and Naive Bayes to Random-Forest Classifier and Multi-layer perceptron.

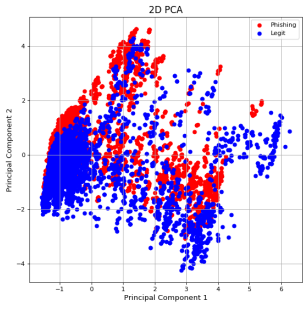
1.3 Preliminary results: Method 1 - Perceptron

Describe the following:

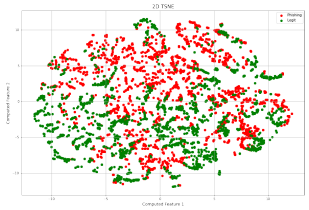
(1) **Off-the-shelf baseline**: For method 2, Perceptron is being used as model. The input settings have been tweaked as follows.
   (a) Penalty for label mismatch as **euclidean distance**
   (b) Regularizer for Penalty as **1e-7**
   (c) Max iterations to run as **100**
   (d) Min change required in the loss in case of early stopping as **1e-7**
   (e) Learning rate as **1e-5**
   (f) The number of iterations to wait for in case of no change as **20**

(2) **Potential modifications to the existing dataset and task**: "nothing to report".

(3) **Preprocessing**: Pre-processing was not required as the dataset was already normalized and processed. As the dataset had no class imbalance and number of sample were enough to train the model no sampling or sketching was done. For feature selection, tried to use TSNE, PCA and correlation matrix to work on reduced dimensions, however it seemed to reduce the accuracy on both the test and the training set as the data did not seem linearly separable



(a) 2D PCA

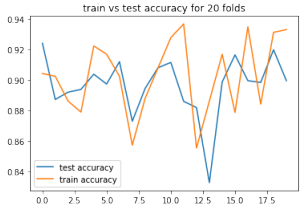


(b) TSNE

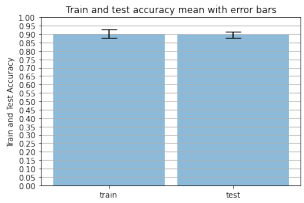
Fig. 1. Reduced Features

(4) **Experimental setting**: Data is separated in to 20 stratified sets using stratified k-fold sampling. A new model with same initial configuration is used for all the folds. These sets are used to train and then test a model. This process is used to test and average out the accuracy and miss rate. We are using miss rate to evaluate our model as in real world, labeling a phishing website as benign will be worse than misidentifying benign sites.

(5) **Results**: The in-built perceptron model ran with average accuracy of 90% with 2% std on the training set and 89% accuracy with 1.5% std on test set. The miss rate was of approximately 2%.

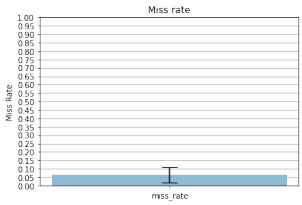


(a) Test and Train Line Graph

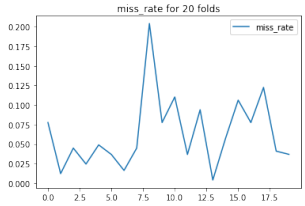


(b) Test and Train Bar Graph

Fig. 2. Test Train Plots



(a) Miss Rate Error Plot



(b) Miss Rate Line Graph

Fig. 3. Miss Rate Plots

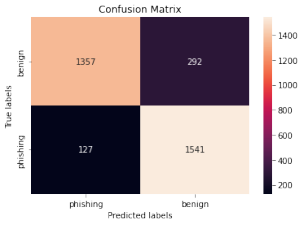


Fig. 4. Confusion Matrix

1.4 Preliminary results: Method 2 - MULTI-LAYER PERCEPTRON

Describe the following:

(1) **Off-the-shelf baseline**: For method 2, Multi-Layer Perceptron has been used. An existing model has been utilized for off-the-shelf implementation purposes, (https://scikit-learn.org/stable/modules/generated/sklearn.neuralnetwork.MLPClassifier.html). A Multi-Layer Perceptron Network, with 1 input layer having 30 neurons (Since we have 30 attributes in our Dataset). 2 Hidden Layers having 12 and 6 neurons respectively. Finally, 1 Output layer. The values for number of Layers and neurons per layer have been decided by creating networks of different architectures and measuring the "Miss Rate" in each case. Finally the metrics were selected where the Miss Rate obtained was Minimum. The following inputs have been used as part of the functioning of the classifier

(a) Maximum number of iterations = **500**
(b) Batch Size = **100**
(c) Hidden Layer Sizes = **(12,6)**
(d) Activation Function = **relu**

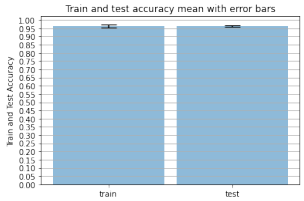
(2) **Potential modifications to the existing dataset and task**: "NOTHING TO REPORT"

(3) **Preprocessing**: As mentioned earlier, the data is already pre-processed and on further visualizing the data, it was observed that there wasn't any class imbalance as well. Therefore, no pre-processing was done on the dataset.

(4) **Experimental setting**: Stratified K-Fold Sampling has been used to slice the DataSet into 20 Stratified sets. For each new set of the dataset, a new MLP Classifier has been used using the previous parameters only. Accuracy and Miss Rate for both, training and testing has been noted for every set of the dataset. Lastly, the accuracy and miss rate have been averaged out to obtain an idea about the overall performance of the classifier's functioning. We have assigned more importance to MISS RATE because, classifying a Phishing website as benign will be having more adverse effect for the user as compared to the case where a benign webiste is classified as phishing.

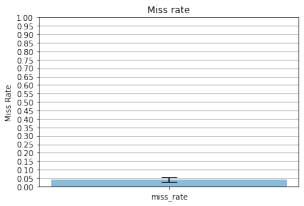
(5) **Results**: The Multi-Layer Perceptron runs with an Average Training accuracy and Standard Deviation of 96.4% and and 0.9% respectively and Average Testing Accuracy and Standard Deviation of 96.2% and 0.3% respectively. The Miss Rate obtained is approximately 2.5%.

(a) Test and Train Line Graph



(b) Test and Train Bar Graph

Fig. 5. Test Train Plots



(a) Miss Rate Error Plot

(b) Miss Rate Line Graph

Fig. 6. Miss Rate Plots

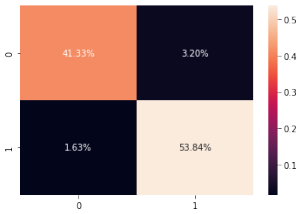
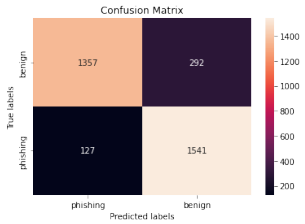


Fig. 7. Confusion Matrix

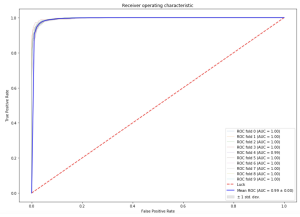
1.5 Preliminary results: Method 3 - Random Forest

Describe the following:

(1) **Off-the-shelf baseline**: The third method employs Random forest classifier. An existing model has been utilized for off-the-shelf implementation purposes,Random Forest is being used as model a estimator, a random forest employs averaging to increase prediction accuracy and reduce overfitting by fitting a number of decision tree classifiers on different subsamples of the dataset. If bootstrap=True (the default), the size of the sub-sample is determined by the max samples argument; otherwise, each tree is constructed using the whole dataset. The following inputs have been used as part of the functioning of the classifier
   (a) To Reduce the overfitting and to improve accuracy **n estimators**
   (b) n estimators = **100**
   (c) maximum number of features to be used in order to increase the overall performance of the model = **max features**
   (d) number of process to be run in parallel = $\mathbf{n}_jobs - 1$
(2) **Potential modifications to the existing dataset and task**: "nothing to report".
(3) **Preprocessing**: As we have mentioned earlier that we took an preproccessed data and thereby found no imbalance while visualizing as well enough number of samples to train the model.
(4) **Experimental setting**: To evaluate the model performance on the provided dataset stratified K-fold is used. Data is separated in to 10 stratified sets using stratified k-fold sampling. A new model is fitted every time on each fold of data. For every dataset AUC value is noted and atlast ploted an ROC curve for each value of AUC on different dataset. Lastly, we have created the confusion matrix and the classification report for the best performance model from K-Fold and AUC value to get the overall idea of performing model.
(5) **Results**: The Random forest classifier has average accuracy of 97% with Miss rate and f1 score respectively as 17% and 98%.



(a) Confusion Matrix



(a) ROC Curve

Fig. 9. ROC CURVE

1.6 Preliminary results: Method 4 - K-nearest-neighbor classification

Describe the following:

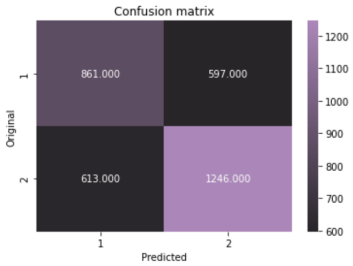
(1) **Off-the-shelf baseline**:Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point. KNeighborsClassifier implements learning based on the K nearest neighbors of each query point, where K is an integer value specified by the user. The K-neighbors classification in KNeighborsClassifier is the most commonly used technique. The optimal choice of the value K is highly data-dependent: in general a larger K suppresses the effects of noise, but makes the classification boundaries less distinct.

(2) **Potential modifications to the existing dataset and task**: "NOTHING TO REPORT"

(3) **Preprocessing**: As previously mentioned, the chosen data-set was pre-processed beforehand and also did not pose any difficulties as it had no null values as well as was well-balanced. Hence, no pre-processing was required.

(4) **Experimental setting**:The selection of number of neighbours (K) is an important parameter needed for model-building, as a low value will lead to over fitting of the model whereas a high value will lead to exponential computational time which is expensive. Here, the value K=3 comes out to be the most optimal as it gives good accuracy compared to other values as well as an odd value is preferred in such studies whenever we have a even number of classes. We then train and test the model to evaluate the performance and garner insights about the accuracy of the model. Furthermore Precision, Recall, F1 and support scores of the model are generated by the classification report.

(5) **Results**: K-Nearest Neighbours has average accuracy of 63% with Miss rate and f1 score respectively as 40% and 0.63.



(a) Confusion Matrix

Fig. 10. Confusion Matrix

1.7 Preliminary results: Method 5 - Decision Tree Classification

Describe the following:

(1) **Off-the-shelf baseline**: For method 5, Decision Tree Classification has been used. In order to determine whether a website is phishy, a Decision Tree Classification model from scikit-learn (https://scikit-learn.org/stable/modules/tree.htmlclassification) is applied. During the installation of decision tree, an optimal feature is selected to split the data based on metrics like information gain, so if there are non-relevant features, they will simply not be used. Further, For hyperparameter optimization of decision tree model, RandomizedSearchCV (sklearn.modeselecti.RandomizedSearchCV) is used.
The parameters for decision tree model are as follows
   (a) Maximum depth of tree = **25**
   (b) Number of features to consider for the best split = 'sqrt'
   (c) criterion = 'gini'
   (d) Splitter = 'best'
   (e) Minimum sample split = **2**
   (f) Minimum sample leaf = **1**

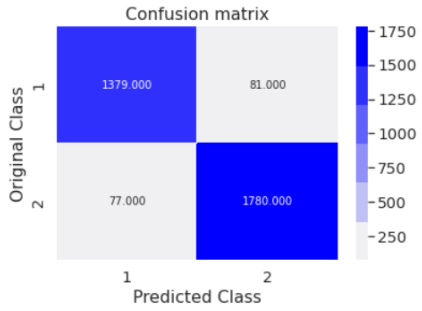
(2) **Potential modifications to the existing dataset and task**:"NOTHING TO REPORT".

(3) **Preprocessing**: Because the chosen data-set was pre-processes and had no null values, pre-processing was not required. Regarding feature selection, in decision tree an optimal feature is selected to split the given data based on metrics like information gain, so if there are non-relevant features, they will simply not be used. However, there is common practice to remove features like the ID field which are bad for decision trees since they end up with the highest information gain value.

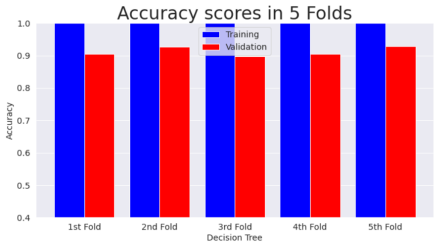
(4) **Experimental setting**: In order to evaluate the Decision tree model performance in the best way is through a confusion matrix, where predicted labels and true labels are matched by showing the predicted value in one axis and the true values in another axis. Further, Precision, Recall and F1 Score will be calculated. It will be challenging to choose which is superior (high precision and low recall or vice-versa). Consequently, a metric that incorporates both like F1 score will be useful. Additionally, Using a stratified K-fold, data will be separated

in 10 sets and for each set, model will be trained. For each model, the AUC value will be calculated and the ROC curve will be plotted. In the end, 5-fold cross-validation will be done, to understand the results better, we can visualize them using a bar graph.

(5) **Results**: The Decision Tree Classification has a average test accuracy of 94.66% with precision, recall and f1 score respectively as 0.95 0.95 0.95. The miss rate was of approximately 4%

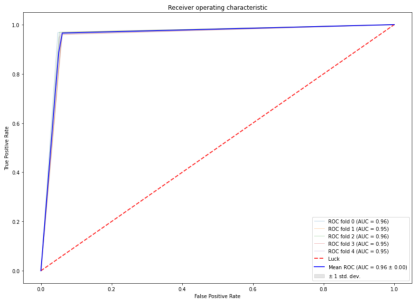


(a) Confusion Matrix



(a) Accuracy Score in 5 Folds

Fig. 12. Accuracy Score in 5 Folds



(a) ROC Curve

Fig. 13. ROC CURVE