CS235 Fall'22 Project Implementation Correctness 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 Implementation Correctness Report: Detection of Phishing Websites. 1, 1 (December 2022), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

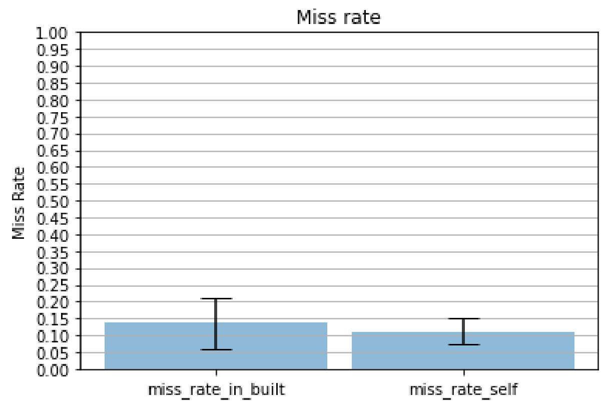
1 PERCEPTRON IMPLEMENTED BY YASH AGGARWAL

- **Algorithm**: A single-layer Perceptron has been implemented. The model consists of a single input layer connected to a perceptron with weights and a bias.
- **Baseline**: As a starting point, sklearn's single-layer Perceptron implementation is used. For the purpose of model evaluation and comparison, the in-built model's prediction accuracy and miss rate are considered.
- **Potential discrepancies**: Some of the input settings for the in-built implementation have been tweaked to match the self implementation. These are as follows:
  1. Max iterations to **100**
  2. Min change required in the loss in case of early stopping as **1e-7**
  3. Learning rate as **1e-5**
  4. The number of iterations to wait for in case of no change as **20**

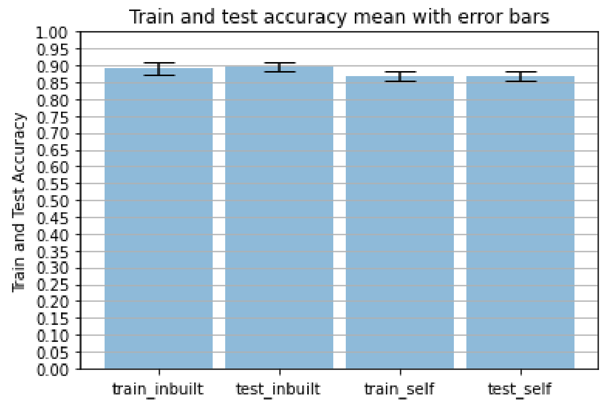
  These adjustments are necessary in order for the built-in implementation to function similarly to the self-implementation. The built-in implementation is much more optimized than self-implementation because it makes use of dynamic learning rate and early stopping. The aforementioned changes are done to eliminate any discrepancy and level the playing field for both techniques.

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

- **Measures of comparison**: The confusion matrix was calculated after the model had been trained, and it provided the measurements of (True Positives, True Negatives, False Positives, and False Negatives). We may estimate how many phishing websites the model categorized as non-phishing by computing the Miss Rate and F1 Score using the obtained metrics. We are interested in miss rate as miss-classifying phishing websites as non phishing would lead users to being vulnerable and trust websites that might be harmful. The inbuilt implementation had a miss rate of 13% ± 7% while the self implementation had a miss rate of 11% ± 3%.
- **Experimental results**: The miss rate and test train accuracy for the k-folds(k=15) was computed and the results are as follows.



(a) Miss Rate



(b) Test train accuracy

Fig. 1. Comparison Results

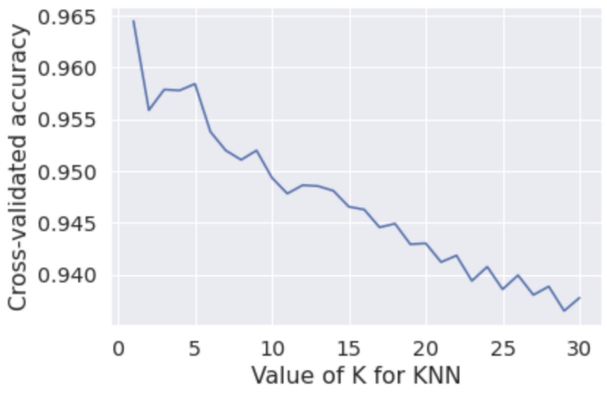
- **Justification of discrepancies**: There is no apparent major disparity. The model was run 15 times, and the average test train accuracy for in-built implementation was calculated to be 89.3% ± 1.3% and for self implementation to be 86.7% ± 1.3%. Further the in-built implementation had a miss rate of 13% ± 7% while the self implementation had a miss rate of 11% ± 3%.

2 K-NEAREST NEIGHBOURS IMPLEMENTED BY SIDDHANT POOJARY

- **Algorithm**: A K-Nearest Neighbour algorithm has been implemented. It is a supervised learning classifier model which employs proximity to make classifications for the grouping of an individual data point.
- **Baseline**: We used the K-Neighbours Classifier from SciKit Learn. 5 Different feature representations were used for the dataset which we obtained from SciKit Pre-processing,Manifold and Decomposition modules. Furthermore, We used 5 different distance functions for each representation which we got from SciKit metrics and Scipy Spatial Distance.
- **Potential discrepancies**: Different models have been created to garner insights into the model on a wide basis. The Feature Representations are as follows:
  1. **Standard Scalar**
  2. **Min-Max Scalar**
  3. **Principal Component Analysis**
  4. **Multi-Dimensional Scaling**
  5. **T-Distributed Stochastic Neighbour Embedding**

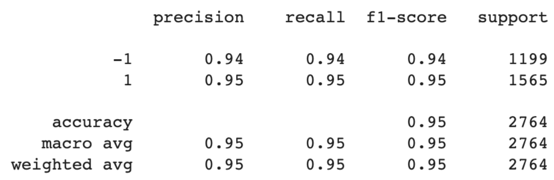
  We combined each representation with 5 different distance metrics namely:
  1. **Manhattan Distance**
  2. **Minkowski Distance**
  3. **Cosine Distance**
  4. **Euclidean Distance**
  5. **Jaccard Distance**

  We implemented this for both off-the-shelf as well as scratch models.
- **Measures of comparison**: We trained the Models for different K-values i.e. from 1 to 31. We found the optimal K-value to be 5 as demonstrated by the plot. Furthermore, We tested different feature representations with different distance metrics and found out that PCA representation along with Manhattan (Cityblock) Distance metric gives out the best results for the data.



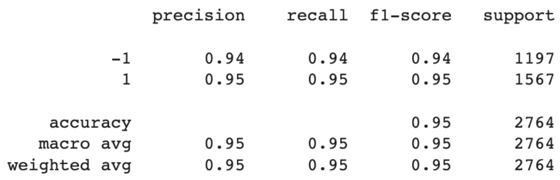
(a) Accuracy for Different Values of K

- **Experimental results**:



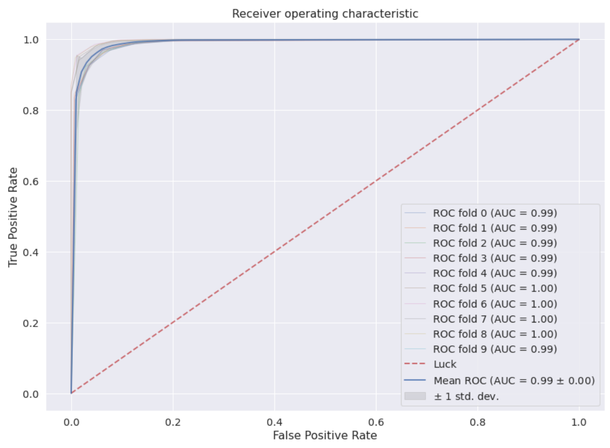
```
              precision    recall  f1-score   support

          -1       0.94      0.94      0.94      1199
           1       0.95      0.95      0.95      1565

    accuracy                           0.95      2764
   macro avg       0.95      0.95      0.95      2764
weighted avg       0.95      0.95      0.95      2764
```

(a) Classification Report of KNN from Shelf

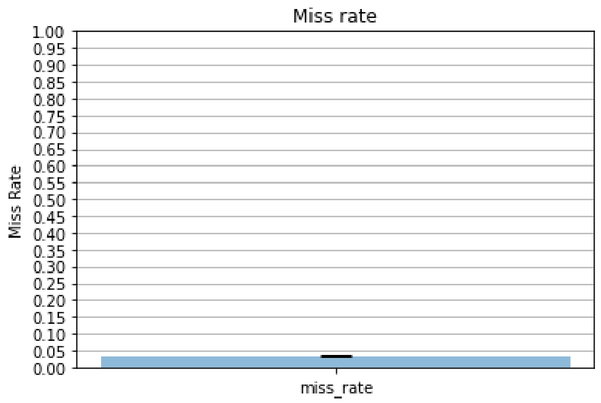


```
              precision    recall  f1-score   support

          -1       0.94      0.94      0.94      1197
           1       0.95      0.95      0.95      1567

    accuracy                           0.95      2764
   macro avg       0.95      0.95      0.95      2764
weighted avg       0.95      0.95      0.95      2764
```

(b) Classification Report of KNN from Scratch



(a) ROC AUC Curve with 10-Fold Implementation



(b) Miss Rate

- **Justification of discrepancies**: Significant Discrepancies have not been observed in our training and testing of models. We tested 30 values of K (1 to 30). Combining 5 feature representations and 5 different distance metrics gave us a combination of 25 models. The baseline method and Scratch implementation gave us similar accuracies of 94.68% and 94.75% respectively.

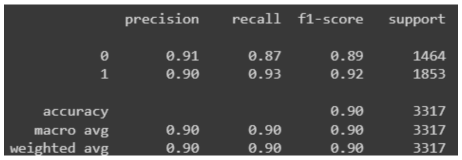
3 MULTILAYER PERCEPTRON (MLP) IMPLEMENTED BY NITYASH GAUTAM

Follow this link for the source code.

- **Algorithm**: A Multilayer Perceptron network consists of three layers (Input layer, Hidden Layer, and Output Layer). Every layer has a certain number of nodes present in it, and except for the input layer, nodes in every layer use a non-linear activation function. MLP functions on a supervised learning technique known as Backpropagation. MLP can distinguish data that is not linearly separable. (https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron)
- **Baseline**: For baseline, Multilayer Perceptron using SciKit Learn has been implemented (https://colab.research.google.com/drive/18kw Apart from accuracy, Miss Rate of the model has been computed to evaluate the model's performance.
- **Potential discrepancies**: The self-implemented algorithm (https://colab.research.google.com/drive/1fWFAYOFhNpOcAD6-SsYZh1nNkWEZjMnS?usp=sharing) and the baseline algorithm are both the same with the same architecture and parameters. Though as reported earlier, the activation function has been changed from 'ReLu' to 'Logistic

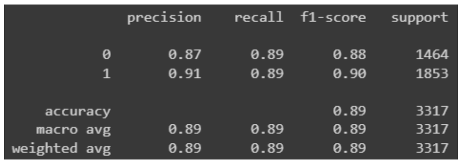
/ Sigmoid', simply due to the implementation constraints of the loss equation corresponding to the 'ReLu' activation function in the self-implementation part. Also, the KFolds cross-validation has been implemented on the baseline model and not on the self-implemented model, because of the dimension mismatch constantly occurring in the passed parameters.

- **Measures of comparison**: Post-training the model, the confusion matrix was computed, which gave the measures of (True Positives, False Positives, True Negatives, and False Negatives). Using these obtained values, the Miss Rate and F1 SCORE are calculated, which gives us an idea of how many Phishing Websites were classified as Non-Phishing by the model.
- **Experimental results**:



```
              precision    recall  f1-score   support

           0       0.91      0.87      0.89      1464
           1       0.90      0.93      0.92      1853

    accuracy                           0.90      3317
   macro avg       0.90      0.90      0.90      3317
weighted avg       0.90      0.90      0.90      3317
```

(a) MLP Scikit Learn Classification Report



```
              precision    recall  f1-score   support

           0       0.87      0.89      0.88      1464
           1       0.91      0.89      0.90      1853

    accuracy                           0.89      3317
   macro avg       0.89      0.89      0.89      3317
weighted avg       0.89      0.89      0.89      3317
```

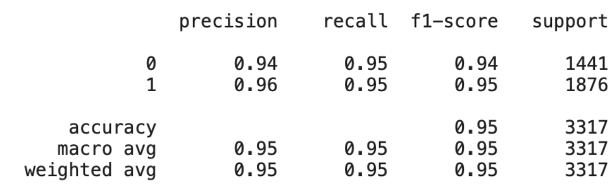
(b) MLP Scratch Implemented model Classification Report

The Miss Rate and F1 Score over the complete dataset has been taken under consideration for the performance comparison of the two implementations. Miss Rate of the baseline and self Implementation are (0.09) and (0.13) respectively and the F1 Score of baseline and self implementation are 0.89 and 0.88 respectively. Hence there is a difference of 44% in Miss Rate and 1.12% in F1 Score.

- **Justification of discrepancies**: Stated above, the baseline implementation works better because of 2 reasons. Firstly, the random initialization of the weights and biases in the self-implementation forced the model to start training with values that required to be updated over large number of epochs, in order to yield the best values for them. Secondly, hyper parameter tuning was also done in the baseline model, where the performance was tested over multiple values of epochs and batch size, which resulted in the better performance.

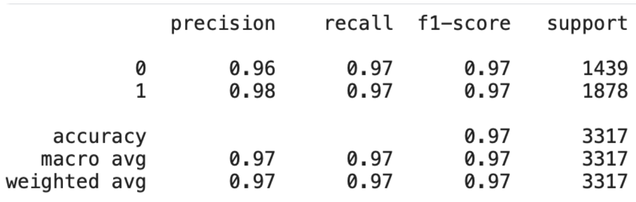
4 RANDOM FOREST CLASSIFIER IMPLEMENTED BY HRITVIK GUPTA

- **Algorithm**: I have used random forest is a estimator that uses averaging to increase accuracy and reduce overfitting after fitting multiple decision tree classifiers to dataset. Moreover, we have implemented 5 Kfold to validate the random forest on dataset.
- **Baseline**: I have used the Random forest classifier from sklearn. And plotted AUC and ROC curve to measure the performance of the classifier on different sub sets of datasets.
- **Potential discrepancies**: The baseline implementation includes some Hyper parameters such as max-Features, Min-sample-leaf, verbose, number of jobs. Thereby, to evaluate the performance of random forest I have used the K-Fold as there will be small discrepancies between the trees in Random forest. So Baseline implementation and from the scratch implementation includes some potential discrepancies.
- **Measures of comparison**: I have used the three different metrics to evaluate the performance the implementation are recall, F1 measure and accuracy. Moreover, I have plotted roc and auc curve of 5 K-Folds on the dataset.
- **Experimental results**:



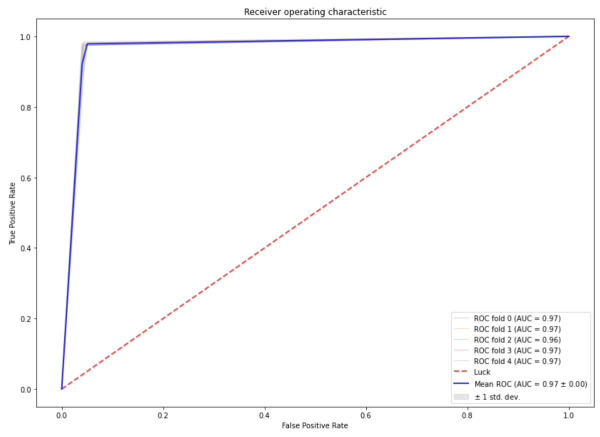
```
              precision    recall  f1-score   support

           0       0.94      0.95      0.94      1441
           1       0.96      0.95      0.95      1876

    accuracy                           0.95      3317
   macro avg       0.95      0.95      0.95      3317
weighted avg       0.95      0.95      0.95      3317
```

(a) Random Forest Scikit Learn Classification Report

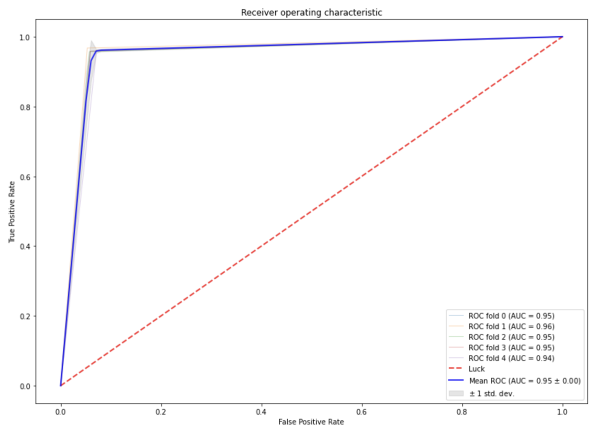


```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97      1439
           1       0.98      0.97      0.97      1878

    accuracy                           0.97      3317
   macro avg       0.97      0.97      0.97      3317
weighted avg       0.97      0.97      0.97      3317
```

(b) Scratch Implemented model Classification Report



(a) Scikit Implemented model ROC AUC Curve with 5 fold implementation



(b) Scratch Implemented model ROC AUC Curve with 5 fold implementation

- **Justification of discrepancies**: There are no as such significant discrepancies. But as we have tested Random forest classifier on 5 k folds of train and test data. And accuracy is comparable and slightly more than that of off the shelf baseline method. In baseline method using scikit learn the accuracy is 95 Percent, and in implmented Random forest get the 97 percent accuracy on same stats.

5 DECISION TREE CLASSIFIER IMPLEMENTED BY SHUBHAM SHARMA

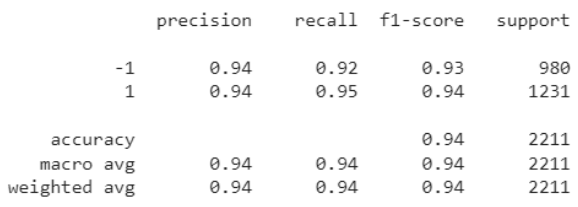
- **Algorithm**: The Decision Tree classification algorithm is used to detect whether a website is phishing or not. The Decision Tree represents all possible solutions based on certain conditions for making a decision. These conditions are based on the features to separate all the labels or classes.
- **Baseline**: For the off-the-self baseline, I have used the scikit-learn decision tree classification model (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html).
  For the visualization of the performance measure, I created a Confusion matrix and then calculate the Accuracy, Precision, Recall, and F1 score. Further, I have also plotted the Receiver Operating Characteristic curve or ROC curve. The area under the ROC curve (ROC AUC) is used for evaluating the performance. The higher the area under the curve (AUC), the better the performance of the model.
- **Potential discrepancies**: In the self-implemented algorithm, I have made just one adjustment to the hyperparameter of the decision tree which is the Number of features to consider for the best split. Now it is changed

from the square root of the maximum number of features to the maximum number of features. Currently, the model is iterating through all the features in order to get the best split. All other parameters are the same as before. The parameters for the decision tree model are as follows:

(1) **Maximum depth of tree** = 25
(2) **Minimum sample split** = 2
(3) **Minimum sample leaf** = 1

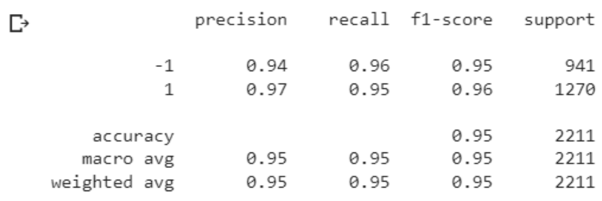
This adjustment was necessary so that the built-in implementation function works similarly to the self-implemented function.

- **Measures of comparison**: In order to compare both implementations of the performance measure, I created a Confusion matrix and then calculate the Accuracy, Precision, Recall, and F1 score for both implementations. Further, I have also plotted the Receiver Operating Characteristic curve or ROC curve by plotting the True Positive (TP) against the False Positive (FP) of 5 K-Folds on the dataset.
- **Experimental results**:



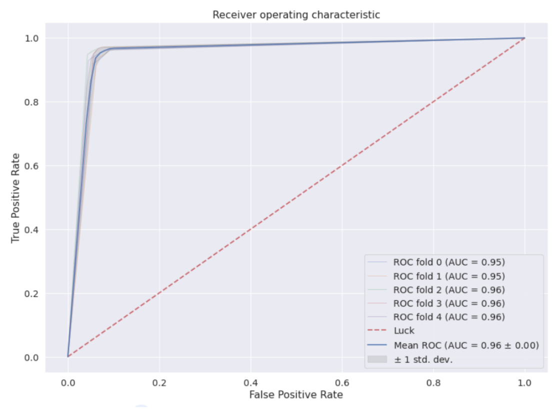
```
              precision    recall  f1-score   support

          -1       0.94      0.92      0.93       980
           1       0.94      0.95      0.94      1231

    accuracy                           0.94      2211
   macro avg       0.94      0.94      0.94      2211
weighted avg       0.94      0.94      0.94      2211
```

(a) Decision Tree Scikit Learn Classification Report

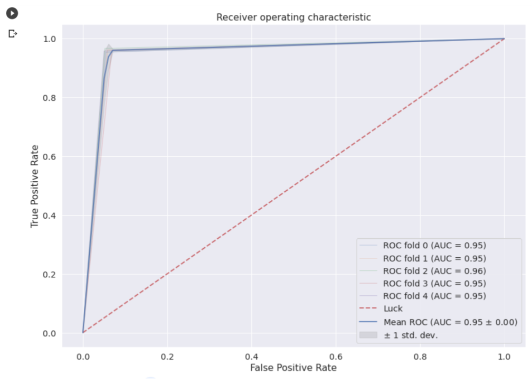


```
              precision    recall  f1-score   support

          -1       0.94      0.96      0.95       941
           1       0.97      0.95      0.96      1270

    accuracy                           0.95      2211
   macro avg       0.95      0.95      0.95      2211
weighted avg       0.95      0.95      0.95      2211
```

(b) Scratch Implemented model Classification Report



(a) Scikit Implemented model ROC AUC Curve with 5 fold implementation



(b) Scratch Implemented model ROC AUC Curve with 5 fold implementation

The off-the-shelf Model Decision Tree Classification has an average test accuracy of 95% with precision, recall and f1 score respectively as 0.94 0.94 0.94. The miss classification rate was approximately 5% and the mean ROC AUC is .96 with a standard deviation of ±1. The self-implemented Model Decision Tree Classification has an average test accuracy of 95.3% with precision, recall and f1 score respectively as 0.95 0.95 0.95. The miss classification rate was of approximately 6.4% and the mean ROC AUC is .95 with a standard deviation of ±1. Comparing both the result shows that both the model are working similar to each other with negligible discrepancy.

- **Justification of discrepancies**: There are no such significant discrepancies between the models. The mean accuracy of the self-implemented model is comparable and slightly more than that of the shelf baseline method. In the baseline method using scikit-learn the accuracy is 94 Percent with ROC AUC .96, whereas in implemented decision tree the mean accuracy is 95 per cent and mean ROC AUC .95 on the same data set.