



Design and Development of a Machine Learning-Based System for Detection of Phishing Websites

Dr. Rajesh Kumar Sharma

Associate Professor

Department of Computer Science and Engineering,
RKDF University Bhopal, Madhya Pradesh, 462036

rajeshsharma.ercs@gmail.com

Abstract: Based on the UCI Phishing Websites dataset, a replicable machine learning system was created to categorise phishing websites. It consists of 11,055 tagged examples and 30 pre-computed website characteristics. Data examination showed 5,785 distinct feature vectors and a lot of recurring patterns. Then, a grouped holdout was used to keep the same feature vectors together while splitting. The experiment contrasted a decision tree, a random forest, and a majority class baseline using a similar training and testing process. The random forest achieved 96.13% accuracy, 95.51% phishing precision, 96.00% phishing recall, 95.75% phishing F1 and 0.9939 ROC AUC on 2,144 heldout data. The confusion matrix yielded 935 properly identified phishing records, 39 missed phishing records, 44 false warnings, and 1,126 correctly categorised real data. Additional threshold, repeated-split and feature-removal analysis further illuminate the stability and limitations of this finding. The provided Python implementation generates the tables, execution outputs and graphs. The programme takes the 30 characteristics that are provided; to categorise a raw URL, an extra matching extraction step and provide external validation would be required.

Keywords: phishing websites, machine learning, random forest, website classification, grouped holdout

1. INTRODUCTION

Phishing sites are fake internet pages that imitate legitimate businesses to trick people into giving up their information. The detection system should differentiate potentially dangerous sites from genuine ones while retaining visibility of both missed threats and superfluous alerts. The challenge is related to browser safety, organisational filtering and user-facing warnings, however an experimental score only makes sense when the dataset and assessment technique are presented properly. Website characteristics refer to the location, domain and structure of the page; these signals are helpful inputs to statistical classifiers (Mohammad et al., 2012).

This article provides a full, executable experiment with measurable outcomes. The proposed contribution is a transparent grouped assessment and implementation on a well-known benchmark. It does not assume that a classifier trained on precomputed historical characteristics can automatically analyse a freshly inserted URL without feature extraction.

A static list of known phishing addresses may detect sites that have been reported, but it cannot, by itself, make a forecast for an unseen address. Feature-based categorisation aims to quantify

attributes of a prospective website. Automated feature extraction and phishing categorisation from website or URL signals has been explored in prior work (Mohammad et al., 2012; Sahingoz et al., 2019). URL-only and page-content systems need different inputs and hence their performances are not directly comparable.

In practice, the motivation for automation is the sheer quantity of possible websites and the impracticality of checking every page by hand. The classifier considered here takes 30 encoded characteristics per record and outputs a label of phishing or authentic. Its architecture makes training, prediction, comparison and error checking reproducible. A false negative is a phishing example that is awarded the valid label; a false positive is an unneeded phishing warning for a real record.

1.1 Objective of the Study

- (a) to develop a working training and evaluation pipeline;
 - (b) to compare a random forest with a single decision tree and a majority baseline;
 - (c) to prevent exact feature patterns from appearing across the train/test boundary; and
 - (d) to report confusion counts, threshold sensitivity and variability across grouped partitions.
- These goals are related to quantifiable results created by the provided code. The essay is arranged as per request in the order of methodology, system development, results, discussion, conclusion and future scope.

The research is based on a publicly available dataset, not on a deployed browser plugin. This option makes the experimental inputs replicable but sets a clear limit around what the system is capable of doing at now. A website is labelled if it has the requisite 30 feature values. Safely and reliably getting such numbers from a live website is a different technical challenge, particularly for features that rely on domain records, or page contents.

2. MATERIALS AND METHODS

2.1 Dataset and Preprocessing

The benchmark was the Phishing Websites dataset in UCI Machine Learning Repository (Mohammad & McCluskey, 2012). This data downloaded has an ARFF structure with 11,055 observations, 30 input variables and a Result column. The baseline target utilises -1 for phishing and 1 for real. For the purpose of assessment in this research, phishing was coded as 1 and genuine as 0. All feature values were decoded from categorical ARFF bytes to integer values. The programme checks for the existence of 31 columns, the lack of missing values in the inputs and labels, and the presence of two predicted target labels.

Table 1. Dataset composition and quality checks

Item	Observed value	Interpretation
Total records	11,055	All supplied observations
Phishing records	4,898 (44.31%)	Positive class
Legitimate records	6,157 (55.69%)	Negative class
Input features	30	Precomputed attributes
Distinct feature vectors	5,785	Patterns before split
Exact duplicate full rows	5,206	Repeated features and labels

Feature patterns with conflicting labels	64	Same inputs, both labels
Missing values	0	No imputation required

We counted duplicate complete rows by utilising all 31 columns and estimated the number of different feature vectors by using just the 30 inputs. The two counts answer separate questions: a feature vector may appear with both labels even if a whole row is not similar. The 64 conflicting patterns were kept. Maybe this is simpler to do after looking at the labels, and the benchmark would alter. Possible ambiguity created by prediction mistakes is addressed in interpreting the prediction errors.

In this experiment, the tree models do not require any standardisation, since their split rules act on the encoded feature values. No rows were lost in silence. The historic dataset used in the download is credited in the references; the research does not claim anything about the current prevalence of phishing websites.

2.2 Feature Extraction

The UCI file already has the characteristics extracted. The code here only reads the 30 numbers, it does not connect to a website or deduce them from a text URL. The feature extraction in the proposed system is thus specified by the data preparation interface before it, which would need a compatible extractor to output the same variables, coding rules and column order as the training dataset. If any of those standards differ, then a forecast may be incorrect even if a model delivers a numeric result.

Table 2. Examples of the supplied feature groups

Feature family	Dataset examples	Potential input needed
Address structure	having_IP_Address, URL_Length, Prefix_Suffix	URL text
Domain and security	having_Sub_Domain, SSLfinal_State, age_of_domain	URL, certificate, domain metadata
HTML and links	URL_of_Anchor, Request_URL, Links_in_tags	Retrieved webpage content
Interaction cues	on_mouseover, RightClick, popUpWidnow	Page script or markup
Popularity and reports	web_traffic, Page_Rank, Statistical_report	External or historical sources

A table collects chosen columns by name; it is an implementation assistance, not a statement that the research assessed how a new website provides each field. Some inputs need the site to be fetched or information from other parties and cannot be rebuilt from the URL string alone. Before a live extractor is created, the precise encoding of each field should be controlled by the original dataset documentation. Past work also differentiates content-based detection and richer feature combinations (Zhang et al., 2007; Xiang et al., 2011).

The programme utilises the names of features present in the ARFF file and separates the Result field from predictors. This prevents inadvertent target leak. The row indices are not fed into the model. After training, feature significance scores are generated to explain how the fitted

forest exploited the provided inputs. They do not demonstrate causation or indicate that a field alone is sufficient to detect a fake site.

2.3 Machine Learning Algorithms

Three algorithms were chosen to establish a basic baseline and two models capable of learning relationships between discrete elements of websites. The DummyClassifier with the most_frequent technique will always predict the bigger class in its training partition. It provides a benchmark for what occurs when one doesn't learn from website attributes. The decision tree separates the feature space with consecutive rules, and is quite straightforward to observe. min_samples_leaf = 2 avoids leaves supported by just one training observation.

The random forest is an ensemble of 200 decision tree models trained using the randomisation of the bootstrap samples and the candidate characteristics. Its class_weight is balanced_subsample, minimum leaf size is 2 and random_state is 42. The basic random forest formulation may be found in Breiman (2001). The selected setup is hard-coded so as to be able to reproduce all provided data. we did not run any test set optimising of these settings

Table 3. Experimental classifier configuration

Classifier	Key settings	Experimental role
Majority baseline	strategy = most_frequent	Reference prediction without feature learning
Decision tree	min_samples_leaf = 2; seed = 42	Single-tree comparison
Random forest	200 trees; leaf $\geq$ 2; balanced_subsample; seed = 42	Proposed model

The three approaches are compared under the same test records and the same training records. Now that common partition is key to making sense of their respective perplexity counts. Their training scores are not presented as proof of performance, as the subject of interest is how each model classifies held-out feature patterns.

2.4 Proposed System Architecture

The system is built as a process of data checks, partitioning, fitting the model, and evaluation. The ARFF reader provides the mapped target and 30 input columns. GroupShuffleSplit keeps matched patterns together by using a hash of each entire feature vector as the group identifier. This split technique assigns groups, it doesn't guaranty a number of records in the test partition (scikit-learn developers).

Table 4. Processing stages and their outputs

Stage	Operation	Output
Input	Load and validate the ARFF file	Integer-valued feature matrix and binary target
Grouping	Hash all 30 input fields	One group ID per feature pattern
Partition	Create a single held-out grouped split	Training and test indices
Fitting	Fit baseline, tree and forest	Three trained classifiers
Prediction	Predict labels and probabilities	Class decisions and scores
Reporting	Calculate metrics and draw plots	CSV, JSON and PNG outputs

The primary test of separation is that the sets of group IDs in training and test are disjoint. The following assessment is just scored on the test labels and is not fitted. The original records remain untouched, duplicates and contradictory labels are kept, just their positioning is regulated. If all feature definitions and encodings are consistent, this architecture may be reused with fresh externally provided data.

The programme produces machine readable output and figures. The separation allows a reader to compare a number in the article to the CSV file in the article, and to see the confusion matrix as a plot. The design offers a clear input interface for a research prototype. It has no user facing web form, no secure live crawler or continuous updating service.

2.5 Performance Evaluation Metrics

The positive class is phishing. True positive (TP) counts accurately identified phishing instances; false negative (FN) counts missed phishing cases. True negative (TN) counts properly accepted genuine records; false positive (FP) counts wrongly flagged legitimate records. The same definition is applied to all tables and figures.

Table 5. Metrics used in the experiment

Metric	Definition	Question addressed
Accuracy	$(TP + TN) / (TP + TN + FP + FN)$	How many total predictions are correct?
Phishing precision	$TP / (TP + FP)$	How many flagged cases are phishing?
Phishing recall	$TP / (TP + FN)$	How many phishing cases are found?
Phishing F1	$2 \times \text{precision} \times \text{recall} / (\text{precision} + \text{recall})$	How do precision and recall balance?
ROC AUC	Area under TPR versus FPR curve	How are positives ranked across thresholds?

Accuracy alone may obscure the nature of errors: two models with comparable accuracy may miss different numbers of phishing incidents. Precision and recall treat phishing as the positive class, which directly reflects the alert quality and detection coverage. F1 is a combination of them but not a replacement for the confusion matrix. ROC AUC is based on prediction probabilities, evaluates the ranking on the chosen test sample, and is not the fraction of true binary predictions.

The default categorisation choice is made using the model's predict technique with a traditional probability cut-off around 0.50. Another threshold exercise applies 0.30 to 0.70 to the same held-out probability scores. These potential thresholds are descriptive, and using one based on its performance on this test set and portraying it as an impartial final estimate would be misusing the holdout. Hence, the initial categorisation of 0.50 is the key result.

3. SYSTEM DEVELOPMENT

3.1 Model Training and Testing

After the goal is transformed, training and testing start. Pandas hash_pandas_object creates group IDs from the whole feature matrix of 30 columns but not row indices. Indices are provided by one GroupShuffleSplit with test_size = 0.20 and random_state = 42. This resulted

in 8911 training and 2144 test observations. The test sample had 974 phishing and 1,170 real records. The split is established on unique groups, hence the size at the record level may not be exactly 20%.

Table 6. Primary training and test allocation

Partition	All records	Phishing	Legitimate
Training	8,911	3,924	4,987
Held-out test	2,144	974	1,170
Total	11,055	4,898	6,157

The majority baseline is trained on the training target. The two tree models are trained on the same X_{train} and y_{train} . Generates predictions and positive class probabilities for X_{test} only. There is no preprocessing transform calculated on the whole dataset in the source. The group overlap is evaluated before the fitting of the model, which is particularly important as the source includes thousands of exact copies.

An precise feature-vector group is a valuable control, but it is not a domain or campaign id. Two URLs within the same attack campaign may yet have separate vectors and may lie on opposing sides of the divide. This benchmark does not give the underlying URLs or collection dates, and hence campaign-disjoint and temporal assessments cannot be performed honestly with the provided file.

3.2 Python Implementation

The executable implementation is in the attached file `phishing_detection.py`. Receives path to ARFF file and output directory. Checks labels. Trains three methods. Writes measurement files. Here are some code snippets illustrating the essential operations. The `.py` file is now ready and may be used for rerunning the experiment The rationale of the `.py` file is explained in the excerpts below.

Listing 1. Input parsing and target mapping

```
records, _ = arff.loadarff(path)
frame = pd.DataFrame(records)
frame = frame.map(
    lambda value: int(value.decode())
    if isinstance(value, bytes) else int(value)
)
features = frame.drop(columns="Result")
target = (frame["Result"] == -1).astype(int)
```

The parser reads the original column names so that future plots recognise features from the source. The target mapping is explicit: a phishing case is positive even if its original UCI Result value is -1. This avoids an inversion of the confusion matrix labels during the interpretation.

Listing 2. Grouped holdout and its separation assertion

```
groups = pd.util.hash_pandas_object(
    features, index=False
).to_numpy()
```

```
train_idx, test_idx = next(
    GroupShuffleSplit(
        n_splits=1, test_size=0.20, random_state=42
    ).split(features, target, groups)
)
assert set(groups[train_idx]).isdisjoint(groups[test_idx])
```

The numeric group hash is just utilised for splitting and not used as a characteristic for prediction. The same group is given the same feature vector, even if the relevant rows have contradicting goal labels.

Listing 3. Proposed random forest classifier

```
model = RandomForestClassifier(
    n_estimators=200,
    min_samples_leaf=2,
    class_weight="balanced_subsample",
    random_state=42,
    n_jobs=-1
)
model.fit(features.iloc[train_idx], target.iloc[train_idx])
prediction = model.predict(features.iloc[test_idx])
```

The identical split is used by a different decision tree and a majority baseline. The parameter selections listed here are the same as those used in the actual programme run and in the Results section. The displayed values should be regenerated instead of manually copying whenever the training file or package versions change.

Listing 4. Scoring and saved evidence

```
tn, fp, fn, tp = confusion_matrix(
    y_test, prediction, labels=[0, 1]
).ravel()
accuracy = accuracy_score(y_test, prediction)
precision = precision_score(y_test, prediction)
recall = recall_score(y_test, prediction)
f1 = f1_score(y_test, prediction)
probability = model.predict_proba(x_test)[: , 1]
roc_auc = roc_auc_score(y_test, probability)
```

The genuine programme additionally produces metrics.csv, thresholds.csv, repeated_group_splits.csv, feature_sensitivity.csv, feature_importances.csv, run_info.json and the results given below. The order of confusion matrix [0, 1] is set so that its cells may be compared immediately to the provided TN, FP, FN and TP values.

3.3 Phishing Website Prediction Process

For an observation that is ready, prediction needs one row with the same 30 feature names and compatible encoded values as the training matrix. The fitted forest applies its trees to the row,

gets the average of their class support and generates a class label and a score for the data. The higher score might be compared to a threshold to trigger an alarm. The script we have written exhibits this method on held-out rows from the same dataset, and provides aggregate evidence rather than providing an interactive endpoint.

Table 7. Prediction process and interpretation

Step	Input or action	What the user receives
1	Validate a 30-field feature record	Accepted or rejected input
2	Apply the trained forest	Phishing probability score
3	Use a specified threshold	Phishing or legitimate classification
4	Record the decision context	Chosen threshold and model version
5	Evaluate with known labels when available	Confusion counts and metrics

If someone gives a URL just, step 1 can not be done with this programme alone. There are features that relate to certificates, site linkages, domain age or external signals. Assigning assumed values to the missing columns would result in an inaccurate seeming forecast. Future URL interface should reproduce feature descriptions of benchmark, and extract failures clearly. The code may be executed from the command line once the numpy, pandas, scipy, scikit-learn and matplotlib are installed. Put Training Dataset.arff in the same directory as phishing_detection.py, then run: `python phishing_detection.py --data "Training Dataset.arff" -out results`. The present execution is done using Python 3.12 and scikit-learn 1.8.0. The CSV spreadsheets saved have more decimal places than the figures rounded above in this post.



Figure 1. Values recorded from the completed execution, formatted for legibility

4. RESULTS AND ANALYSIS

4.1 Performance of Machine Learning Models

Table 8 shows the three models on the same group-separated holdout of 2144 records. The majority baseline only predicts the valid class, as the legitimate observations are more frequent

than the others in its training partition. Its accuracy of 0.5457 is not indicative of phishing detection, its phishing recall is zero. Both feature based models get much greater recall and F1 on this test.

Table 8. Model performance on the same grouped test sample

Model	Accuracy	Phishing precision	Phishing recall	Phishing F1	ROC AUC
Majority baseline	0.5457	0.0000	0.0000	0.0000	0.5000
Decision tree	0.9501	0.9455	0.9446	0.9450	0.9711
Random forest	0.9613	0.9551	0.9600	0.9575	0.9939

The random forest gains approximately 1.12 percentage points of accuracy and 1.54 percentage points of phishing recall over the decision tree in this particular split. The corresponding phishing F1 increases from 0.9450 to 0.9575. These differences are descriptive for this run. No confidence interval or formal significance test was calculated for the difference between classifiers.

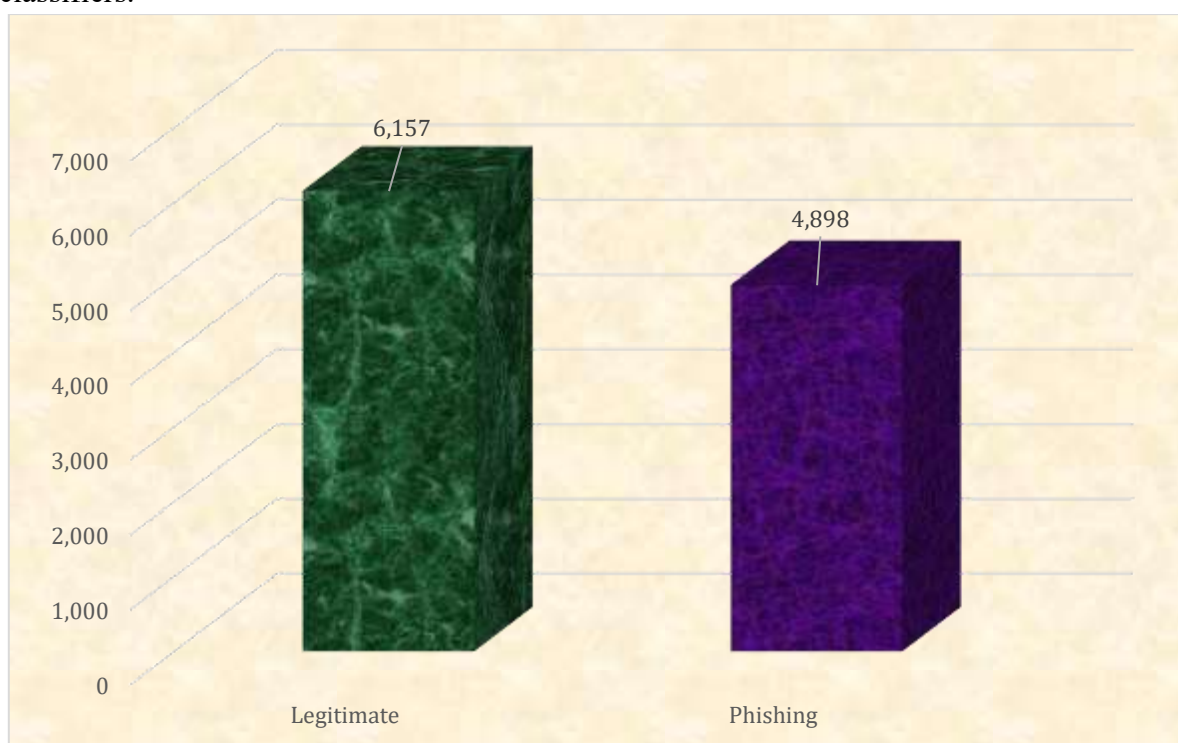


Figure 2. Observed class distribution of all 11,055 records

The initial class distribution is relatively imbalanced and no single label dominates. However, a majority predictor provides falsely acceptable accuracy if its inability to identify phishing is masked. This issue is addressed by reporting accuracy together with positive-class scores.

4.2 Confusion Matrix and Classification Results

The forest recognised 935 phishing records accurately, classifying 39 phishing records as real. Of the valid entries, 1126 were appropriately identified and 44 were marked as phishing. So on the test set of 2,144 records the model produced 83 mistakes. The practical ramifications of a missed phishing site vs. an unwanted warning are somewhat different; the four counts should be preserved whenever results are compared.

Table 9. Random forest confusion counts

Actual category	Predicted legitimate	Predicted phishing	Total
Legitimate	1,126 (TN)	44 (FP)	1,170
Phishing	39 (FN)	935 (TP)	974
Total	1,165	979	2,144

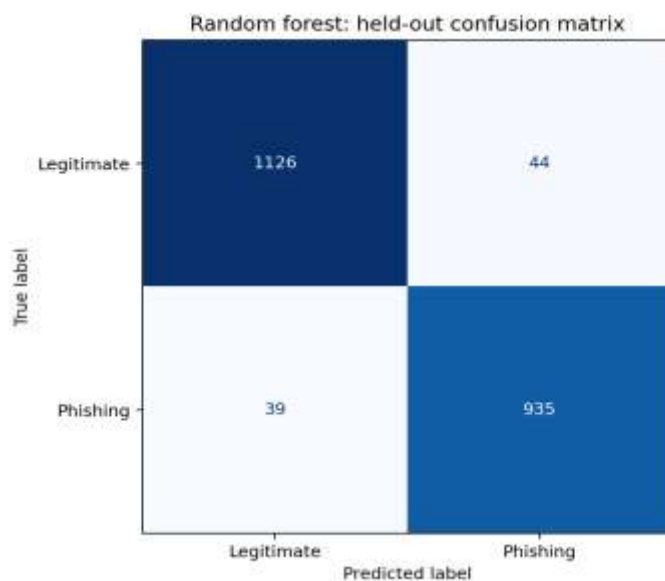


Figure 3. Random forest confusion matrix from the held-out predictions

Phishing recall is 0.9600 rounded from $935 / 974$. $935 / (935 + 44) = 0.9551$ is the accuracy of phishing. This figure allows one to infer these computations without the need of a stand-alone accuracy score. The continuous phishing scores provide a second view of classifier behaviour. The ROC curve in Figure 4 plots the true positive rate against the false positive rate as the classification threshold changes. Its 0.9939 area summarises ranking on the held-out sample; it is distinct from the 0.9613 binary accuracy measured at the default decision rule.

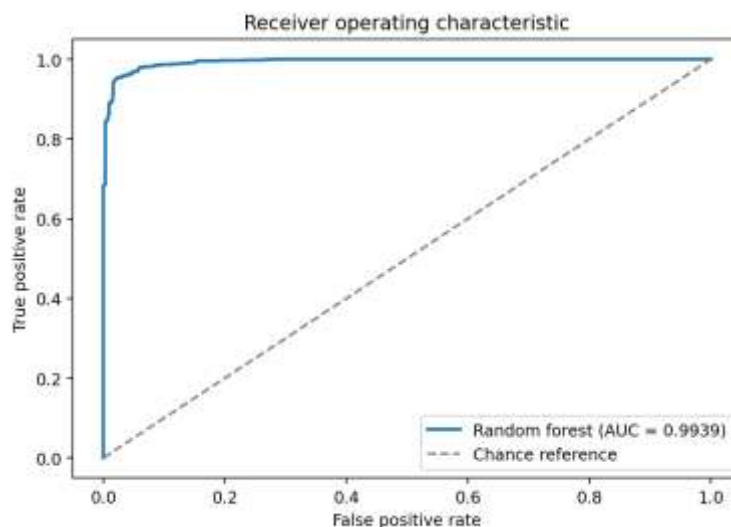


Figure 4. Held-out receiver operating characteristic curve

The precision-recall curve in Figure 5 illustrates the trade-off between detection coverage and alert confidence. Its dashed reference is sample specific, reflecting the percentage of phishing in this test set. The curves do not guaranty the same ranking behaviour if phishing tactics, website content or class predominance changes after the historical dataset was gathered.

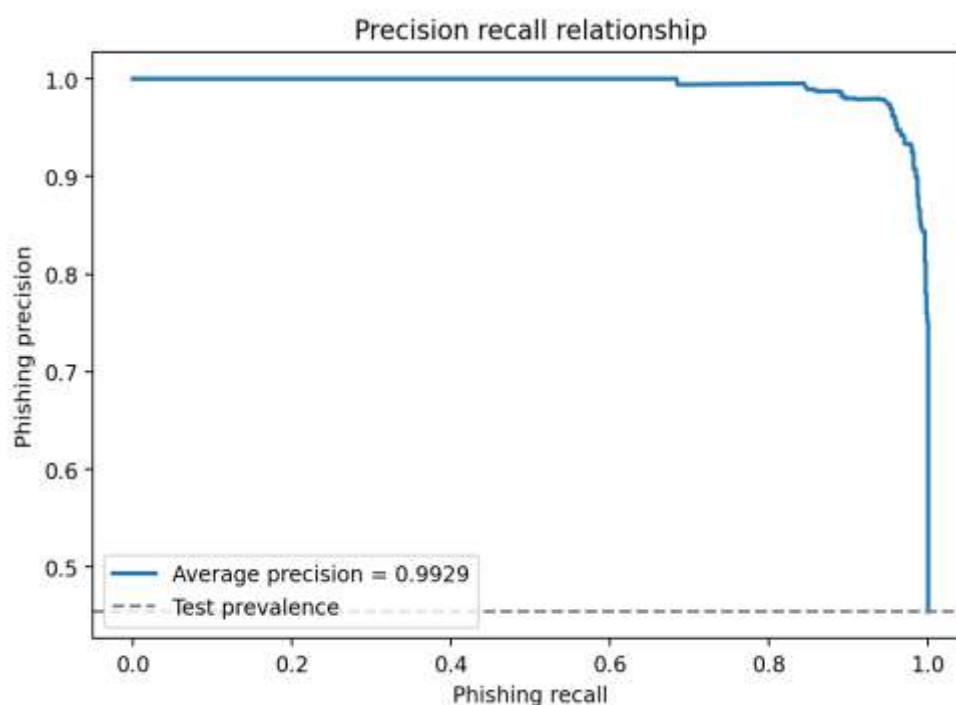


Figure 5. Held-out precision–recall curve and test prevalence reference

4.3 Comparative Analysis of Algorithms

The model comparison should consider the quantity and the kind of errors. The decision tree missed 54 phishing incidents and created 53 false alarms, the forest missed 39 and caused 44 false alarms. This division was reduced by the forest to both sorts of mistake. The majority baseline didn't catch any phishing scenario. The confusion counts help explain why the models had varied recall and accuracy levels even when all three were tested on the identical data .

Table 10. Error counts by classifier

Classifier	TN	FP	FN	TP	Total errors
Majority baseline	1,170	0	974	0	974
Decision tree	1,117	53	54	920	107
Random forest	1,126	44	39	935	83

In Table 11 we provide a descriptive threshold check for the forest using the original held out set scores. At 0.30, the number of missed phishing incidents drops to 13 while false warnings increase to 110. False warnings fell to 18 while missed phishing incidents rose to 91 at 0.70. This trade-off implies that the desired threshold will rely on the relative weight that a future application would place on these two implications.

Table 11. Forest threshold sensitivity on the held-out set

Threshold	Precision	Recall	F1	False alerts	Missed phishing
0.30	0.8973	0.9867	0.9399	110	13
0.40	0.9307	0.9795	0.9545	71	20
0.50	0.9551	0.9600	0.9575	44	39
0.60	0.9787	0.9435	0.9608	20	55
0.70	0.9800	0.9066	0.9419	18	91

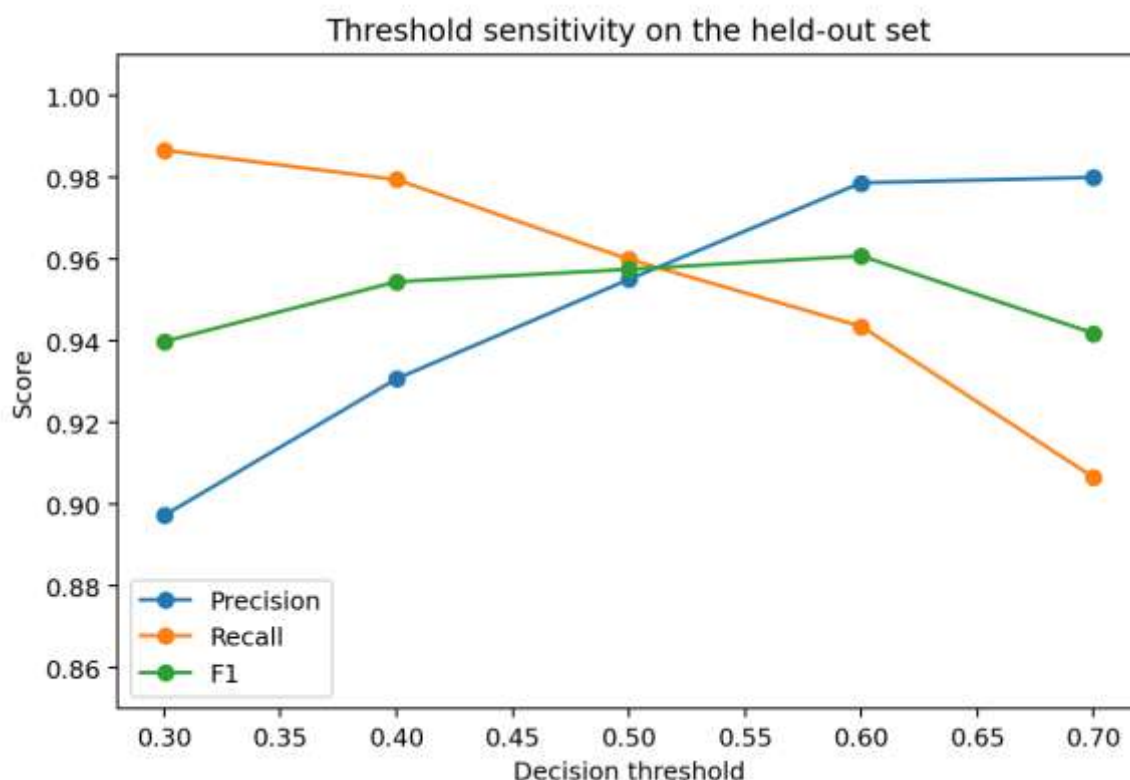


Figure 6. Precision, recall and F1 at five illustrative thresholds

The threshold graph shows variations on the same data used for the main assessment. It is meant to be used for understanding operational trade-offs and not to pick and certify a new final threshold from the same test set. If threshold selection is to be a goal then an extra validation division inside training data is required followed by an undisturbed final assessment. The main 0.50 result in Table 8 is not subject to this selection problem.

The forest score is a model result, not necessarily a calibrated real-world likelihood. An operational warning may have a variable false alert rate if installed in a location where phishing is less prevalent, or attacker activity changes. The calibration assessment and determination of an alert threshold in a real-world scenario need distinct representative data and a defined error-cost policy.

The same random forest configuration was trained five separate times on different grouped holdouts to test the dependency on the specific grouped allocation. The seeds were 21, 42, 63,

84 and 105. The test sets are different and may overlap; the five measurements are to be interpreted as a sensitivity study, and not as five statistically separate deployment experiments.

Table 12. Random forest results across five grouped allocations

Seed	Test N	Phishing N	Accuracy	Precision	Recall	F1
21	2,203	1,000	0.9496	0.9674	0.9200	0.9431
42	2,144	974	0.9613	0.9551	0.9600	0.9575
63	2,226	1,012	0.9573	0.9562	0.9496	0.9529
84	2,224	955	0.9591	0.9509	0.9539	0.9524
105	2,267	1,035	0.9502	0.9502	0.9401	0.9451

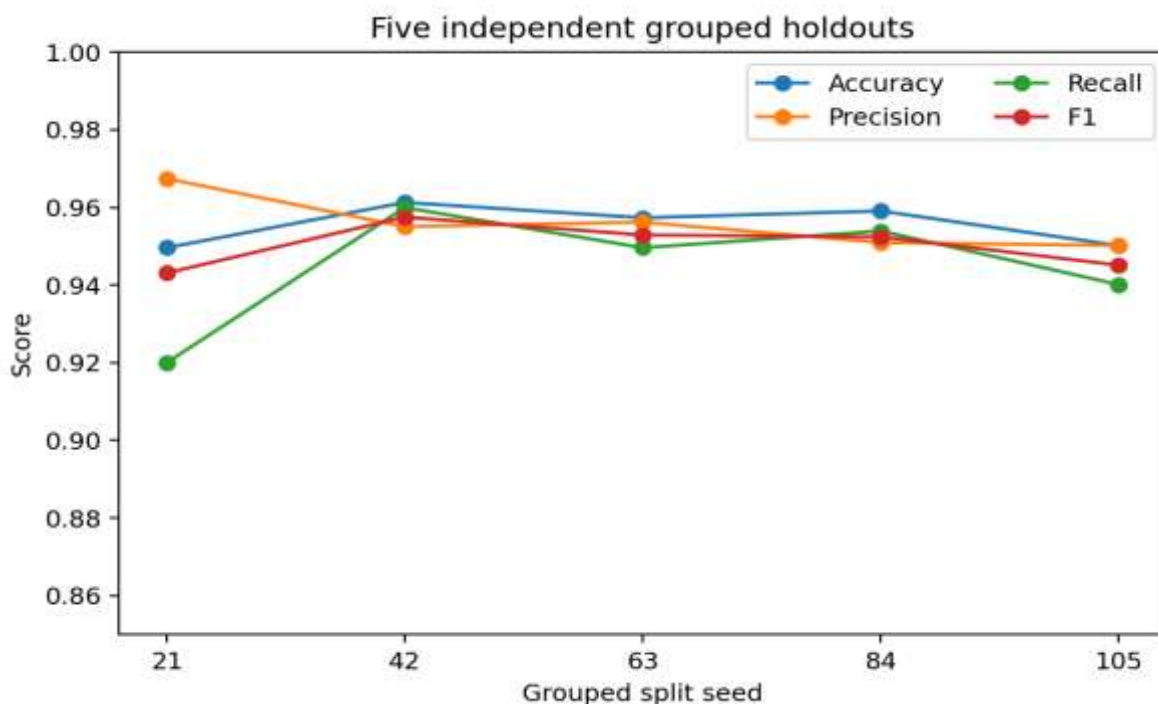


Figure 7. Performance sensitivity across five group-separated splits

The accuracy of these divides varies from 0.9496 to 0.9613 while the phishing recall runs from 0.9200 to 0.9600. The means are accuracy of 0.9555 and recall of 0.9447; The seed-42 run is the pre-specified main table outcome; the other allocations demonstrate that its 0.9600 recall is at the top of this observed range.

The forest impurity-based relevance ranking is based on the most often or successfully utilised given characteristics in the tree splits. The top two are SSLfinal_State (0.3452) and URL_of_Anchor (0.2423). The presence of impurities may cause associated qualities to have unequal weights, such that the ranking becomes a description rather than a causal explanation.

Table 13. Ten largest forest impurity importances

Feature	Importance
SSLfinal_State	0.3452
URL_of_Anchor	0.2423
web_traffic	0.0734
having_Sub_Domain	0.0669

Prefix_Suffix	0.0473
Links_in_tags	0.0439
SFH	0.0202
Request_URL	0.0181
Domain_registration_length	0.0162
Links_pointing_to_page	0.0154

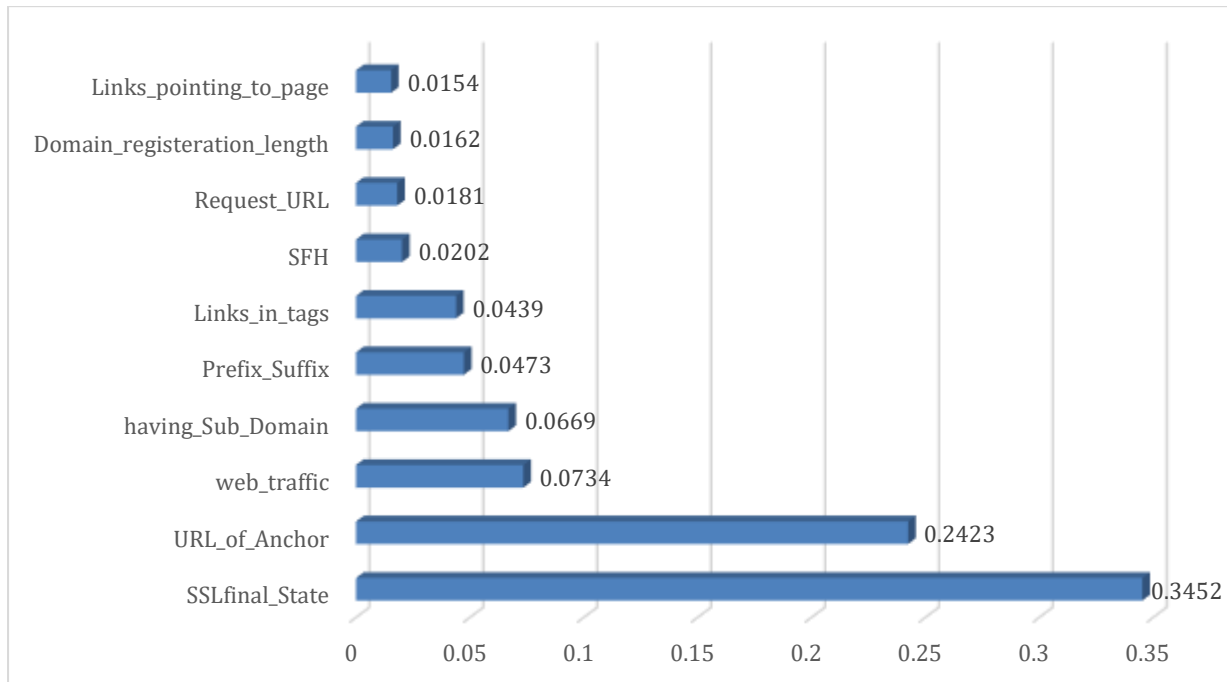


Figure 8. Ten leading input attributes in the primary fitted forest

The identical forest setup was trained on a fixed feature-removal experiment sans SSLfinal_State and URL_of_Anchor. Performance subsequently dropped from 0.9613 to 0.8745 accuracy and from 0.9575 to 0.8676 phishing F1. This is a sensitivity result on a single benchmark split. The experiment is exploratory and should not be seen as a prospectively verified effect, since the two excluded fields were chosen after looking at the fitted significance scores.

Table 14. Exploratory feature-removal comparison

Feature set	Features	Accuracy	Phishing recall	Phishing F1
All supplied features	30	0.9613	0.9600	0.9575
Remove two leading fields	28	0.8745	0.9045	0.8676

The magnitude of the drop highlights an implementation issue: a real system that cannot recreate these inputs may behave considerably differently than the benchmark. It also helps explain why it would be misleading to describe this trained model as a raw-URL detector. Some important attributes are connected to the page content or security status. To get similar values, you may need to get the page and its metadata.



All model comparisons above are on the same historical data. The model score is proof of classification performance under the feature generation and group split of that dataset, not proof of performance versus unknown current sites. Repeating the test using independently acquired URLs, extraction rules, dates and domain groups might reinforce the external interpretation.

5. DISCUSSION

The primary finding is that the 200-tree random forest beat a single tree and a majority baseline in terms of stronger phishing recall and fewer overall mistakes on a group-separated holdout. The detection result is tangible with its 935 true positives and 39 false negatives. The difference between forest and tree was seen on the identical test data, which supports a direct descriptive comparison. This is not a proof that random forests always beat alternatives, since rankings might vary depending on feature definitions, partitions, updates or hyperparameters.

There is a discernible source of optimism in this file and the group policy handles it. Out of 11,055 observations, there are 5,785 different input vectors and more than five thousand precise full-row duplicates. A classifier benefits from previously observed patterns while it is intended to be graded on unseen ones, when matching vectors appear in both groups. The split blocks chosen have the same overlap. The 64 feature patterns that still have contradicting labels show that the 30 predictors do not answer every choice in the recorded data.

Results also reveal a trade-off between false alarms and missing assaults. Lower thresholds resulted in higher recall but also more false positives in the observed test sample. The 0.50 rule provides a clear starting point but selection of thresholds for individual applications in the future would require separate validation data. Here, the forest's high ROC AUC captures the ranking, but it does not obviate the need to consider the real warning count that would be anticipated in practice.

The methodological limitations include the age of the dataset, missing raw URLs, no fields for domain or collection date, and no external validation. The feature names of the source dataset are known, but the present implementation does not create all the information needed for a freshly visited site. Several important predictors rely on information beyond the content of a URL. Any system interface suggesting differently would overestimate the work done. Other phishing systems have incorporated content and richer characteristics purposefully, emphasising the need to describe the evidence consumed by each detector (Zhang et al., 2007; Xiang et al., 2011).

6. CONCLUSION

In this study, a working system was constructed for the classification of 30 pre-computed website features in the UCI Phishing Websites dataset. The holdout was group divided, thus the training and test sets were not similar in terms of input patterns. The random forest accurately categorised 2,061 of 2,144 held-out data with 96.13% accuracy, 95.51% phishing precision, 96.00% phishing recall and 95.75% phishing F1. The post contains data source, algorithms, code extracts, real tables and figures required to audit those findings.

The conclusion is constrained by the historical benchmark and the documented characteristics. The model can only assess a raw website address after it has been fed into a suitable extractor. The repeated split and feature removal tests indicate good performance across numerous allocations and a noticeable dependency on the features provided.

7. FUTURE SCOPE

The next phase in development is to build and validate an extractor for each of the 30 fields from a live website under controlled collecting settings. Its results should be compared to the coded definitions in the dataset and should be used to identify missing or unavailable fields not to fill in values. A new corpus should keep raw URLs, resolved domains, collection dates and labels so that tests may filter similar campaigns and utilise a later time period than training. A future research might evaluate URL-only, webpage-content and mixed feature sets; pick decision thresholds within the training data; estimate calibration; analyse latency; and provide external accuracy, recall and false alarm counts. The repeated group holdouts in this paper are a check on the internal sensitivity of the method, not a substitute for such an independent review. You should test explicitly its behaviour on unavailable sites, redirection, changing content, missing information, before releasing a user-facing warning system.

REFERENCES

- [1] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
- [2] Mohammad, R., & McCluskey, L. (2012). *Phishing Websites* [Data set]. UCI Machine Learning Repository. <https://doi.org/10.24432/C51W2X>
- [3] Mohammad, R. M., Thabtah, F., & McCluskey, L. (2012). An assessment of features related to phishing websites using an automated technique. In *2012 International Conference for Internet Technology and Secured Transactions* (pp. 492–497). IEEE. <https://ieeexplore.ieee.org/document/6470857>
- [4] Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications*, 117, 345–357. <https://doi.org/10.1016/j.eswa.2018.09.029>
- [5] scikit-learn developers. (n.d.). *GroupShuffleSplit* [Software documentation]. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupShuffleSplit.html
- [6] Xiang, G., Hong, J., Rose, C. P., & Cranor, L. (2011). CANTINA+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security*, 14(2), Article 21. <https://doi.org/10.1145/2019599.2019606>
- [7] Zhang, Y., Hong, J. I., & Cranor, L. F. (2007). CANTINA: A content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web* (pp. 639–648). ACM. <https://doi.org/10.1145/1242572.1242659>
- [8] Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Systems with Applications*, 117, 345–357. DOI: 10.1016/j.eswa.2018.09.029.
- [9] Chiew, K. L., Tan, C. L., Wong, K. S., Yong, K. S. C., & Tiong, W. K. (2019). A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. *Information Sciences*, 484, 153–166. DOI: 10.1016/j.ins.2019.01.064.
- [10] Yang, P., Zhao, G., & Zeng, P. (2019). Phishing website detection based on multidimensional features driven by deep learning. *IEEE Access*, 7, 15196–15209. DOI: 10.1109/ACCESS.2019.2892066.
- [11] Jain, A. K., & Gupta, B. B. (2018). Towards detection of phishing websites on client-side using machine learning based approach. *Telecommunication Systems*, 68, 687–700. DOI: 10.1007/s11235-017-0414-0.



- [12] Aljofey, A., Jiang, Q., Rasool, A., Chen, H., Liu, W., Qu, Q., & Wang, Y. (2022). An effective detection approach for phishing websites using URL and HTML features. *Scientific Reports*, 12, 8842. DOI: 10.1038/s41598-022-10841-5.
- [13] Ahammad, S. K. H., Kale, S. D., Upadhye, G. D., Pande, S. D., Babu, E. V., Dhumane, A. V., & Jang Bahadur, D. K. (2022). Phishing URL detection using machine learning methods. *Advances in Engineering Software*, 173, 103288. DOI: 10.1016/j.advengsoft.2022.103288.
- [14] Hannousse, A., & Yahiouche, S. (2021). Towards benchmark datasets for machine learning based website phishing detection: An experimental study. *Engineering Applications of Artificial Intelligence*, 104, 104347. DOI: 10.1016/j.engappai.2021.104347.
- [15] Al-Sarem, M., Saeed, F., Al-Mekhlafi, Z. G., Mohammed, B. A., Al-Hadhrani, T., Alshammari, M. T., Alreshidi, A., & Alshammari, T. S. (2021). An optimized stacking ensemble model for phishing websites detection. *Electronics*, 10(11), 1285. DOI: 10.3390/electronics10111285.
- [16] Prasad, A., & Chandra, S. (2024). PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. *Computers & Security*, 136, 103545. DOI: 10.1016/j.cose.2023.103545.
- [17] Xiang, G., Hong, J., Rose, C. P., & Cranor, L. (2011). CANTINA+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security*, 14(2), Article 21, 1–28. DOI: 10.1145/2019599.2019606.
- [18] Mohammad, R. M., Thabtah, F., & McCluskey, L. (2014). Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, 25(2), 443–458. DOI: 10.1007/s00521-013-1490-z.
- [19] Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based associative classification data mining. *Expert Systems with Applications*, 41(13), 5948–5959. DOI: 10.1016/j.eswa.2014.03.019.
- [20] Marchal, S., François, J., State, R., & Engel, T. (2014). PhishStorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4), 458–471. DOI: 10.1109/TNSM.2014.2377295.
- [21] Marchal, S., Saari, K., Singh, N., & Asokan, N. (2016). Know your phish: Novel techniques for detecting phishing sites and their targets. 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), 323–333. DOI: 10.1109/ICDCS.2016.10.
- [22] Zouina, M., & Outtaj, B. (2017). A novel lightweight URL phishing detection system using SVM and similarity index. *Human-centric Computing and Information Sciences*, 7, 17. DOI: 10.1186/s13673-017-0098-1.
- [23] Mao, J., Tian, W., Li, P., Wei, T., & Liang, Z. (2017). Phishing-Alarm: Robust and efficient phishing detection via page component similarity. *IEEE Access*, 5, 17020–17030. DOI: 10.1109/ACCESS.2017.2743528.
- [24] Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1245–1254. DOI: 10.1145/1557019.1557153.
- [25] Le, A., Markopoulou, A., & Faloutsos, M. (2011). PhishDef: URL names say it all. *Proceedings of IEEE INFOCOM 2011*. DOI: 10.1109/INFCOM.2011.5934995.
- [26] Wei, W., Ke, Q., Nowak, J., Korytkowski, M., Scherer, R., & Woźniak, M. (2020). Accurate and fast URL phishing detector: A convolutional neural network approach. *Computer Networks*, 178, 107275. DOI: 10.1016/j.comnet.2020.107275.
- [27] Alani, M. M., & Tawfik, H. (2022). PhishNot: A cloud-based machine-learning approach to phishing URL detection. *Computer Networks*, 218, 109407. DOI: 10.1016/j.comnet.2022.109407.