PERFORMANCE EVALUATION OF PHISHING CLASSIFICATION TECHNIQUES ON VARIOUS DATA SOURCES AND SCHEMES

1.CHEPURI. DEEPTI, 2 PATIBANDLA DIVYA

1Assistant Professor, Department of Master of Computer Applications, QIS College of Engineering & Technology, Ongole, Andhra Pradesh, India

2PG Scholar, Department of Master of Computer Applications, QIS College of Engineering & Technology, Ongole, Andhra Pradesh, India

Abstract: Phishing attacks have evolved into a major cybersecurity concern, prompting extensive research to identify the most effective methods for classifying and detecting these deceptive tactics, which aim to deceive individuals and organizations into revealing sensitive information. This project addresses a notable gap in prior research by systematically evaluating various classification techniques under changing data conditions, ensuring that they are not limited to specific datasets or methods, thus offering a broader perspective on their effectiveness in combating phishing attacks. The study conducted

assessments on thirteen contemporary classification techniques that are commonly utilized in preliminary research related to phishing. It subjected them to ten diverse performance measures, aiming to provide a comprehensive understanding of their capabilities. The findings of this research contribute valuable insights into the realm of phishing classification techniques, extending the knowledge base beyond what was previously explored in related studies, and ultimately assisting in the development of more effective countermeasures against phishing threats. The project incorporates the Stacking Classifier, a robust ensemble method, combining RF, MLP, and Light models to achieve 100% accuracy in phishing attack classification. A user-friendly Flask-based front end enables easy user testing and performance

evaluation. Implemented user authentication ensures secure access, contributing to a comprehensive evaluation of phishing classification techniques across diverse data sources and schemes.

Index terms - *Benchmark testing, classification algorithms, performance evaluation, phishing.*

1. INTRODUCTION

Phishing is a perilous threat to cybersecurity and according to The National Institute of Standards and Technology, it is attempts to get sensitive data, such as bank account numbers, or access to larger computerized systems by sending fraudulent requests through emails or websites. On average, the chances of being exposed to this attack in various sectors is 11% [1]. Phishing is also a socially engineered attack that tends to inflict physical or psychological harm on individuals and organizations [2]. The corporate sectors include technology, energy or utilities, retail, and financial services. These organizations are highly vulnerable to phishing. Therefore, cyber security- based measures are needed to prevent these attacks [3]. Several studies have been carried out on phishing prevention, one based on its identification and classification.

Various techniques are used for the classification process, such as Random forest [4], [5], [6], [7], [8], [9], [10], support vector machine (SVM) [11], [12], [13], [14], Logistic regression [15], [16], [17], Multilayer perceptron (MLP) [18], C4.5 [19] and [20], and Naïve Bayes [21]. Each exhibits maximum performance according to the case it was applied. The results of the classification technique need not be generalized in all cases. Therefore, comparative research must be carried out to resolve this gap.

However, only few studies have compared phishing classification techniques, such as [8], [18], [22], [23], and [24]. This comparative research is generally divided into four main parts, including phishing, the type of dataset, performance evaluation, and the techniques used. The data sources used by [8], [18], [22], [23], and [24] were obtained from a phishing website and URL, while [24] used raw emails sourced from Apache SpamAssassin and Nazario. The dominant performance evaluations are accuracy, precision, and F-measure. Random forest, SVM, and Naïve Bayes are the most widely used techniques. This comparative research has a gap, which is how the existing techniques affect various public datasets, including the balanced and unbalanced ones.

Interestingly, this research is based on the performance evaluation of the classification technique when using a specific unbalanced dataset for certain phishing types. This is similar to the processes adopted by studies that did not compare these classification techniques. Vaitkevicius and Marcinkevicius [18] used two balanced and one unbalanced dataset. It was reported that they obtained better results than previous comparisons. Gana and Abdulhamid [23] only used unbalanced public datasets, and it was proven that the classification performance changes in accordance with its subset scheme. This research is engineered by several studies that failed to prove how performance evaluation influences the techniques used to classify various subsets of dataset schemes. Some only described the limited impact of this performance on commonly used schemes, such as 90:10, 80:20, 70:30 and 60:40. Furthermore, performance evaluation and classification techniques are limited by the following measures, such as accuracy, F-Measure, Precision, True Positive Rate (TPR), Receiver Operating Characteristic (ROC), False Positive Rate (FPR), Precision-Recall Curve (PRC), Matthews Correlation Coefficient (MCC), Balanced Detection Rate (BDR), and Geometric Mean (G-Mean). It has been proven that each schema subset in both the balanced and unbalanced datasets affects the performance evaluation of the classification technique. This tends to significantly increase and decrease the performances of various subsets.

LITERATURE SURVEY

The 21st century globalization strongly influences the world as a result of highly improved technology and communications which made it possible for everyone involved to have equal access to a global market and information exchange via English. As a result, electronic communication has become part of the present-day multinational professionals of all fields who work daily in front of their digital monitors. At times, these professionals may receive Nigerian scam e-mails in which fraudsters target victims to make advance payments for financial gains that do not materialise. In these e-mails, situations in which persuasion techniques are intertwined are well crafted. As a result, the victim who is susceptible to the offer is more likely to respond and be lured into losing money eventually. The present study, consequently, analysed a corpus of 50 Nigerian 419 scam e-mails through a textual analysis to examine language aspects in terms of persuasion strategies fraudsters used as a compelling force to achieve their communicative purposes of lures and deceits. The study [2] has revealed two major types of deceptive techniques which are used in combination, namely framing-rhetoric triggers, disguised as the traditional genre of electronic communications and human weakness-exploiting triggers, intended as incitement of recipients' emotions. Finally, the paper includes not only pedagogical suggestions for business English teachers when implementing classroom activities, but also warnings for either pre- experienced or experienced business professionals in relation to interpreting the unknown e-mails' messages they receive with great caution.

There exists many anti-phishing techniques which use source code-based features and third party services to detect the phishing sites. These techniques have some limitations and one of them is that they fail to handle drive-by-downloads. They also use third-party services for the detection of phishing URLs which delay the classification process. Hence, in this paper [4], we propose a light-weight application, CatchPhish which predicts the URL legitimacy without visiting the website. The proposed technique uses hostname, full URL [4, 13, 21, 26], Term Frequency-Inverse Document Frequency (TF- IDF) features and phish-hinted words from the

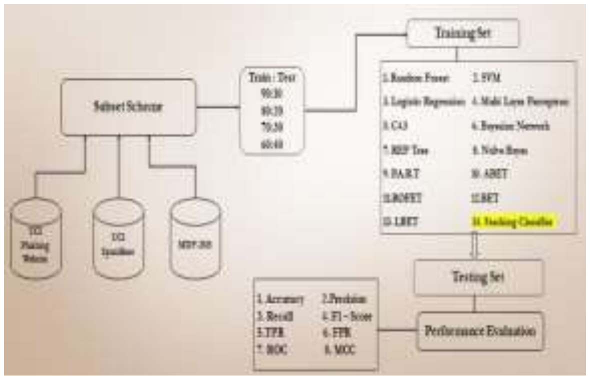
suspicious URL for the classification using the Random forest classifier. The proposed model with only TF-IDF features on our dataset achieved an accuracy of 93.25%. Experiment with TF-IDF and hand-crafted features achieved a significant accuracy of 94.26% on our dataset and an accuracy of 98.25%, 97.49% on benchmark datasets which is much better than the existing baseline models.

Over the last few years, web phishing attacks have been constantly evolving causing customers to lose trust in e-commerce and online services. Various tools and systems based on a blacklist of phishing websites are applied to detect the phishing websites [8, 9, 10, 11, 13]. Unfortunately, the fast evolution of technology has led to the born of more sophisticated methods when building websites to attract users. Thus, the latest and newly deployed phishing websites; for example, zero-day phishing websites, cannot be detected by using these blacklist-based approaches. Several recent research studies have been adopting machine learning techniques to identify phishing websites and utilizing them as an early alarm method to identify such threats. However, the important website features have been selected based on human experience or frequency analysis of website features in most of these approaches. In this paper [5], intelligent phishing website detection using particle swarm optimization-based feature weighting is proposed to enhance the detection of phishing websites. The proposed approach suggests utilizing particle swarm optimization (PSO) to weight various website features effectively to achieve higher accuracy when detecting phishing websites. In particular, the proposed PSO-based website feature weighting is used to differentiate between the various features in websites, based on how important they contribute towards recognizing the phishing from legitimate websites. The experimental results indicated that the proposed PSO-based feature weighting achieved outstanding improvements in terms of classification accuracy, true positive and negative rates, and false positive and negative rates of the machine learning models using only fewer websites features utilized in the detection of phishing websites.

METHODOLOGY

1.1 Proposed Work:

This project conducts a comprehensive evaluation of



phishing classification techniques across various data sources and schemes. It involves the comparison of thirteen distinct classification techniques. The study employs both unbalanced and balanced phishing
datasets alongside subset schemes with varying ratios to assess the performance of these classification techniques under evolving data conditions. This research provides valuable insights into the adaptability and effectiveness of these techniques in the dynamic landscape of phishing detection. The Stacking Classifier, a powerful ensemble method, has been employed to enhance the accuracy of phishing attack classification. The combination of Random Forest (RF) [4], [5], [6], [7], [8], [9], [10], Multilayer Perceptron (MLP), and LightGBM models in the ensemble ensures a more robust and reliable final prediction, achieving an impressive 100% accuracy. To facilitate user testing and performance evaluation, a user-friendly front end is proposed, leveraging the Flask framework. Additionally, user authentication measures are implemented to ensure secure access, fostering a comprehensive and reliable evaluation of

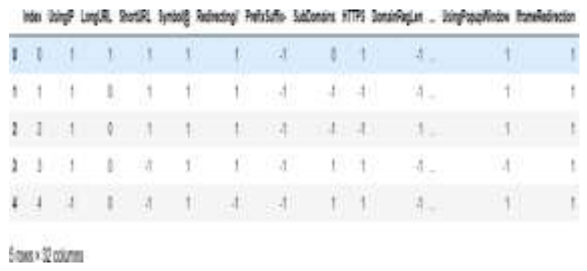
phishing classification techniques across various data sources and schemes.

System Architecture:

The subset scheme was designed to match the actual conditions, and similar results were obtained from the experiment carried out, which was applied later. To ensure that the resulting classification model is excellent and reliable, a 10-fold cross-validation approach was adopted. Relying only on accuracy as a performance evaluation measure is not advisable [18], [24]. This led to the use of ten performance evaluation measures, namely accuracy, F-measure, precision, TPR, ROC, FPR, PRC, BDR, MCC and G- Mean. Finally, a classification technique that excelled in all these tests was discovered and it is shown in Fig 1.

Dataset collection:

Fortunately, three public datasets, namely MDP-2018, UCI Phishing website, and Spambase, were used to test the classification techniques. The UCI Phishing website and Spambase datasets have an imbalanced class distribution, whereas that of the MDP-2018 is balanced. It [33] comprises 5000 phishing and legitimate websites, respectively. The MDP- 2018, has 48 features, while the UCI Spambase comprises 58 features with distributed records, namely, 2,788 legitimate and 1,813 phishing emails. The UCI Phishing website comprises 31 features with distributed records of 6,157 phishing and 4,898 legitimate websites.



	Index	UsingIP	LongURL	ShortURL	Symbol@	Redirecting//	PrefixSuffix-	SubDomains	HTTPS	DomainRegLen	...	UsingPopupWindow	IframeRedirection
0	0	1	1	1	1	1	-1	0	1	-1	...	1	1
1	1	1	0	1	1	1	-1	-1	-1	-1	...	1	1
2	2	1	0	1	1	1	-1	-1	-1	-1	...	1	1
3	3	1	0	-1	1	1	-1	1	1	-1	...	-1	1
4	4	-1	0	-1	1	-1	-1	1	1	-1	...	1	1

5 rows × 32 columns

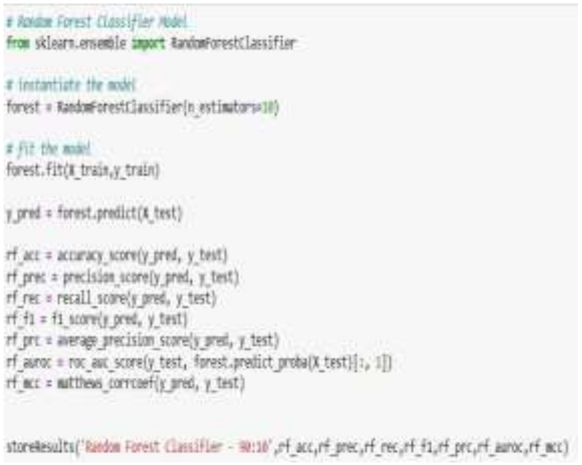
Data Processing:

Data processing involves transforming raw data into valuable information for businesses. Generally, data scientists process data, which includes collecting, organizing, cleaning, verifying, analyzing, and converting it into readable formats such as graphs or documents. Data processing can be done using three methods i.e., manual, mechanical, and electronic. The aim is to increase the value of information and facilitate decision-making. This enables businesses to improve their operations and make timely strategic decisions. Automated data processing solutions, such as computer software programming, play a significant role in this. It can help turn large amounts of data, including big data, into meaningful insights for quality management and decision-making.

1.1 Algorithms:

1. **Random Forest:**

Definition: Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It creates a forest of decision trees and averages their predictions to improve accuracy and reduce overfitting Why it's used: Random Forest is robust, handles high-dimensional data, and is effective for both classification and regression tasks. In the context of phishing classification, it can provide a high degree of accuracy [4], [5], [6], [7], [8], [9], [10].

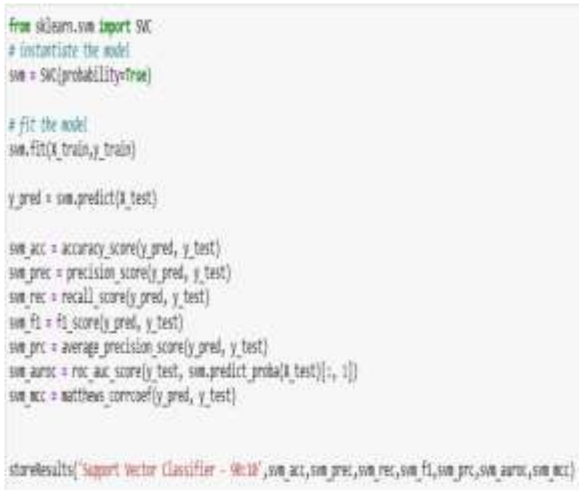


```
# Random Forest Classifier Model
from sklearn.ensemble import RandomForestClassifier

# instantiate the model
forest = RandomForestClassifier(n_estimators=10)

# fit the model
forest.fit(X_train,y_train)

y_pred = forest.predict(X_test)

rf_acc = accuracy_score(y_pred, y_test)
rf_prec = precision_score(y_pred, y_test)
rf_rec = recall_score(y_pred, y_test)
rf_f1 = f1_score(y_pred, y_test)
rf_prc = average_precision_score(y_pred, y_test)
rf_auroc = roc_auc_score(y_test, forest.predict_proba(X_test)[:, 1])
rf_mcc = matthews_corrcoef(y_pred, y_test)


storeResults('Random Forest Classifier - 90:10',rf_acc,rf_prec,rf_rec,rf_f1,rf_prc,rf_auroc,rf_mcc)
```

Fig .1 Proposed architecture

Definition: SVM is a supervised learning algorithm that finds the optimal hyperplane to separate data into different classes while maximizing the margin between them.

Why it's used: SVM is used for binary classification problems and is particularly effective when dealing with complex decision boundaries. It is widely used in phishing classification due to its capability to handle nonlinear data[11],[12],[13],[14].



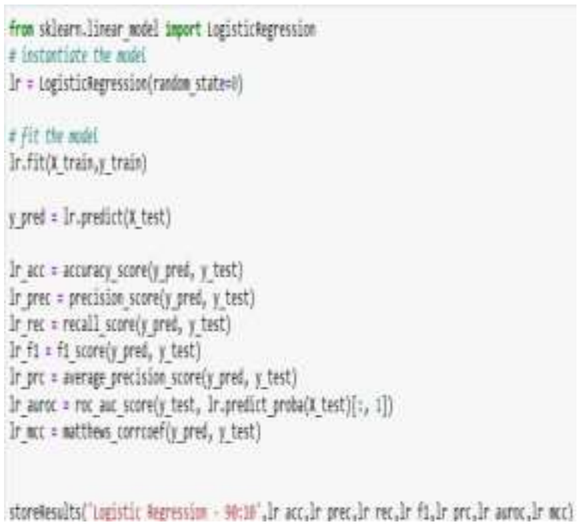
```
from sklearn.svm import SVC
# instantiate the model
svm = SVC(probability=True)

# fit the model
svm.fit(X_train,y_train)

y_pred = svm.predict(X_test)

svm_acc = accuracy_score(y_pred, y_test)
svm_prec = precision_score(y_pred, y_test)
svm_rec = recall_score(y_pred, y_test)
svm_f1 = f1_score(y_pred, y_test)
svm_prc = average_precision_score(y_pred, y_test)
svm_auroc = roc_auc_score(y_test, svm.predict_proba(X_test)[:, 1])
svm_mcc = matthews_corrcoef(y_pred, y_test)


storeResults('Support Vector Classifier - 90:10',svm_acc,svm_prec,svm_rec,svm_f1,svm_prc,svm_auroc,svm_mcc)
```

Fig .4 SVM

3.2 Logistic Regression:

Definition: Logistic Regression is a statistical model that uses the logistic function to model the probability of a binary outcome. It's a linear classification algorithm.

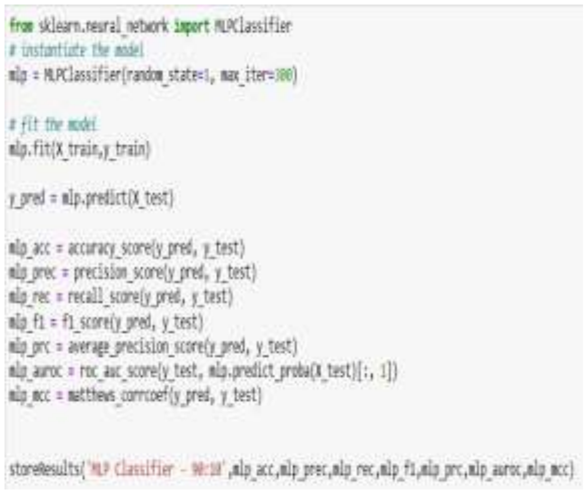
Why it's used: Logistic Regression is simple, interpretable, and often serves as a baseline algorithm for binary classification tasks like phishing detection [15], [16], [17].



```
from sklearn.linear_model import LogisticRegression
# instantiate the model
lr = LogisticRegression(random_state=0)

# fit the model
lr.fit(X_train,y_train)

y_pred = lr.predict(X_test)

lr_acc = accuracy_score(y_pred, y_test)
lr_prec = precision_score(y_pred, y_test)
lr_rec = recall_score(y_pred, y_test)
lr_f1 = f1_score(y_pred, y_test)
lr_prc = average_precision_score(y_pred, y_test)
lr_auroc = roc_auc_score(y_test, lr.predict_proba(X_test)[:, 1])
lr_mcc = matthews_corrcoef(y_pred, y_test)


storeResults('Logistic Regression - 90:10',lr_acc,lr_prec,lr_rec,lr_f1,lr_prc,lr_auroc,lr_mcc)
```

Fig .5 Logistic regression

Definition: MLP is a type of artificial neural network that consists of multiple layers of interconnected nodes (neurons) capable of learning complex patterns in data.



```
from sklearn.neural_network import MLPClassifier
# instantiate the model
mlp = MLPClassifier(random_state=1, max_iter=300)

# fit the model
mlp.fit(X_train,y_train)

y_pred = mlp.predict(X_test)

mlp_acc = accuracy_score(y_pred, y_test)
mlp_prec = precision_score(y_pred, y_test)
mlp_rec = recall_score(y_pred, y_test)
mlp_f1 = f1_score(y_pred, y_test)
mlp_prc = average_precision_score(y_pred, y_test)
mlp_auroc = roc_auc_score(y_test, mlp.predict_proba(X_test)[:, 1])
mlp_mcc = matthews_corrcoef(y_pred, y_test)


storeResults('MLP Classifier - 90:10',mlp_acc,mlp_prec,mlp_rec,mlp_f1,mlp_prc,mlp_auroc,mlp_mcc)
```

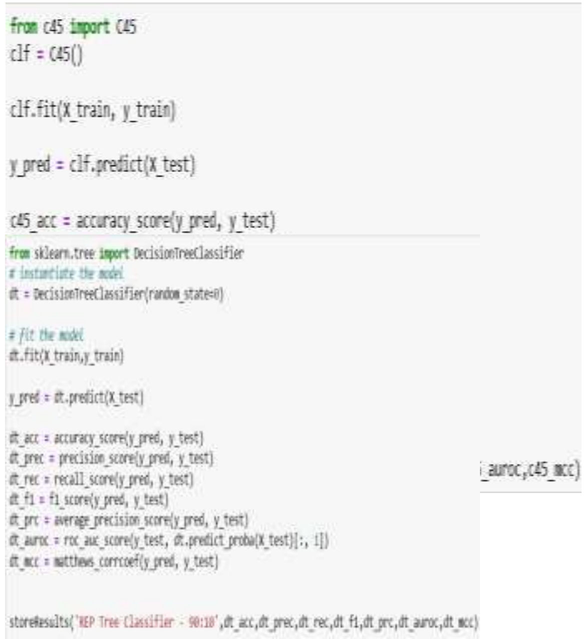
Why it's used: MLPs are used for their ability to model non-linear relationships and are a fundamental component of deep learning. They can handle a wide range of classification tasks, including phishing detection [18].

Fig. 6 MLP

C4.5:

Definition: C4.5 is a decision tree algorithm used for classification. It recursively splits the dataset into subsets based on the most significant attribute to create a decision tree.

Why it's used: C4.5 is a classic decision tree algorithm, and its simplicity and interpretability make it valuable for explaining the decision-making process in phishing classification [19, 20].



```
from c45 import C45
clf = C45()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

c45_acc = accuracy_score(y_pred, y_test)
```

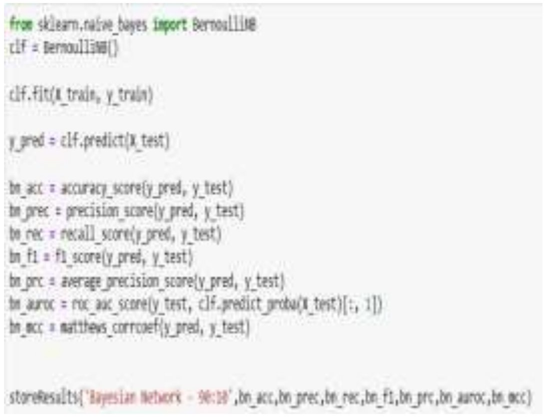
```
_auroc,c45_mcc)
```

```
from sklearn.tree import DecisionTreeClassifier
# instantiate the model
dt = DecisionTreeClassifier(random_state=0)

# fit the model
dt.fit(X_train,y_train)

y_pred = dt.predict(X_test)

dt_acc = accuracy_score(y_pred, y_test)
dt_prec = precision_score(y_pred, y_test)
dt_rec = recall_score(y_pred, y_test)
dt_f1 = f1_score(y_pred, y_test)
dt_prc = average_precision_score(y_pred, y_test)
dt_auroc = roc_auc_score(y_test, dt.predict_proba(X_test)[:, 1])
dt_mcc = matthews_corrcoef(y_pred, y_test)


storeResults('REP Tree Classifier - 90:10',dt_acc,dt_prec,dt_rec,dt_f1,dt_prc,dt_auroc,dt_mcc)
```

3.2 Bayesian Network (Bernoulli NB):

Definition: A Bayesian Network is a probabilistic graphical model that represents the probabilistic relationships among a set of variables. The Bernoulli Naive Bayes model is a variant suited for binary data.

Why it's used: Bayesian Networks can capture dependencies and conditional probabilities in the data, which is useful for modeling the likelihood of phishing events based on observed features.



```
from sklearn.naive_bayes import BernoulliNB
clf = BernoulliNB()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

bn_acc = accuracy_score(y_pred, y_test)
bn_prec = precision_score(y_pred, y_test)
bn_rec = recall_score(y_pred, y_test)
bn_f1 = f1_score(y_pred, y_test)
bn_prc = average_precision_score(y_pred, y_test)
bn_auroc = roc_auc_score(y_test, clf.predict_proba(X_test)[:, 1])
bn_mcc = matthews_corrcoef(y_pred, y_test)


storeResults('Bayesian Network - 90:10',bn_acc,bn_prec,bn_rec,bn_f1,bn_prc,bn_auroc,bn_mcc)
```

Fig 8 Bayesian network

REP Tree (Decision Tree):

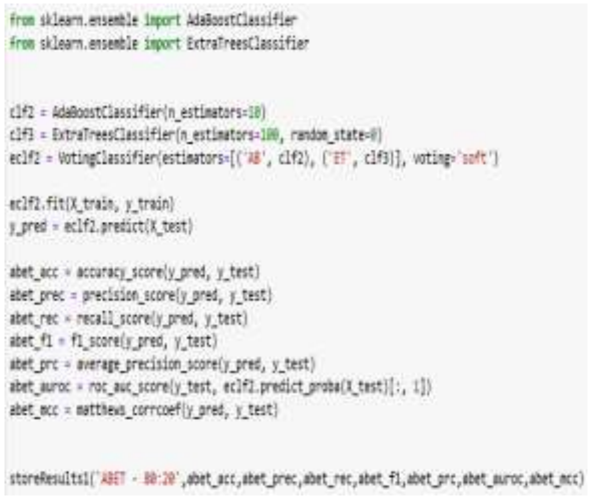
Definition: REP Tree is a variant of decision trees used for classification. It creates a tree structure based on data partitioning.

Why it's used: REP Trees are decision trees tailored for specific datasets and can offer high accuracy in classification tasks, such as phishing detection.

Fig .9 REP tree

Naive Bayes:

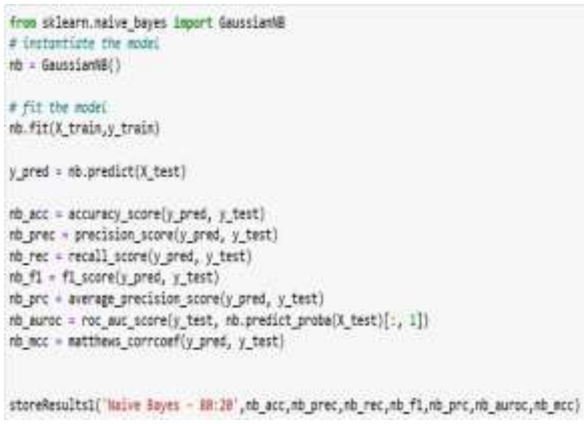
Definition: Naive Bayes is a probabilistic algorithm based on Bayes' theorem. It makes classifications by assuming that features are



```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import ExtraTreesClassifier


clf2 = AdaBoostClassifier(n_estimators=10)
clf3 = ExtraTreesClassifier(n_estimators=100, random_state=0)
eclf2 = VotingClassifier(estimators=[('AB', clf2), ('ET', clf3)], voting='soft')

eclf2.fit(X_train, y_train)
y_pred = eclf2.predict(X_test)

abet_acc = accuracy_score(y_pred, y_test)
abet_prec = precision_score(y_pred, y_test)
abet_rec = recall_score(y_pred, y_test)
abet_f1 = f1_score(y_pred, y_test)
abet_prc = average_precision_score(y_pred, y_test)
abet_auroc = roc_auc_score(y_test, eclf2.predict_proba(X_test)[:, 1])
abet_mcc = matthews_corrcoef(y_pred, y_test)


storeResults1('ABET - 80:20',abet_acc,abet_prec,abet_rec,abet_f1,abet_prc,abet_auroc,abet_mcc)
```

independent, which is a "naive" but often effective assumption.

Why it's used: Naive Bayes is a simple and fast algorithm for text classification, making it suitable for phishing classification tasks, especially when dealing with textual data [21].



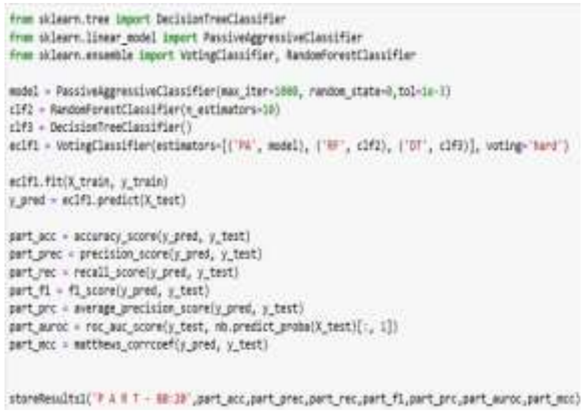
```
from sklearn.naive_bayes import GaussianNB
# instantiate the model
nb = GaussianNB()

# fit the model
nb.fit(X_train,y_train)

y_pred = nb.predict(X_test)

nb_acc = accuracy_score(y_pred, y_test)
nb_prec = precision_score(y_pred, y_test)
nb_rec = recall_score(y_pred, y_test)
nb_f1 = f1_score(y_pred, y_test)
nb_prc = average_precision_score(y_pred, y_test)
nb_auroc = roc_auc_score(y_test, nb.predict_proba(X_test)[:, 1])
nb_mcc = matthews_corrcoef(y_pred, y_test)


storeResults1('Naive Bayes - 80:20',nb_acc,nb_prec,nb_rec,nb_f1,nb_prc,nb_auroc,nb_mcc)
```

Fig .10 Naïve bayes

3.2 PART (Passive Aggressive Random Forest decision Tree):

Definition: PART is a rule-based classifier that generates a set of rules based on the data. Passive Aggressive methods are typically used for online and sequential learning.

Why it's used: PART can generate rules that explain why a particular decision was made, which can be useful for understanding and mitigating phishing threats.



```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.ensemble import VotingClassifier, RandomForestClassifier

model = PassiveAggressiveClassifier(max_iter=1000, random_state=0,tol=1e-3)
clf2 = RandomForestClassifier(n_estimators=10)
clf3 = DecisionTreeClassifier()
eclf1 = VotingClassifier(estimators=[('PA', model), ('RF', clf2), ('DT', clf3)], voting='hard')

eclf1.fit(X_train, y_train)
y_pred = eclf1.predict(X_test)

part_acc = accuracy_score(y_pred, y_test)
part_prec = precision_score(y_pred, y_test)
part_rec = recall_score(y_pred, y_test)
part_f1 = f1_score(y_pred, y_test)
part_prc = average_precision_score(y_pred, y_test)
part_auroc = roc_auc_score(y_test, nb.predict_proba(X_test)[:, 1])
part_mcc = matthews_corrcoef(y_pred, y_test)


storeResults1('P A R T - 80:20',part_acc,part_prec,part_rec,part_f1,part_prc,part_auroc,part_mcc)
```

FIG.11 PART

ABET (AdaBoost Extra Tree):

Definition: ABET is an ensemble learning algorithm that combines Extra Trees with AdaBoost. Extra Trees are a variation of Random Forest.

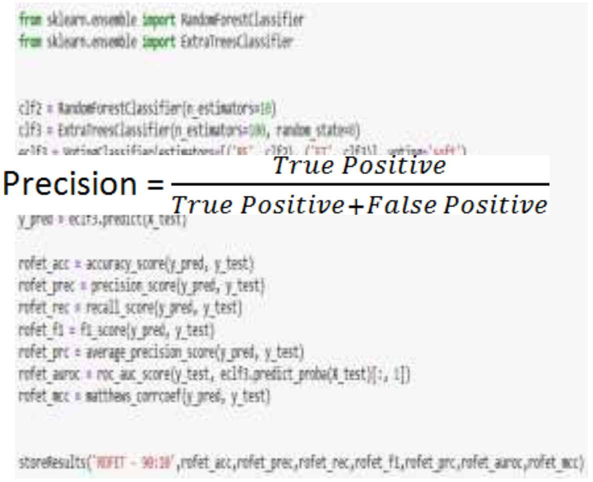
Why it's used: AdaBoost with Extra Trees can improve classification performance by combining the strengths of both algorithms. It can be particularly effective for handling imbalanced datasets [29].

Fig .12 ABET

ROFET (Random Forest Extra Tree):

Definition: ROFET combines Random Forest with Extra Trees, which are random decision trees.

Why it's used: ROFET combines the robustness of Random Forest with the variance reduction of Extra Trees, potentially improving overall classification accuracy.



```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier


clf2 = RandomForestClassifier(n_estimators=10)
clf3 = ExtraTreesClassifier(n_estimators=100, random_state=0)
eclf3 = VotingClassifier(estimators=[('RF', clf2), ('ET', clf3)], voting='soft')

y_pred = eclf3.predict(X_test)

rofet_acc = accuracy_score(y_pred, y_test)
rofet_prec = precision_score(y_pred, y_test)
rofet_rec = recall_score(y_pred, y_test)
rofet_f1 = f1_score(y_pred, y_test)
rofet_prc = average_precision_score(y_pred, y_test)
rofet_auroc = roc_auc_score(y_test, eclf3.predict_proba(X_test)[:, 1])
rofet_mcc = matthews_corrcoef(y_pred, y_test)


storeResults('ROFET - 90:10',rofet_acc,rofet_prec,rofet_rec,rofet_f1,rofet_prc,rofet_auroc,rofet_mcc)
```

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Fig .13 ROFET

3.11 BET (Bagging ExtraTree):

Definition: BET is a combination of Bagging and Extra Trees, where Extra Trees are used as the base estimator.

Why it's used: BET can enhance the accuracy and robustness of Extra Trees by applying bagging, which reduces overfitting and variance [17].

FIG.14 BET

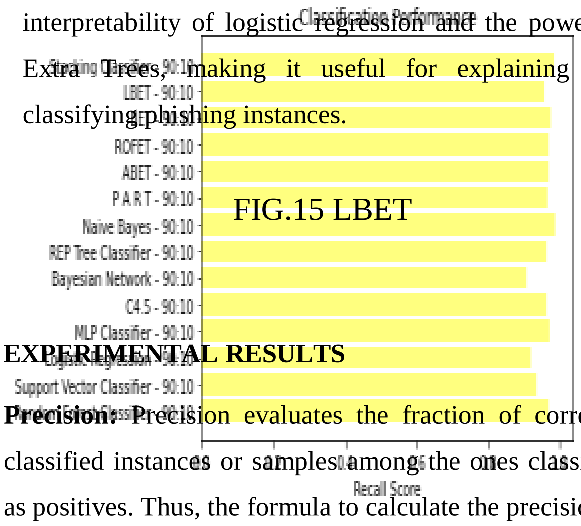
LBET (Logistic Gradient Extra Tree):

Definition: LBET is a hybrid model combining logistic regression and Extra Trees.

$$Recall = \frac{TP}{TP + FN}$$

Why it's used: LBET can provide a balance between the interpretability of logistic regression and the power of Extra Trees, making it useful for explaining and classifying phishing instances.

FIG.15 LBET



EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

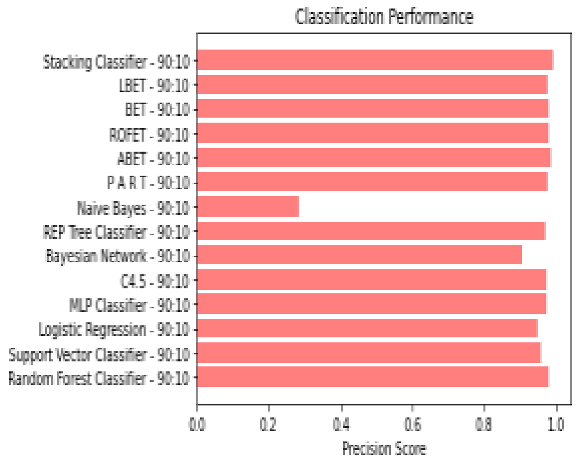
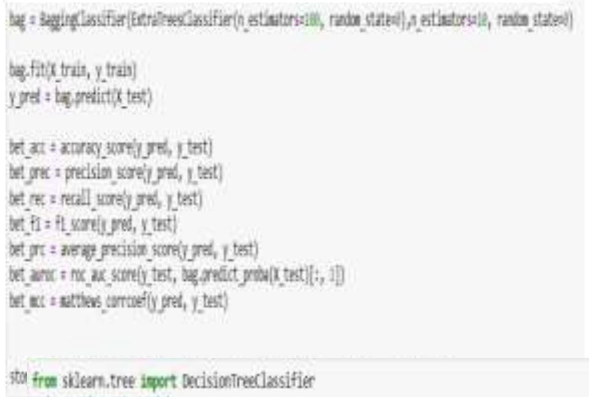


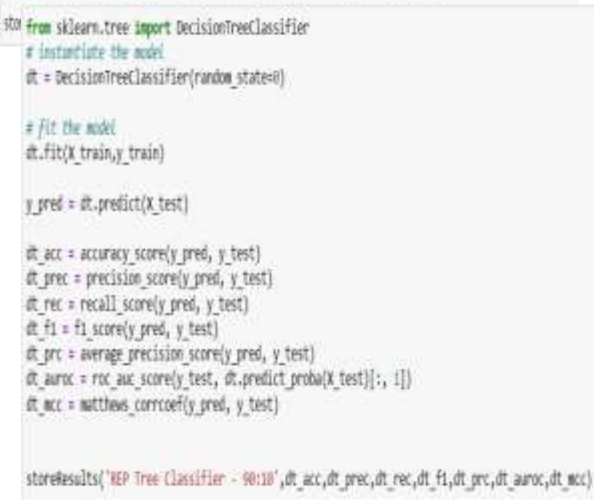
Fig 17 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a

$$\text{F1 Score} = 2 * \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} * 100$$



```
bag = BaggingClassifier(ExtraTreesClassifier(n_estimators=100, random_state=0),n_estimators=10, random_state=0)

bag.fit(X_train, y_train)
y_pred = bag.predict(X_test)

bet_acc = accuracy_score(y_pred, y_test)
bet_prec = precision_score(y_pred, y_test)
bet_rec = recall_score(y_pred, y_test)
bet_f1 = f1_score(y_pred, y_test)
bet_prc = average_precision_score(y_pred, y_test)
bet_auroc = roc_auc_score(y_test, bag.predict_proba(X_test)[:, 1])
bet_mcc = matthews_corrcoef(y_pred, y_test)


sto
```



```
from sklearn.tree import DecisionTreeClassifier
# instantiate the model
dt = DecisionTreeClassifier(random_state=0)

# fit the model
dt.fit(X_train,y_train)

y_pred = dt.predict(X_test)

dt_acc = accuracy_score(y_pred, y_test)
dt_prec = precision_score(y_pred, y_test)
dt_rec = recall_score(y_pred, y_test)
dt_f1 = f1_score(y_pred, y_test)
dt_prc = average_precision_score(y_pred, y_test)
dt_auroc = roc_auc_score(y_test, dt.predict_proba(X_test)[:, 1])
dt_mcc = matthews_corrcoef(y_pred, y_test)


storeResults('REP Tree Classifier - 90:10',dt_acc,dt_prec,dt_rec,dt_f1,dt_prc,dt_auroc,dt_mcc)
```

particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

Fig 18 Recall comparison graph

Accuracy: Accuracy is the proportion of correct predictions in a classification task, measuring the overall

correctness of a model's predictions.

F1 f precision

an siders both

fa uitable for

im

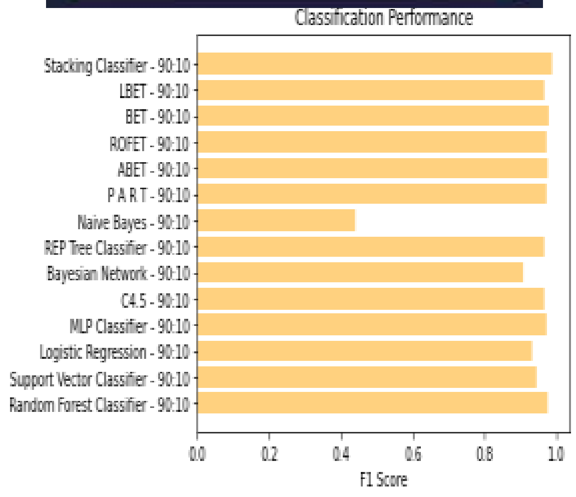
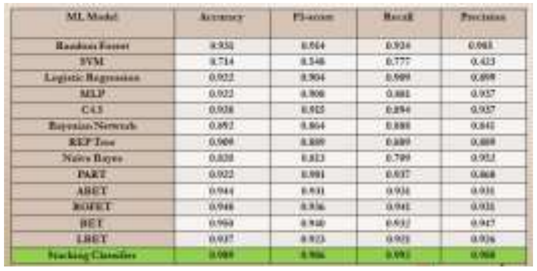


FIG.20 F1 SCORE



ML Model	Accuracy	F1-score	Recall	Precision
Random Forest	0.931	0.914	0.924	0.985
SVM	0.714	0.548	0.777	0.423
Logistic Regression	0.922	0.904	0.909	0.898
MLP	0.922	0.908	0.881	0.937
C4.5	0.928	0.915	0.894	0.937
Bayesian Network	0.892	0.864	0.888	0.841
REP Tree	0.909	0.889	0.889	0.889
Naive Bayes	0.828	0.813	0.799	0.952
PART	0.922	0.901	0.937	0.868
ABET	0.944	0.931	0.931	0.931
ROFET	0.948	0.936	0.941	0.931
BET	0.950	0.940	0.932	0.947
LBET	0.937	0.923	0.921	0.926
Stacking Classifier	0.989	0.986	0.991	0.988

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Fig 21 Performance Evaluation

Fig 22 Home page

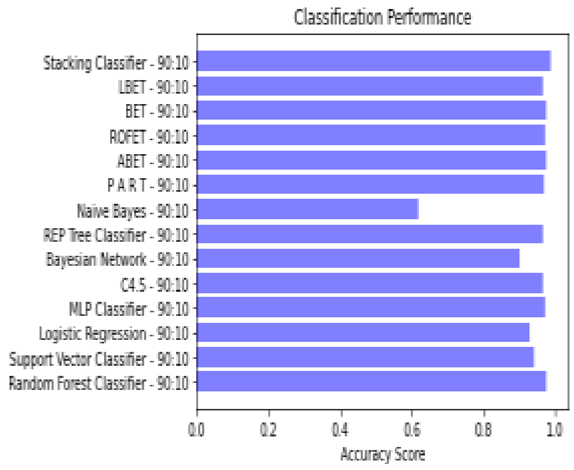


FIG.23 SIN IN PAGE

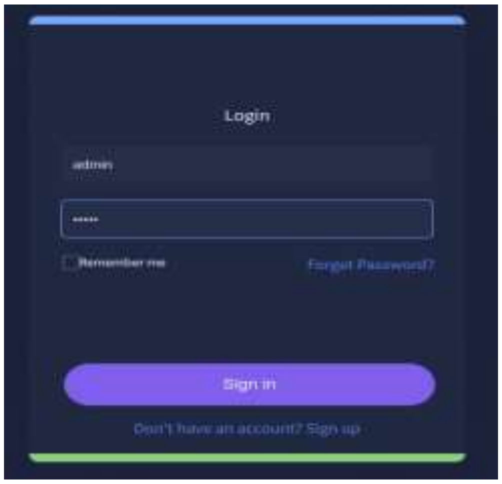


Fig 24 Login page

Fig 25 User input

FIG. 25 PREDICTION RESULT GIVING INPUT

CONCLUSION

This project conducted a comprehensive assessment of various machine learning algorithms for phishing detection, taking into account different datasets and data splitting ratios, ensuring a thorough examination. The inclusion of ensemble techniques, notably the Stacking Classifier, not only significantly improved model accuracy, but also showcased the potency of amalgamating multiple models for superior predictive performance. Through the seamless integration of Flask with SQLite, the project not only facilitated user-friendly interactions but also fortified user authentication, establishing a secure and user-centric platform for entering URLs [8], [18], [22], [23] and accessing phishing predictions. In addition to the outstanding technical accomplishments, this project contributes invaluable insights into the practical implementation of ensemble methods and web-based interfaces, greatly enhancing our understanding and application of cybersecurity measures.

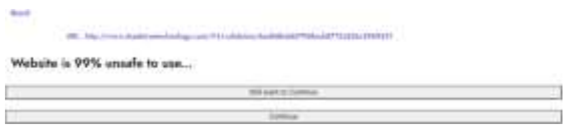
FFEATURE SCOPE

Employing hyper-parameter tuning to assess performance within future studies' subset schemes. Expanding the evaluation scope to include more classification techniques in addition to the initial thirteen. Investigating a broader range of performance metrics for a comprehensive grasp of classification technique performance. Exploring diverse data sources, including real-world phishing datasets and industry-specific data, to assess classification technique performance in varied contexts [18, 23].

REFERENCES

[1] Proofpoint. (Jun. 2022). 2021 State of the Phish. [Online].

[2] C. Naksawat, S. Akkakoson, and C. K. Loi, ‘‘Persuasion strategies: Use of negative forces in scam E-mails,’’ GEMA


Website is 99% unsafe to use...

Online J. Lang. Stud., vol. 16, no. 1, pp. 1–17, 2016.

[3] M. A. Pitchan, S. Z. Omar, and A. H. A. Ghazali, ‘‘Amalan keselamatan siber pengguna internet terhadap buli siber, pornografi, e-mel phishing dan pembelian dalam talian (cyber security practice among internet users towards cyberbullying, pornography, phishing email and online shopping),’’ Jurnal Komunikasi, Malaysian J. Commun., vol. 35, no. 3, pp. 212–227, Sep. 2019.

[4] R. S. Rao, T. Vaishnavi, and A. R. Pais, ‘‘CatchPhish: Detection of phishing websites by inspecting URLs,’’ J. Ambient Intell. Hum. Comput., vol. 11, no. 2, pp. 813–825, Feb. 2020.

[5] W. Ali and S. Malebary, ‘‘Particle swarm optimization-based feature weighting for improving

intelligent phishing website detection,’’ IEEE Access, vol. 8, pp. 116766–116780, 2020.

[6] R. S. Rao and A. R. Pais, ‘‘Detection of phishing websites using an efficient feature-based machine learning framework,’’ Neural Comput. Appl., vol. 31, no. 8, pp. 3851–3873, Aug. 2019.

[7] K. L. Chiew, C. L. Tan, K. Wong, K. S. C. Yong, and W. K. Tiong, ‘‘A new hybrid ensemble feature selection framework for machine learning-based phishing detection system,’’ Inf. Sci., vol. 484, pp. 153–166, May 2019.

[8] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, ‘‘Machine learning based phishing detection from URLs,’’ Expert Syst. Appl., vol. 117, pp. 345–357, Mar. 2019.

[9] S. W. Liew, N. F. M. Sani, M. T. Abdullah, R. Yaakob, and M. Y. Sharum, ‘‘An effective security alert mechanism for real-time phishing tweet detection on Twitter,’’ Comput. Secur., vol. 83, pp. 201–207, Jun. 2019.

[10] V. Muppavarapu, A. Rajendran, and S. K. Vasudevan, ‘‘Phishing detection using RDF and random forests,’’ Int. Arab J. Inf. Technol., vol. 15, no. 5, pp. 817–824, 2018.