WEB PHISHING DETECTION

Submitted BY;
NAME: Adhithyan S
College:Dhanalakshmi srinivsan college of engineering coimbatore
Department:Artificial intelligence and data science

Abstract

The Web Application for Phishing Website Detection is a machine learning-powered tool designed to identify and classify potentially malicious websites that aim to deceive users and extract sensitive information. Phishing attacks continue to pose a significant threat to individuals and organizations worldwide, highlighting the critical need for robust detection mechanisms.

This project leverages state-of-the-art machine learning algorithms, feature extraction techniques, and web scraping methodologies to analyze the content and structure of URLs and web pages. The application provides users with a simple and intuitive interface where they can input a URL for analysis. The system then extracts relevant features from the URL and web page content, preprocesses the data, and feeds it into a trained machine learning model.

The model predicts whether the website is likely to be a phishing attempt or a legitimate entity based on the extracted features. The application aims to empower users to make informed decisions when navigating the web, helping them avoid falling victim to phishing scams and protecting their sensitive information.

Through this project, we contribute to the ongoing efforts to combat cyber threats and enhance cybersecurity awareness. The Web Application for Phishing Website Detection serves as a valuable tool for individuals, businesses, and cybersecurity professionals seeking to safeguard themselves against the evolving landscape of online threats

Introduction:

In today's digital age, the use of online services and activities has led to an increase in cyber threats, with phishing attacks being among the most prevalent and damaging. Phishing involves the use of deceptive tactics to trick individuals into Giving sensitive information such as login credentials, financial data, or personal details. So, there is a critical need for effective mechanisms to detect these malicious activities.

Overview:

The Web Application for Phishing Website Detection is an approach to address the rising threat of phishing attacks by leveraging machine learning techniques and web scraping methodologies. This project aims to provide users with a tool that can analyze URLs and web page content to identify potential phishing attempts.

Purpose of the Application:

The primary purpose of the Web Application for Phishing Website Detection is to empower users to make informed decisions when browsing the web. By quickly assessing the legitimacy of a website, users can mitigate the risk of falling victim to phishing scams and protect their sensitive information from compromise.

Audience:

The intended audience for this application includes individuals, businesses, and cybersecurity professionals who are concerned about online security threats. This tool helps the users who want to verify the authenticity of a website before interacting with it, thereby enhancing their security.

Scope:

The scope of the project encompasses the development of a web-based application that offers a user-friendly interface for analyzing URLs. The application extracts relevant features from both the URL itself and the content of the corresponding web page. These features are then used as input to a machine learning model trained to classify websites as either legitimate or phishing.

Phishing Attacks: Overview and Statistics:

The concept of phishing attacks,, common techniques employed by attackers, and the various forms phishing attacks can take (e.g., email phishing, website spoofing, SMS phishing). Additionally, it presents statistics and trends regarding the prevalence and impact of phishing attacks, highlighting the scale of the problem and the need for robust countermeasures.

Need for Phishing Detection Solutions:

Here, the discussion centers on the inherent vulnerabilities in traditional methods of identifying phishing attempts, such as manual inspection or reliance on blacklists., Automated solutions that can swiftly identify phishing threats. It also outlines the potential consequences of falling victim to phishing attacks, including financial loss, data breaches, and reputational damage.

Machine Learning in Phishing Detection:

This section explores the role of machine learning (ML) algorithms in combating phishing attacks. It tells how ML techniques, such as supervised learning, unsupervised learning, and ensemble methods, can be leveraged to analyze features extracted from URLs and web page content to find patterns of phishing behavior. Moreover, it discusses the advantages of ML-based approaches, such as adaptability to evolving threats, scalability, and the ability to handle large

volumes of data efficiently. Additionally, it highlights notable ML algorithms commonly used in phishing detection applications, along with their strengths and limitations.

Feature Extraction:

Feature extraction is a crucial step in the process of building a machine learning model for phishing detection. It involves identifying and selecting relevant attributes or characteristics from the raw data that could be used as inputs to the model. In data set feature extraction is performed using various techniques to extract meaningful information from both the URL itself and the HTML content of the webpages.

URL Features:

- URL Length: The length of the URL is calculated to determine if it exceeds a certain threshold, which may indicate suspicious behavior.
- Hyphens in URL: The presence of hyphens in the URL is checked, as certain patterns in the URL structure could be indicative of phishing attempts.
- Subdomain Count: The number of subdomains in the URL is counted, as URLs with numerous subdomains may be associated with phishing activities.
- Phishing TLD: The top-level domain (TLD) of the URL is examined to determine if it matches known phishing TLDs, which can help identify potentially malicious websites.

HTML Content Features:

- Presence of HTML Elements: Features such as the presence of input fields, buttons, images, links, passwords, email inputs, hidden elements, audio, and video elements are extracted from the HTML content of the webpages.

- Number of HTML Elements: The count of various HTML elements such as inputs, buttons, images, links, paragraphs, scripts, meta tags, tables, and so on is calculated to capture the structural characteristics of the webpage.
- Length of Title and Text: The length of the title and text content of the webpage is measured, as phishing websites may exhibit specific patterns in the length of their titles and textual content.
- Heading Tags: The presence of heading tags (h1, h2, h3) is checked, as they may provide insights into the organization and structure of the webpage.
- Form and Input Fields: The presence of forms and specific types of input fields (e.g., text areas, text inputs) is examined, as they are commonly used in phishing attempts to collect sensitive information from users.
- Other HTML Elements: Features related to the presence of footer, navigation (nav), objects, pictures, and iframes are also extracted to capture additional structural information about the webpage.

Data Exploration and Analysis (EDA - Exploratory Data Analysis):

Data Exploration and Analysis is a crucial step in understanding the characteristics and patterns present in the dataset before applying any machine learning algorithms. It involves examining the structure, content, and distribution of the data to gain insights that can inform feature selection, preprocessing techniques, and model selection.

Key Components of Data Exploration and Analysis:

Data Understanding: This step involves gaining a comprehensive understanding of the dataset, including its size, shape, types of variables, and the target variable (if applicable). It helps in identifying potential issues such as missing values, outliers, or imbalanced classes.

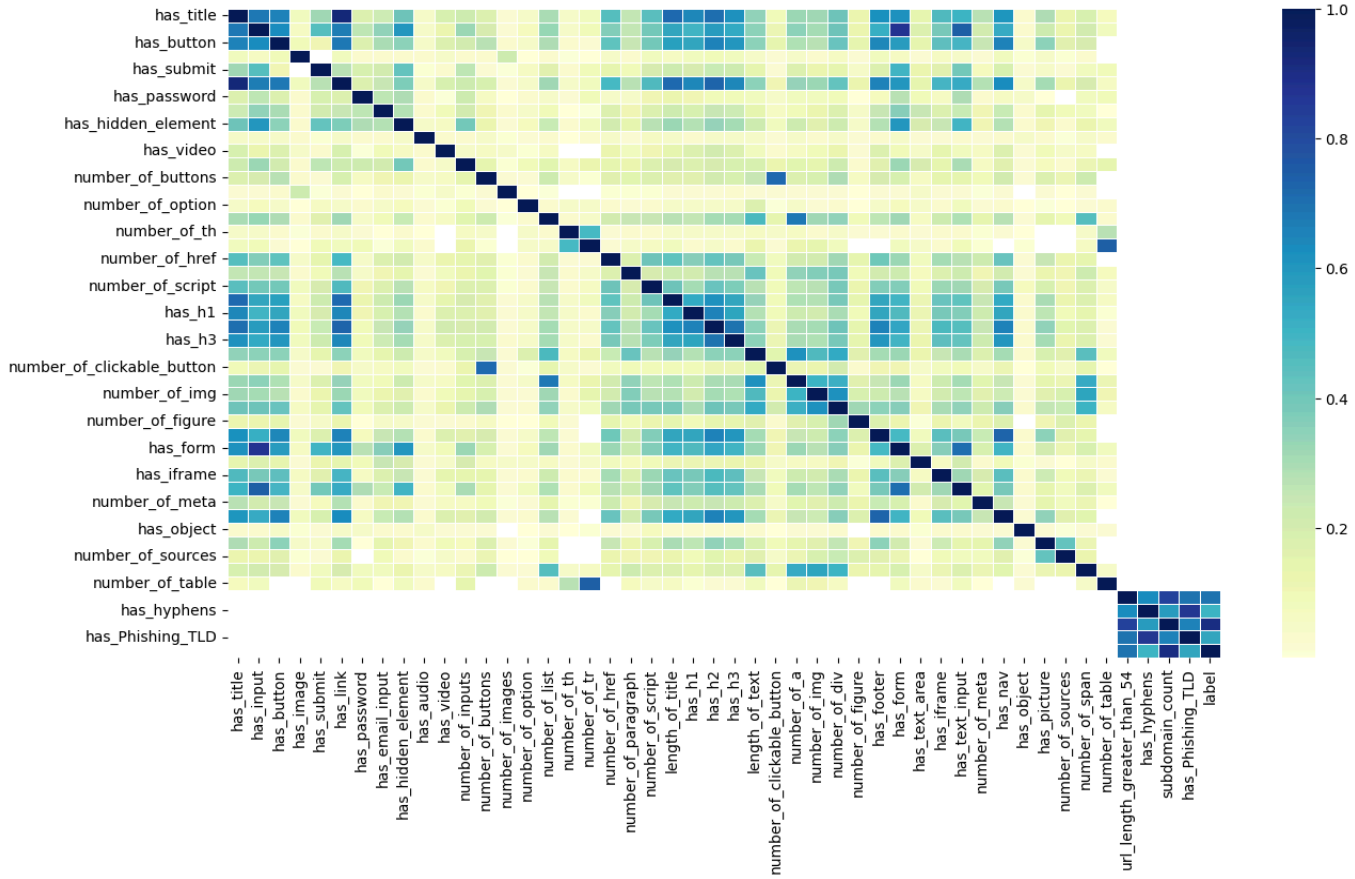
- `Df.shape ---> (26585, 50)`
- Feature columns used in the dataset

```
'URL', 'has_title', 'has_input', 'has_button', 'has_image',
       'has_submit', 'has_link', 'has_password', 'has_email_input',
       'has_hidden_element', 'has_audio', 'has_video', 'number_of_inputs',
       'number_of_buttons', 'number_of_images', 'number_of_option',
       'number_of_list', 'number_of_th', 'number_of_tr', 'number_of_href',
       'number_of_paragraph', 'number_of_script', 'length_of_title',
'has_h1',
       'has_h2', 'has_h3', 'length_of_text', 'number_of_clickable_button',
       'number_of_a', 'number_of_img', 'number_of_div', 'number_of_figure',
       'has_footer', 'has_form', 'has_text_area', 'has_iframe',
       'has_text_input', 'number_of_meta', 'has_nav', 'has_object',
       'has_picture', 'number_of_sources', 'number_of_span',
'number_of_table',
       'url_length', 'has_hyphens', 'domain', 'subdomain_count',
       'Phishing_TLD', 'label'
```

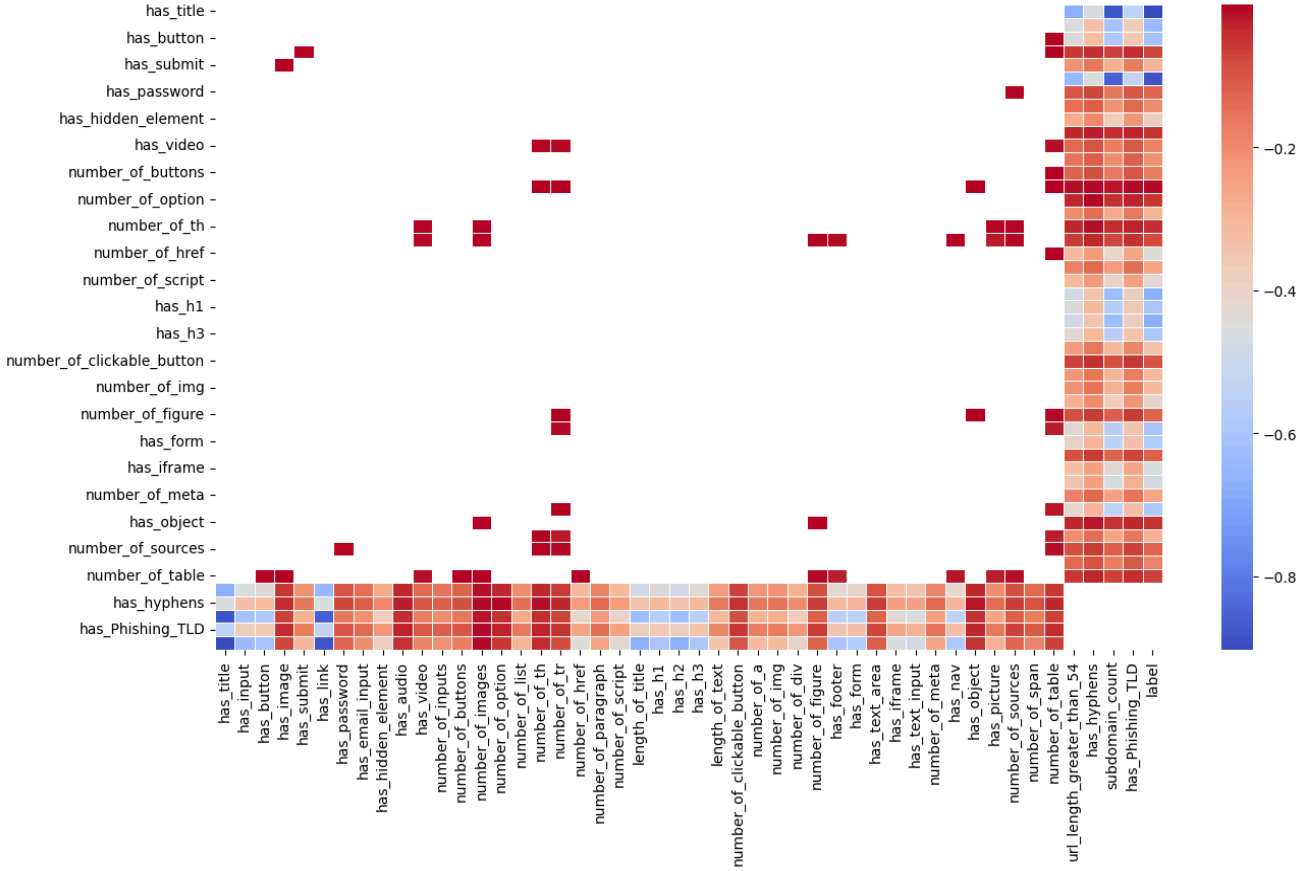
Descriptive Statistics: Descriptive statistics such as mean, median, mode, standard deviation, and quartiles are calculated to summarize the central tendency, dispersion, and distribution of numerical variables. Frequency tables and histograms may also be used to visualize the distribution of categorical variables.

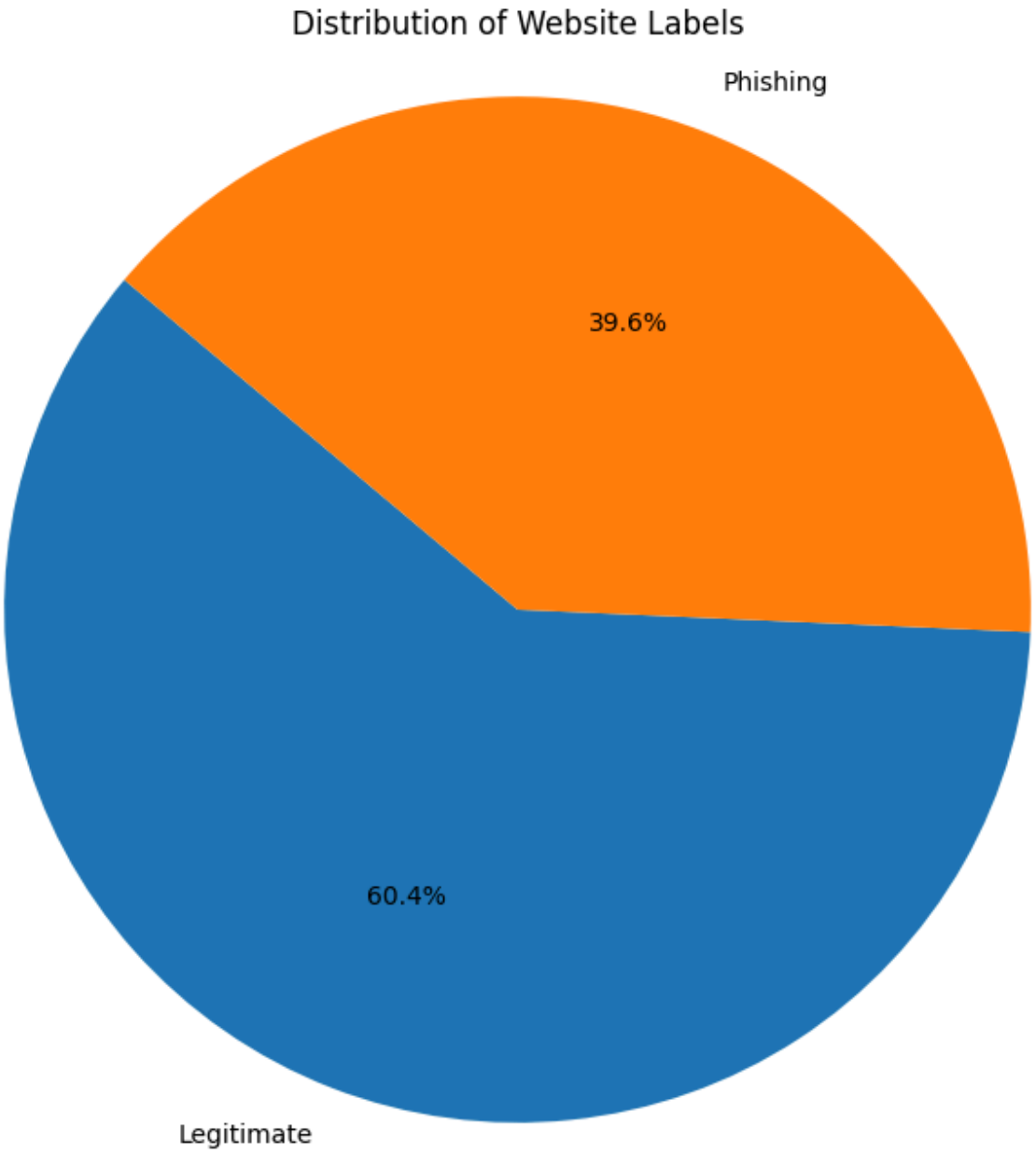
Data Visualization: Data visualization techniques such as histograms, box plots, scatter plots, and heatmaps are employed to visually explore relationships between variables, identify patterns, detect outliers, and uncover potential correlations between features and the target variable.

Positive correlation heatmap



Negative correlation heatmap





Handling Missing And Null Values: Missing values in the dataset are identified and handled using techniques such as imputation (replacing missing values with the mean, median, or mode of the variable) or deletion (removing rows or columns with missing values).

- In my data set there is no null value
- Using df.isnull().sum() i find out that there is no missing or null values

Correlation Analysis:

- Correlation matrix computed to identify correlations between numerical features.
- Strong positive correlations (>0.9) or strong negative correlations
- Features with strong correlations may lead to
- , so one of the correlated features may need to be removed to avoid redundancy.
- I have removed thehas_title because has_title has strong positive relation greater than 0.9 between some features ('has_title' and 'has_link' has strong positive correlation 'has_title'and 'label' also have strong positive correlation ,So i have removed the has_title to avoid multicollinearity.

Feature Engineering:

- I have transformed existing features to improve model performance.
- Techniques such as Min-Max scaling ,standard scaling ,one-hot encoding, binning, scaling, and normalization may be applied to preprocess the data.
- I have used the Min-max scaling
- Min-max scaling is applied to numerical features to scale them to a range of [0, 1]. This scaling method is chosen because it preserves the original distribution of the data and is suitable for features with varying scales.
- Columns scaled: 'number_of_inputs', 'number_of_buttons', 'number_of_images', 'number_of_option', 'number_of_list', 'number_of_TH', 'number_of_TR', 'number_of_href', 'number_of_paragraph', 'number_of_script', 'length_of_title', 'length_of_text',

'number_of_clickable_button', 'number_of_a', 'number_of_img', 'number_of_div', 'number_of_figure', 'number_of_sources', 'number_of_span', 'number_of_table', 'url_length_greater_than_54', 'subdomain_count'.

- These columns are scaled because they have varying scales and magnitudes, which can affect the performance of machine learning algorithms. Scaling brings all features to a similar scale, ensuring that no single feature dominates the others during model training

Statistical Summary Of Columns that Needed Scaling(Before Scaling)

	number_of_clickable_button	number_of_a	number_of_img	number_of_div	number_of_figure	number_of_table	number_of_inputs	number_of_buttons	number_of_images	number_of_option	number_of_list	number_of_th	number_of_tr	number_of_href	numbe
count	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	
mean	1.536280	106.908031	26.339364	162.167500	2.118601	0.414933	3.666015	4.158661	0.066617	6.114313	57.014933	0.297875	1.427647	10.327704	
std	12.543582	270.958896	68.465269	322.668585	13.864411	4.602446	13.541662	19.149299	2.873912	97.717824	154.977610	5.247308	13.900155	18.297292	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	19.000000	2.000000	27.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.000000	
75%	0.000000	138.000000	29.000000	213.000000	0.000000	0.000000	3.000000	3.000000	0.000000	0.000000	68.000000	0.000000	0.000000	15.000000	
max	1189.000000	22838.000000	3571.000000	15241.000000	852.000000	286.000000	616.000000	1219.000000	334.000000	13418.000000	9312.000000	391.000000	622.000000	785.000000	

```
[ ] df[['number_of_meta','subdomain_count',]].describe()
```

	number_of_meta	subdomain_count
count	26585.000000	26585.000000
mean	9.764491	1.657438
std	29.330174	0.892998
min	0.000000	1.000000
25%	0.000000	1.000000
50%	5.000000	1.000000
75%	16.000000	3.000000
max	2881.000000	6.000000

Statistical Summary Of Columns that Needed Scaling (After Scaling)

	number_of_clickable_button	number_of_a	number_of_img	number_of_div	number_of_figure	number_of_table	number_of_inputs	number_of_buttons	number_of_images	number_of_option	number_of_list	number_of_th	number_of_tr	number_of_href	numbe
count	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	26585.000000	
mean	0.001292	0.004681	0.007376	0.010640	0.002487	0.001451	0.005951	0.003412	0.000199	0.000456	0.006123	0.000762	0.002295	0.013156	
std	0.010550	0.011864	0.019173	0.021171	0.016273	0.016092	0.021983	0.015709	0.008605	0.007283	0.016643	0.013420	0.022348	0.023309	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000832	0.000560	0.001772	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.005096	
75%	0.000000	0.006043	0.008121	0.013975	0.000000	0.000000	0.004870	0.002461	0.000000	0.000000	0.007302	0.000000	0.000000	0.019108	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

```
[ ] df[['number_of_meta','subdomain_count']].describe()
```

	number_of_meta	subdomain_count
count	26585.000000	26585.000000
mean	0.003389	0.131488
std	0.010181	0.178600
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.001736	0.000000
75%	0.005554	0.400000
max	1.000000	1.000000

Model Training and Hyperparameter Tuning

Machine Learning Models Used:

In this project, we explored various machine learning models for phishing detection. We considered models such as decision trees, random forests, support vector machines (SVM), logistic regression, and others. Each model was evaluated based on its suitability for the problem at hand, considering factors such as interpretability, scalability, and performance.

Machine Learning Models Overview:

Logistic Regression:

- **Principle:** Logistic regression is a linear classification algorithm that models the probability of a binary outcome based on one or more predictor variables. It uses the logistic function to map the output to a probability value between 0 and 1.
- **Characteristics:** Simple, interpretable, and efficient. It assumes a linear relationship between the features and the log-odds of the outcome.
- Suitability: Suitable for binary classification tasks like phishing detection, especially when the relationship between features and the target is approximately linear.

Decision Trees:

- **Principle**: Decision trees recursively partition the feature space into regions, where each partition is chosen to maximize the purity of the resulting subsets with respect to the target variable.
- **Characteristics:** Can capture non-linear relationships and interactions between features. Prone to overfitting if not pruned.

- **Suitability:** Effective for interpreting feature importance and detecting complex decision boundaries, making them suitable for phishing detection tasks.

Random Forest:

- **Principle**: Random forest is an ensemble learning technique that constructs multiple decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees.
- **Characteristics**: Combines the predictions of multiple weak learners to improve accuracy and robustness. Less prone to overfitting compared to individual decision trees.
- **Suitability:** Well-suited for phishing detection due to its ability to handle high-dimensional data and mitigate overfitting, resulting in better generalization performance.

Gradient Boosting:

- **Principle:** Gradient boosting builds an ensemble of decision trees sequentially, where each tree corrects the errors of its predecessor. It minimizes a loss function by iteratively fitting new trees to the residual errors.
- **Characteristics**: Builds strong predictive models by focusing on the residual errors of previous iterations. Can be sensitive to hyperparameters and prone to overfitting if not tuned properly.
- Suitability: Effective for tasks where maximizing predictive accuracy is paramount, such as phishing detection. However, it requires careful tuning and may have longer training times compared to random forests.

Support Vector Classifier (SVC):

- **Principle:** SVC finds the hyperplane that best separates the classes in the feature space by maximizing the margin between the closest data points (support vectors).
- Characteristics: Effective in high-dimensional spaces and can capture complex decision boundaries using different kernel functions (e.g., linear, polynomial, radial basis function).
- **Suitability:** Suitable for phishing detection tasks where the data may not be linearly separable in the original feature space. However, it may suffer from scalability issues with large datasets.

Gaussian Naive Bayes:

- **Principle**: Naive Bayes is a probabilistic classifier based on Bayes' theorem and the assumption of conditional independence between features.
- **Characteristics:** Simple, fast, and requires a small amount of training data. It's called "naive" because it assumes that all features are independent, which may not hold true in practice.
- **Suitability:** Despite its simplicity, Naive Bayes can perform well for text classification tasks, but its performance may vary for more complex datasets like phishing detection.

K-Nearest Neighbors (KNN):

- **Principle:** KNN is a non-parametric algorithm that classifies new data points based on the majority class of their nearest neighbors in the feature space.

- **Characteristics:** Simple and lazy learning approach where the model does not explicitly learn a function during training. It relies on the entire training dataset for predictions.
- **Suitability:** While KNN can be effective for certain datasets, it may not be suitable for high-dimensional or imbalanced datasets commonly encountered in phishing detection tasks.

AdaBoost:

- **Principle**: AdaBoost (Adaptive Boosting) is an ensemble learning method that combines multiple weak learners (e.g., decision trees) sequentially, with each subsequent model focusing on the examples that the previous models misclassified.
- **Characteristics**: Focuses on improving the performance of misclassified instances iteratively, leading to better generalization and reduced bias.
- **Suitability:** Effective for boosting the performance of weak learners and reducing bias, making it suitable for tasks like phishing detection where the classes may be imbalanced or difficult to separate.

Mentioning Suitability:

- Each model's suitability for the task of phishing detection and online security depends on various factors such as dataset characteristics, computational resources, interpretability requirements, and performance metrics.
- It's essential to experiment with different models, tune hyperparameters, and evaluate performance rigorously to determine the most suitable approach for a given problem domain

Hyperparameter Tuning Techniques:

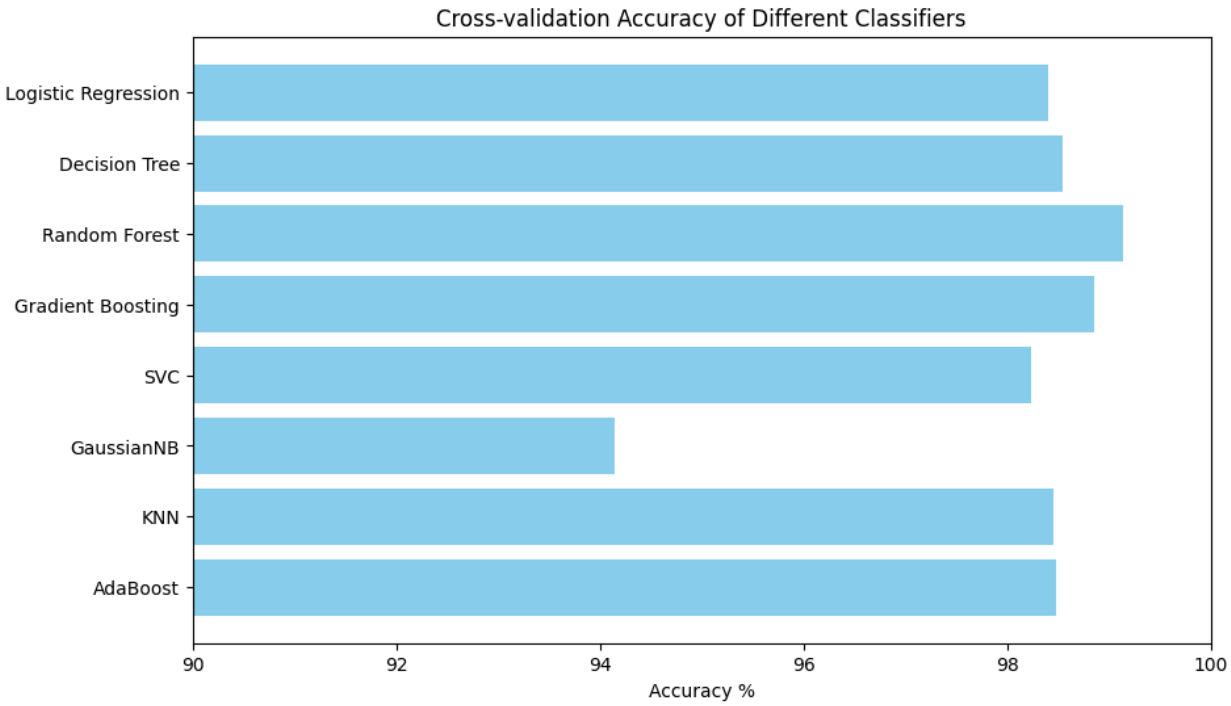
Hyperparameter tuning is crucial for optimizing model performance. We employed techniques such as grid search to explore different combinations of hyperparameters for each model. Grid search exhaustively searches through a predefined grid of hyperparameter values,.

- We begin by defining a dictionary named `params`, which contains hyperparameter grids for various classifiers. Each key corresponds to a classifier, and the associated value is another dictionary containing the hyperparameters and their respective values to be tuned

```
params ={
  'LogisticRegression': {'C': [0.1, 1.0, 10.0],'max_iter':[1000]},
  'DecisionTreeClassifier': {'max_depth': [None, 10, 20]},
  'RandomForestClassifier': {'n_estimators': [100, 200, 300], 'max_depth': [None, 10, 20]},
  'GradientBoostingClassifier': {'n_estimators': [100, 200, 300], 'learning_rate': [0.01, 0.1, 1.0]},
  'SVC': {'C': [0.1, 1.0, 10.0], 'kernel': ['linear', 'rbf']},
  'GaussianNB': {},
  'KNeighborsClassifier': {'n_neighbors': [3, 5, 7]},
  'AdaBoostClassifier': {'n_estimators': [50, 100, 200], 'learning_rate': [0.01, 0.1, 1.0]}
```

.

- We then iterate over each classifier specified in `params`. For each classifier, we utilize `RandomizedSearchCV` to perform hyperparameter tuning. This involves randomly sampling hyperparameters from the specified distributions and conducting cross-validation to identify the best combination.



- After the iteration through every classifier we got the best params for each classifier and the best cross-validation Accuracy for every classifier in that **RandomForestClassifier has almost 99.14 Cross-val-accuracy**

Model Selection Process:

:

- Following hyperparameter tuning, we create a list named `models` containing instances of the classifiers with the best parameters found during hyperparameter tuning. This ensures that we use the optimized settings for each model.

```
models =[

LogisticRegression(max_iter=1000,C=10.0),DecisionTreeClassifie
r(max_depth=10), RandomForestClassifier(n_estimators = 200,
```

```
max_depth= None),GradientBoostingClassifier(n_estimators =
300,learning_rate = 0.1),SVC(kernel='linear',C=10.0),
     GaussianNB(), KNeighborsClassifier(n_neighbors=
3),AdaBoostClassifier(n_estimators=200,learning_rate= 1.0)
]
```

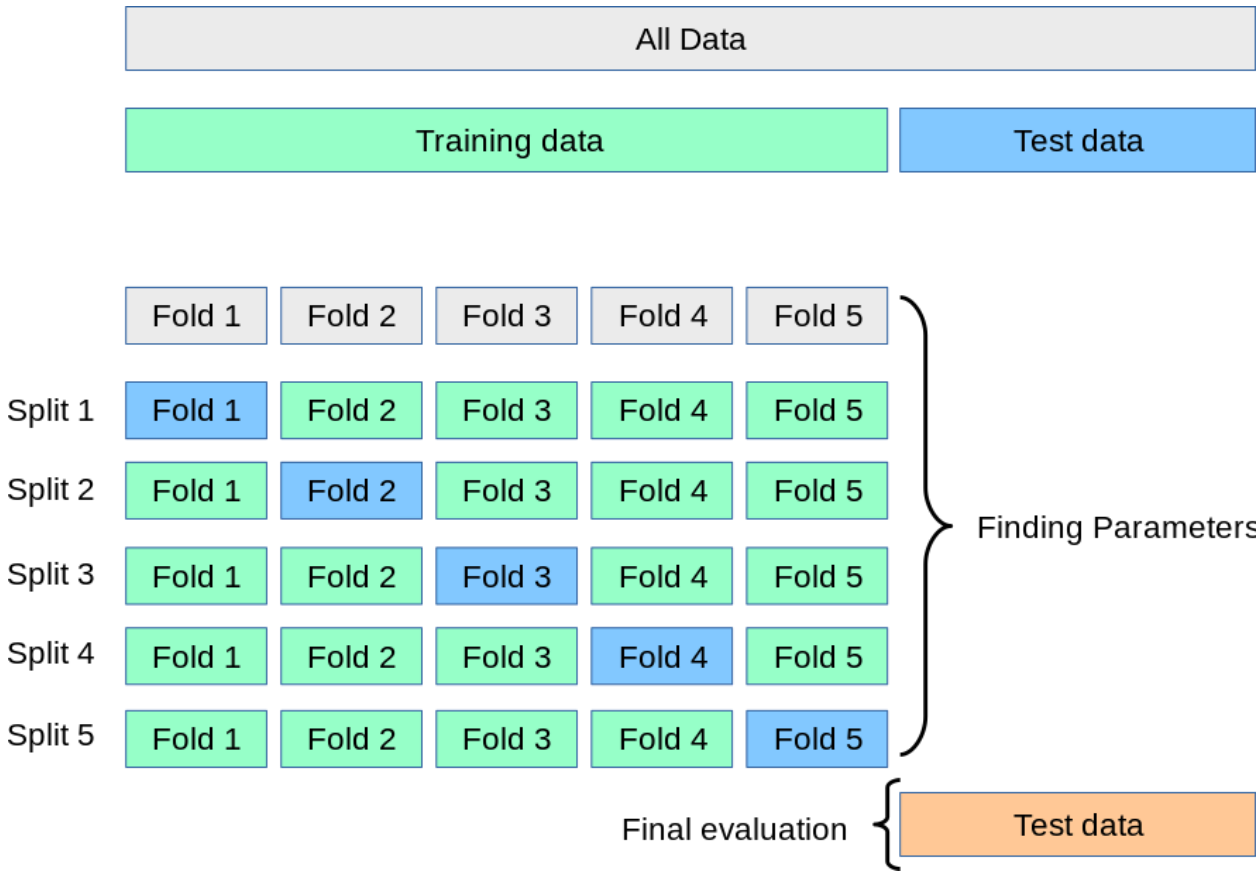
- We define a function named `compare_models()` to evaluate each model using cross-validation. Within this function, we will calculate the **Mean cross-validation accuracy score** for each model and print both the cross-validation accuracy scores and the mean accuracy across all folds.
- **RandomForestClassifier has the best mean Cross val Accuracy**

- Cross-validation is a technique used in machine learning to assess the performance of a predictive model. It involves partitioning the dataset into subsets, training the model on a subset of the data, and then evaluating it on the remaining subset. This process is repeated multiple times, with different subsets used for training and evaluation each time. The goal is to ensure that the model's performance is robust and not overly influenced by the specific data points used for training and testing.

We have used K-Fold cross validation here

K-Fold Cross validation: The training set is further divided into K subsets, or folds, of approximately equal size. The model is trained K times, each time using K-5 folds for training and one fold for validation. This ensures that each data point is used for validation exactly once.

K-Fold Cross Validation diagram



- From the model comparison, we observe that the **RandomForestClassifier** achieved the highest cross-validation accuracy score. Therefore, we select this model for further training. This selection is based on its superior performance in accurately classifying phishing instances compared to other models.

Model Training:

We split the dataset into training and testing sets using `train_test_split()`, with 80% of the data allocated for training and 20% for testing. This ensures that

we have independent datasets for training and evaluating the model's performance.

The selected model, RandomForestClassifier with optimized hyperparameters, is then instantiated with the best parameters. We fit this model on the training data using `fit()`

```
X_train, X_test, Y_train, Y_test =
train_test_split(X, Y, test_size=0.2,
random_state=42)
```

Random state: The `random_state` parameter is used in machine learning algorithms, particularly in tasks involving data splitting, shuffling, or randomization. It is a parameter that controls the randomness of the algorithm's operations, ensuring reproducibility of results.

In the context of `train_test_split()` function, setting the `random_state` parameter to a specific value ensures that the data splitting process is deterministic. When you provide a particular integer value to `random_state`, the data is split in the same way every time the code is run with that specific value, resulting in consistent train-test splits**.**

Evaluation Metrics:

Accuracy:

- **Definition:**Accuracy represents the proportion of correctly classified instances out of the total instances.
- **Formula:** Accuracy = (TP + TN) / (TP + TN + FP + FN)
- **Interpretation:** Accuracy measures the overall correctness of the model's predictions.

Precision:

- **Definition:** Precision measures the proportion of true positive predictions among all positive predictions made by the model.
- **Formula**: Precision = TP / (TP + FP)
- **Interpretation:** Precision indicates the model's ability to avoid false positives.

Recall:

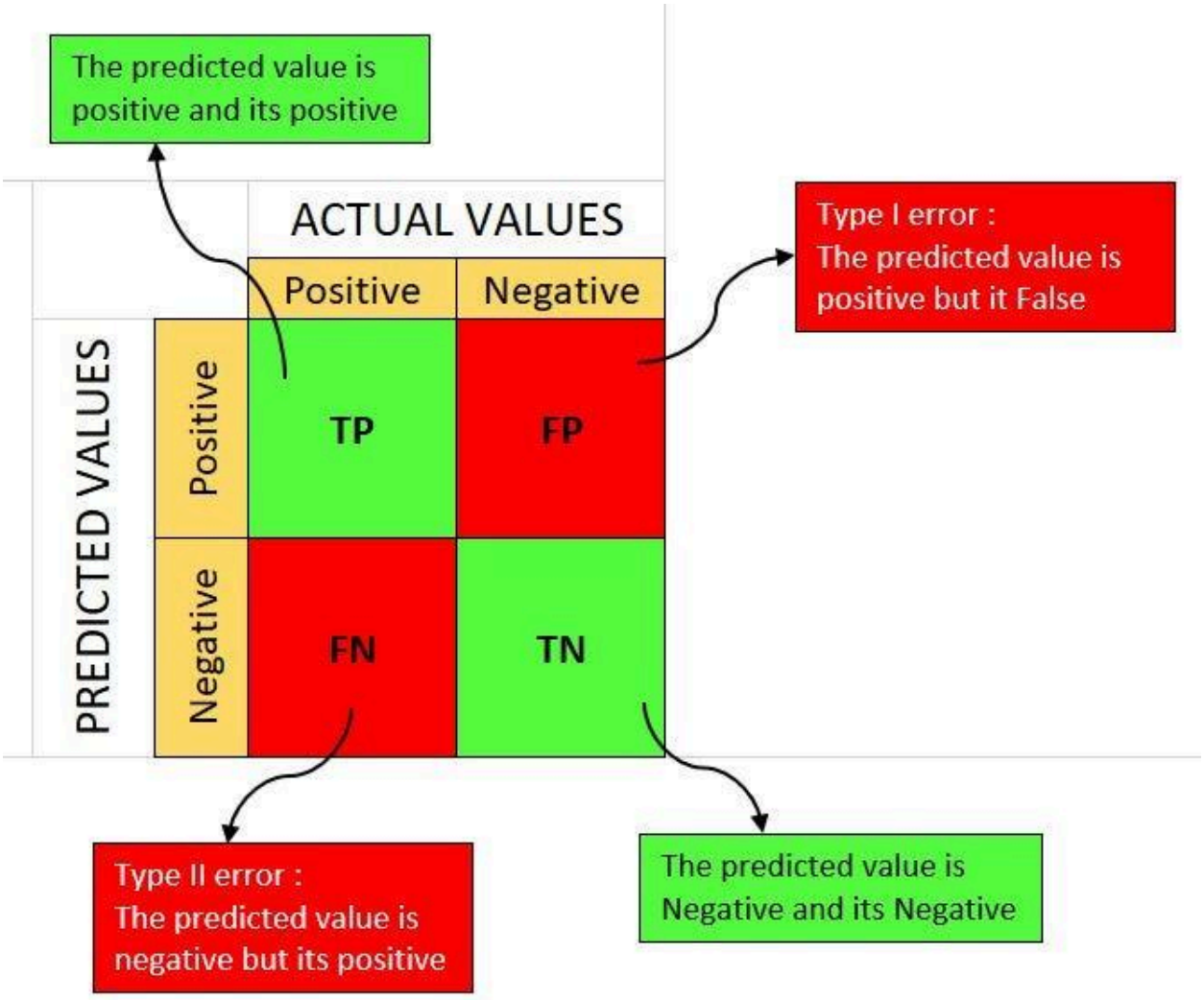
- **Definition:** Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances that are correctly predicted by the model.
- **Formula:** Recall = TP / (TP + FN)
- **Interpretation**: Recall reflects the model's ability to capture all positive instances.

F1 Score:

- **Definition:** F1 score is the harmonic mean of precision and recall, providing a balance between them.
- Formula: F1 Score = 2 * ((Precision * Recall) / (Precision + Recall))
- **Interpretation**: F1 score combines precision and recall into a single metric, useful for imbalanced datasets.

Confusion Matrix:

- **Definition:** A confusion matrix is a table that visualizes the performance of a classification model, showing the counts of true positives, true negatives, false positives, and false negatives.
- **Interpretation:** It provides insights into the model's behavior, helping to identify areas of improvement such as misclassifications and errors.



Model Evaluation:

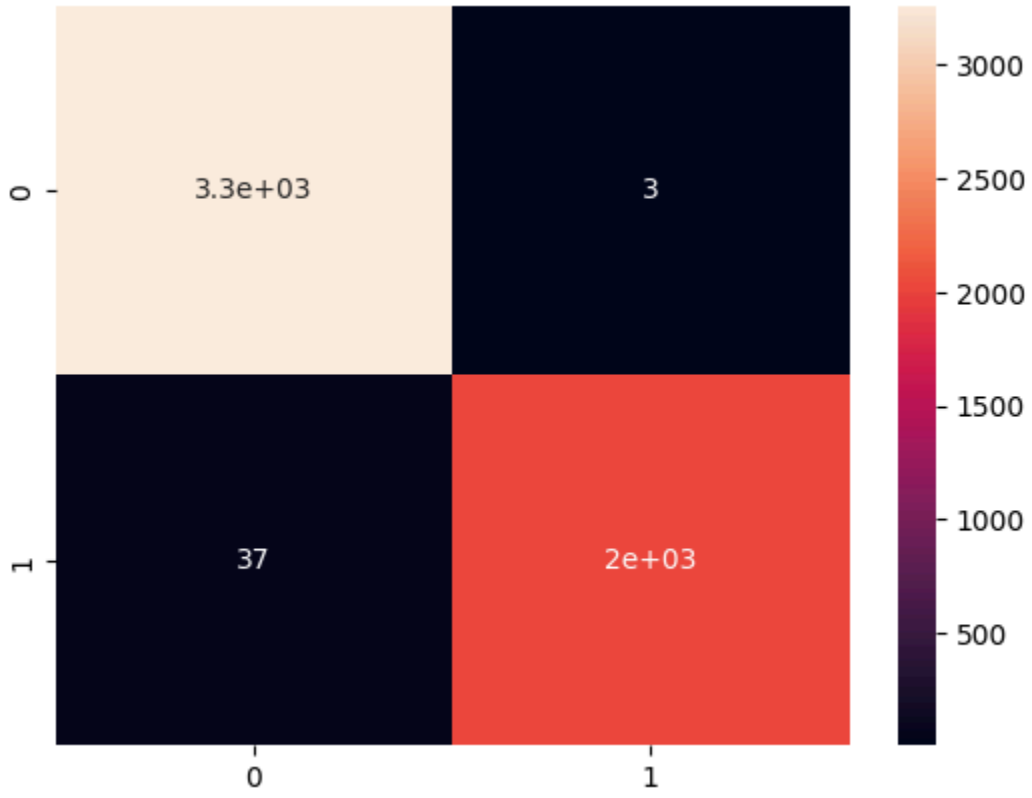
Testing Data Evaluation:

- To evaluate our model using the testing dataset, we compare the predicted labels to the actual labels and calculate various evaluation metrics.
- These metrics, including accuracy, precision, recall, and F1 score, give us insights into how well our model performs in identifying phishing websites

All the scores that we have got from training the RandomClassifier model on both training and testing

Metrics	Dataset	Scores
Accuracy	Train	0.998213
Accuracy	Test	0.992477
Precision	Train	0.998346
Precision	Test	0.998513
Recall	Train	0.997167
Recall	Test	0.981969
F1 Score	Train	0.997756

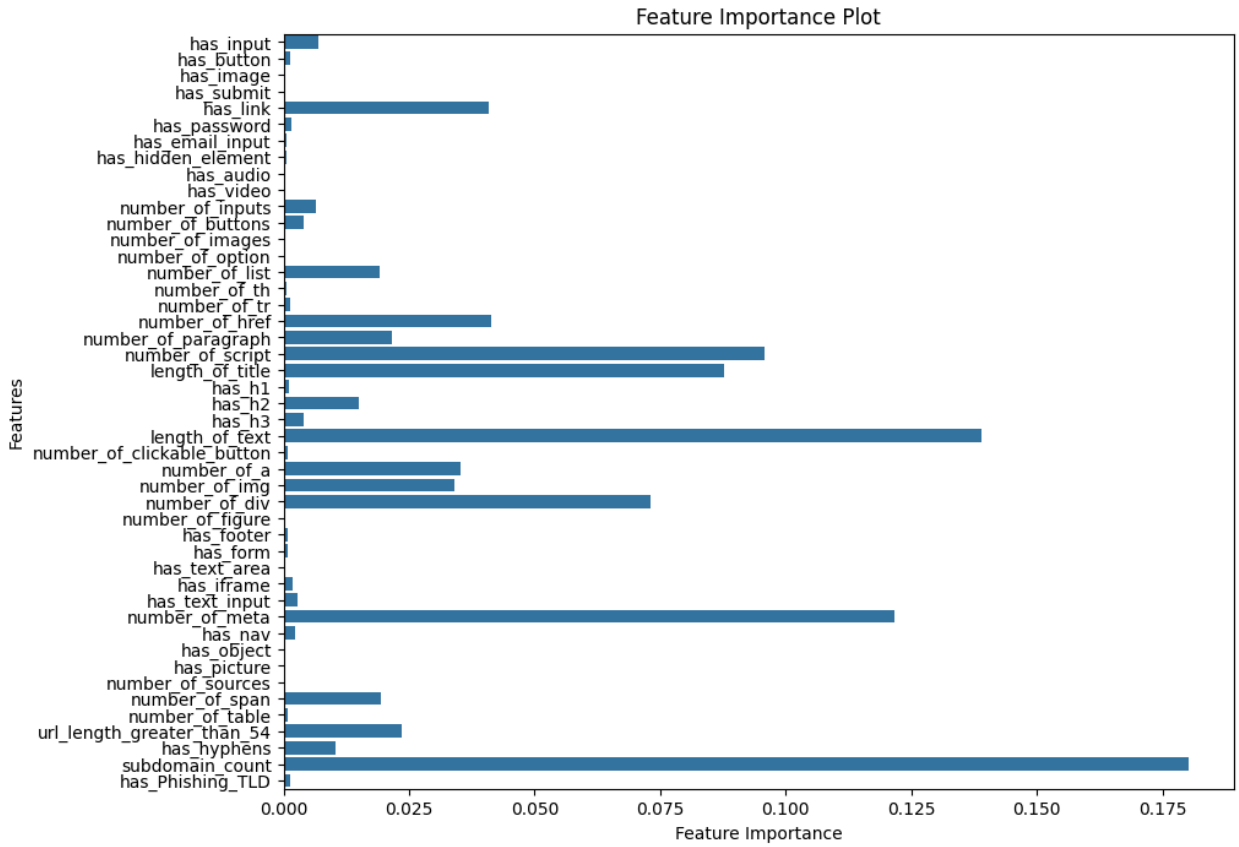
CONFUSION MATRIX



(confusion matrix of my test data prediction)

Feature Importance:

In machine learning, feature importance refers to the relative contribution of each input variable (feature) to the model's predictions. Understanding feature importance is crucial for gaining insights into the underlying patterns learned by the model and identifying which features have the most influence on the target variable.



- The feature importance plot provides valuable insights into the key factors that contribute to the detection of web phishing attempts. The plot shows that features related to the presence and properties of various form elements, such as "has_input", "has_button", "has_image", and "has_submit", are among the most important indicators of potential phishing activity. Additionally, features associated with the hyperlink structure, such as "has_link" and "number_of_links", also play a significant role in the classification process.

- The analysis suggests that the presence and characteristics of interactive elements on a webpage, as well as the way links are structured, are crucial in distinguishing legitimate websites from those engaging in phishing attempts.

Future Enhancement:

Incorporating JavaScript Features:

- JavaScript plays a crucial role in modern web applications, and its presence can reveal valuable insights into website behavior and functionality.
- Future enhancements could involve extending feature extraction capabilities to include **JavaScript attributes such as event handlers, DOM manipulations, and AJAX requests.**
- By incorporating **JavaScript features,** we can capture dynamic interactions and behaviors that may not be evident from static HTML content alone.
- This enhancement would require leveraging tools and techniques for parsing and analyzing JavaScript code, such as headless browsers or JavaScript interpreters.
- By enriching our feature set with **JavaScript attributes, we can enhance the model's ability to detect sophisticated phishing techniques that rely on dynamic content generation** and user interaction.

Model Persistence using Joblib:

Saving the Trained Model:

- After training our phishing detection model, we utilize the Joblib library to persist the trained model to disk.
- The `joblib.dump()` function is employed to serialize the model object and save it as a binary file in the desired location.

Incorporating the Model into Streamlit Application:

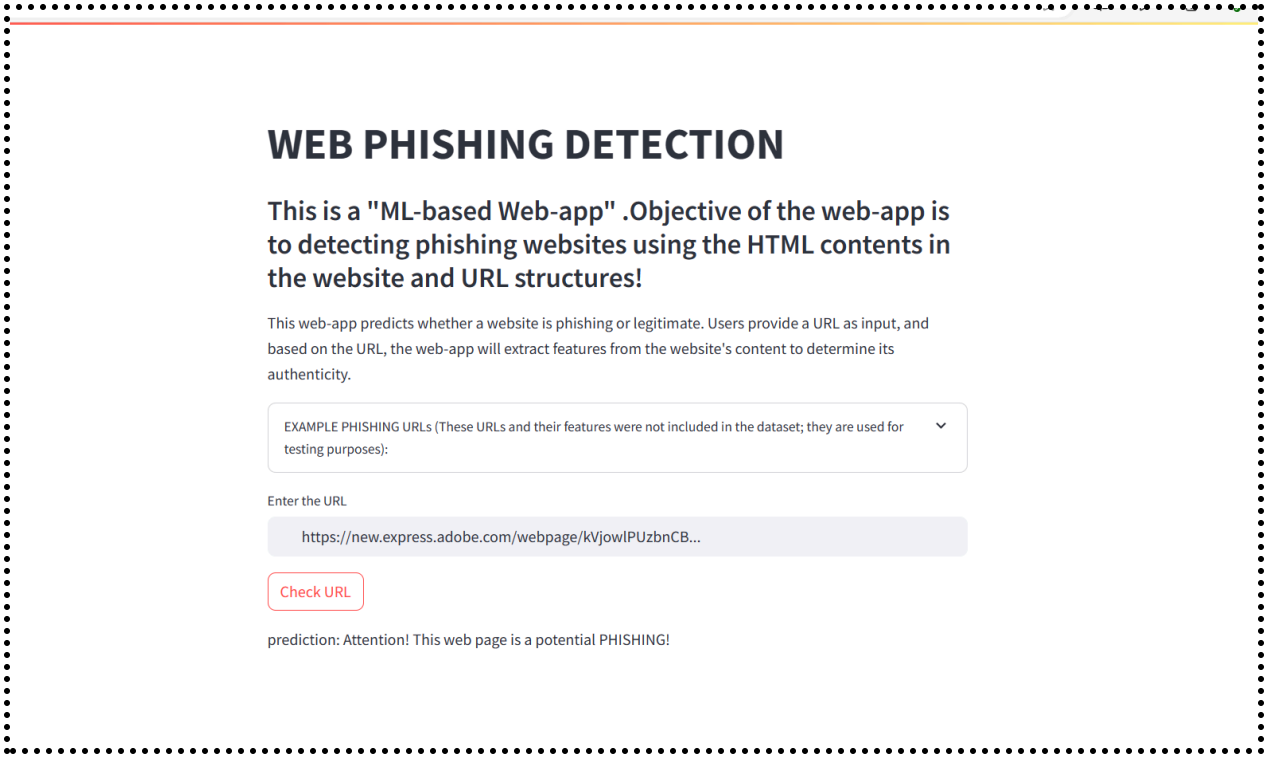
Streamlit Integration:

- Streamlit is a powerful Python library for building interactive web applications with minimal code, making it an excellent choice for deploying machine learning models.
- We can integrate our trained phishing detection model with Streamlit to create a user-friendly interface for users to input URLs and receive real-time predictions.

Model Loading and Inference:

- Within the Streamlit application, we load the pre-trained machine learning model along with any necessary preprocessing steps.
- When a user submits a URL through the interface, the application preprocesses the input data, passes it through the model for prediction, and displays the result to the user.

User interface design



- The user interface of the web phishing detection application is straightforward and intuitive. Users can simply enter a URL in the designated input field, and then click the "Check URL" button to initiate the phishing detection process.
- The application's objective is clearly stated at the top of the interface - it aims to detect phishing websites by analyzing the HTML content and URL structure of the provided website. This makes the purpose of the tool transparent to users.

- Upon entering a URL and clicking the "Check URL" button, the application will process the website's features and provide a prediction on whether the site is a potential phishing attempt or not.

The prediction is displayed prominently, allowing users to easily understand the assessment.

- The inclusion of an example phishing URL section further enhances the usability of the application, as it provides users with sample URLs they can use for testing and demonstration purposes. This helps users familiarize themselves with the tool and understand its capabilities.

- Overall, the user interface is designed to be simple, user-friendly, and focused on the core functionality of web phishing detection. This makes the application accessible and easy to use for a wide range of users, including those who may not have extensive technical expertise.

—----------Thank you—----------