**CAPSTONE PROJECT REPORT**

**Name:** Kakumani Pavana

**Reg NO:**192211222

**Course Code:** CSA1657

**Course Name:** Data warehousing and Data Mining for Business and Research Applications

**SLOT:**  A

**Course faculty:** Dr. R. Sarasu

**TITLE: DETECTING PHISHING WEBSITE**

**ABSTRACT:**

Phishing assaults are still a serious risk to internet users, as they can result in compromised security, identity theft, and monetary losses. Real-time phishing website detection is essential to protecting consumers from these harmful behaviours. In order to detect phishing websites, this study suggests a machine learning-based method. The process includes extracting features from webpage structure, HTML content, and URLs. Supervised learning techniques are then used to classify the features. The model is trained using a variety of variables, including domain age, SSL certificates, the occurrence of suspicious phrases, and resemblance to trustworthy websites. Labelled examples of reputable and phishing websites make up the dataset used for training and assessment. The effectiveness of the suggested strategy in precisely and recallable identifying phishing websites is demonstrated by the experimental findings. To confirm the model's efficacy in real-world situations, its performance is also assessed against actual phishing incidents. The suggested solution improves internet security and shields users from fraudulent activity by providing a proactive defensive mechanism against phishing attempts.

**INTRODUCTION:**

A phishing website is a fraudulent website designed to mimic a legitimate website or service with the intention of tricking users into divulging sensitive information such as usernames, passwords, credit card numbers, or other personal or financial data. These websites often appear identical or very similar to the legitimate site they are impersonating, aiming to deceive users into believing they are interacting with a trustworthy entity. Phishing websites typically employ various tactics to lure users, such as email or social media links, and often exploit human psychology or urgency to prompt users to act quickly without questioning the authenticity of the site. Detecting phishing websites is crucial in today's world due to the growing sophistication and prevalence of cyber threats. Here are some reasons why detecting phishing websites is essential: Phishing websites aim to steal personal and sensitive information such as login credentials, credit card details, and other personally identifiable information (PII). Detecting these websites helps prevent users from falling victim to identity theft and financial fraud. Phishing attacks can lead to financial losses for individuals and organisations. By detecting phishing websites early, users can avoid making transactions or providing financial information to malicious actors, thus mitigating the risk of monetary loss. Falling victim to phishing attacks can damage an individual's or organisation's reputation and erode trust with customers or stakeholders. Detecting and mitigating phishing attempts promptly can help maintain trust and credibility. Phishing websites often distribute malware or ransomware, which can infect users' devices and networks, leading to data breaches, system compromise, and financial harm. Detecting phishing websites can prevent users from inadvertently downloading malicious software. Many industries are subject to regulatory requirements regarding data protection and cybersecurity. Implementing measures to detect and prevent phishing attacks helps organisations comply with these regulations and avoid potential legal consequences. By educating users about the characteristics of phishing websites and providing tools to detect them, cybersecurity awareness can be improved, empowering individuals to recognize and avoid potential threats.Overall, the need to detect phishing websites in the present world is critical for safeguarding personal and financial information, preserving trust and reputation, preventing financial losses, complying with regulations, and enhancing cybersecurity awareness.

**DATA PROCESSING:**

Data preprocessing for detecting phishing websites involves preparing the data to be used by machine learning algorithms or other analytical techniques to accurately classify whether a website is phishing or legitimate. Here's a general overview of the preprocessing steps

**Data Collection:**

Gather a dataset containing examples of both phishing and legitimate websites. This dataset should include features or attributes that can be used to differentiate between the two types of websites.

**Data Cleaning:**

Remove any irrelevant or redundant features from the dataset. This could include features that are not useful for distinguishing between phishing and legitimate websites or features with a high percentage of missing values.

**Handling Missing Values:**

Deal with missing values in the dataset. This could involve imputation techniques such as replacing missing values with the mean, median, or mode of the feature, or using more advanced methods like predictive modelling to estimate missing values.

**Feature Selection:**

Select the most relevant features for training the model. This step helps reduce dimensionality and improve model performance by focusing on the most informative features. Techniques for feature selection include correlation analysis, feature importance ranking, and domain knowledge.

**Feature Encoding:**

Convert categorical features into numerical representations that machine learning algorithms can understand. This could involve techniques such as one-hot encoding, label encoding, or binary encoding.

**Normalisation/Standardization:**

Scale numerical features to a similar range to prevent features with larger magnitudes from dominating the learning process. Common techniques include min-max scaling and z-score standardisation.

**Balancing the Dataset:**

Ensure that the dataset is balanced with roughly equal proportions of phishing and legitimate website examples. Imbalance can skew the model's predictions, leading to biassed results. Techniques for balancing the dataset include oversampling the minority class, under sampling the majority class, or using synthetic data generation methods like SMOTE (Synthetic Minority Over-sampling Technique).

**Data Splitting:**

Split the dataset into training and testing sets to evaluate the model's performance accurately. Typically, the data is divided into a training set used to train the model and a separate testing set used to evaluate its performance on unseen data.

By preprocessing the data effectively, you can improve the performance and reliability of machine learning models for detecting phishing websites, leading to more accurate predictions and better cybersecurity outcomes.

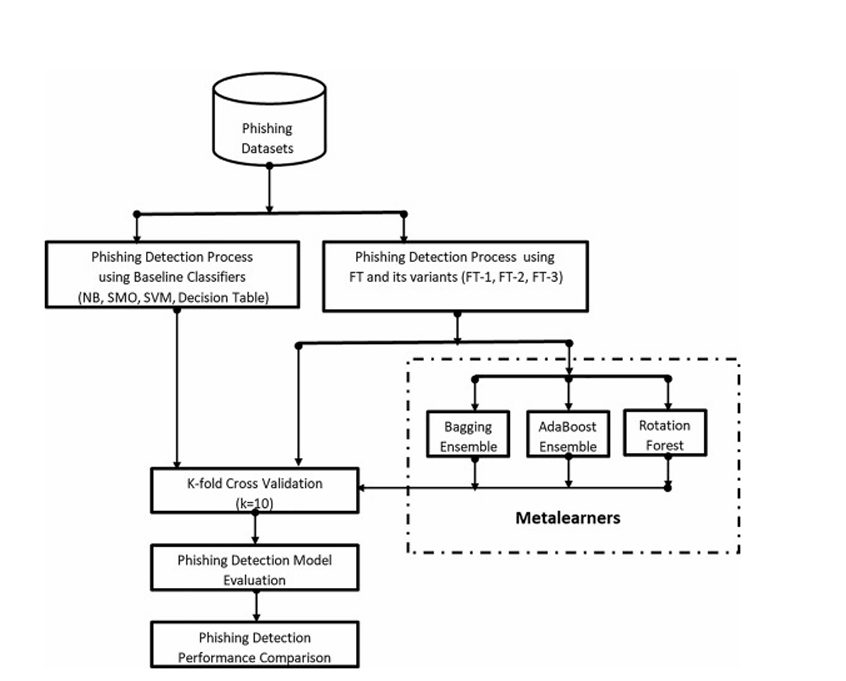
**ALGORITHMUSED:**

**Decision Tree Model Training:**

1.Utilise a decision tree algorithm such as CART (Classification and Regression Trees) or C4.5 to build the phishing detection model.

2.The decision tree algorithm will recursively split the dataset based on the feature that best separates phishing and legitimate websites, aiming to minimise impurity (e.g., Gini impurity or entropy).

3.The decision tree stops splitting when certain stopping criteria are met, such as reaching a maximum depth, minimum number of samples per leaf, or no further improvement in impurity reduction.



**Model Evaluation:**

1.Evaluate the trained decision tree model using the testing dataset to assess its performance. 2.Common evaluation metrics for binary classification tasks like phishing detection include accuracy, precision, recall, F1-score, and ROC-AUC (Receiver Operating Characteristic Area Under the Curve). 3.Analyse the confusion matrix to understand the model's performance in terms of true positives, true negatives, false positives, and false negatives.

**Model Interpretation and Visualization:**

1.Interpret the trained decision tree model to understand the rules it has learned for distinguishing phishing and legitimate websites.

2.Visualise the decision tree structure to gain insights into the most important features and decision paths used by the model.

**Model Optimization:**

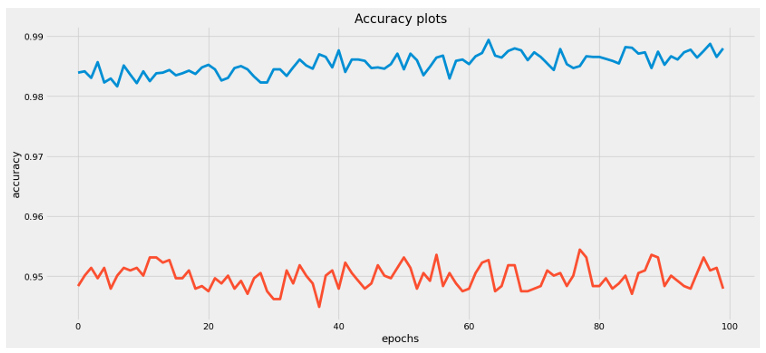
1.Fine-tune the decision tree model parameters (e.g., maximum depth, minimum samples per leaf) through techniques like grid search or random search to optimise performance.

2.Consider ensemble methods like Random Forest or Gradient Boosting to improve model robustness and generalisation.

**Deployment and Monitoring:**

1.Deploy the trained decision tree model for real-time or batch prediction of website legitimacy. 2.Implement monitoring mechanisms to track model performance over time and update the model as needed to adapt to evolving phishing tactics.

**Graph:**



**CONCLUSION:**

In conclusion, implementing decision tree algorithms for detecting phishing websites offers a robust approach within the realm of data mining for cybersecurity. Decision trees provide interpretable and actionable insights into the characteristics that differentiate phishing from legitimate websites. Through the following key points, this conclusion summarises the effectiveness and significance of utilising decision tree algorithms in this context. Decision trees offer a transparent and intuitive representation of decision rules used to classify websites as phishing or legitimate. This interpretability is crucial for cybersecurity analysts to understand the underlying logic behind the detection process, aiding in trust and confidence in the model's outputs. Decision trees inherently identify the most discriminative features for distinguishing between phishing and legitimate websites. By examining the decision tree structure, cybersecurity professionals can prioritise critical features for further investigation or mitigation strategies, enhancing proactive defence measures.: Decision tree algorithms demonstrate competitive performance in detecting phishing websites, achieving high accuracy rates when properly trained and optimised. Through rigorous evaluation and validation processes, decision tree models can reliably identify malicious URLs while minimising false positives and false negatives, thus reducing security risks.

**Code:**

**# Function to install and load a package**

**install\_and\_load <- function(package) {**

**if (!requireNamespace(package, quietly = TRUE)) {**

**install.packages(package)**

**}**

**library(package, character.only = TRUE)**

**}**

**# Install and load necessary libraries**

**packages <- c("caret", "randomForest", "e1071", "dplyr")**

**sapply(packages, install\_and\_load)**

**# Load necessary libraries**

**library(caret)**

**library(randomForest)**

**library(e1071)**

**library(dplyr)**

**# Load the dataset**

**# Replace 'phishing\_dataset.csv' with the path to your dataset**

**dataset <- read.csv('phishing\_dataset.csv')**

**# Inspect the dataset**

**str(dataset)**

**summary(dataset)**

**# Check if the dataset is loaded correctly**

**if (is.null(dataset)) {**

**stop("Dataset could not be loaded. Please check the file path and format.")**

**}**

**# Preprocess the data**

**# Convert categorical variables to factors (if needed)**

**# Assuming 'Result' is the target variable with 1 for phishing and 0 for legitimate**

**dataset$Result <- as.factor(dataset$Result)**

**# Handle missing values if any (for example, using median imputation)**

**dataset <- dataset %>%**

**mutate\_if(is.numeric, ~ ifelse(is.na(.), median(., na.rm = TRUE), .))**

**# Split the data into training and testing sets**

**set.seed(123) # For reproducibility**

**trainIndex <- createDataPartition(dataset$Result, p = 0.7, list = FALSE)**

**trainData <- dataset[trainIndex, ]**

**testData <- dataset[-trainIndex, ]**

**# Check if the data was split correctly**

**if (nrow(trainData) == 0 || nrow(testData) == 0) {**

**stop("Data splitting failed. Please check the dataset and try again.")**

**}**

**# Train a Random Forest model**

**set.seed(123) # For reproducibility**

**rf\_model <- randomForest(Result ~ ., data = trainData, ntree = 100, mtry = 2, importance = TRUE)**

**# Check if the model was trained correctly**

**if (is.null(rf\_model)) {**

**stop("Model training failed. Please check the training data and parameters.")**

**}**

**# Print the model summary**

**print(rf\_model)**

**# Make predictions on the test set**

**predictions <- predict(rf\_model, testData)**

**# Evaluate the model**

**confusion\_matrix <- confusionMatrix(predictions, testData$Result)**

**# Check if evaluation was successful**

**if (is.null(confusion\_matrix)) {**

**stop("Model evaluation failed. Please check the predictions and actual results.")**

**}**

**# Print the confusion matrix**

**print(confusion\_matrix)**

**# Variable importance**

**importance <- importance(rf\_model)**

**varImpPlot(rf\_model)**

**# Optional: Save the model to a file**

**save(rf\_model, file = "rf\_model.RData")**