|  |  |
| --- | --- |
| **Case Study Details** | |
| **Domain** | **Cyber Security** |
| **Title of the Case Study** | **Phishing URL Detection** |
| **Tools Used** | **Python** |

|  |
| --- |
| **Name of the Dataset: URL Phishing Detection Dataset** |
| **Dataset URL (Active in online): Not active** |
| **Dataset Description**:  This dataset contains various features related to URLs, such as domain properties, structure, redirection behavior, and security indicators. It is used to classify whether a URL is legitimate or phishing. |
| **Features in Dataset: (include all feature names and their descriptions as per the information available at the source of dataset (Kaggle / UCI Data Repository etc)**   |  |  | | --- | --- | | **Domain** | The domain name of the URL | | **Have\_IP** | Whether the URL contains an IP address instead of a domain name (1 = Yes, 0 = No) | | **Have\_At** | Whether the URL contains an '@' symbol (1 = Yes, 0 = No) | | **URL\_Length** | Length of the URL | | **URL\_Depth** | Number of directory levels in the URL path | | **Redirection** | Whether the URL redirects (1 = Yes, 0 = No) | | **https\_Domain** | Indicates if the domain includes "https" (1 = Yes, 0 = No) | | **TinyURL** | Indicates if the URL is a shortened URL (1 = Yes, 0 = No) | | **Prefix/Suffix** | Checks if there is a prefix or suffix in the domain (1 = Yes, 0 = No) | | **DNS\_Record** | Whether the domain has a DNS record (1 = Yes, 0 = No) | | **Web\_Traffic** | Website traffic ranking (1 = High, 0 = Low) | | **Domain\_Age** | Age of the domain in years | | **Domain\_End** | Expiry status of the domain (1 = Active, 0 = Expired) | | **iFrame** | Whether the webpage uses iFrames (1 = Yes, 0 = No) | | **Mouse\_Over** | Whether the webpage triggers unexpected actions on mouse hover (1 = Yes, 0 = No) | | **Right\_Click** | Whether the webpage disables right-click (1 = Yes, 0 = No) | | **Web\_Forwards** | Number of times the webpage forwards to another URL | | **Label** | Target variable (1 = Phishing, 0 = Legitimate) | |
| **Number of Features in Dataset: 18** |
| **Number of Samples (records) in Dataset: 10000** |
| **Is the dataset is having null values: No** |
| **Is the dataset is having missing values: No** |
| **Is the dataset is in encoded format of PCA values: No** |
| **Is it essential to pre-process the dataset for the case study: Yes / No**  Yes, preprocessing is essential for effective analysis and machine learning.  **If Yes, how you want to preprocess? Give details:**  1. Feature Scaling: Normalize numerical features like URL\_Length, URL\_Depth, Domain\_Age.  2. Handling Imbalanced Data: If the phishing vs. legitimate URLs are imbalanced, apply oversampling/undersampling.  3. Encoding Categorical Features: Convert Domain into numerical representations if used.  4. Feature Engineering: Create additional derived features such as keyword analysis in URLs. |
| **List out the possible opportunities for analysis on this dataset based on the available features**  -> Phishing Detection Model: Train classification models (Logistic Regression, Random Forest, SVM) to detect phishing URLs.  -> Feature Importance Analysis: Identify the most influential features contributing to phishing detection.  -> URL Pattern Analysis: Compare legitimate vs. phishing URL structures.  -> Website Traffic and Domain Age Impact: Study how traffic and age affect legitimacy.  -> Model Performance Comparisons: Compare ML models based on accuracy, precision, recall, and F1-score. |

|  |
| --- |
| **Title of the Case Study:**  **Phishing URL Detection** |
| **List of Objectives:**  1. To analyze URLs and classify them as phishing or legitimate.  2. To perform Exploratory Data Analysis (EDA) to understand data distribution.  3. To visualize correlations and trends in the dataset.  4. To apply different machine learning models for classification.  5. To identify the most accurate model for predicting phishing URLs.  6. To deploy a trained model that can classify a given URL as phishing or not. |
| **Approach: What features are going to be considered, processed, or feature-engineered to derive a specific outcome after applying one or more models?**  -> Features considered: The case study focuses on detecting phishing URLs by analyzing various URL-based features such as presence of IP addresses, "@" symbols, URL length, redirections, HTTPS usage, domain age, and web traffic. After performing Exploratory Data Analysis (EDA) and visualizing key patterns, we trained multiple models including Decision Tree, Random Forest, SVM, and XGBoost. XGBoost achieved the highest accuracy, making it the final model for deployment. The trained model takes a URL as input and predicts whether it is phishing or legitimate, helping in cybersecurity threat detection.  -> Preprocessing: Removing unnecessary columns (e.g., "Domain"), handling missing values, and normalizing the data.  -> Feature Engineering: Transforming raw features into meaningful insights for training models.  -> Model Selection: Evaluating Decision Tree, Random Forest, SVM, and XGBoost classifiers.  -> Final Model: XGBoost performed best, so it was selected for final training. |

|  |
| --- |
| **Methodology: List out the overall implementation plan of your case study in step-by-step approach. (Data Preprocessing, Feature selection, Feature engineering, model selection, model building, model training approach, model testing, evaluation of metrics etc)** |
| -> Data Loading – Importing the dataset (5.urldata.csv).  -> Exploratory Data Analysis (EDA) – Understanding data structure, missing values, and feature distributions.  -> Data Preprocessing – Dropping irrelevant columns and checking for null values.  -> Visualization – Correlation heatmaps, phishing class distribution, and bar/pie charts for insights.  -> Model Training – Applying Decision Tree, Random Forest, SVM, and XGBoost classifiers.  -> Model Evaluation – Comparing accuracy, precision, recall, and F1-score for all models.  -> Final Model Selection – XGBoost achieved the highest accuracy, so it was chosen for final deployment.  -> Prediction – Accepting user input (URL) and predicting whether it is phishing or not. Data Loading – Importing the dataset (5.urldata.csv).  -> Exploratory Data Analysis (EDA) – Understanding data structure, missing values, and feature distributions.  -> Data Preprocessing – Dropping irrelevant columns and checking for null values.  -> Visualization – Correlation heatmaps, phishing class distribution, and bar/pie charts for insights.  -> Model Training – Applying Decision Tree, Random Forest, SVM, and XGBoost classifiers.  -> Model Evaluation – Comparing accuracy, precision, recall, and F1-score for all models.  -> Final Model Selection – XGBoost achieved the highest accuracy, so it was chosen for final deployment.  -> Prediction – Accepting user input (URL) and predicting whether it is phishing or not. |

|  |  |
| --- | --- |
| Task | Loading Dataset |
| Step-1 | Loading Dataset with name, display its descriptive information |
| Description | The dataset contains URL-based features necessary for phishing detection. We observed 10000 number of rows and 18 number of columns. |
| Code | #Loading the data  data0 = pd.read\_csv('/content/5.urldata.csv')  data0.head()  data0.shape |
| Result | Successfully loaded the dataset and displayed its first few rows.  (10000, 18) |
| Description about results in detailed way | Output the details regarding the dataset after loading the dataset. |

|  |  |
| --- | --- |
| Task | Exploratory Data Analysis |
| Step -2 | Exploratory Data Analysis |
| Description | We performed EDA to understand the data distribution, feature importance, and identify any anomalies. Various visualizations such as histograms, boxplots, and correlation heatmaps were used to analyze patterns. |
| Code | #Dropping the Domain column  data = data0.drop(['Domain'], axis = 1).copy() # No need much for model training and predictions  data0.columns  data.isnull().sum() |
| Result | Index(['Have\_IP', 'Have\_At', 'URL\_Length', 'URL\_Depth',  'Redirection', 'https\_Domain', 'TinyURL', 'Prefix/Suffix', 'DNS\_Record',  'Web\_Traffic', 'Domain\_Age', 'Domain\_End', 'iFrame', 'Mouse\_Over',  'Right\_Click', 'Web\_Forwards', 'Label'],  dtype='object') |
| Description about results in detailed way | The dataset contains some highly correlated features. Certain features like URL length and domain age show significant differentiation between phishing and legitimate websites. |

|  |  |
| --- | --- |
| Task | Splitting the Dataset & Handling Missing Values |
| Step - 3 | Dataset Splitting |
| Description | Separating & assigning features and target columns to X & y and splitting the dataset. |
| Code | # Sepratating & assigning features and target columns to X & y  y = data['Label'] # target  X = data.drop('Label',axis=1)  X.shape, y.shape  # Splitting the dataset into train and test sets: 80-20 split  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,                                                      test\_size = 0.2, random\_state = 12)  X\_train.shape, X\_test.shape |
| Result | ((10000, 16), (10000,))  ((8000, 16), (2000, 16)) |
| Description about results in detailed way | The dataset was split into training and testing sets with different splits to evaluate performance across different models. Null values were either replaced or removed based on their significance. |

|  |  |
| --- | --- |
| Task | **Training the Models** |
| Step - 4 | Model Training (Decision Tree, Random Forest, SVM, XGBoost) |
| Description | Multiple machine learning models, including Decision Tree, Random Forest, SVM, and XGBoost, were trained using the training dataset to evaluate their accuracy in detecting phishing URLs. |
| Code | # Decision Tree model  from sklearn.tree import DecisionTreeClassifier  # instantiate the model  tree = DecisionTreeClassifier(max\_depth = 5)  # fit the model  tree.fit(X\_train, y\_train)  #predicting the target value from the model for the samples  y\_test\_tree = tree.predict(X\_test)  y\_train\_tree = tree.predict(X\_train)  #computing the accuracy of the model performance  acc\_train\_tree = accuracy\_score(y\_train,y\_train\_tree)  acc\_test\_tree = accuracy\_score(y\_test,y\_test\_tree)  print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc\_train\_tree))  print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc\_test\_tree))  storeRes('Decision Tree',acc\_test\_tree)  # Random Forest model  from sklearn.ensemble import RandomForestClassifier  forest = RandomForestClassifier(max\_depth=5)  forest.fit(X\_train, y\_train)  y\_test\_forest = forest.predict(X\_test)  y\_train\_forest = forest.predict(X\_train)  storeRes('Random Forest',acc\_test\_forest)  #Support vector machine model  from sklearn.svm import SVC  # instantiate the model  svm = SVC(kernel='linear', C=1.0, random\_state=12)  #fit the model  svm.fit(X\_train, y\_train)  acc\_train\_svm = accuracy\_score(y\_train,y\_train\_svm)  acc\_test\_svm = accuracy\_score(y\_test,y\_test\_svm)  print("SVM: Accuracy on training Data: {:.3f}".format(acc\_train\_svm))  print("SVM : Accuracy on test Data: {:.3f}".format(acc\_test\_svm))  storeRes('SVM',acc\_test\_svm)  #XGBoost Classification model  from xgboost import XGBClassifier  # instantiate the model  xgb = XGBClassifier(learning\_rate=0.4,max\_depth=7)  #fit the model  xgb.fit(X\_train, y\_train)  y\_pred = xgb.predict(X\_test) |
| Result |  |
| Description about results in detailed way | Model performance was compared based on accuracy, precision, recall, and F1-score. XGBoost performed the best with the highest accuracy. |

|  |  |
| --- | --- |
| Task | Selecting XGBoost Model |
| Step -5 | Final Model Selection |
| Description | After comparing different models, XGBoost was chosen as the final model because it achieved the highest accuracy in detecting phishing URLs. |
| Code | import joblib  import numpy as np  import re  from urllib.parse import urlparse  # Feature extraction function  def extract\_features(url):      parsed\_url = urlparse(url)      hostname = parsed\_url.hostname if parsed\_url.hostname else ""      path = parsed\_url.path if parsed\_url.path else ""      features = {}      features['Have\_IP'] = 1 if re.match(r'^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$', hostname) else 0      features['Have\_At'] = 1 if '@' in url else 0      features['URL\_Length'] = len(url)      features['URL\_Depth'] = path.count('/')      features['Redirection'] = 1 if '//' in url[7:] else 0      features['https\_Domain'] = 1 if 'https' in parsed\_url.scheme else 0      features['TinyURL'] = 1 if 'tinyurl' in url else 0      features['Prefix/Suffix'] = 1 if '-' in hostname else 0      features['DNS\_Record'] = 1 if hostname != '' else 0      features['Web\_Traffic'] = 1 if len(url) > 20 else 0  # This logic might need revision      features['Domain\_Age'] = 1 if len(hostname.split('.')) > 2 else 0      features['Domain\_End'] = 1 if hostname.endswith(('.com', '.org', '.net')) else 0      features['iFrame'] = 1 if '<iframe' in url else 0      features['Mouse\_Over'] = 1 if 'mouseover' in url else 0      features['Right\_Click'] = 1 if 'rightclick' in url else 0      features['Web\_Forwards'] = 1 if 'forward' in url else 0        return list(features.values())  # Load the XGBoost model  def load\_model(model\_path='xgb\_model.pkl'):      try:          model = joblib.load(model\_path)          return model      except FileNotFoundError:          print("Model file not found!")          return None  # Predict URL safety  def predict\_url\_safety(url, model\_path='xgb\_model.pkl'):      model = load\_model(model\_path)      if model is None:          return "Model not found. Train the model first."        features = extract\_features(url)      features\_array = np.array(features).reshape(1, -1)  # Reshape to match model input      prediction = model.predict(features\_array)        return "safe" if prediction[0] == 0 else "not safe" |
| Result | XGBoost was finalized for deployment. |
| Description about results in detailed way | XGBoost outperformed other models in terms of accuracy and efficiency, making it the best choice for real-time phishing detection. |

|  |  |
| --- | --- |
| Task | Predicting Output |
| Step -6 | Making Predictions |
| Description | We performed EDA to understand the data distribution, feature importance, and identify any anomalies. |
| Code | # Example usage  if \_\_name\_\_ == "\_\_main\_\_":      test\_url = 'http://www.accsystemprblemhelp.site/checkpoint.htm'  # Example URL      #test\_url='https://www.youtube.com/watch?v=dE4tG1mFZ-Q'      result = predict\_url\_safety(test\_url)      print(f"The URL '{test\_url}' is predicted to be {result}.")  #     print(result) |
| Result | The URL '<http://www.accsystemprblemhelp.site/checkpoint.htm>' is predicted to be safe. |
| Description about results in detailed way | The input URL is analyzed based on selected features, and the model predicts its legitimacy with high accuracy. |

Visualization:





