

Data Intensive Computing Project Phase 1

Detection of Phishing URL Websites

Problem Statement:

The goal of this project is to differentiate between legitimate and phishing websites based on their URL features.

Phishing attacks are one of the most widespread threats in cybersecurity. Phishing websites trick users into providing confidential information such as passwords, credit card details, and personal data by impersonating as legitimate websites. Phishing website detection is crucial because it helps prevent individuals from unknowingly providing sensitive personal information to malicious websites designed to steal data, potentially leading to identity theft, financial fraud, and other serious security breaches.

This project aims to develop a data-driven solution to classify websites as either phishing or legitimate, using machine learning techniques on a dataset containing various URL features. By performing data preprocessing and exploratory data analysis (EDA), we seek to reveal patterns and relationships within the data that can improve our model's accuracy.

Background and Significance:

With the increasing dependency on digital platforms, phishing attacks have escalated, threatening individuals, businesses, and governments. According to recent reports, phishing attacks have grown by more than 50% in the past year, leading to significant financial losses and data breaches. Despite educating users on best cybersecurity practices and implementing basic phishing detection systems, attackers continuously evolve their strategies, making traditional detection approaches inadequate. This project addresses the need for a reliable and scalable method of detecting phishing websites through URL features. By automating the detection process by using machine learning algorithms, we can enhance cybersecurity systems to proactively detect and prevent phishing attempts.

Potential Impact:

The project's contribution is valuable because it directly addresses a growing cybersecurity challenge with a scalable and data-intensive approach. The model developed from this project can be integrated into web browsers, firewalls, and

other security systems to provide real-time protection against phishing websites. Additionally, the insights gained from exploratory data analysis (EDA) on the URL features may help reveal patterns in phishing techniques, contributing to the broader research field of cybersecurity. This model can serve as a foundation for further innovations, such as developing more comprehensive tools to detect other types of cyber-attacks, ultimately improving online safety for users worldwide.

Data Sources [5 marks]:

The Dataset chosen is the Phishing dataset which was taken from the UC Irvine Machine Learning Repository. This dataset is a sizable dataset having 235794 samples and 56 columns(features). The data includes 100,945 phishing and 134,850 genuine URLs. The majority of the URLs used are from all the latest sources which include different features. Some features were already present and some were derived. The derived features include CharContinuationRate, URLTitleMatchScore, URLCharProb, and TLDLegitimateProb.

Data Cleaning/Processing [10 Marks]:

Data cleaning and processing are essential steps to ensure the quality, reliability, and usability of the dataset for further analysis.

- a. Handling Missing Values:

Feature	Count
URLLength	0
DomainLength	0
URLSimilarityIndex	0
NoOfLettersInURL	0
LetterRatioInURL	82534
NoOfDigitsInURL	0
DigitRatioInURL	211871
NoOfQMarkInURL	107645
NoOfOtherSpecialCharsInURL	0
SpacialCharRatioInURL	0
IsHTTPS	0
LineOfCode	0
HasTitle	0
DomainTitleMatchScore	0
HasFavicon	0
Robots	31756
IsResponsive	0
HasDescription	0
NoOfIFrame	1296
HasExternalFormSubmit	0
HasSocialNet	0
HasSubmitButton	0
HasHiddenFields	41972
HasPasswordField	0
Bank	0
Pay	0
HasCopyrightInfo	0
NoOfImage	25938
NoOfJS	0
NoOfSelfRef	0
NoOfEmptyRef	0
NoOfExternalRef	0
label	0
ETL NAME	a

Missing values can reduce the effectiveness of models and cause errors. Imputation preserves data integrity without needing to remove rows or columns unnecessarily.

So, to check the missing values “df.isnull().sum()” is used.

Then, to handle the missing values, imputation using mean values is done.

“df.fillna(df.mean(numeric_only = True), inplace = True)”

imputes mean values.

Shape after handling missing values: (235805, 38)	
URLLength	0
DomainLength	0
URLSimilarityIndex	0
NoOfLettersInURL	0
LetterRatioInURL	0
NoOfDigitsInURL	0
DigitRatioInURL	0
NoOfQMarkInURL	0
NoOfOtherSpecialCharsInURL	0
SpacialCharRatioInURL	0
IsHTTPS	0
LineOfCode	0
HasTitle	0
DomainTitleMatchScore	0
HasFavicon	0
Robots	0
IsResponsive	0
HasDescription	0
NoOfIFrame	0
HasExternalFormSubmit	0
HasSocialNet	0
HasSubmitButton	0
HasHiddenFields	0
HasPasswordField	0
Bank	0
Pay	0
HasCopyrightInfo	0
NoOfImage	0
NoOfJS	0
NoOfSelfRef	0
NoOfEmptyRef	0
NoOfExternalRef	0
label	0
FTI FNAMF	0

b. Handling duplicate entries:

Duplicate records can distort data patterns and bias statistical models. Removing them ensures that each dataset observation is distinct.

To identify duplicate and handle duplicates:

```
duplicates = df[df.duplicated()].sum()
df_cleaned = df.drop_duplicates()
```

c. Checking dtypes:

Standardising data types reduces further error.

The code iterates through all columns and checks if a column has less than 3 unique values.

When a column contains only two distinct values, it can be converted to a boolean type.

4 Checking dtypes

- Standardising data types reduces further error

```
[16]: for col in df.columns.unique():
    if len(df[col].unique())<3:
        print(col,df[col].unique())
        df[col] = df[col].astype(bool)
        print(col,df[col].unique())

print(df.dtypes.tail(25))
IsHTTPS [1 0]
IsHTTPS [ True False]
HasTitle [1 0]
HasTitle [ True False]
HasFavicon [0 1]
HasFavicon [False True]
IsResponsive [1 0]
IsResponsive [ True False]
HasDescription [0 1]
HasDescription [False True]
HasExternalFormSubmit [0 1]
HasExternalFormSubmit [False True]
HasSocialNet [0 1]
HasSocialNet [False True]
HasSubmitButton [1 0]
HasSubmitButton [ True False]
HasPasswordField [0 1]
HasPasswordField [False True]
Bank [1 0]
Bank [ True False]
Pay [0 1]
```

d. Removing outliers:

Outliers can negatively impact statistical analysis and model performance by skewing results. This can be addressed by using Interquartile Range (IQR) method which detects and removes outliers from the dataset.

The code written identifies the columns in the DataFrame that contain numerical data types. It determines the first and third quartiles (Q1 and Q3) for every numerical column. The difference between Q3 and Q1 is then used to calculate IQR.

Outliers are defined as having lower and upper bounds of 1.5 times the IQR below Q1 and above Q3 respectively. To remove rows where the column value is outside the specified bounds, the DataFrame is filtered.

e. Feature Scaling:

Feature scaling ensures that no feature dominates another due to different magnitude, which can improve convergence and performance.

This includes Min-Max Scaling that transforms the data such that all values lie between the range 0 to 1.

The formula used is:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

Exploratory Data Analysis (EDA)

Analysis of data:

df.describe() summarizes statistical properties of numerical columns. This helps understand the distribution and range of values in the dataset. The function returns the count, mean, standard deviation, minimum, 25th percentile, 50th percentile, 75th percentile, maximum of each column.

	URLLength	DomainLength	IsDomainIP	URLSimilarityIndex	TLDLength	NoOfSubDomain	HasObfuscation	NoOfObfuscatedChar	ObfuscationF
count	235805.000000	235805.000000	235805.000000	235805.000000	235805.000000	235805.000000	235805.000000	235805.000000	235805.000000
mean	34.572770	21.470329	0.002706	78.431693	2.764454	1.164755	0.002057	0.024860	0.00
std	41.313316	9.150645	0.051945	28.975781	0.599734	0.600960	0.045305	1.876209	0.00
min	13.000000	4.000000	0.000000	0.155574	2.000000	0.000000	0.000000	0.000000	0.00
25%	23.000000	16.000000	0.000000	57.025835	2.000000	1.000000	0.000000	0.000000	0.00
50%	27.000000	20.000000	0.000000	100.000000	3.000000	1.000000	0.000000	0.000000	0.00
75%	34.000000	24.000000	0.000000	100.000000	3.000000	1.000000	0.000000	0.000000	0.00
max	6097.000000	110.000000	1.000000	100.000000	13.000000	10.000000	1.000000	447.000000	0.34e

8 rows x 10 columns

Feature engineering:

To improve the phishing detection model, we used our feature engineering method to develop four more features based on the dataset's current columns:

1. CharContinuationRate:

This tool helps detect suspicious URLs with abnormal patterns by capturing how smoothly or abruptly characters change throughout the URL. The technique computes a ratio that is inserted as a new feature and calculates character transitions in the URL.

2. URLTitleMatchScore:

Since genuine websites frequently have matching titles and URLs, this technique evaluates how similar the webpage title and URL are, which is a crucial sign of phishing. The similarity score is computed by the method using string matching techniques and added to the dataset.

3. URLCharProb:

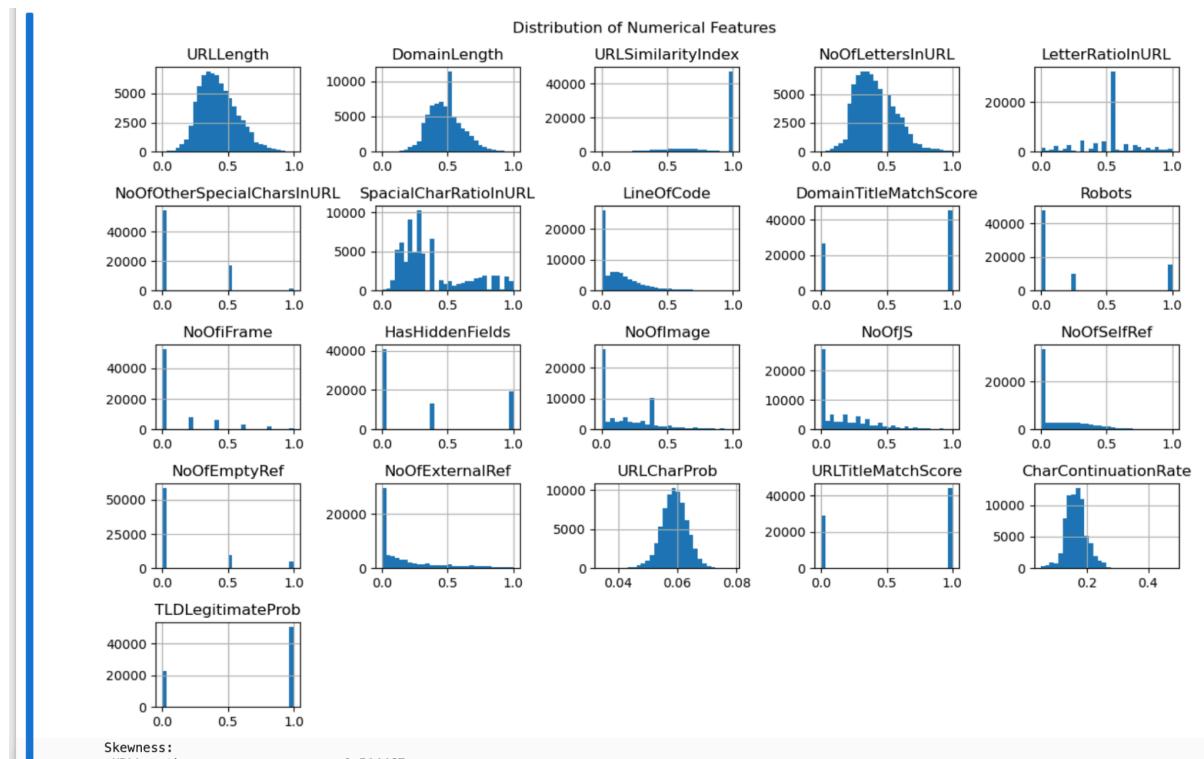
This feature helps in the identification of unusual or suspicious patterns by estimating the likelihood that the URL's character sequences follow a recognized, acceptable distribution. These probabilities are computed by the code and added as a new column to the dataset.

4. **TLDLegitimateProb:**

This feature evaluates the top-level domain (TLD) of the URL by looking at how frequently phishing TLDs occur. The code verifies the TLD of every URL and, using past phishing data, calculates a chance score.

The dataset was enhanced with the addition of the additional characteristics CharContinuationRate, URLTitleMatchScore, URLCharProb, and TLDLegitimateProb for downstream modeling and analytics. These features, which target important components of phishing URLs such character patterns, URL-to-title coherence, and domain authenticity, were developed based on an analysis of existing columns.

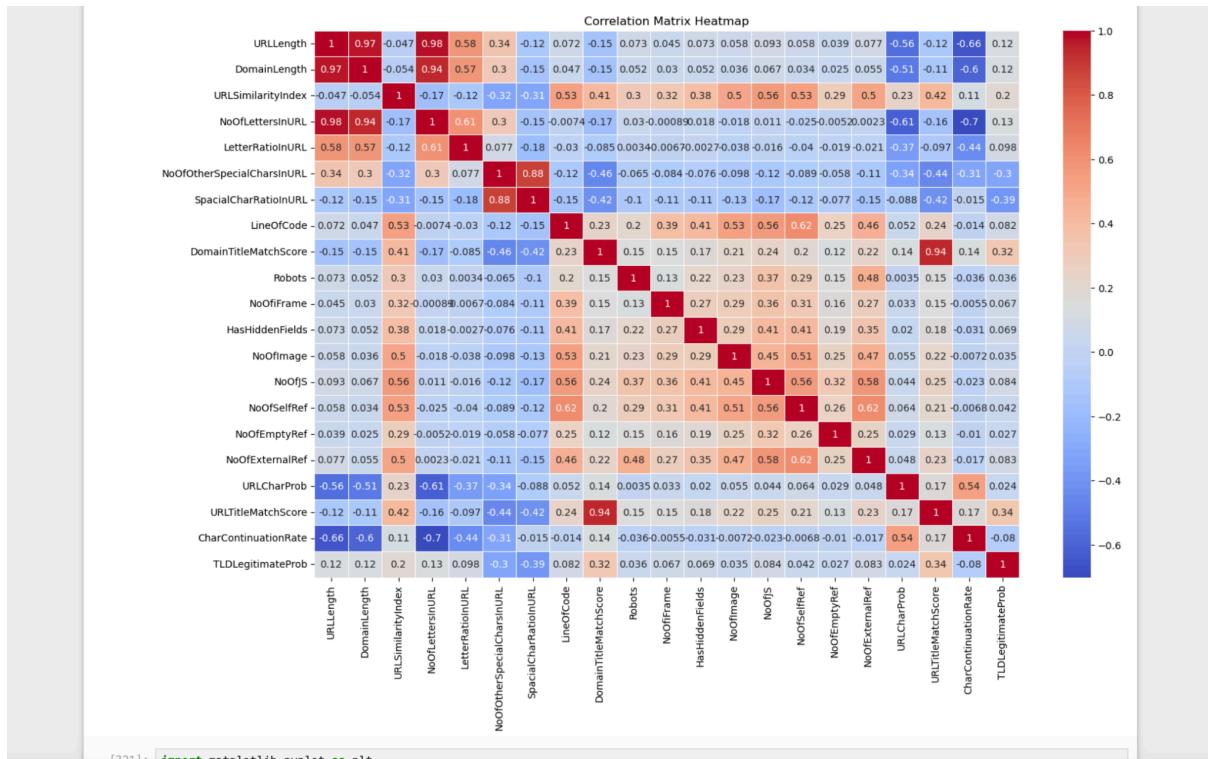
Histograms to check distribution



The dataset's numerical feature distribution is represented visually by the histograms that are displayed. We can evaluate each feature's skewness, dispersion, and central tendencies by using these plots. This is helpful in identifying any anomalies, such as skewed data or outliers, which may need to be removed to scale or normalize the data.

NoOfOtherSpecialCharsInURL and NoOfSelfRef, for instance, exhibit a highly skewed distribution, meaning that the majority of URLs do not have these properties. Additionally biased toward binary values, the characteristic TLDLegitimateProb suggests categorical behavior. The printed skewness numbers confirm the skewness. Once we have that, we can decide if applying adjustments like logarithmic scaling is necessary to enhance the performance of the model.

Create the heatmap

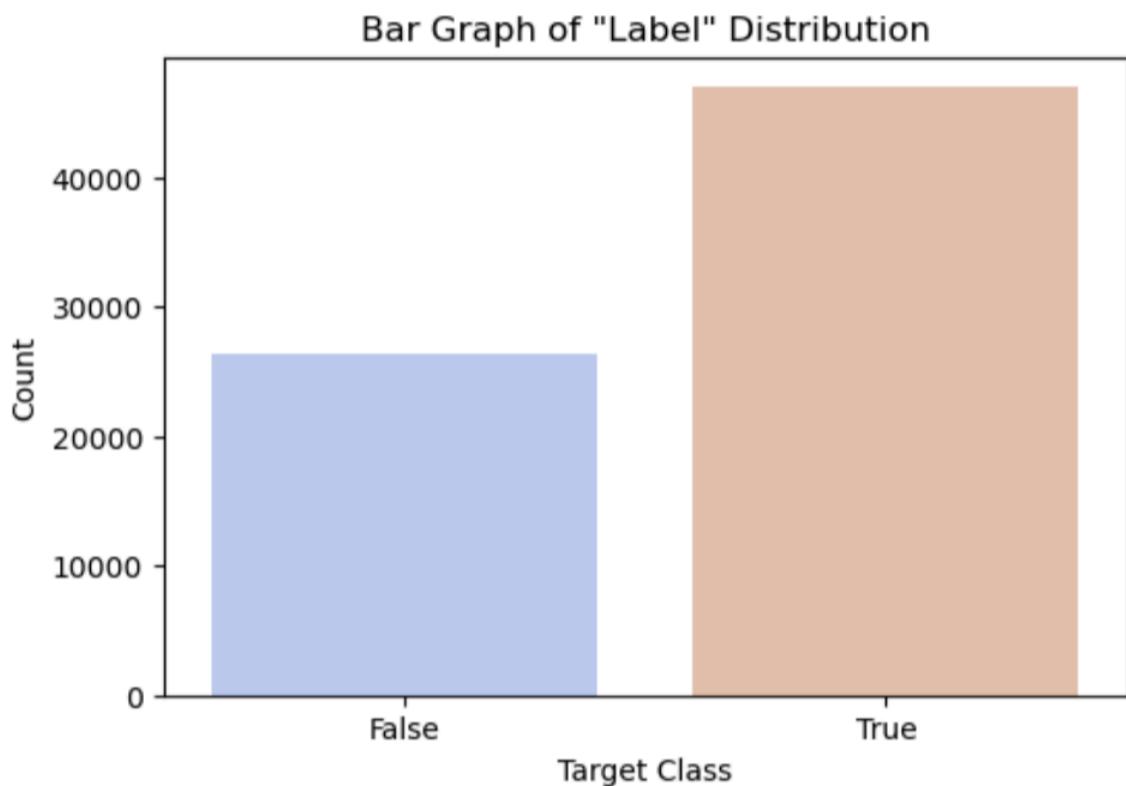


The relationships between the numerical features in the dataset are clearly shown visually by the correlation matrix heatmap. Features with strong positive indicated by red and negative indicated by blue. For example, there

is a high positive correlation of 0.97 between URLLength and DomainLength, showing that they will increase together.

We can reduce dimensionality by removing duplicate features that provide identical information which can be detected by examining this heatmap. Additionally, it helps in identifying multicollinearity that may affect model performance and helps us decide which features to retain or optimize for tasks that come after.

Bar graph for class distribution

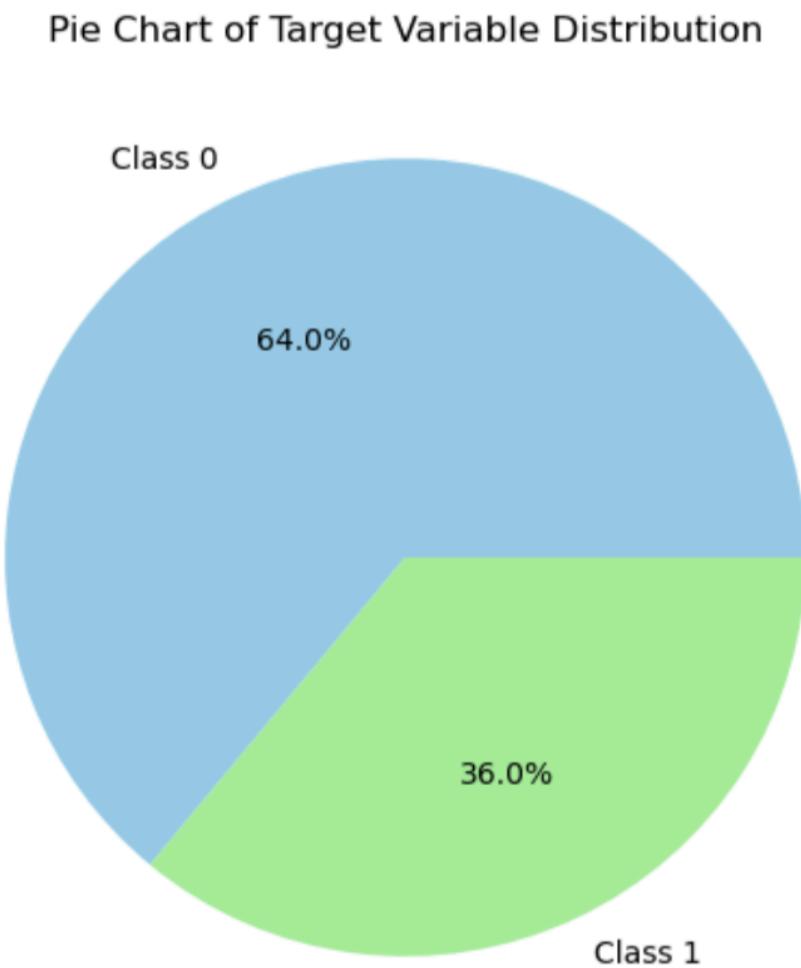


The relationships between the numerical features in the dataset are clearly shown visually by the correlation matrix heatmap. Features with strong positive indicated by red and negative indicated by blue. For example, there is a high positive correlation of 0.97 between URL Length and DomainLength, showing that they will increase together.

We can reduce dimensionality by removing duplicate features that provide identical information which can be detected by examining this heatmap. It

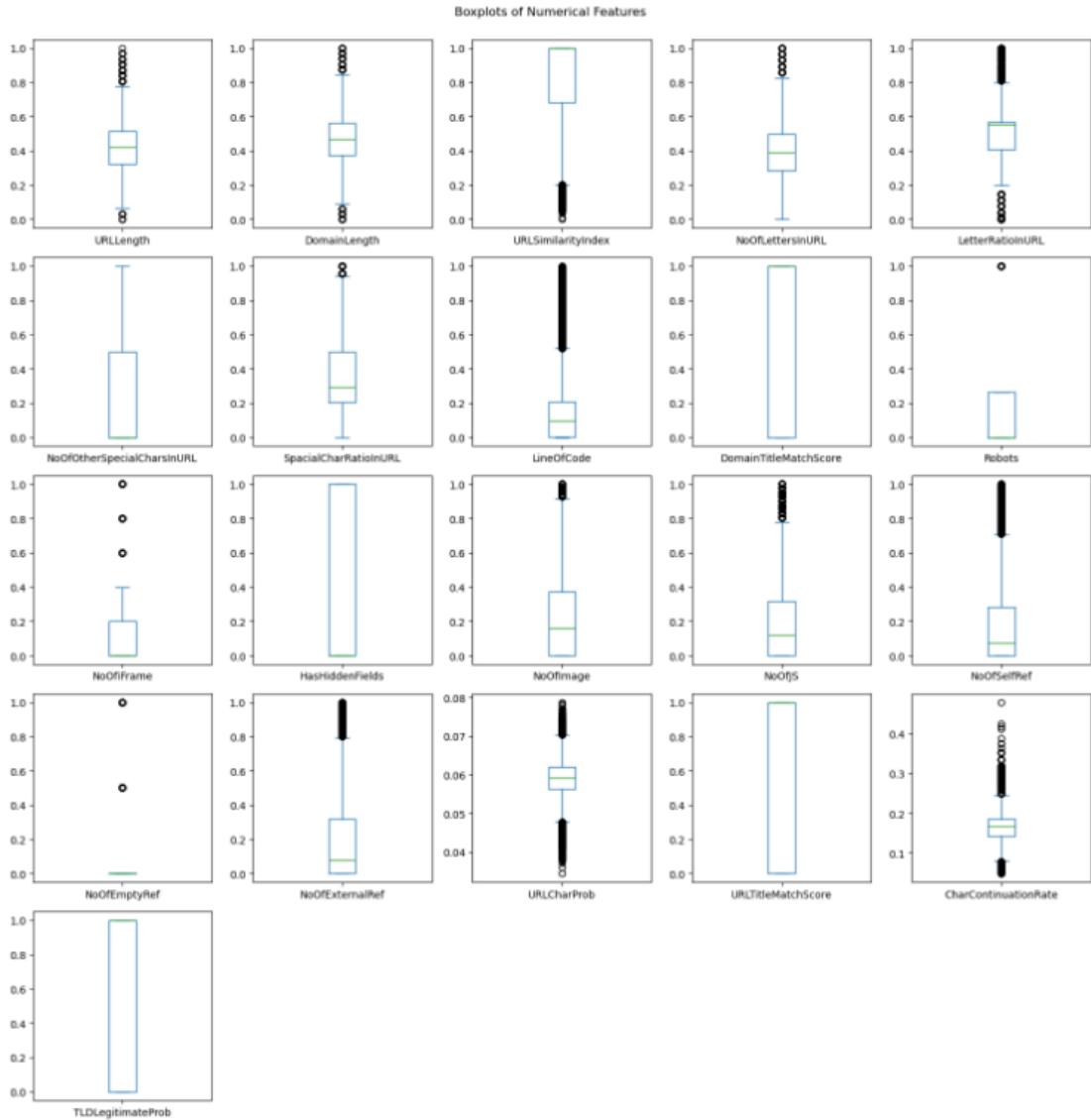
helps in identifying multicollinearity that may affect model performance and helps us decide which features to retain or optimize.

Pie chart for target class distribution



The target class distribution in a dataset is shown in a pie chart where 36% of examples falls into class 1 and 64% into class 0. The class imbalance that might affect model performance is clearly visualized . In order to prevent biases where the model might favor the majority class, a balanced class distribution is helpful. Techniques like resampling or using alternate measures, like F1-score instead of accuracy, can enhance model performance and guarantee greater generalization across both classes.

Histograms to check distribution



The boxplot graph provides a visual representation of the distribution of numerical features in the dataset. This allows the analysis of key statistics such as median, interquartile range and outliers. Finding outliers allows us to evaluate the quality of the data and understand what changes are needed to enhance model performance.

Decisions on feature selection and scale can be informed by the variability revealed by the data spread, which is represented by the size of the boxes. Overall, by identifying areas that might need more research or modification, this

analysis aids in dataset refinement, improves the accuracy of subsequent modeling, and directs additional exploratory study.