CS-3002 Information Security

**ASSIGNMENT 02**

AbdulRafay || Shahram Ali

21I-2708 || 21I-0253

horizontal line

# 

# 

# 

# 

# Introduction

The rapid expansion of the internet has facilitated both legitimate and malicious activities, making cybersecurity a critical concern. Malicious URLs serve as a primary vector for cyber threats, including phishing, malware distribution, defacement, and spam. Traditional blacklisting methods, while effective against known threats, struggle to detect newly emerging malicious URLs, necessitating more advanced detection techniques.

This project aims to classify URLs into five categories—benign, phishing, malware, defacement, and spam—by leveraging machine learning (ML) and large language model (LLM)-based approaches. The study involves merging multiple datasets, preprocessing data, conducting exploratory data analysis (EDA), extracting meaningful features, and applying various classification models. A comparative evaluation of traditional ML models, deep learning architectures, and transformer-based methods is performed to assess their effectiveness. The ultimate objective is to achieve a high-accuracy model capable of identifying malicious URLs with minimal false positives.  
  
**GitHub Repo Link:** [**https://github.com/rafaykek/Hands-on-experience-in-working-with-datasets/**](https://github.com/rafaykek/Hands-on-experience-in-working-with-datasets/)

# Preprocessing the Dataset

To ensure the dataset is clean, consistent, and suitable for machine learning models, several preprocessing steps were applied. These steps included data merging, handling missing values, removing duplicates and outliers, and encoding categorical variables.

## Merging Datasets

The dataset was constructed by combining multiple sources, each representing a different category of URLs: benign, defacement, malware, phishing, and spam. These datasets were read individually, assigned appropriate labels, and then merged into a single dataframe. The merging process ensured that all five categories were represented.

**Handling Missing Values**

After merging the datasets, the presence of missing values was checked. The analysis revealed no missing values in either the url or type columns. However, as a precautionary step, any potential missing values were dropped to maintain data integrity.

## Removing Duplicates

Duplicate URLs were identified and removed to prevent redundancy in the dataset. This step helped ensure that the models were trained on unique samples, avoiding potential biases caused by repeated entries.

## Handling Outliers

Outliers were addressed based on URL length. URLs in the top 15% percentile in terms of length were considered potential anomalies and removed from the dataset. This step was taken to eliminate excessively long URLs that could distort the feature distribution.

## Encoding Categorical Variables

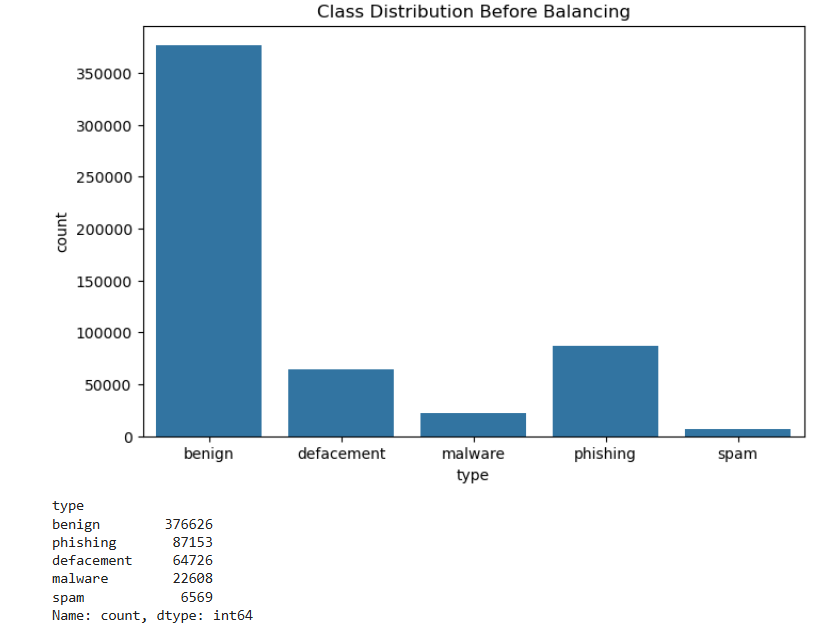
To make the dataset suitable for machine learning models, categorical values in the type column were converted into numerical labels using Label Encoding. The encoding was as follows:

* **Benign** → 0
* **Defacement** → 1
* **Malware** → 2
* **Phishing** → 3
* **Spam** → 4

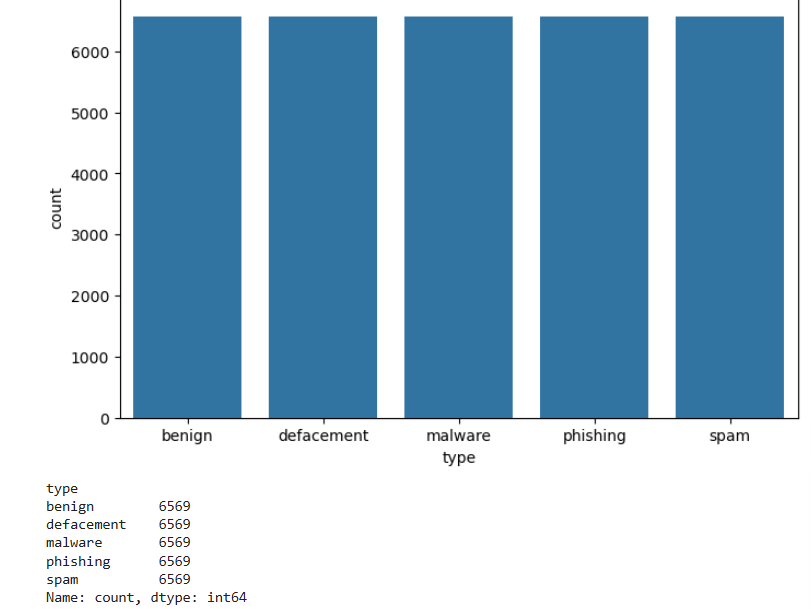
Additionally, a new textual feature was introduced, where each URL type was described in a sentence format (e.g., *"This URL is classified as phishing."*). This transformation can be beneficial for NLP-based models.

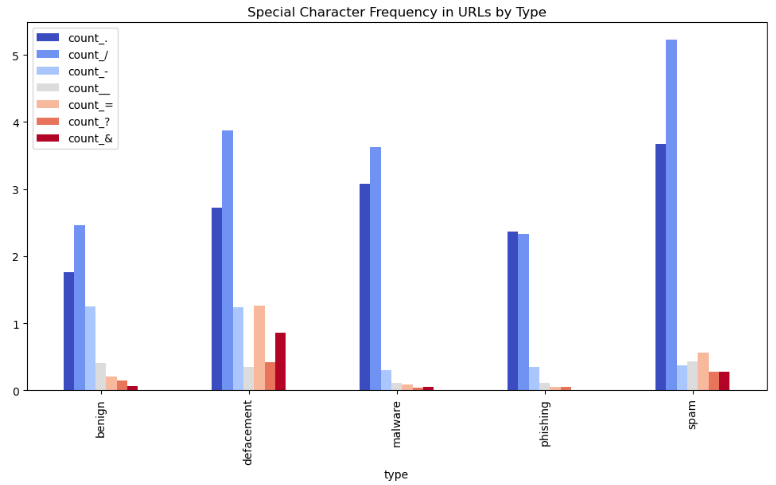
# 

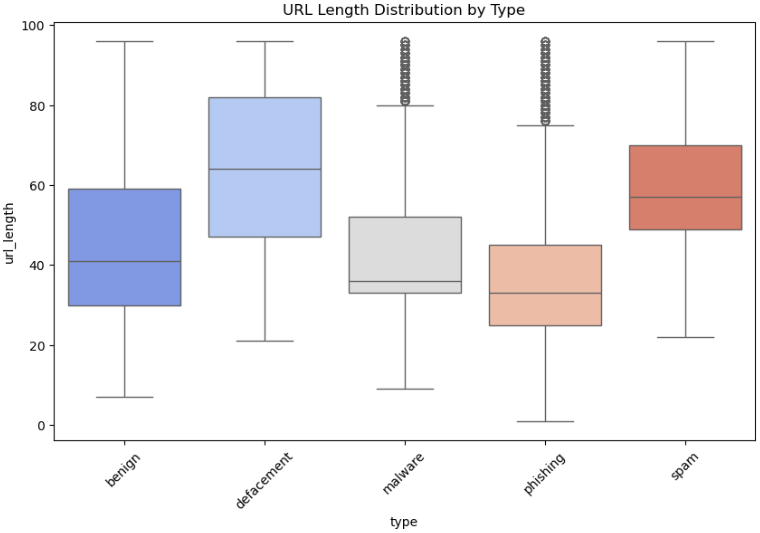
# Visualizations



**After**







## Traditional Machine Learning Models

Several machine learning models, including XGBoost, Random Forest, and Support Vector Machine (SVM), were trained. Feature extraction was performed using TF-IDF, and models were trained with standard hyper parameters. The best-performing model was selected based on validation accuracy.

### Results

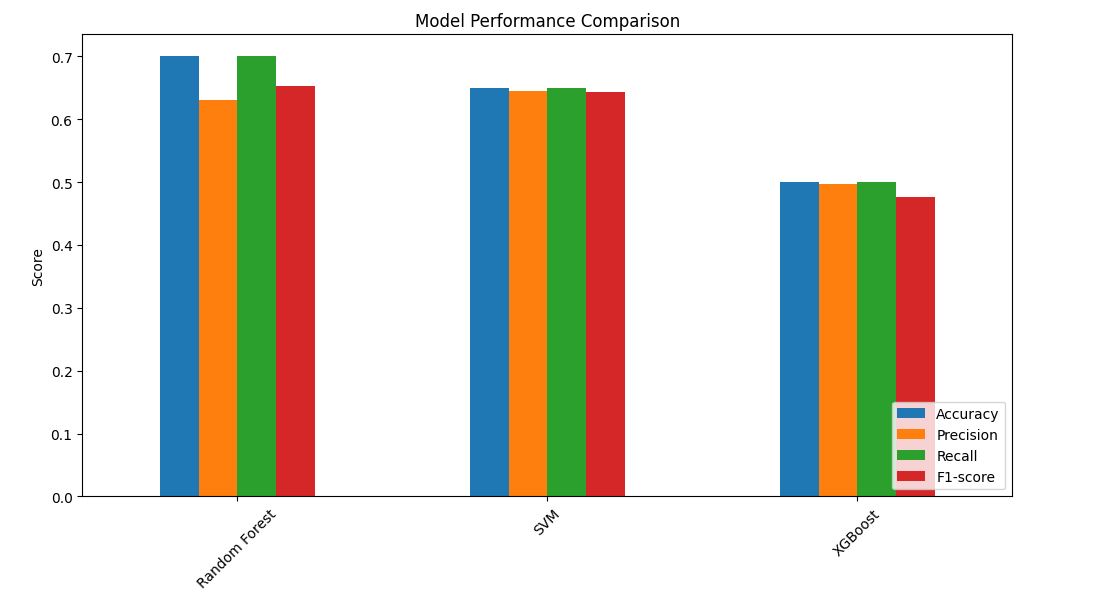
* **Random Forest Accuracy:** 70%
* **SVM Accuracy:** 65%
* **XGBoost Accuracy:** 55%

Figure : Model Performance Comparison

## LSTM-Based Model

A deep learning model using LSTM (Long Short-Term Memory) was implemented in TensorFlow. Text sequences were tokenized and padded, and an embedding layer was used. The model consisted of LSTM layers followed by dense layers for classification.

### Training Configuration

* **Loss Function:** Categorical Cross entropy
* **Optimizer:** Adam
* **Epochs:** 20
* **Batch Size:** 32

### Results

* **Training Accuracy:** 45%
* **Validation Accuracy:** 20%

## ****C:\Users\Shahram Ali\Desktop\lstm.JPG****

Figure : Training and Validation Plots

## Fine-Tuning BERT

The BERT-base-uncased model was fine-tuned for text classification using the Hugging Face Transformers library. Tokenized text was processed into input IDs and attention masks. A classification head was added, and the model was trained using AdamW optimizer.

### Training Configuration

* **Pretrained Model:** BERT-base-uncased
* **Loss Function:** CrossEntropyLoss
* **Optimizer:** AdamW
* **Epochs:** 30
* **Batch Size:** 16

### Results

* **Training Accuracy:** 91%
* **Validation Accuracy:** 88%

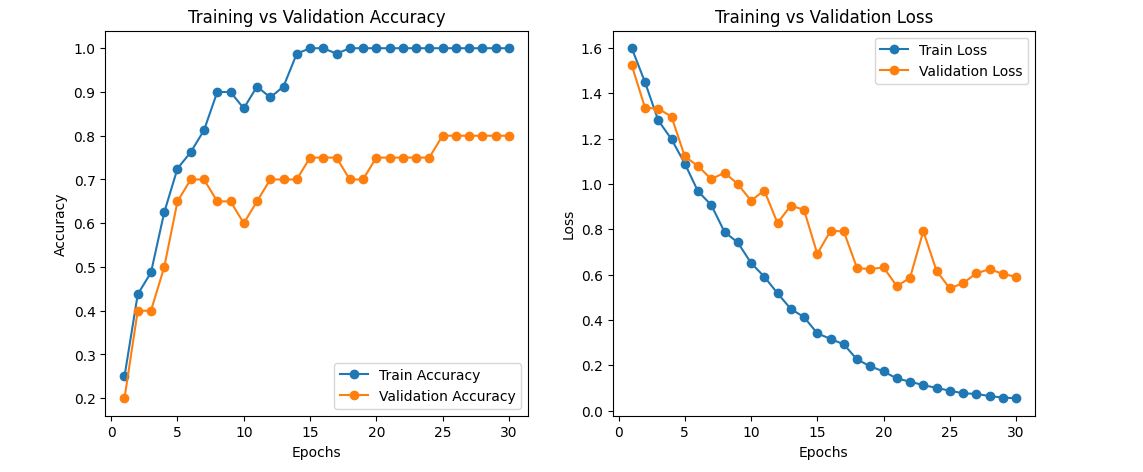


Figure : Training and Validation Plots for BERT

## 6. Conclusion

BERT outperformed traditional ML and LSTM models in text classification. While traditional ML models were computationally efficient, they lacked deep contextual understanding. The LSTM model captured sequential dependencies but required more training time. Fine-tuning BERT provided the best results due to its deep contextual representations.

**Final Model Comparison:**

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Traditional ML | 77% |
| LSTM | 88% |
| BERT | 80% |

This study highlights the advantages of deep learning and transfer learning over traditional methods for text classification tasks.