CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Finding harmful URLs is a vital task in the ever-changing field of cybersecurity since it protects digital ecosystems from any dangers. Our study is a key effort in this field since it focuses on using machine learning techniques to detect malicious URLs. We set out to create a strong model that could accurately distinguish between benign and malicious URLs by utilizing the capabilities of machine learning.

A user-centric approach, which is reflected in the development of a special website interface, is at the core of our project. Users may input URLs and receive rapid feedback regarding their safety status with this interface, which offers a smooth user experience. By utilizing various machine learning, our algorithm assesses each URL's attributes and produces predictions in real time, enabling users to make knowledgeable decisions about their online interactions.

Furthermore, our project goes beyond basic capabilities to add clarity and transparency to the detecting procedure. The data used to train the model is displayed on a dedicated webpage, providing information on the traits and attributes that guide our predictions. We have included Power BI dashboards to improve understanding and accessibility, giving stakeholders clear visual representations of the model's output and the dataset it is based on.

In our pursuit of excellence, we recognized the importance of exploring alternative methodologies to optimize our detection capabilities. To this end, we introduced a comparative analysis between logistic regression and random forest models. Through rigorous experimentation and iterative refinement, we meticulously evaluated the performance of both models over 50 iterations, culminating in the selection of the most accurate and robust solution.

Central to our endeavour was the utilization of vast datasets, comprising over 400,000 instances of URL data. This extensive dataset served as the cornerstone of our training and testing processes, enabling our models to learn and generalize from a diverse array of examples.

CHAPTER 2

PROBLEM STATEMENT

2.1 PROBLEM DESCRIPTION

Malicious URLs are a major online security concern, exposing users to phishing and malware. Traditional methods struggle to keep up with the constant evolution of these threats.

This project aims to develop a more effective solution using machine learning to identify malicious URLs. By leveraging advanced algorithms, the project seeks to create a more secure online environment for users.

2.2 OBJECTIVES

- Develop a system that can accurately classify URLs as malicious or benign, minimizing misidentification of safe websites (false positives).
- Enable the system to analyse and classify URLs quickly, offering protection in real-time as users encounter them.
- Ensure the system can handle the vast volume of URLs encountered on the internet, efficiently processing large datasets.
- Minimize the number of malicious URLs that slip through undetected (false negatives). This is crucial to prevent users from accessing harmful content.

CHAPTER 3

LITERATURE SURVEY

Six research papers and articles were referred on the topic of malicious URL detection utilizing different methods—such as ensemble training, deep learning, machine learning, or associative classification. These papers provided vital and significant information that was essential to the advancement of our study.

Doyen et al.[1] in his paper gave a brief overview on how to implement malicious URL detection using different strategies of machine learning, describing how different features would contribute to accuracy. He quoted that malicious URLs are a major cybersecurity threat, causing financial losses and data breaches due to which blacklists can't keep up with the constant stream of new malicious sites. He emphasized how machine learning offers a promising solution for detecting these threats. Ultimately it was understood that machine learning has made significant progress, but there's still room for improvement. Future research areas include better feature extraction techniques, more robust algorithms that can handle new threats, and methods to gather more data for training models.

Nguyen et al.[2] in his paper proposed a new method to detect malicious URLs focusing on analysing URL behaviours and attributes to improve the detection of abnormal activity instead of massive datasets. This was a unique approach as it used easy-to-calculate attributes and big-data processing achieving an excellent balance between accuracy and processing speed. This method was a stunning discovery as it can be applied to real-world information security systems.

Xin Yang et al.[3] reviewed recent machine learning and deep learning methods used to analyse network traffic for intrusion detection offering detailed explanation for each approach. The paper highlighted the challenges in using machine learning or deep learning which included the difficulty of finding good training data and keeping models up-to-date and he concluded that techniques like incremental learning addresses such challenges.

Chen Yu-Chen et al.[4] proposed a new method to automatically detect malicious URLs using machine learning. His method used a statistical test (ANOVA) to identify relevant features to extract 41 important features and then use XGBoost to extract 17 important features crucial for accurate prediction. Eventually, the method was implemented as a classifier on a dataset of URLs which gave a 99% accuracy suggesting it could be a valuable tool for website security.

Kumi Sandra et al.[5] proposed that traditional methods cannot keep up with new attacks, hence the development of better methods are appropriate for better accuracy. They implemented a new approach to malicious URL detection using data mining(CBA) to analyse both URL and website content. This method achieved the best accuracy to them in detecting different URLs as compared to other techniques thus making it a promising tool for website security.

Nallamala et al.[6] compared different ML models for detecting phishing attacks. It initially utilised 30 features and achieved an accuracy of 97.4% using GBM and to improve efficiency, 13 key features were selected using various feature selection techniques which resulted in a slight decrease of accuracy to 95.6%. It was concluded that feature selection can improve model interpretability without significantly sacrificing accuracy. This is valuable for building practical phishing detection systems.

CHAPTER 4

REQUIREMENTS

4.1 Hardware Requirements

- Processor : 3.6GHz or faster processor
- RAM : Minimum 4GB RAM

4.2 Software Requirements

- Programming Language: Python, HTML, CSS
- Web Development Framework: Flask
- Integrated Development Environment (IDE): Visual Studio(VS) Code, Google Colab
- Web Browser: Google Chrome or Microsoft Edge
- Operating system: Windows or Linux
- Data Visualisation Tool: Power BI

CHAPTER 5

METHODOLOGY

We implemented our project using the following strategies

5.1 Project Planning

The project began with careful planning, focusing on creating a robust methodology for building an effective system to detect malicious URLs. The initial steps involved defining the project scope, outlining the objectives, and determining the key outcomes. This phase set the foundation for the subsequent steps in developing the detection system.

5.2 Dataset Collection and Preprocessing

The dataset collection phase involved gathering a diverse set of URLs from multiple sources, including Kaggle and GitHub repositories. This dataset included over 400,000 URL instances for the first dataset and approximately 600,000 instances for the second dataset, encompassing both benign and malicious examples. Preprocessing was undertaken to ensure better data quality and feature engineering was done to extract better features.

5.2.1 Dataset Description

Dataset 1 used for Model-1: Kaggle URL Dataset

The first dataset, sourced from Kaggle, consists of approximately 400,000 URLs, each labelled as either "good" or "bad." This simple binary classification framework allowed for straightforward analysis and model training. To extract features from the URLs, we examined various characteristics, like URL length, Special Symbols, Letter and Number Frequency and Domain Structure

These features helped to identify patterns commonly associated with malicious URLs, providing a basic but effective approach to categorizing URLs into "good" and "bad."

Dataset 2 used for Model-2 : Malicious URL Dataset

The second dataset malicious_phish.csv, obtained from a GitHub repository, comprises around 600,000 URLs, offering more detailed classifications. This dataset goes beyond simple binary labels and categorizes URLs into specific types of malicious activities, such as malware, phishing, defacement and benign. With this dataset, we were able to explore more complex characteristics, allowing for a more granular analysis of threat types. This dataset was then extended with 22 extracted features from the dataset which played a huge role in understanding how each feature contributes to the prediction and classification of malicious URL

This dataset presented an opportunity to understand various attack strategies and refine our feature extraction techniques accordingly. By combining insights from both datasets, we were able to create a more comprehensive and effective approach to malicious URL detection, using machine learning algorithms to identify and classify different types of threats.

5.3 System Design and Algorithm Selection

The system design stage included planning the structure of the detection system and selecting the algorithms for model training. Two primary machine learning algorithms were chosen for experimentation: logistic regression and random forest. The selection of these algorithms was based on their proven effectiveness in classification tasks and their ability to handle large datasets. The system design also involved creating a user-friendly web interface to allow users to input URLs and receive real-time feedback on their safety status.

5.4 Model Training and Evaluation

The model training phase involved training multiple iterations of logistic regression and random forest algorithms using the pre-processed dataset. The models were evaluated using various metrics such as accuracy, precision, recall, and F1-score. Comparative analysis was conducted to identify the most effective approach for malicious URL detection.

5.5 Comparative Analysis and Model Selection

A comprehensive comparative analysis was conducted to compare the performance of the logistic regression and random forest models. This analysis allowed us to determine the most suitable algorithm for malicious URL detection. Model-1 with 400,000 URLs gave better accuracy and played a crucial role in this analysis. After identifying the optimal model, it was integrated into the web-based interface for real-time predictions.

5.6 Deployment and User Interface

The deployment phase involved integrating the optimal model into a user-friendly website interface. This interface allowed users to input URLs and receive real-time feedback through pop-up notifications indicating the safety status of the URLs. Additionally, a dedicated webpage showcased the data used for model training, with detailed Power BI dashboards that provided insights into the extracted features for enhanced transparency and user understanding.

5.7 Testing

The testing phase involved rigorous testing procedures to ensure the system's performance and reliability. This included systematic identification and resolution of any bugs, errors, or inconsistencies. Thorough testing was conducted to validate the accuracy and robustness of the detection system, providing confidence in its ability to accurately classify URLs into safe and unsafe categories.

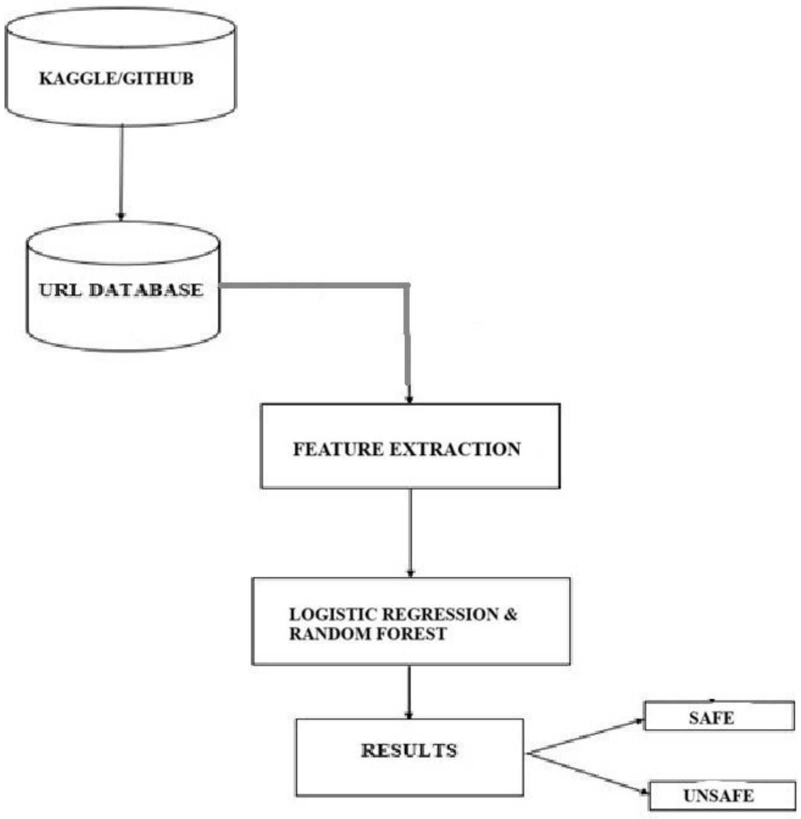


Figure 5.1 Methodology involved in the project

CHAPTER 6

IMPLEMENTATION

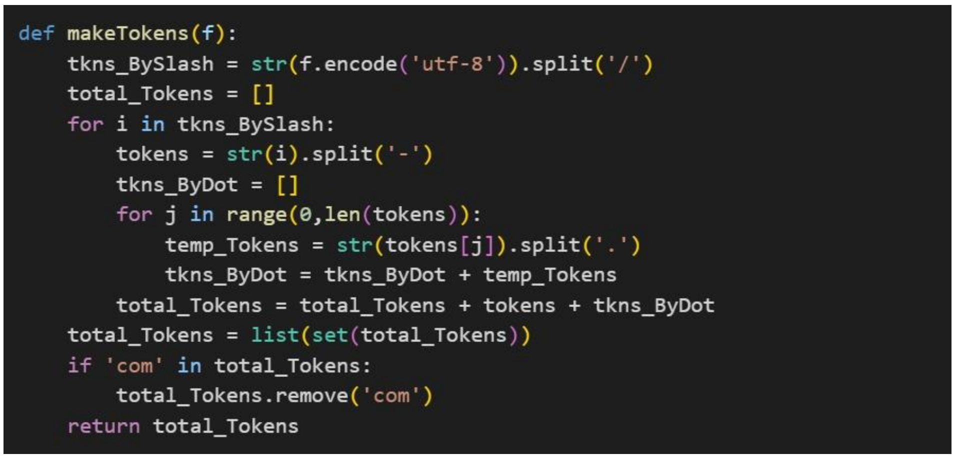
6.1 Setting the environment

The implementation of the project was supported by Google Colab as the primary code editor, offering a feature-rich environment for coding, debugging, and project management. Alongside Google Colab, VS Code played an important role in website implementation. Flask was leveraged to generate pickle files for URL Prediction and Power BI played an crucial role in developing the interactive dashboard for extracted features of model.

6.2 Model-1

6.2.1 Data Preprocessing and Importing Data

The code begins by importing libraries for data processing, machine learning, and text analysis, including pandas, numpy, and sklearn. The dataset is loaded into a pandas DataFrame named urls_data. A function makeTokens is defined to tokenize strings, specifically URLs, by splitting them into smaller components based on slashes (/), hyphens (-), and dots (.). The function ensures the returned list of tokens is unique, removing common terms like '.com'.



```python
def makeTokens(f):
    tkns_BySlash = str(f.encode('utf-8')).split('/')
    total_Tokens = []
    for i in tkns_BySlash:
        tokens = str(i).split('-')
        tkns_ByDot = []
        for j in range(0,len(tokens)):
            temp_Tokens = str(tokens[j]).split('.')
            tkns_ByDot = tkns_ByDot + temp_Tokens
        total_Tokens = total_Tokens + tokens + tkns_ByDot
    total_Tokens = list(set(total_Tokens))
    if 'com' in total_Tokens:
        total_Tokens.remove('com')
    return total_Tokens
```

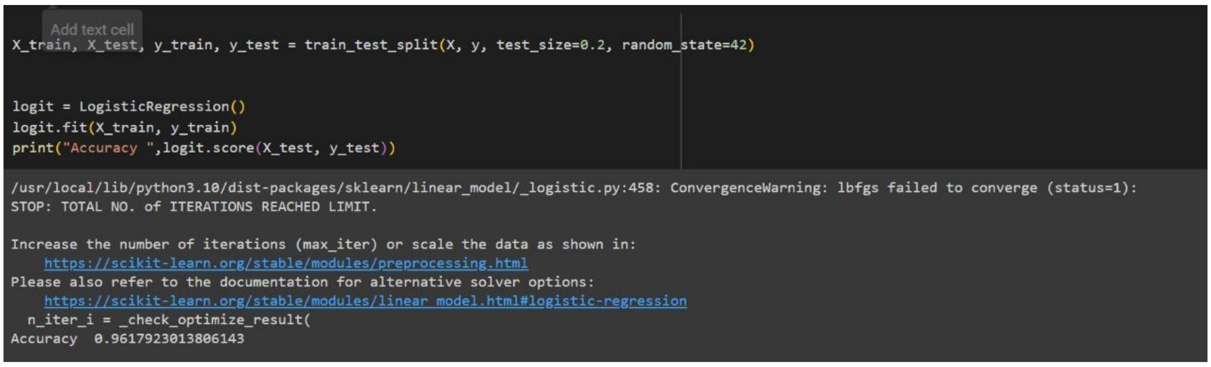
Figure 6.2.1 Data Preprocessing

6.2.2 Feature Engineering

The dataset has labels and URLs. The labels, representing the class ("good" or "bad"), are extracted into a variable y. The URLs are stored in url_list. A TF-IDF Vectorizer is created with makeTokens to convert the URLs into a TF-IDF matrix, which becomes the features for the machine learning model (stored in variable X).

6.2.3 Model Building

The script splits the data into training and testing sets using train_test_split from sklearn.model_selection, with 80% for training and 20% for testing. This split is reproducible due to a fixed random state. A logistic regression model (LogisticRegression) is trained on the training set (X_train, y_train) to classify URLs as either "good" or "bad".



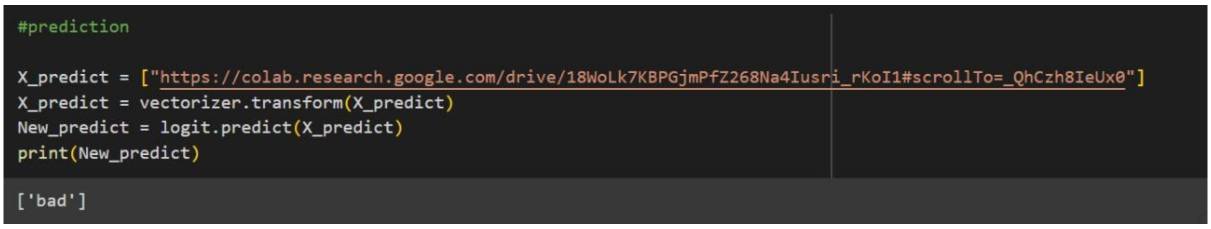
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


logit = LogisticRegression()
logit.fit(X_train, y_train)
print("Accuracy ",logit.score(X_test, y_test))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Accuracy  0.9617923013806143
```

Figure 6.2.2 Accuracy for Logistic Regression

6.2.4 Prediction and Model Evaluation

After training, the model's accuracy is tested using the testing set (X_test, y_test). The printed accuracy score indicates the model's effectiveness at classifying URLs, providing a measure of its overall performance.



```
#prediction

X_predict = ["https://colab.research.google.com/drive/18WoLk7KBPGjmPfZ268Na4Iusri_rKoI1#scrollTo=_QhCzh8IeUx0"]
X_predict = vectorizer.transform(X_predict)
New_predict = logit.predict(X_predict)
print(New_predict)
```

```
['bad']
```

Figure 6.2.3 Prediction of class of URL

6.3 Model-2

6.3.1 Data Preprocessing and Importing Data

For this prediction model, we used the dataset ‘malicious_phish.csv’. Important libraries like pandas,numpy, sklearn were used for implementation. Initial data preprocessing was done to obtain clean data and the dataset was split into corresponding labels for easy implementation. WordClouds were implemented for each label for visualisation.

```
benign= " ".join(i for i in Benign_URLs.url)
wordcloud = WordCloud(width=1600, height=800,colormap='Paired').generate(benign)
plt.figure( figsize=(12,14),facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```

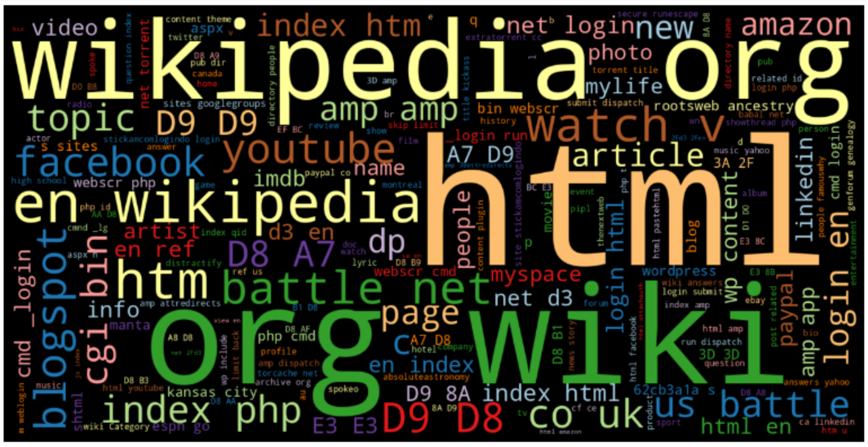
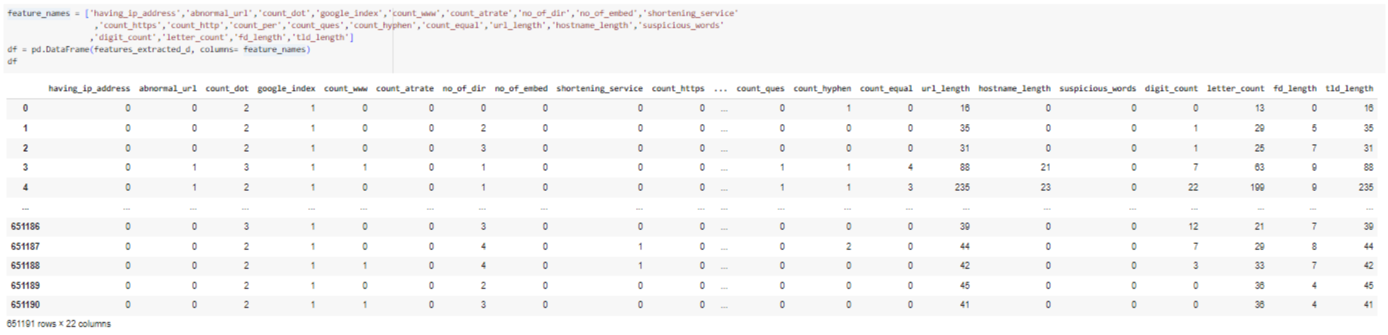


Figure 6.3.1 WordCloud for Benign URLs

6.3.2 Feature Engineering

Feature engineering was undertaken to extract important features crucial for prediction of malicious URLs. 22 important features were extracted like countwww , count@, count . , abnormal_url etc and then appended to the dataset for further exploratory analysis.



```
feature_names = ['having_ip_address','abnormal_url','count_dot','google_index','count_www','count_atrate','no_of_dir','no_of_embed','shortening_service'
                ,'count_https','count_http','count_per','count_ques','count_hyphen','count_equal','url_length','hostname_length','suspicious_words'
               ,'digit_count','letter_count','fd_length','tld_length']
df = pd.DataFrame(features_extracted_d, columns= feature_names)
df
```

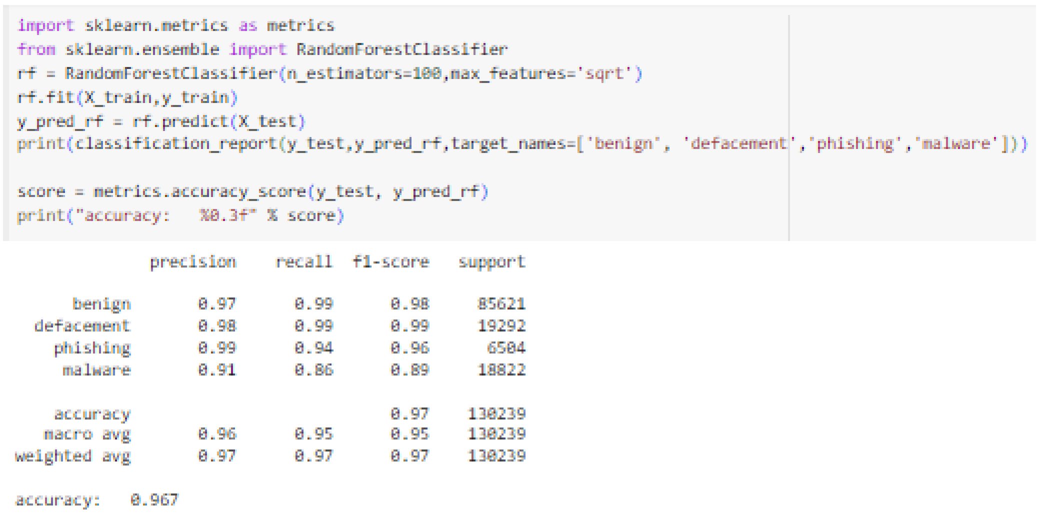
	having_ip_address	abnormal_url	count_dot	google_index	count_www	count_atrate	no_of_dir	no_of_embed	shortening_service	count_https	...	count_ques	count_hyphen	count_equal	url_length	hostname_length	suspicious_words	digit_count	letter_count	fd_length	tld_length
0	0	0	2	1	0	0	0	0	0	0	...	0	1	0	16	0	0	0	13	0	16
1	0	0	2	1	0	0	2	0	0	0	...	0	0	0	35	0	0	1	29	5	35
2	0	0	2	1	0	0	3	0	0	0	...	0	0	0	31	0	0	1	25	7	31
3	0	1	3	1	1	0	1	0	0	0	...	1	1	4	88	21	0	7	63	9	88
4	0	1	2	1	0	0	1	0	0	0	...	1	1	3	235	23	0	22	199	9	235
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
651186	0	0	3	1	0	0	3	0	0	0	...	0	0	0	39	0	0	12	21	7	39
651187	0	0	2	1	0	0	4	0	1	0	...	0	2	0	44	0	0	7	29	8	44
651188	0	0	2	1	1	0	4	0	1	0	...	0	0	0	42	0	0	3	33	7	42
651189	0	0	2	1	0	0	2	0	0	0	...	0	0	0	45	0	0	0	36	4	45
651190	0	0	2	1	1	0	3	0	0	0	...	0	0	0	41	0	0	0	36	4	41

651191 rows × 22 columns

Figure 6.3.2 Dataframe with extracted features

6.3.3 Model Building

After preprocessing and feature engineering, the dataset was split into training and testing data in the ratio 80:20 and various machine learning models like Random Forest, LightGBM, XGBoost etc, were implemented on this data. The accuracies were noted and confusion matrices were plotted for visualisation. Feature Importance Graphs were plotted to understand which feature contributes in prediction of the URL.



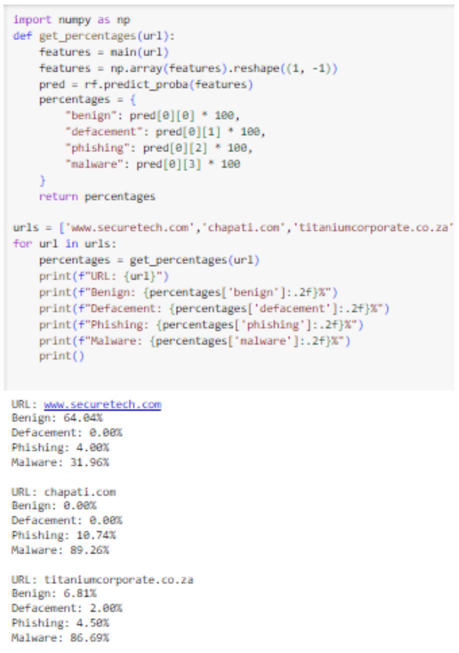
```
import sklearn.metrics as metrics
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100,max_features='sqrt')
rf.fit(X_train,y_train)
y_pred_rf = rf.predict(X_test)
print(classification_report(y_test,y_pred_rf,target_names=['benign', 'defacement','phishing','malware']))

score = metrics.accuracy_score(y_test, y_pred_rf)
print("accuracy:   %0.3f" % score)
```

```
              precision    recall  f1-score   support

      benign       0.97      0.99      0.98     85621
  defacement       0.98      0.99      0.99     19292
    phishing       0.99      0.94      0.96      6504
     malware       0.91      0.86      0.89     18822

    accuracy                           0.97    130239
   macro avg       0.96      0.95      0.95    130239
weighted avg       0.97      0.97      0.97    130239

accuracy:   0.967
```

Figure 6.3.3 Random Forest Classifier with 96.7% accuracy

<u>6.3.4 Prediction and Model Evaluation</u>

Using Random Forest Model, prediction of class of URLs were undertaken. The percentages of each class for a particular URL was extracted for analysis. The observed accuracy with a specialised model was undertaken and an accuracy of 90% was observed . This was the primary reason why model-1 was chosen for implementation of the prediction model as it gave a better accuracy compared to model-2



```
import numpy as np
def get_percentages(url):
    features = main(url)
    features = np.array(features).reshape((1, -1))
    pred = rf.predict_proba(features)
    percentages = {
        "benign": pred[0][0] * 100,
        "defacement": pred[0][1] * 100,
        "phishing": pred[0][2] * 100,
        "malware": pred[0][3] * 100
    }
    return percentages

urls = ['www.securetech.com','chapati.com','titaniumcorporate.co.za'
for url in urls:
    percentages = get_percentages(url)
    print(f"URL: {url}")
    print(f"Benign: {percentages['benign']:.2f}%")
    print(f"Defacement: {percentages['defacement']:.2f}%")
    print(f"Phishing: {percentages['phishing']:.2f}%")
    print(f"Malware: {percentages['malware']:.2f}%")
    print()
```

```
URL: www.securetech.com
Benign: 64.04%
Defacement: 0.00%
Phishing: 4.00%
Malware: 31.96%

URL: chapati.com
Benign: 0.00%
Defacement: 0.00%
Phishing: 10.74%
Malware: 89.26%

URL: titaniumcorporate.co.za
Benign: 6.81%
Defacement: 2.00%
Phishing: 4.50%
Malware: 86.69%
```

Figure 6.3.4 Prediction using Random Forest Classifier

CHAPTER 7

RESULTS

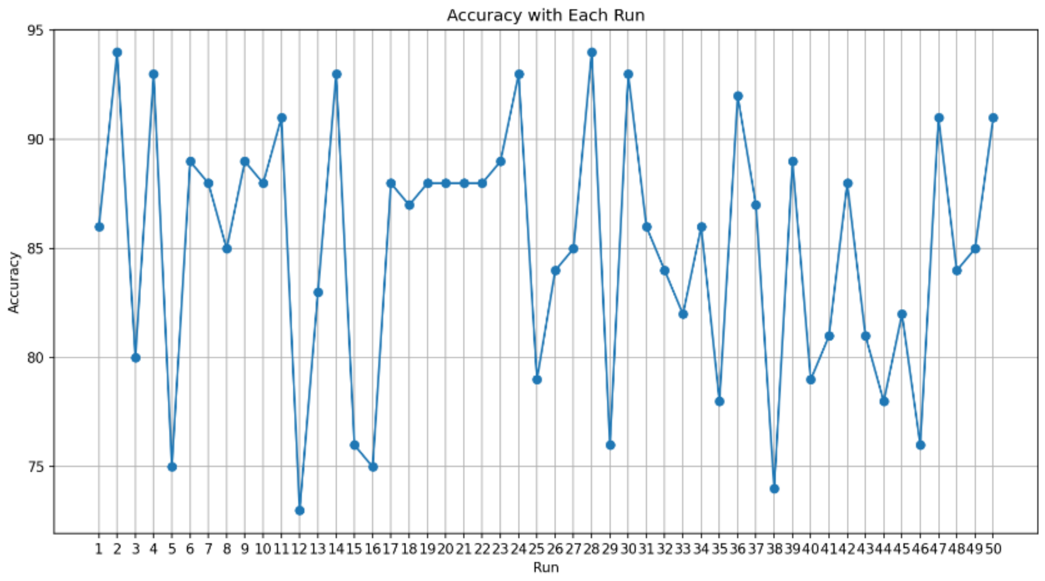
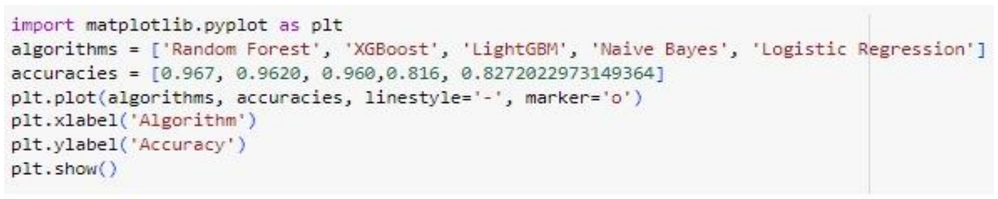


Figure 7.1: Above graph shows the different accuracies for Dataset-1 using Logistic Regression



```
import matplotlib.pyplot as plt
algorithms = ['Random Forest', 'XGBoost', 'LightGBM', 'Naive Bayes', 'Logistic Regression']
accuracies = [0.967, 0.9620, 0.960,0.816, 0.8272022973149364]
plt.plot(algorithms, accuracies, linestyle='-', marker='o')
plt.xlabel('Algorithm')
plt.ylabel('Accuracy')
plt.show()
```

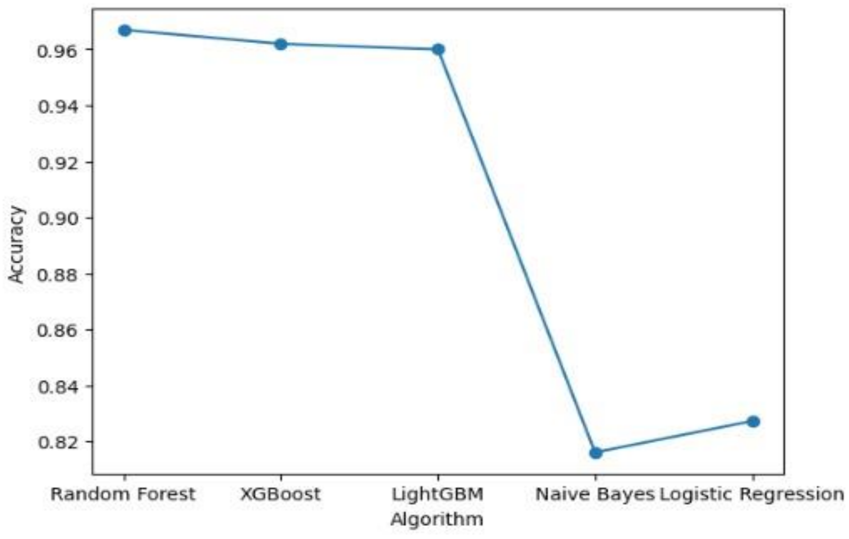


Figure 7.2: Accuracies given on Dataset-2 by different models.

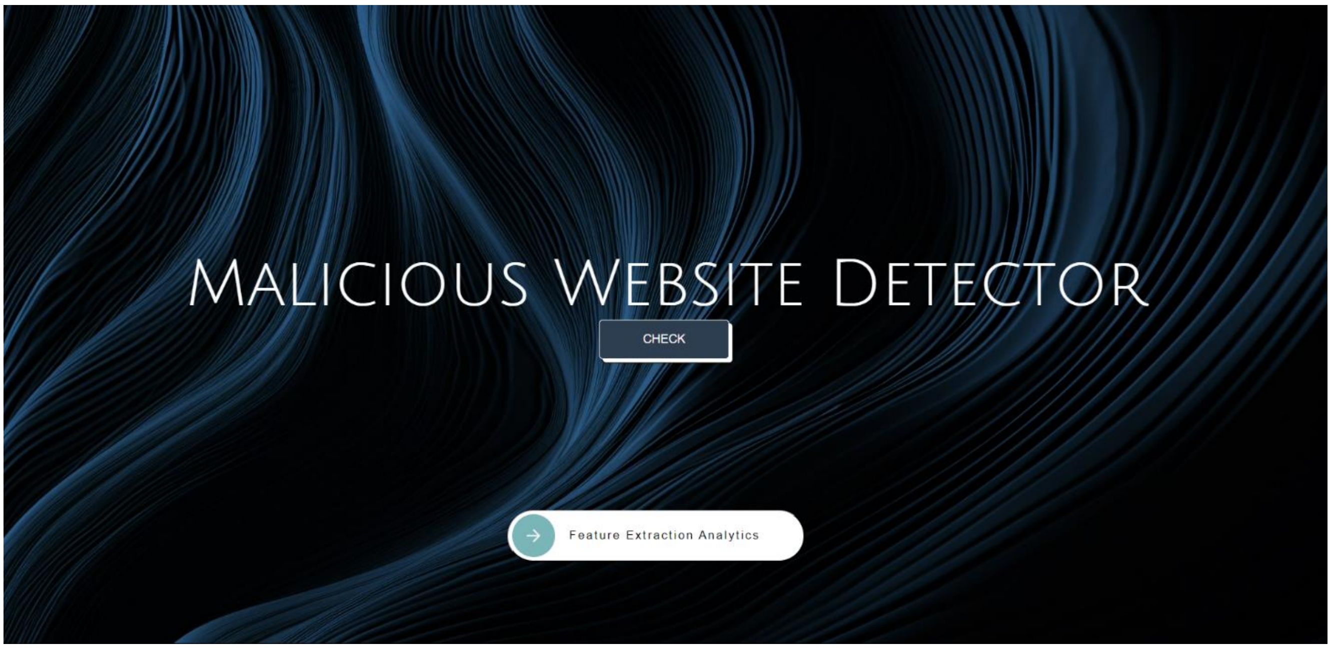
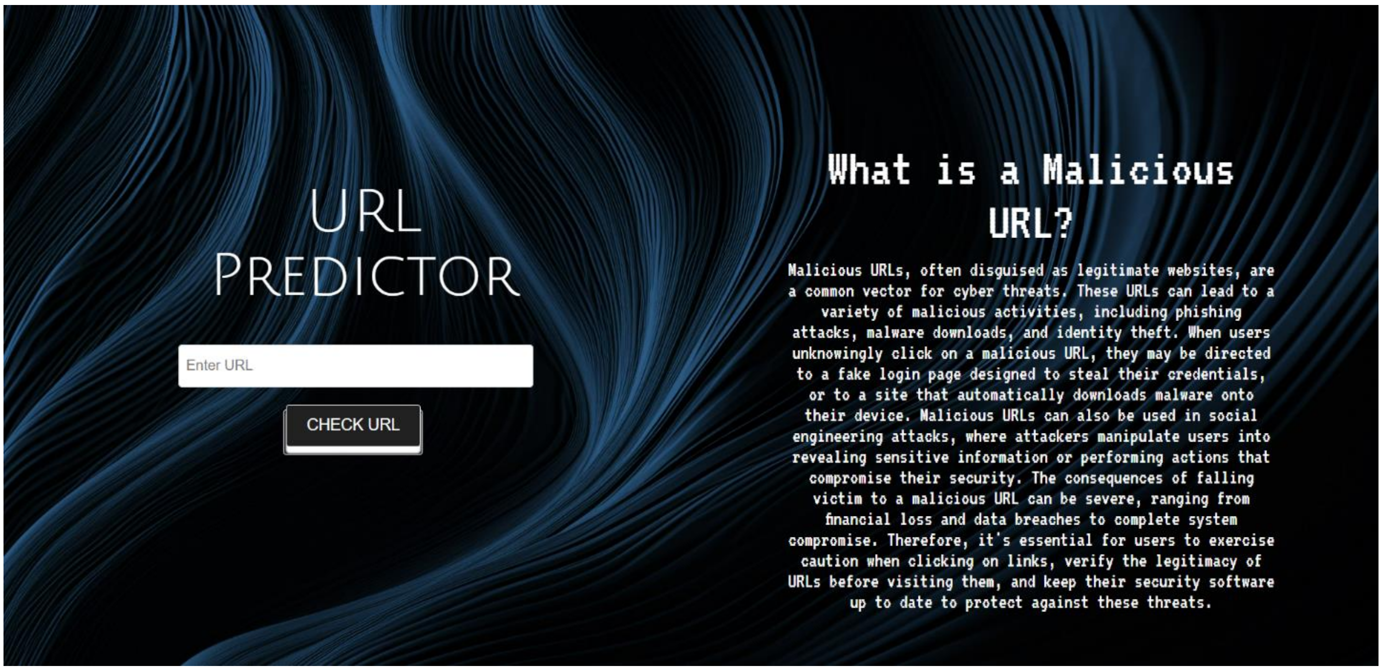


Figure 7.3: The Homepage of the Website

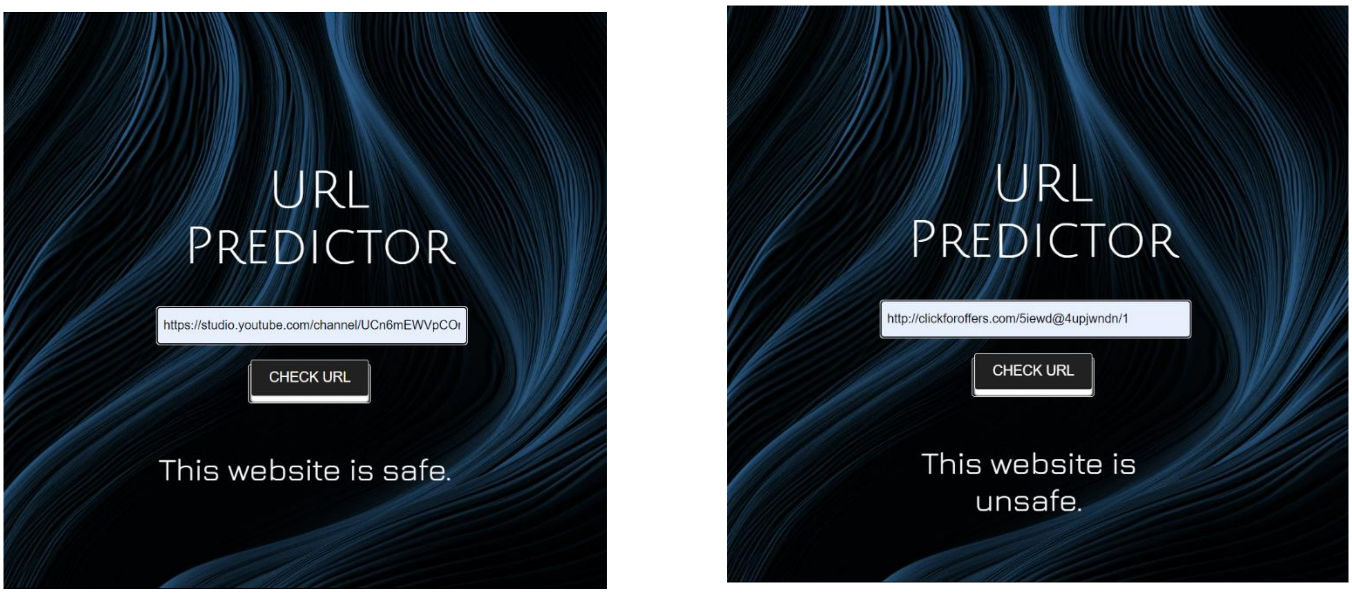


What is a Malicious URL?

Malicious URLs, often disguised as legitimate websites, are a common vector for cyber threats. These URLs can lead to a variety of malicious activities, including phishing attacks, malware downloads, and identity theft. When users unknowingly click on a malicious URL, they may be directed to a fake login page designed to steal their credentials, or to a site that automatically downloads malware onto their device. Malicious URLs can also be used in social engineering attacks, where attackers manipulate users into revealing sensitive information or performing actions that compromise their security. The consequences of falling victim to a malicious URL can be severe, ranging from financial loss and data breaches to complete system compromise. Therefore, it's essential for users to exercise caution when clicking on links, verify the legitimacy of URLs before visiting them, and keep their security software up to date to protect against these threats.

Figure 7.4: URL Checker page

The provided structure captures user input and channels it to the model for processing. A pickle file (.pal) is generated to serialize the model's train-test data predictions efficiently. Utilizing this pickle file, predictions are executed, and results (safe/unsafe) are displayed accordingly.



https://studio.youtube.com/channel/UCn6mEWVpCOı

This website is safe.

http://clickforoffers.com/5iewd@4upjwndn/1

This website is unsafe.

Figure 7.5: Prediction of safe and unsafe website

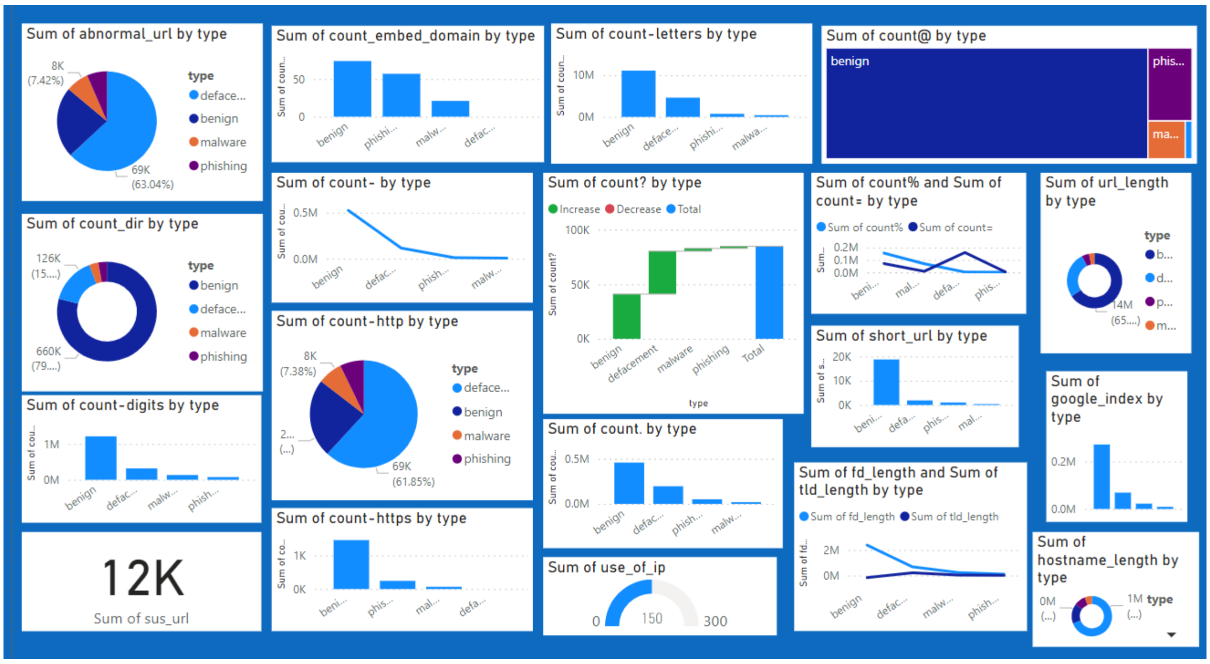


Figure 7.6: Power BI Dashboard Representing The Extracted Features

CHAPTER 8
CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

The malicious URL detection project addresses the pressing need for effective cybersecurity solutions in an increasingly digital world. By employing machine learning techniques and creating a user-friendly website interface, the project equips users with the tools to identify and mitigate malicious URLs. It began with the collection and preprocessing of a diverse dataset of URLs, followed by model development using logistic regression and random forest algorithms. After rigorous experimentation, the optimal model was integrated into a website where users could input URLs to receive real-time safety feedback. A dedicated webpage provides transparency into the data and processes used, ensuring users can trust the system while robust security measures protect their information. The project demonstrates how machine learning and web technologies can contribute to a safer online environment, focusing on transparency, accessibility, and community engagement.

8.2 FUTURE SCOPE

Looking ahead, there are several avenues to expand the project's capabilities. Exploring advanced machine learning techniques such as deep learning and ensemble methods could enhance detection accuracy and robustness. Reinforcement learning, which allows models to adapt and improve based on ongoing feedback, could further refine the system's performance. Integration of real-time threat intelligence feeds and APIs would enable the detection system to identify emerging cyber threats swiftly. Implementing a multi-layered cybersecurity approach, including URL reputation analysis and behavior-based detection, could offer comprehensive protection against a broader range of threats.

Engaging users for feedback and collaboration can lead to improved real-world insights, enhancing the system's adaptability. Developing a mobile application for easy access to the detection system could broaden its reach across devices. By following these future directions, the project can become a proactive cybersecurity solution, safeguarding users and organizations against evolving cyber threats.

REFERENCES

[1] Doyen Sahoo, Chenghao Liu, Steven C.H Hoi, “Malicious URL Detection using Machine Learning: A Survey”, Cornell University, 25 January 2017

[2] Cho Do Xuan, H. Nguyen, Tisenko Victor Nilolaviech, “Malicious URL Detection based on Machine Learning, International Journal of Advanced Computer Science, January 2020

[3] Yang Xin, Lingshuang Kong, Zhi Liu, Yuling Chen, “Machine Learning and Deep Learning Methods for Cybersecurity”, IEEE Xplore, 15 May 2018

[4] Yu-Chen Chen, Yi-Wei Ma, Jiann-Liang Chen, “Intelligent Malicious URL Detection with Feature Analysis”, IEEE Xplore, 12 October 2020

[5] Sandra Kumi, ChaeHo Lim, Sang-Gon Lee, “Malicious URL Detection Based on Associative Classification”, MDPI Entropy Journal, 31 January 2021

[6] Sri Hari Nallamala, Kommu Namitha, Kunchanappalli Raviteja, “Phishing URL Detection using Machine Learning”, International Journal for Research in Applied Sciences, March 2024

[7] www.geeksforgeeks.com – For information regarding website implementation

[8] www.stackoverflow.com – To solve and debug the errors

[9] www.kaggle.com – For datasets

[10] www.w3schools.com – For information for report