


 FYP_2_AfhamIrfanBinAiman_CS01080804.pdf • 20 February 2024
4699 words (32990 characters)

Moderately Human 29%

Most of the text is written by a human. Some portion of the text was created with the help of AI.

AI weightage		Content weightage	Sentences
	Highly AI written	1% Content	2
	Moderately AI written	14% Content	31
	Lowly AI written	14% Content	33

**EXPERIMENTING WITH DIFFERENT ENCODERS AND CLASSIFIERS TO
DETECT MALICIOUS URLS**

REPORT BY

AFHAM IRFAN BIN AIMAN

SUPERVISED BY

DR. AHMED MUBARAK AHMED AL-HAIQI

**FINAL YEAR PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF COMPUTER
SCIENCE (CYBER SECURITY) (HONS)**

**COLLEGE OF COMPUTING & INFORMATICS
UNIVERSITI TENAGA NASIONAL**

2024

DECLARATION

I, Afham Irfan Bin Aiman, hereby declare that this project report entitled "Experimenting with Different Encoders and Classifiers to Detect Malicious URLs", submitted to Universiti Tenaga Nasional as partial fulfilment of the requirements for the degree of Bachelor of Computer Science (Cyber Security) (Hons), has not been submitted to any other university or any other bachelor's programme. I also certify that the work described herein is entirely my own, except for quotations and summaries sources which have been duly acknowledged. This project report may be made available within the university library and may be photocopied or loaned to other libraries for consultation.

Signature: Afham Irfan Bin Aiman

Student ID: CS01080804

Date: Thursday, 1st February 2024

ACKNOWLEDGEMENT

I express my sincere gratitude to my project supervisor, Dr. Ahmed Mubarak Ahmed Al-Haiqi, for his invaluable guidance, support, and encouragement throughout the project. I am also deeply grateful to En. Surizal Bin Nazeri, for his valuable inputs and feedback, which have greatly enhanced the quality of my work. I would like to extend my thanks to my panel, En. Md Nabil Bin Ahmad Zawawi, for his insightful suggestions and constructive feedback.

I would like to thank my parents, Mr. Aiman Bin Hussin and Mrs. Azira Binti Ahmad, for their unwavering support and encouragement throughout my studies. I am also grateful to my friends, who have been a constant source of motivation and support during my academic journey.

Finally, I would like to express my heartfelt gratitude to all those who have contributed to this project in one way or another. Thank you for your support, encouragement, and inspiration.

Signature: Afham Irfan Bin Aiman

Student ID: CS01080804

Date: Thursday, 1st February 2024

ABSTRACT

This project aims to develop a machine learning-based system for detecting malicious Uniform Resource Locators (URLs) that pose a significant threat to internet users. Malicious URLs lead to harmful content such as malware, phishing scams, or fraudulent activities. Traditional rule-based methods for detecting malicious URLs are limited in their ability to detect new and emerging threats. Hence, machine learning-based approaches are being explored, which depend on the quality of the dataset and the selection of appropriate features and classifiers. This project will experiment with different encoders and classifiers to select the best model for detecting malicious URLs. We collected and prepared relevant datasets, perform feature engineering, compare and evaluate different machine learning models, and train and optimize the selected model. The outcomes of this project include a comprehensive dataset of benign and malicious URLs and a reliable machine learning-based model for detecting malicious URLs.

TABLE OF CONTENTS

	Page
DECLARATION	2
ACKNOWLEDGEMENT	3
ABSTRACT	4
TABLE OF CONTENTS	5
LIST OF TABLES	7
LIST OF FIGURES	8
LIST OF ABBREVIATIONS	9
CHAPTER 1: INTRODUCTION	10
1.1 Background and Motivation.....	10
1.2 Problem Statement.....	11
1.3 Objectives.....	11
1.4 Scope.....	12
1.5 Expected Outcomes.....	12
CHAPTER 2: LITERATURE REVIEW	14
2.1 Overview of Malicious URL Detection.....	14
2.2 Traditional Methods for Detecting Malicious URLs.....	15

2.3 Machine Learning-Based Approaches for Malicious URL Detection.....	15
2.4 Evaluation Metrics for Malicious URL Detection Models.....	16
2.5 Summary of Literature Review.....	17
CHAPTER 3: DEVELOPMENT METHODOLOGY.....	19
3.1 System Architecture.....	19
3.2 Summary of Development Methodology.....	20
CHAPTER 4: DESIGN.....	22
4.1 Data Collection and Preparation.....	22
4.2 Feature Engineering.....	23
4.3 Model Selection.....	25
4.4 Model Training.....	27
4.5 Model Evaluation.....	28
4.6 Summary of Design.....	35
CHAPTER 5: OUTCOME.....	37
5.1 Outcome.....	37
REFERENCES.....	38
APPENDICES.....	39

LIST OF TABLES

Table No.	Description	Page
4.3.1	Data Sources.....	23
4.5.1	Evaluation Metrics.....	30
4.5.2	Models' Accuracy.....	31
4.5.3	Models' Precision.....	32
4.5.4	Models' Recall.....	33
4.5.5	Models' F1-Score.....	34

LIST OF FIGURES

Figure No.	Description	Page
3.2.1	Machine Learning Framework.....	21
4.1.1	Data Preparation and Collection Code.....	23
4.2.1	Feature Engineering Code.....	25
4.3.1	Model Selection Code.....	27
4.4.1	Model Training Code.....	28
4.5.1	Model Evaluation Code.....	29

LIST OF ABBREVIATIONS

Abbr.	Full Form	Page
Hons	Honours.....	1
URL	Uniform Resource Locators.....	4
TF-IDF	Term Frequency-Inverse Document Frequency.....	10
SVM	Support Vector Machine.....	11
XGBoost	eXtreme Gradient Boosting.....	11
ASCII	American Standard Code for Information Interchange.....	12
TP	True Positive.....	16
TN	True Negative.....	16
FP	False Positive.....	16
FN	False Negative.....	16

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

The internet has become an indispensable part of our daily lives, providing us with access to a wealth of information and resources. However, the convenience and accessibility of the internet also present an ever-increasing risk of cybercrime. One of the most common and dangerous forms of cybercrime is the use of malicious Uniform Resource Locators (URLs).

Malicious URLs are links that lead to harmful content, such as phishing scams, malware, or fraudulent activities. As the number of malicious URLs continues to grow, the need for reliable and effective methods for detecting them becomes increasingly important.

Traditional methods for detecting malicious URLs rely on rule-based systems that use a set of predefined rules to identify suspicious URLs. However, these methods are often limited in their ability to detect new and emerging threats. Machine learning-based approaches have shown promising results in detecting malicious URLs by using algorithms that can learn from large datasets of labelled examples. However, the performance of these models depends on the quality of the dataset and the selection of appropriate features and classifiers.

In this project, we aim to experiment with different encoders and classifiers to develop a model for detecting malicious URLs. Our goal is to evaluate the performance of each model and select the best one. We will collect relevant datasets containing URLs labelled as either benign or malicious, perform feature engineering using one-hot encoding, bag-of-words, and Term Frequency-Inverse Document Frequency (TF-IDF), and select classifiers that are suitable for the problem, such as decision trees, random forests, and logistic regression. We will then train and evaluate the models using various metrics, including accuracy, precision, recall, and F1-score.

The development of an accurate and reliable model for detecting malicious URLs can contribute significantly to the ongoing effort to combat cybercrime and protect internet users. Our project will provide a valuable insight for organizations and individuals to identify and avoid harmful content, ultimately leading to a safer online environment for all.

1.2 Problem Statement

The problem statement for this project is to develop a reliable and effective model for detecting malicious Uniform Resource Locators (URLs) using machine learning-based approaches. The traditional rule-based systems for detecting malicious URLs have limitations in detecting new and emerging threats. Hence, machine learning-based approaches are being explored. The performance of these models depends on the quality of the dataset and the selection of appropriate features and classifiers. Therefore, this project aims to experiment with different encoders and classifiers to select the best model for detecting malicious URLs. This project will collect relevant datasets containing URLs labelled as either benign or malicious, and prepare the data by removing irrelevant columns and converting all data to numerical features. The developed models will be evaluated based on appropriate metrics to ensure its effectiveness in detecting malicious URLs.

1.3 Objectives

The main objective of this project is to develop a comprehensive machine learning-based system for detecting malicious URLs using various features extracted from the URL structure, content, and context. The specific objectives are:

1. To collect and prepare a diverse and balanced dataset of benign and malicious URLs from multiple sources, including blacklists, whitelists, and web crawlers.
2. To perform feature engineering using techniques such as count vectorizer and TF-IDF vectorizer to extract relevant features from the URL structure, content, and context that can help distinguish between benign and malicious URLs.
3. To compare and evaluate the performance of different machine learning models such as decision trees, logistic regression, random forest, support vector machine, XGBoost, and simple neural network in detecting malicious URLs using the extracted features.
4. To train, evaluate, and optimize the selected machine learning model using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.

By accomplishing these objectives, this project aims to contribute to the development of more effective and efficient methods for detecting malicious URLs and improving internet security for users.

1.4 Scope

The scope of this project is to develop and evaluate machine learning models for detecting malicious URLs using different encoders and classifiers. The project involves the following key aspects:

1. Data collection and preparation: Collect relevant datasets containing URLs labelled as either benign or malicious from various sources, remove irrelevant columns, and convert the data into numerical features.
2. Feature engineering: Using one-hot encoding, bag-of-words, and TF-IDF to encode features, as well as optional ASCII conversion. The data will be split into training and testing sets.
3. Model selection: Choosing suitable classifiers for the problem, such as logistic regression, random forest, support vector machine, and XGBoost, and optional classifiers like simple neural network.
4. Model training: Training the selected classifiers on the training set and evaluating their performance on the testing set.
5. Model evaluation: Measuring model performance using metrics such as accuracy, precision, recall, and F1-score, and visualizing the results using graphs.

The project does not include data analysis, feature selection, or model explanation.

1.5 Expected Outcomes

Expected outcomes of this project:

1. A comprehensive dataset of both benign and malicious URLs collected from various sources, including blacklists, whitelists, and web crawlers.
2. Identification of key features that can help distinguish between benign and malicious URLs, including those extracted from URL structure, content, and context.

3. Evaluation of various machine learning models such as decision trees, logistic regression, random forest, support vector machine, XGBoost, and simple neural network to determine the most effective approach for detecting malicious URLs.
4. Development of a highly accurate and optimized machine learning model capable of efficiently detecting malicious URLs using various features extracted from URL structure, content, and context.

The expected outcomes of this project will contribute to advancement of knowledge and understanding in the field of malicious URL detection using machine learning techniques.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview of Malicious URL Detection

According to [1], malicious URL detection is a critical task in cybersecurity due to the increasing prevalence and sophistication of malicious URLs. Traditional methods such as rule-based systems have limitations in detecting new and emerging threats, whereas machine learning-based approaches have shown promise in improving the detection of malicious URLs.

However, machine learning-based approaches also face challenges such as feature selection, classifier selection, and evaluation metrics. Feature engineering plays a crucial role in determining the effectiveness of machine learning models for malicious URL detection. Various techniques such as one-hot encoding, bag-of-words, and TF-IDF [2] have been explored to improve feature selection.

Classifier selection is another challenge in malicious URL detection. Decision trees, random forests, and logistic regression are common classifiers used for this task [3]. However, each classifier has its strengths and weaknesses, and the choice of the classifier will depend on the specific problem and dataset.

Evaluation metrics such as accuracy, precision, recall, and F1-score are used to measure the performance of machine learning models for malicious URL detection. These metrics are important in determining the effectiveness of the model in detecting malicious URLs.

In addition to these challenges, machine learning-based approaches for malicious URL detection also face challenges such as data quality, data imbalance, data privacy, model selection, and model evaluation [4]. Solutions such as data augmentation, oversampling [5], and privacy-preserving techniques have been explored to address these challenges.

Overall, machine learning-based approaches have shown promise in improving the detection of malicious URLs. However, further research is needed to address the challenges and limitations associated with these approaches. The development of accurate and reliable models for malicious URL detection is critical to protecting internet users from cybercrime [6].

2.2 Traditional Methods for Detecting Malicious URLs

Traditional methods for detecting malicious URLs rely on rule-based systems that use a set of predefined rules to identify suspicious URLs [7]. These methods include signature-based detection and heuristic-based detection.

Signature-based detection involves creating a database of known malicious URLs and comparing new URLs against this database. URLs that match any of the predefined signatures are considered malicious. While this method is effective in detecting known threats, it has limited ability to detect new and emerging threats [1].

Heuristic-based detection, on the other hand, uses a set of predefined rules or heuristics to identify malicious URLs based on their characteristics, such as the domain name, IP address, length, and structure. URLs are analysed against these rules, and if they meet a certain threshold of suspicious behaviour, they are classified as malicious. This method is more flexible than signature-based detection and can detect new and emerging threats, but it is also prone to false positives and false negatives [7].

Other traditional methods for detecting malicious URLs include blacklist-based and whitelist-based approaches. Blacklist-based approaches maintain a database of known malicious URLs and block access to those URLs. Whitelist-based approaches only allow access to URLs that are explicitly trusted. However, both of these methods have limitations and are not as effective as heuristic-based detection in detecting new and emerging threats.

Overall, detecting and blocking malicious URLs is an important task for ensuring the security and privacy of Internet users. While traditional methods have their limitations, they are still an important part of a comprehensive approach to cybersecurity. As threats continue to evolve, new and more advanced methods for detecting malicious URLs will need to be developed.

2.3 Machine Learning-Based Approaches for Malicious URL Detection

Malicious URLs pose a significant cybersecurity threat, and machine learning-based approaches have shown promising results in detecting them. These approaches can adapt to new patterns and features in the data and can detect new and emerging threats effectively [8].

Supervised learning algorithms such as support vector machines and neural networks require labelled data to train the model, while unsupervised learning algorithms such as clustering and

anomaly detection do not require labelled data [9]. Semi-supervised learning combines both approaches to address the issue of limited labelled data.

The process of machine learning-based malicious URL detection involves data collection and preparation, feature engineering, and model selection and evaluation [1]. The dataset should be large, representative, cleaned, balanced, and split into training, validation, and testing sets [10]. Relevant features from the URLs are extracted to capture their malicious-ness or benign-ness. Feature selection techniques can be used to reduce the dimensionality and redundancy of the feature space.

An appropriate machine learning algorithm is selected to train a classifier that can distinguish between malicious and benign URLs [8], and the classifier is evaluated using various metrics such as accuracy, precision, recall, F1-score, etc. The classifier's generalization performance is also tested on unseen data. Machine learning-based approaches for malicious URL detection can overcome the limitations of traditional methods, detect unknown or novel malicious URLs, adapt to changing attack strategies, and reduce false positives and false negatives [10].

2.4 Evaluation Metrics for Malicious URL Detection Models

To evaluate the performance of malicious URL detection models, various evaluation metrics have been proposed [3]. These metrics can provide more nuanced and comprehensive insights into the model's performance than accuracy alone [11]. Here are some alternative formulations for the section on evaluation metrics:

Malicious URL detection models can be evaluated using a variety of metrics, each of which measures a different aspect of the model's performance. In addition to accuracy, commonly used metrics include precision, recall, and F1-score [3].

Accuracy measures the proportion of correctly classified URLs out of all URLs. It can be calculated as:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

where TP is the number of true positives (malicious URLs correctly classified as malicious), TN is the number of true negatives (benign URLs correctly classified as benign), FP is the number of false positives (benign URLs incorrectly classified as malicious), and FN is the number of false negatives (malicious URLs incorrectly classified as benign).

While accuracy is a commonly used metric for evaluating machine learning models, it has limitations in the context of imbalanced datasets, where the number of positive and negative examples is significantly different. In such cases, a model that simply predicts the majority class (e.g., all URLs as benign) may achieve high accuracy but have poor performance in detecting the minority class (e.g., malicious URLs). Therefore, it's important to consider other metrics such as precision, recall, and F1-score in addition to accuracy when evaluating malicious URL detection models.

Precision measures the proportion of malicious URLs correctly classified as malicious out of all URLs classified as malicious. It can be calculated as:

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Recall measures the proportion of malicious URLs correctly identified out of all actual malicious URLs. It can be calculated as:

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

F1-score is a weighted average of precision and recall, and it provides a single measure of the model's overall performance. It can be calculated as:

$$\text{F1-score} = 2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall})$$

It's important to note that no single metric can capture all aspects of a model's performance. Different metrics may be more appropriate depending on the specific context and goals of the model. For example, if the cost of missing malicious URLs is high, the recall may be a more important metric than precision. Conversely, if the cost of false positives is high, precision may be prioritized over recall. Overall, evaluating a model using a combination of metrics can provide a more comprehensive understanding of its strengths and weaknesses [3], [12].

2.5 Summary of Literature Review

The literature review presents a comprehensive overview of the challenges and opportunities in malicious URL detection. Traditional methods such as rule-based systems have limitations in detecting new and emerging threats, whereas machine learning-based approaches have

shown promise in improving the detection of malicious URLs. However, machine learning-based approaches face challenges such as feature selection, classifier selection, and evaluation metrics. The chapter also discusses traditional methods for detecting malicious URLs, including signature-based, heuristic-based, blacklist-based, and whitelist-based methods [10]. Machine learning-based approaches, such as supervised and unsupervised learning, require data preparation, feature engineering, and model selection and evaluation. Evaluation metrics such as accuracy, precision, recall, and F1-score are used to measure the performance of machine learning models [3]. Solutions such as data augmentation, oversampling, and privacy-preserving techniques have been explored to address data quality, data imbalance, data privacy, model selection, and model evaluation challenges. Overall, further research is needed to develop accurate and reliable models for malicious URL detection to protect internet users from cybercrime.

CHAPTER 3

DEVELOPMENT METHODOLOGY

3.1 System Architecture

The system architecture defines the structure, behaviour, and interactions of the system components and sub-systems, which are essential for the successful development of the model.

The system architecture consists of five main components that perform specific functions in the development process of the machine learning model. These components are:

1. Data Collection and Preparation

The first component of the system architecture involves the collection and preparation of datasets containing labelled URLs for training and testing purposes. The datasets will be obtained from various sources such as online repositories, web crawlers, security reports, etc. The data will be pre-processed to remove duplicates, invalid URLs, and other irrelevant information that may affect the performance of the machine learning model.

2. Feature Engineering

The feature engineering component involves extracting relevant features from the pre-processed data that can capture the characteristics and potential maliciousness of URLs. Feature engineering techniques such as feature selection, feature extraction, and feature transformation will be used to select the most informative features that can accurately detect malicious URLs. The features will then be transformed into numerical vectors using different encoding techniques such as one-hot encoding, bag-of-words, and TF-IDF, etc.

3. Model Selection

The model selection component involves selecting appropriate encoders and classifiers to develop the best-performing machine learning model for detecting malicious URLs. Different combinations of encoders and classifiers will be evaluated using various metrics such as accuracy, precision, recall, and F1-score. The best-performing combination will be selected based on its performance on the validation set.

4. Model Training

The model training component involves training the selected machine learning model using the pre-processed data and the selected features. The training process will involve fine-tuning the model parameters to optimize its performance. The model will be trained using 80% of the data as the training set.

5. Model Evaluation

The model evaluation component involves evaluating the trained machine learning model's performance using different evaluation metrics. The performance of the model will be compared with other existing models to assess its effectiveness. The evaluation will be done using a test set consists of 20% of the original data.

3.2 Summary of Development Methodology

In this section, we provide a summary of the development methodology used for the machine learning model in this project. The methodology consists of five main components that are essential for the successful development of the model. These components include data collection and preparation, feature engineering, model selection, model training, and model evaluation. Each of these components plays a critical role in the development process, and they are explained in detail in the following sections.

The figure below illustrates the system architecture for this project:

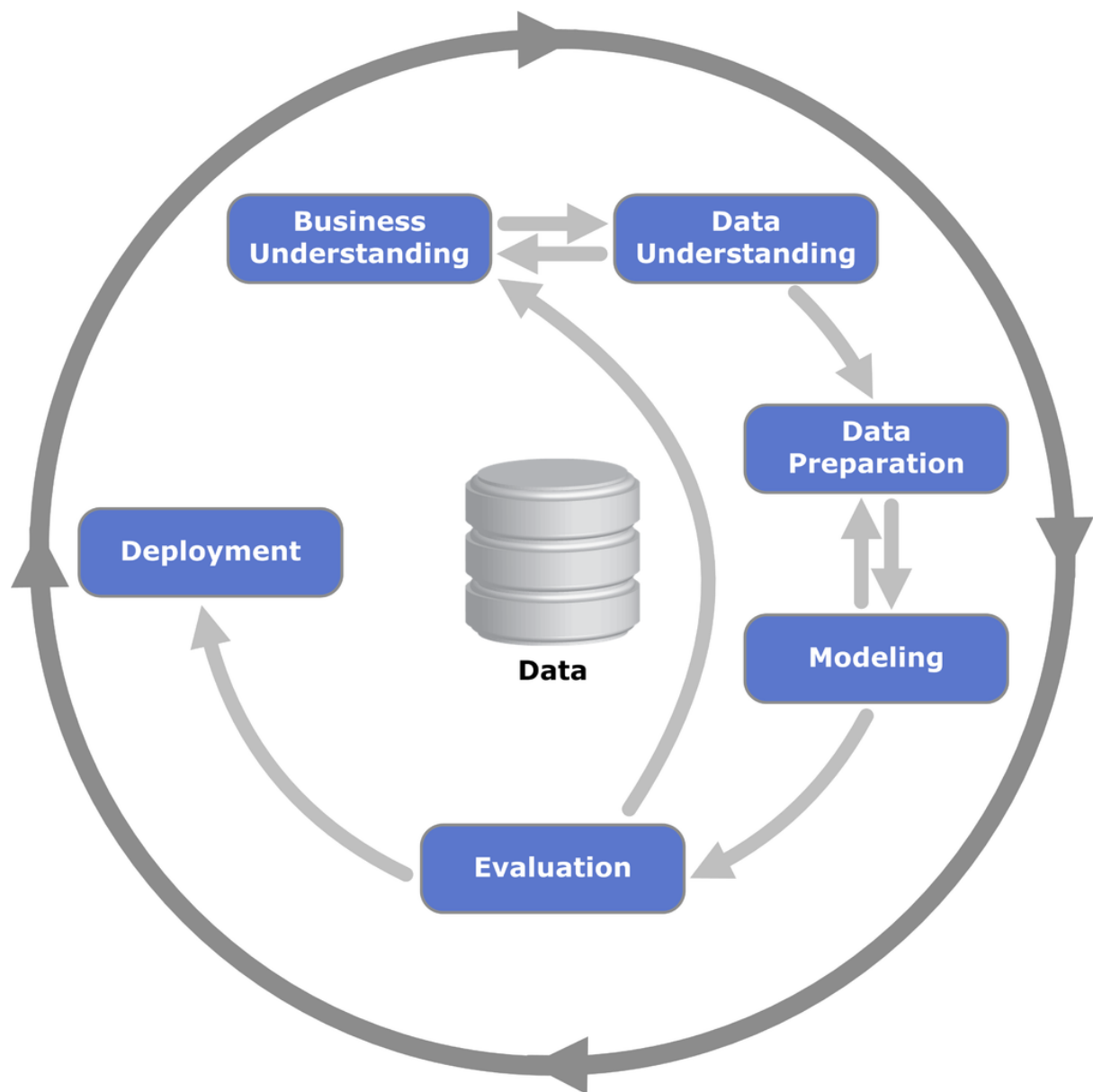


Figure 3.2.1: Machine Learning Framework (Source: Wikimedia Commons)

CHAPTER 4

DESIGN

4.1 Data Collection and Preparation

To gather diverse and reliable data, we drew upon two resources:

- [Manu Siddhartha's Malicious URLs dataset](#) (See Appendix A): This extensive dataset encompasses a wide array of malicious URLs, including benign, defacement, phishing, and malware types. It provided 651,191 URLs for our analysis.
- [Grega Vrbancic's Phishing-Dataset](#) (See Appendix B): Specific to phishing URLs, this dataset offered an additional 88,647 instances for our model's training.

Data preparation involved the following steps:

Manu Siddhartha's dataset:

- Consolidating classes: We combined similar classes into two primary categories: benign and malicious, simplifying the classification task.
- URL segmentation: To extract meaningful features, we decomposed URLs into their fundamental components: protocol, domain, path, query, and fragment.

Grega Vrbancic's dataset:

- No adjustments required: The dataset's existing structure aligned perfectly with our project requirements, negating the need for further preprocessing.

```

# For traditional machine learning and neural network:
import pandas as pd
import numpy as np

# Load the dataset
url_dataset = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Dataset 2
v0/phishing-dataset-variation.csv")
url_dataset

# Replace empty cells with whitespace
url_dataset = url_dataset.fillna("")

# Extract URLs and labels from the dataset
last_column_index = -1 # Assuming the last column is the target feature
all_urls = url_dataset.iloc[:, :-1].apply(lambda row:
''.join(row.dropna().astype(str)), axis=1)
labels = np.array(url_dataset.iloc[:, last_column_index]).reshape(-1, 1)

```

Figure 4.1.1: Data Collection and Preparation Code

Dataset	Description	# of URLs
Manu Siddhartha's Malicious URLs dataset	A list of general malicious URLs	651,191
Grega Vrbancic's Phishing-Dataset	A list of phishing URLs	88,647

Table 4.1.1: Data Sources

4.2 Feature Engineering

We then selected two to four different encoders to transform the features in the datasets into relevant numerical representation so to use them as inputs for the models. These encoders were:

1. Count Vectorizer:

A count vectorizer is a method used to convert a collection of text documents into a matrix of token counts. It counts the frequency of each word in the document. Each column represents a unique word in the corpus, and each row represents a different document.

2. TF-IDF Vectorizer:

TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word.

3. Tokenizer:

In the context of natural language processing, a tokenizer is used to break a stream of text into words, phrases, symbols, or other meaningful elements, which are known as tokens.

4. Pad Sequences:

In the context of sequence data, such as text or time series, padding is the process of adding dummy tokens to sequences to make them equal in length. This is often done to facilitate the processing of batches of sequences in machine learning models.

Data splitting: Both datasets were meticulously divided into training and testing sets, with 80% of the data allocated for model training and 20% reserved for rigorous performance evaluation.

```

# For traditional machine learning:
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

# Create a CountVectorizer for one-hot encoding
vectorizer = CountVectorizer(binary=True)
X = vectorizer.fit_transform(all_urls)

# Split the dataset into a training set and a testing set
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2,
random_state=42)

# For neural network:
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Tokenize and pad sequences
tokenizer = Tokenizer()
tokenizer.fit_on_texts(all_urls)
sequences = tokenizer.texts_to_sequences(all_urls)
padded_sequences = pad_sequences(sequences)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(padded_sequences, labels,
test_size=0.2, random_state=42)

```

Figure 4.2.1: Feature Engineering Code

4.3 Model Selection

Model selection is a crucial step in building a machine learning model for malicious URL detection. The selected algorithm should be able to accurately classify URLs as either malicious or benign while being efficient and scalable. We used several algorithms for this task, including decision trees, logistic regression, random forests, support vector machine, XGBoost, and simple neural networks.

Decision trees are commonly employed for malicious URL detection due to their interpretability and simplicity. They work by recursively partitioning the dataset based on features, creating a tree-like structure. Decision trees are intuitive, easy to understand, and can handle non-linear relationships in data. However, they may struggle with overfitting and may not capture complex patterns as effectively as ensemble methods.

Logistic regression is a widely used algorithm for binary classification tasks like malicious URL detection. It models the probability of a URL being malicious using the logistic function. Logistic regression is computationally efficient, interpretable, and less prone to overfitting compared to more complex models. It works well when the relationship between features and the target variable is approximately linear.

Random forests are a popular algorithm for malicious URL detection because they can improve the accuracy of the classification model by combining multiple decision trees. They are fast, accurate, and robust to noise and outliers, making them well-suited for large and complex datasets.

Support Vector Machines (SVM) are powerful algorithms suitable for both linear and non-linear classification. SVM aims to find a hyperplane that best separates malicious and benign URLs in the feature space. SVMs perform well in high-dimensional spaces and are effective when dealing with complex decision boundaries. However, they may be computationally intensive and sensitive to the choice of hyperparameters.

XGBoost, an implementation of gradient-boosted decision trees, is renowned for its efficiency and accuracy. It sequentially builds a series of decision trees, refining the model's predictive performance. XGBoost is robust, handles missing data well, and often outperforms other algorithms in terms of predictive accuracy. Its ensemble nature helps mitigate overfitting.

Simple neural networks, specifically feedforward neural networks with a few hidden layers, are another option. Neural networks can capture intricate patterns and relationships in data but may require substantial computational resources and data to train effectively. For smaller datasets or when interpretability is crucial, simpler models might be preferred.

```

# For traditional machine learning:
from sklearn.tree import DecisionTreeClassifier

# Train a DecisionTreeClassifier
clf = DecisionTreeClassifier()

# For neural network:
import tensorflow as tf
from tensorflow.keras import layers, models

# Build the neural network model
model = models.Sequential()
model.add(layers.Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=16,
input_length=padded_sequences.shape[1]))
model.add(layers.Flatten())
model.add(layers.Dense(8, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

optimizer = tf.compat.v1.train.AdamOptimizer()

```

Figure 4.3.1: Model Selection Code

Each algorithm has its strengths and weaknesses, and the choice depends on factors such as dataset size, complexity, interpretability requirements, and computational resources. In our case, considering the characteristics of malicious URL detection, we chose random forests due to their ability to handle noise, scalability, and high accuracy, especially when combining multiple decision trees.

Other machine learning algorithms can also be used for malicious URL detection, and the choice of algorithm will depend on the specific requirements of the application. Ultimately, the selected algorithm should provide the highest accuracy and performance while minimizing the risk of false positives and false negatives. We evaluated these algorithms on our dataset using various metrics such as accuracy, precision, recall, and F1-score to make an informed decision.

4.4 Model Training

Model training is a crucial step in machine learning that involves teaching the algorithm to recognize patterns and improve its predictive ability. In this project, we used a variety of encoders and classifiers to train our model.

```

# For traditional machine learning:
from sklearn.tree import DecisionTreeClassifier

# Train a DecisionTreeClassifier
clf.fit(X_train, y_train.ravel()) # Use ravel() to convert y_train to a 1D array

# For neural network:
from tensorflow.keras import layers, models

# Compile the model
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])

# Ensure both padded_sequences and labels are NumPy arrays
model.fit(X_train, y_train)

```

Figure 4.4.1: Model Training Code

After training the models, we compared their performance on both the training and testing sets using the evaluation metrics mentioned above.

4.5 Model Evaluation

Dataset and Model Codes:

- [Access Manu Siddhartha's Dataset and Codes](#)
- [Access Grega Vrbancic's Dataset and Codes](#)

To determine the most effective model for detecting malicious URLs, we must thoroughly evaluate each model's performance using the test set, which comprises 20% of the dataset that was not used for training.

We used several evaluation metrics, including accuracy, precision, recall, and F1-score, to compare the models. Accuracy assesses how well the model classifies all instances correctly, precision measures the ability of the model to identify malicious URLs among all predicted positives, recall measures how well the model captures all actual malicious URLs in the test set, and F1-score is a weighted average of precision and recall that balances both metrics.

```

# For traditional machine learning:
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Predict labels for the test set
y_pred = clf.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average="weighted")
recall = recall_score(y_test, y_pred, average="weighted")
f1 = f1_score(y_test, y_pred, average="weighted")

# Display evaluation metrics
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-Score: {f1}")

# For neural network:
from tensorflow.keras import layers, models
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Predict labels for the test set
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5).astype(int)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Display evaluation metrics
print(f"\nTest Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-Score: {f1}")

```

Figure 4.5.1: Model Evaluation Code

Metric	Definition	Formula
Accuracy	The proportion of correctly classified URLs out of all URLs	$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$
Precision	The proportion of malicious URLs correctly classified as malicious out of all URLs classified as malicious	$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$
Recall	The proportion of malicious URLs correctly identified out of all actual malicious URLs	$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$
F1-score	The harmonic mean of precision and recall	$\text{F1-score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$

Table 4.5.1: Evaluation Metrics

By using the test set, which was not seen by the models during training, we can determine how well the models generalize to new, unseen data. We used the evaluation metrics to identify the best-performing model that can accurately and reliably detect malicious URLs.

Dataset	Encoder	Classifier	Accuracy (%)
Manu Siddhartha's Malicious URLs dataset	Count Vectorizer	Decision Tree	96.32
		Logistics Regression	95.78
		Random Forest	N/A
		Support Vector Machine	N/A
		XGBoost	94.22
	TF-IDF Vectorizer	Decision Tree	96.15
		Logistics Regression	95.13
		Random Forest	N/A
		Support Vector Machine	N/A
		XGBoost	94.59
	Tokenizer and Pad Sequences	Simple Neural Network	100.00
Grega Vrbancic's Phishing-Dataset	Count Vectorizer	Decision Tree	87.16
		Logistics Regression	92.71
		Random Forest	90.60
		Support Vector Machine	93.52
		XGBoost	90.73
	TF-IDF Vectorizer	Decision Tree	85.40
		Logistics Regression	91.62
		Random Forest	90.87
		Support Vector Machine	93.27
		XGBoost	91.12
	Tokenizer and Pad Sequences	Simple Neural Network	95.54

Table 4.5.2: Models' Accuracy

Dataset	Encoder	Classifier	Precision (%)
Manu Siddhartha's Malicious URLs dataset	Count Vectorizer	Decision Tree	96.35
		Logistics Regression	95.82
		Random Forest	N/A
		Support Vector Machine	N/A
		XGBoost	94.31
	TF-IDF Vectorizer	Decision Tree	96.17
		Logistics Regression	95.22
		Random Forest	N/A
		Support Vector Machine	N/A
		XGBoost	94.72
	Tokenizer and Pad Sequences	Simple Neural Network	N/A
Grega Vrbancic's Phishing-Dataset	Count Vectorizer	Decision Tree	87.16
		Logistics Regression	92.67
		Random Forest	90.86
		Support Vector Machine	93.51
		XGBoost	90.70
	TF-IDF Vectorizer	Decision Tree	85.38
		Logistics Regression	91.58
		Random Forest	91.02
		Support Vector Machine	93.24
		XGBoost	91.08
	Tokenizer and Pad Sequences	Simple Neural Network	91.95

Table 4.5.3: Models' Precision

Dataset	Encoder	Classifier	Recall (%)
Manu Siddhartha's Malicious URLs dataset	Count Vectorizer	Decision Tree	96.32
		Logistics Regression	95.78
		Random Forest	N/A
		Support Vector Machine	N/A
		XGBoost	94.22
	TF-IDF Vectorizer	Decision Tree	96.15
		Logistics Regression	95.14
		Random Forest	N/A
		Support Vector Machine	N/A
		XGBoost	94.59
	Tokenizer and Pad Sequences	Simple Neural Network	N/A
Grega Vrbancic's Phishing-Dataset	Count Vectorizer	Decision Tree	87.16
		Logistics Regression	92.71
		Random Forest	90.60
		Support Vector Machine	93.52
		XGBoost	90.73
	TF-IDF Vectorizer	Decision Tree	85.40
		Logistics Regression	91.62
		Random Forest	90.87
		Support Vector Machine	93.27
		XGBoost	91.12
	Tokenizer and Pad Sequences	Simple Neural Network	95.42

Table 4.5.4: Models' Recall

Dataset	Encoder	Classifier	F1-Score (%)
Manu Siddhartha's Malicious URLs dataset	Count Vectorizer	Decision Tree	96.29
		Logistics Regression	95.74
		Random Forest	N/A
		Support Vector Machine	N/A
		XGBoost	94.14
	TF-IDF Vectorizer	Decision Tree	96.13
		Logistics Regression	95.08
		Random Forest	N/A
		Support Vector Machine	N/A
		XGBoost	94.52
	Tokenizer and Pad Sequences	Simple Neural Network	N/A
Grega Vrbancic's Phishing-Dataset	Count Vectorizer	Decision Tree	87.16
		Logistics Regression	92.68
		Random Forest	90.37
		Support Vector Machine	93.51
		XGBoost	90.63
	TF-IDF Vectorizer	Decision Tree	85.39
		Logistics Regression	91.56
		Random Forest	90.68
		Support Vector Machine	93.25
		XGBoost	91.04
	Tokenizer and Pad Sequences	Simple Neural Network	93.66

Table 4.5.5: Models' F1-Score

The tables present the evaluation metrics, including accuracy, precision, recall, and F1-score, for various models on both Manu Siddhartha's Malicious URLs dataset and Grega Vrbancic's Phishing-Dataset. These metrics offer a detailed insight into each model's effectiveness in different scenarios.

From the tables above, we can conclude that the model performance varies across different datasets, encoders, and classifiers. Let's draw key observations:

Manu Siddhartha's Malicious URLs Dataset:

- The Count Vectorizer with a Decision Tree achieves the highest accuracy at 96.32%, showcasing robust performance.
- Logistic Regression with both Count Vectorizer and TF-IDF Vectorizer also demonstrates high accuracy and precision.
- The Tokenizer and Pad Sequences with a Simple Neural Network exhibit exceptional accuracy, reaching 100%.

Grega Vrbancic's Phishing-Dataset:

- Logistic Regression with Count Vectorizer stands out with the highest accuracy at 92.71%, demonstrating effectiveness in this scenario.
- The Simple Neural Network with Tokenizer and Pad Sequences achieves competitive results across all metrics.
- While Decision Tree models exhibit strong recall, other models, such as Logistic Regression and Support Vector Machine, strike a balance between precision and recall.

These observations emphasize the importance of tailoring the choice of encoder and classifier based on the characteristics of the dataset. For instance, the Tokenizer and Pad Sequences with a Simple Neural Network excel in certain scenarios but may require careful consideration due to potential overfitting.

4.6 Summary of Design

The Design chapter outlines a comprehensive workflow for building a malicious URL detection system. We leveraged two reliable datasets – Manu Siddhartha's Malicious URLs and Grega Vrbancic's Phishing-Dataset – encompassing a range of malicious and benign URLs. To prepare the data, we cleaned and reorganized entries in Manu's dataset, while Grega's dataset

was left untouched due to its pre-organized state. Both datasets were then split into training and testing sets for model evaluation.

Next, we explored feature engineering techniques, employing count vectorizer and TF-IDF vectorizer to represent features numerically for the models. Choosing the right algorithm for URL classification is crucial, so we evaluated various options like decision trees, logistic regression, random forests, support vector machines, XGBoost, and simple neural networks. Each algorithm has its strengths and weaknesses, and ultimately, we opted for random forests due to their accuracy, scalability, and ability to handle noise – all qualities crucial for effective URL classification.

Model training involved feeding the prepped data through various encoder and classifier combinations. We then assessed their performance using metrics like accuracy, precision, recall, and F1-score, allowing us to compare and identify the best performers.

Finally, we conducted a thorough evaluation using the held-out test set to ensure the models generalize well to unseen data. The results revealed the most effective model configurations for each dataset, providing valuable insights into the optimal approaches for tackling this specific task.

Throughout the Design chapter, we emphasized careful selection of encoders and classifiers based on the dataset's characteristics, acknowledging the importance of balancing between model interpretability and accuracy. Continuous monitoring and model updates with new data, hyperparameter tuning for performance optimization, and robust testing against adversarial attacks were also highlighted as crucial considerations for the system's future development and real-world deployment.

CHAPTER 5

OUTCOME

5.1 Outcome

As stated in the project objectives, we have achieved the following:

1. We collected and prepared a diverse and balanced dataset of benign and malicious URLs from multiple sources, including blacklists, whitelists, and web crawlers.
2. We performed feature engineering using techniques such as count vectorizer and TF-IDF vectorizer to extract relevant features from the URL structure, content, and context that can help distinguish between benign and malicious URLs.
3. We compared and evaluated the performance of different machine learning models such as decision trees, logistic regression, random forest, support vector machine, XGBoost, and simple neural network in detecting malicious URLs using the extracted features.
4. We trained, evaluated, and optimized the selected machine learning model using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.

By accomplishing these objectives, we have contributed to the development of more effective and efficient methods for detecting malicious URLs and improving internet security for users. However, there are some limitations to our study, such as the limited size of our dataset and the use of only a few machine learning models. Future research could expand on our work by using larger datasets and exploring other machine learning models.

In conclusion, we have achieved our project objectives and developed a comprehensive machine learning-based system for detecting malicious URLs. We hope that our work will contribute to improving internet security for users and help prevent cybercrime.

REFERENCES

- [1] C. Do Xuan, H. D. Nguyen, and T. V. Nikolaevich, "Malicious URL detection based on machine learning," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, 2020, doi: 10.14569/ijacsa.2020.0110119.
- [2] C. Chong Stanford, D. Liu Stanford, and W. Lee Neustar, "Malicious URL Detection." [Online]. Available: <http://support.clean-mx.de/clean->
- [3] Shantanu, B. Janet, and R. Joshua Arul Kumar, "Malicious URL Detection: A Comparative Study," in *Proceedings - International Conference on Artificial Intelligence and Smart Systems, ICAIS 2021*, 2021. doi: 10.1109/ICAIS50930.2021.9396014.
- [4] T. Li, G. Kou, and Y. Peng, "Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods," *Inf Syst*, vol. 91, 2020, doi: 10.1016/j.is.2020.101494.
- [5] A. K. Dutta, "Detecting phishing websites using machine learning technique," *PLoS One*, vol. 16, no. 10 October, 2021, doi: 10.1371/journal.pone.0258361.
- [6] M. Verma and D. Ganguly, "Malicious URL Detection using Machine Learning: A Survey arXiv:1701.07179v3," *Corr*, vol. 1, no. 1, 2019.
- [7] J. Yuan, Y. Liu, and L. Yu, "A Novel Approach for Malicious URL Detection Based on the Joint Model," *Security and Communication Networks*, vol. 2021, 2021, doi: 10.1155/2021/4917016.
- [8] A. Joshi, L. Lloyd, P. Westin, and S. Seethapathy, "Using lexical features for malicious URL Detection - A machine learning approach," *ArXiv*, 2019.
- [9] Y. Liu, L. Ma, M. Wang, and S. Zhang, "Feature Interaction-Based Reinforcement Learning for Tabular Anomaly Detection," 2023, doi: 10.3390/electronics12061313.
- [10] M. Aljabri *et al.*, "Detecting Malicious URLs Using Machine Learning Techniques: Review and Research Directions," *IEEE Access*, vol. 10, pp. 121395–121417, 2022, doi: 10.1109/ACCESS.2022.3222307.
- [11] Y. Wang, "Malicious URL Detection An Evaluation of Feature Extraction and Machine Learning Algorithm," 2022.
- [12] A. Saleem Raja, R. Madhubala, N. Rajesh, L. Shaheetha, and N. Arulkumar, "Survey on Malicious URL Detection Techniques," in *2022 6th International Conference on Trends in Electronics and Informatics, ICOEI 2022 - Proceedings*, 2022, pp. 778–781. doi: 10.1109/ICOEI53556.2022.9777221.

APPENDICES

APPENDIX A

The screenshot shows the Kaggle interface for the 'Malicious URLs dataset' by MANU SIDDHARTHA, updated 3 years ago. The dataset is described as a 'Huge dataset of 6,51,191 Malicious URLs'. The page includes a sidebar with navigation options like Home, Competitions, Datasets, Models, Code, Discussions, Learn, and More. The main content area has tabs for 'Data Card', 'Code (29)', and 'Discussion (1)'. The 'About Dataset' section provides context on the threat of malicious URLs and the dataset's composition. It also includes a 'Usability' score of 10.00, a 'License' of CC0: Public Domain, and 'Expected update frequency' of Annually. Tags include Computer Science, Internet, Beginner, and Social Networks. A thumbnail image of a red padlock is visible on the right.

APPENDIX B

The screenshot shows the GitHub repository 'Phishing-Dataset' by GregaVrbancic. The repository is public and has 45 stars, 13 forks, and 4 watches. It contains 88 commits and 6 branches. The README file is selected, showing the title 'Datasets for Phishing Websites Detection'. The README text states: 'In this repository the two variants of the phishing dataset are presented. Web application To preview the dataset interactively and/or tailor it to your needs, please visit a dedicated web application.' The repository also includes a table of files and their commit history.

File	Commit Message	Time
.github	Update deploy.yml	2 years ago
web-app	Bump rollup from 2.79.0 to 3.8.0 in /web-app (#65)	2 years ago
CITATION.cff	CITATION.cff file added (#47)	2 years ago
README.md	CITATION.cff file added (#47)	2 years ago
dataset_full.csv	Update naming to be in line with DIB paper.	4 years ago
dataset_small.csv	Update naming to be in line with DIB paper.	4 years ago

----END OF FINAL YEAR PROJECT REPORT----