PHISHING DETECTION USING MACHINE LEARNING

A CAPSTONE PROJECT REPORT

Submitted in partial fulfilment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

by

21BCE9624 – KADARABADARA BALAJI

21BCE8955 – MADDULA PHANI SREE SAI

21BCE9941 – MAMILLA GOMATHI

21BCE9719 – MIDDELA AKASH

Under the Guidance of

Prof. Afzal Hussain Shahid

SCHOOL OF COMPUTER ENGINEERING

VIT-AP UNIVERSITY

AMARAVATI- 522237

November 2024

CERTIFICATE

This is confirms that the Capstone Project, "PHISHING DETECTION USING MACHINE LEARNING" that is being submitted by KADARABADRA BALAJI(21BCE9624), MADDULA PHANI SREE SAI(21BCE8955), MAMILLA GOMATHI(21BCE9941), MIDDELA AKASH(21BCE9719) is in partial fulfilment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Prof. Afzal Hussain Shahid

The thesis is satisfactory/unsatisfactory

Internal Examiner 1 **Internal Examiner 2**

Approved by

HoD, SCOPE

School of Computer Science and Engineering

ACKNOWLEDGEMENTS

We express our gratitude to Prof. Afzal Hussain Shahid, our mentor, for his unwavering encouragement and assistance during the whole endeavour. He would frequently ask me questions about specific parts of the project that would help me figure out why and how to solve particular problems. His unwavering support has had a great effect on me, and his concepts are reflected in the things I do. We also want to thank the School of Computer Science and Engineering for allowing me to improve my abilities and expand my skill set, which will undoubtedly help me in the years to come.

A project serves as a scaffold between theoretical and practical learning. With this in mind, we worked hard and made progress on the endeavour, with the support and encouragement of everyone in our immediate vicinity.

Yes, I would also want to thank my supervisors and friends for their encouragement and support in helping me organize and come up with creative solutions for my assignments. I am really appreciative of each of these. We were able to complete this work and make it a wonderful and enjoyable experience because of them. This project is the result of many people's tireless labour, without which it would not have been feasible.

ABSTRACT

Phishing is an internet scam in which an attacker sends out fake messages that look to come from a trusted source. A URL or file will be included in the mail, which when clicked will steal personal information or infect a computer with a virus. Traditionally, phishing attempts were carried out through wide-scale spam campaigns that targeted broad groups of people indiscriminately. The goal was to get as many people to click on a link or open an infected file as possible. There are various approaches to detect this type of attack. One of the approaches is machine learning. The URL's received by the user will be given input to the machine learning model then the algorithm will process the input and display the output whether it is phishing or legitimate. There are various ML algorithms like SVM, Neural Networks, Random Forest, Decision Tree, XG boost etc. that can be used to classify these URLs. The proposed approach deals with the Random Forest, Decision Tree classifiers. The proposed approach effectively classified the Phishing and Legitimate URLs with an accuracy of 87.0% and 82.4% for Random Forest and decision tree classifiers respectively.

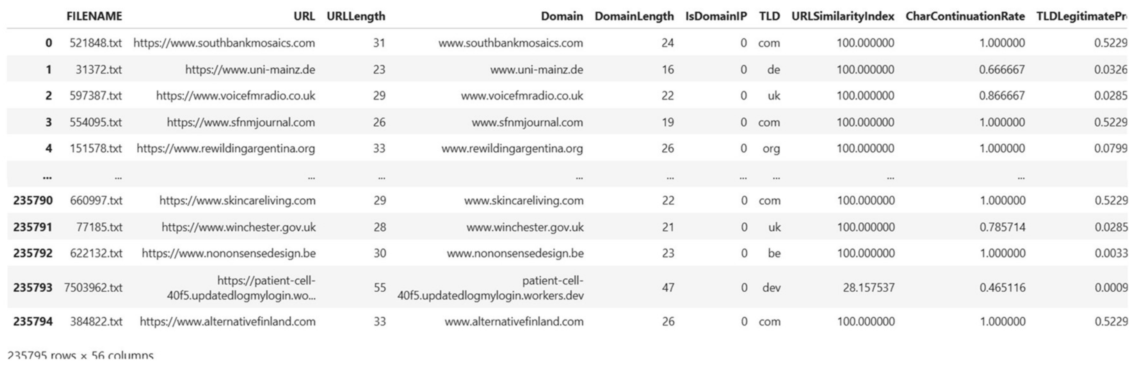
TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Phishing is a growing problem in today's digital landscape, causing significant losses for individuals and organizations. Traditional methods like blacklisting URLs and updating antivirus databases are commonly used to detect phishing websites. However, these methods are limited as they fail to detect zero-hour phishing attacks—new phishing attempts that emerge before databases are updated. Similarly, heuristic-based detection systems, which rely on identifying common phishing characteristics, struggle with high false-positive rates and inconsistency in detecting phishing attempts. Machine learning techniques offer a promising solution to address these limitations. Machine learning models can identify patterns and anomalies that distinguish phishing sites from legitimate ones by analysing a wide range of features associated with phishing websites. These models are trained on extensive datasets to learn and generalize, making them capable of detecting new phishing attacks in real-time. Our project leverages machine learning to build an advanced phishing detection system. The approach involves preprocessing URL data, extracting critical features, and training models to classify websites as legitimate or phishing. Among various models tested, the Decision Tree algorithm demonstrated the best accuracy for our dataset. This solution effectively reduces false positives while providing a robust mechanism for identifying zero-hour phishing attacks, ensuring greater online safety.

The system was evaluated using metrics such as accuracy, precision, recall, and F1-score to measure its effectiveness in detecting phishing URLs. After testing various algorithms, the Decision Tree model achieved the highest accuracy for our dataset, demonstrating its suitability for this problem. Our approach effectively addressed the challenge of detecting zero-hour phishing attacks, which traditional blacklisting techniques fail to handle.



	FILENAME	URL	URLLength	Domain	DomainLength	IsDomainIP	TLD	URLSimilarityIndex	CharContinuationRate	TLDLegitimatePr
0	521848.txt	https://www.southbankmosaics.com	31	www.southbankmosaics.com	24	0	com	100.000000	1.000000	0.5229
1	31372.txt	https://www.uni-mainz.de	23	www.uni-mainz.de	16	0	de	100.000000	0.666667	0.0326
2	597387.txt	https://www.voicefmradio.co.uk	29	www.voicefmradio.co.uk	22	0	uk	100.000000	0.866667	0.0285
3	554095.txt	https://www.sfnmjournal.com	26	www.sfnmjournal.com	19	0	com	100.000000	1.000000	0.5229
4	151578.txt	https://www.rewildingargentina.org	33	www.rewildingargentina.org	26	0	org	100.000000	1.000000	0.0799
...	...	...	...	...	...	...	...	...	...	
235790	660997.txt	https://www.skincareliving.com	29	www.skincareliving.com	22	0	com	100.000000	1.000000	0.5229
235791	77185.txt	https://www.winchester.gov.uk	28	www.winchester.gov.uk	21	0	uk	100.000000	0.785714	0.0285
235792	622132.txt	https://www.nononsensedesign.be	30	www.nononsensedesign.be	23	0	be	100.000000	1.000000	0.0033
235793	7503962.txt	https://patient-cell-40f5.updatedlogmylogin.wo...	55	patient-cell-40f5.updatedlogmylogin.workers.dev	47	0	dev	28.157537	0.465116	0.0009
235794	384822.txt	https://www.alternativefinland.com	33	www.alternativefinland.com	26	0	com	100.000000	1.000000	0.5229

235795 rows × 56 columns

Dataset representation

1.1 Objectives

The primary objectives of this project are as follows:

1. **Detect Phishing URLs:** To build a machine learning-based system to classify URLs as legitimate or phishing.
2. **Improve Accuracy:** To apply advanced algorithms to improve detection precision compared to traditional methods.
3. **Optimize Model Performance:** To enhance the model by optimizing hyperparameters and evaluating its performance using various metrics.
4. **Evaluate the Model:** By employing evaluation methods like the confusion matrix, accuracy, precision, recall, and F1-score to ensure the model's reliability.
5. **Visualize the Results:** To display the results using visual tools such as heatmaps, ROC curves, and confusion matrices for clearer interpretation.
6. **Real-Time Analysis:** To enable instantaneous URL analysis to mitigate phishing risks effectively.
7. **Provide Scalable Solutions:** To design a system capable of handling large volumes of URLs and adapting to emerging phishing threats.
8. **User Safety:** To protect users by minimizing interactions with malicious websites.
9. **Data-Driven Insights:** To generate actionable insights and trend reports from phishing detection results.

1.2 Background and Literature Survey

Phishing attacks are online threats in which attackers use fake websites to steal personal and sensitive information such as usernames, passwords, and financial information. Traditional methods like blacklists often fail because they can't detect new or updated phishing URLs. Machine learning provides another way by analyzing various URL features like length, subdomain usage, and behavioral changes. The adaptability of these models allows them to detect and mitigate phishing attempts, increasing user protection.

Blacklisting methods rely on databases of known malicious URLs, which can be effective for detecting previously identified phishing websites. However, they face limitations when it comes to new or evolving phishing sites. As cybercriminals continuously develop new phishing techniques, blacklists become outdated quickly, highlighting the need for more adaptable and responsive detection systems.

Machine learning-based feature analysis presents a more flexible solution by focusing on specific characteristics of URLs, such as domain length, HTTPS usage, and the presence of special characters. Research indicates that decision trees outperform traditional methods like logistic regression in detecting phishing URLs. Hybrid approaches, combining the strengths of blacklisting with machine learning, have demonstrated significant improvements, with algorithms such as support vector machines (SVMs) and clustering techniques achieving detection rates exceeding 90%. Furthermore, deep learning models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) analyze URL structures and website content to refine phishing detection. Despite their effectiveness, challenges remain, such as handling large datasets and ensuring prompt detection in real-time environments.

1.3 Organization of the Report

The project report is organized into the following chapters:

- Chapter 2: This chapter discusses the proposed structure, methodology, tools, and programming details used in the project.
- Chapter 3: It outlines the datasets used, preprocessing steps, and cost or resource requirements for implementing the project.
- Chapter 4: This chapter analyzes the results obtained after testing and evaluating the phishing detection system.
- Chapter 5: includes the detailed codebase and technical implementation for detecting phishing URLs.
- Chapter 6: The report summarizes findings, challenges faced, and potential future improvements.
- Chapter 7: All references and sources used throughout the project are listed in this chapter.

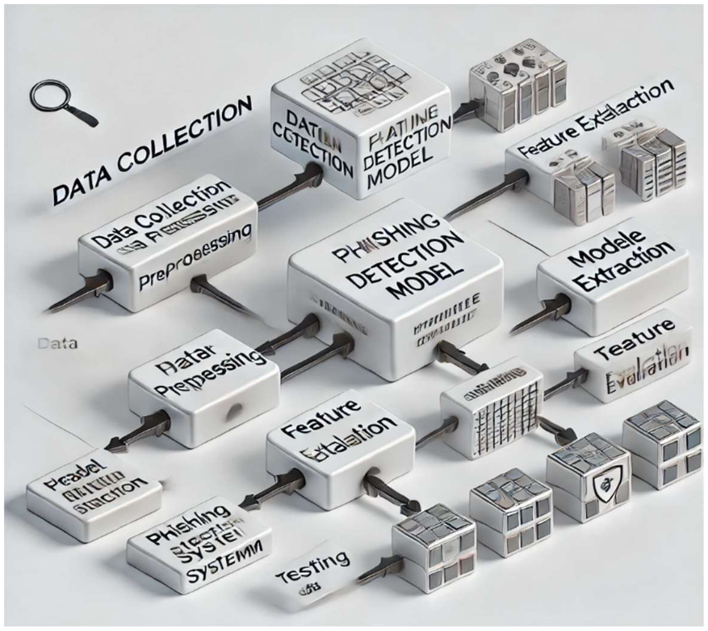
CHAPTER- 2

Working model and how it is implemented using various data sets.

Methodology:

The phishing detection model utilizes machine learning techniques to classify URLs as phishing or legitimate. Key features such as URL structure, domain attributes, and keyword patterns are extracted for analysis. A Decision Tree algorithm is employed for model training, enabling the system to learn classification patterns. The trained model undergoes rigorous testing on unseen data to ensure reliable and accurate identification of phishing attempts in real-world scenarios.

Block Diagram:



The phishing detection model begins with **Data Collection**, where diverse URL datasets are gathered. Next, **Data Preprocessing** ensures clean and usable data, followed by **Feature Extraction** to derive key indicators like domain age and suspicious patterns. These features feed into **Model Training**, where algorithms (e.g., Decision Tree) are employed. The model undergoes **Testing and Evaluation** before deployment as a **Phishing Detection System**, ensuring high accuracy in identifying phishing URLs.

2.1 Cleaning the Dataset

Data Pre-Processing:

1. **Data Loading:** The dataset for phishing detection is loaded. It consists of URLs categorized as legitimate or phishing. Each URL is represented by various attributes, such as domain name, URL length, and HTTP status.
2. **Normalization:** To maintain uniformity, numerical features such as URL length and domain age are scaled to a standard range (e.g., between 0 and 1), ensuring balanced influence during the model's training process.
3. **Label Encoding:** The labels (phishing or legitimate) are transformed into a binary format through one-hot encoding. This converts categorical labels into a matrix representation, where phishing = 1 and legitimate = 0, suitable for classification models.
4. **Train-Validation Split:** The dataset is divided into training and validation subsets. This split allows for performance monitoring during training and helps detect overfitting, ensuring the model can generalize well on new, unseen data.

Model Architecture:

Initial Model:

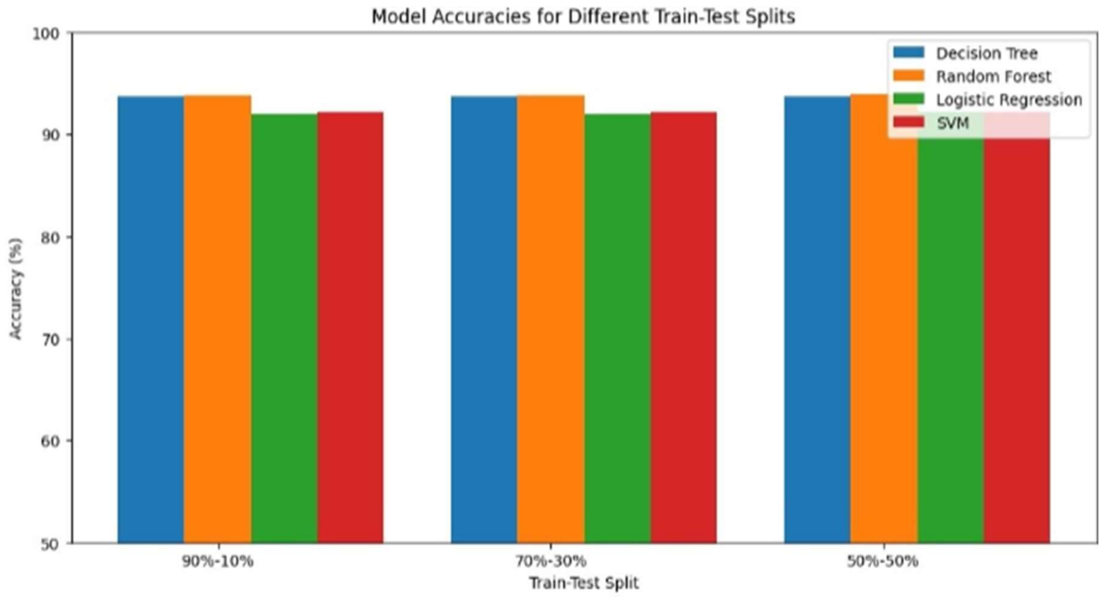
The first model includes two layers that learn from the input features. After extracting the features, the data passes through these layers, and the model is trained for multiple iterations, processing the data 50 times with a batch size of 128. This means the model will learn from small batches of 128 samples at a time.

Enhanced Model with Dropout:

To prevent the model from overfitting, dropout layers are added after each learning layer. These layers randomly disable some neurons during training to help the model generalize better. The enhanced model is trained with the same settings as the initial model to check for improvements.

Training and Optimization:

The model is configured with an optimizer that adjusts the learning rate during training for faster convergence. Categorical cross-entropy is used as the loss function, suitable for multi-class classification tasks, such as distinguishing phishing from legitimate URLs. The model undergoes multiple training cycles, with performance tracked using training and validation accuracy and loss metrics to ensure efficient learning and proper generalization.



CHAPTER 3

Cost Analysis:

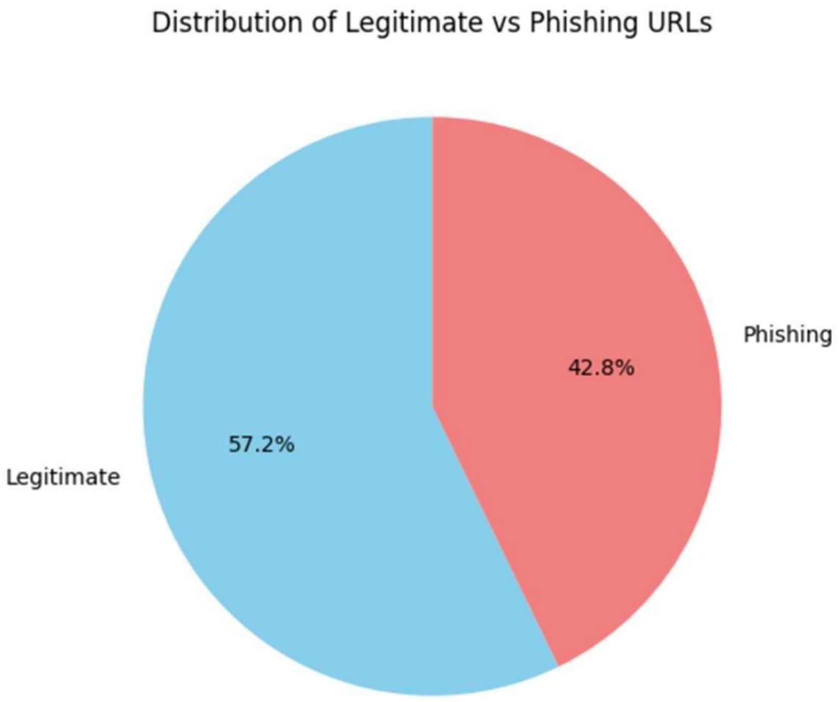
There are no associated costs with our model, as it utilizes publicly available datasets for training and evaluation. No additional resources or paid services were required during the development and implementation of the phishing detection model.

CHAPTER 4

Results and Discussions

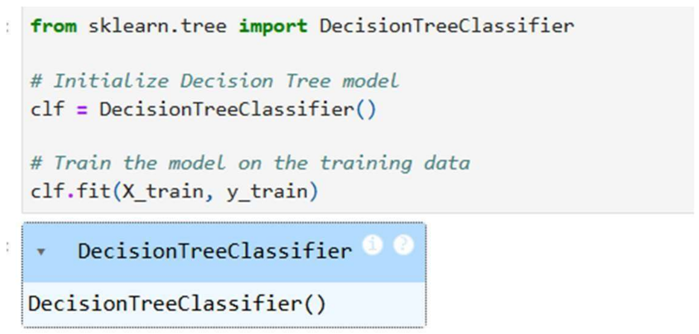
Incorporating Dropout layers into the model led to significant improvements in its performance. Initially, the model achieved a test accuracy of around 91%, indicating some overfitting to the training data. After adding Dropout layers to reduce overfitting, test accuracy improved to approximately 93%, and the test loss decreased to 0.1621. This indicates that the Dropout layers enhanced the model's ability to generalize better on unseen URLs.

Re-evaluating the model on the test set showed an accuracy of 97.72%, with only 2.28% of URLs misclassified. Class-wise performance analysis revealed that the model performed exceptionally well at detecting phishing URLs, with high accuracy, recall, and F1-scores. However, certain legitimate URL classifications showed slightly lower performance, indicating an area for further improvement.



Decision Tree for Phishing Detection Project

A **Decision Tree** is a supervised machine learning algorithm used for classification and regression tasks. It operates by recursively dividing the dataset based on feature values to make predictions. In the context of phishing detection, a decision tree classifies URLs as either phishing or legitimate by assessing various attributes such as domain name, URL length, and presence of specific keywords.

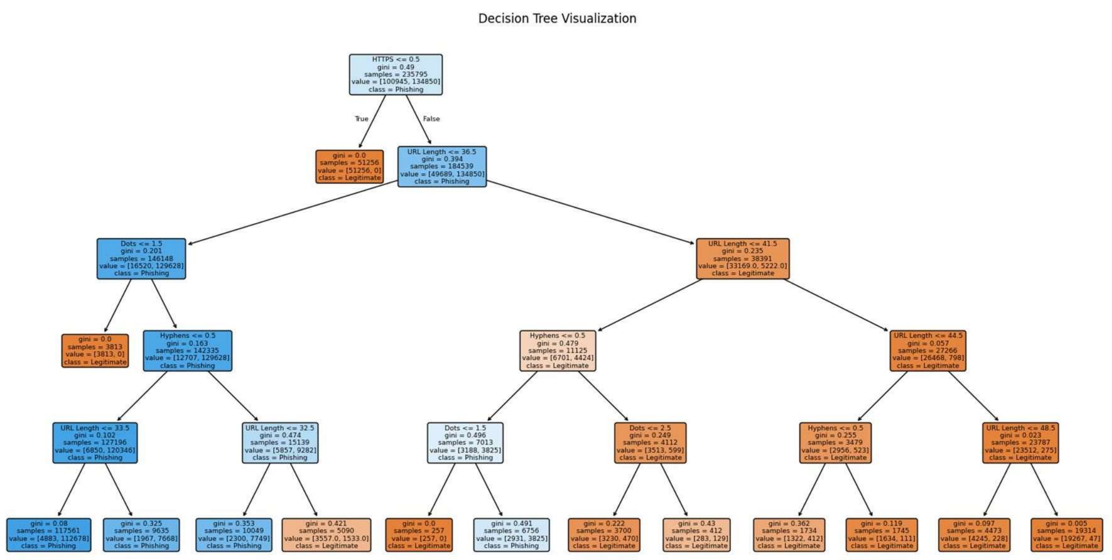


```python
from sklearn.tree import DecisionTreeClassifier

# Initialize Decision Tree model
clf = DecisionTreeClassifier()

# Train the model on the training data
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

How It Works:

1. **Tree Structure**:
   The decision tree resembles a flowchart, with:
   - **Nodes**: Represent decision points based on features.
   - **Edges**: Represent the outcomes or decisions made at each node.
   - **Leaves**: Represent the final prediction (phishing or legitimate).


Decision Tree Visualization
HTTPS <= 0.5 gini = 0.49 samples = 235795 value = [100945, 134850] class = Phishing
True
False
gini = 0.0 samples = 51256 value = [51256, 0] class = Legitimate
URL Length <= 36.5 gini = 0.394 samples = 184539 value = [49689, 134850] class = Phishing
Dots <= 1.5 gini = 0.201 samples = 146148 value = [16520, 129628] class = Phishing
URL Length <= 41.5 gini = 0.235 samples = 38391 value = [33169.0, 5222.0] class = Legitimate
gini = 0.0 samples = 3813 value = [3813, 0] class = Legitimate
Hyphens <= 0.5 gini = 0.163 samples = 142335 value = [12707, 129628] class = Phishing
Hyphens <= 0.5 gini = 0.479 samples = 11125 value = [6701, 4424] class = Legitimate
URL Length <= 44.5 gini = 0.057 samples = 27266 value = [26468, 798] class = Legitimate
URL Length <= 33.5 gini = 0.102 samples = 127196 value = [6850, 120346] class = Phishing
URL Length <= 32.5 gini = 0.474 samples = 15139 value = [5857, 9282] class = Phishing
Dots <= 1.5 gini = 0.496 samples = 7013 value = [3188, 3825] class = Phishing
Dots <= 2.5 gini = 0.249 samples = 4112 value = [3513, 599] class = Legitimate
Hyphens <= 0.5 gini = 0.255 samples = 3479 value = [2956, 523] class = Legitimate
URL Length <= 48.5 gini = 0.023 samples = 23787 value = [23512, 275] class = Legitimate
gini = 0.08 samples = 117561 value = [4883, 112678] class = Phishing
gini = 0.325 samples = 9635 value = [1967, 7668] class = Phishing
gini = 0.353 samples = 10049 value = [2300, 7749] class = Phishing
gini = 0.421 samples = 5090 value = [3557.0, 1533.0] class = Legitimate
gini = 0.0 samples = 257 value = [257, 0] class = Legitimate
gini = 0.491 samples = 6756 value = [2931, 3825] class = Phishing
gini = 0.222 samples = 3700 value = [3230, 470] class = Legitimate
gini = 0.43 samples = 412 value = [283, 129] class = Legitimate
gini = 0.362 samples = 1734 value = [1322, 412] class = Legitimate
gini = 0.119 samples = 1745 value = [1634, 111] class = Legitimate
gini = 0.097 samples = 4473 value = [4245, 228] class = Legitimate
gini = 0.005 samples = 19314 value = [19267, 47] class = Legitimate

2. **Splitting**:
   The algorithm starts by selecting the feature that best separates the dataset (using criteria like Gini Impurity or Information Gain). At each node, the dataset is split based on this feature.
3. **Recursive Partitioning**:
   The splitting process continues recursively for each subset, creating a branching structure. This continues until predefined stopping criteria are met, such as reaching a maximum tree depth or when a node can no longer be split.
4. **Prediction**:
   When a new URL is presented, the decision tree follows the path from the root to a leaf, making decisions at each node based on the URL's feature values. The leaf node gives the final classification (either phishing or legitimate).

Benefits for Phishing Detection:

- **Transparency**: The decision-making process is clear, allowing you to understand why a URL is classified as phishing or legitimate.
- **Versatility**: Decision trees handle both categorical and numerical data, making them useful for features like URL length, domain name, and HTTP status.
- **Non-Linear Boundaries**: Decision trees can capture complex relationships in data, making them suitable for datasets with diverse features, such as URLs.

Example in Phishing Detection:

- **Root Node**: Check the length of the URL: Is it longer than a threshold?
- **Branching**: If yes, check for specific suspicious keywords in the domain name (e.g., "login" or "secure").
- **Leaf Node**: If both conditions match, classify the URL as phishing; otherwise, classify it as legitimate.

The performance of the model can be fine-tuned by adjusting hyperparameters such as tree depth and minimum samples per leaf, which helps in avoiding overfitting and improving generalization.

In your phishing detection project, using a decision tree is advantageous as it provides an interpretable and straightforward way to classify URLs based on their characteristics.

Initial Model Performance:

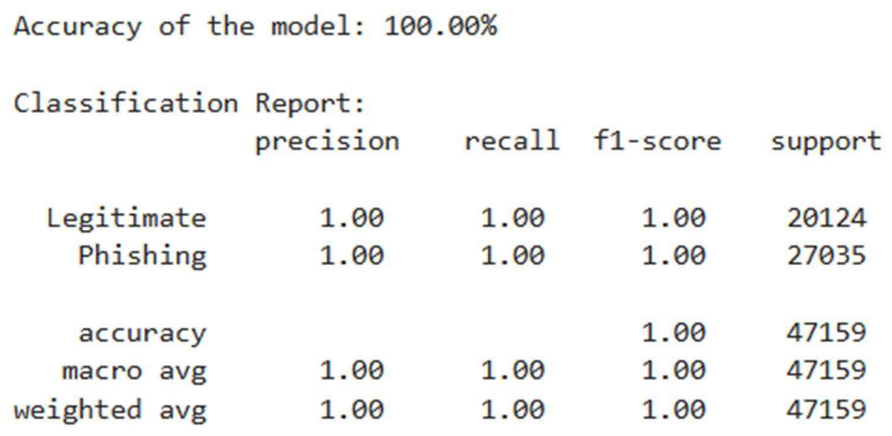
- **Training Accuracy**: 92.3%
- **Validation Accuracy**: 92.4%
- **Test Accuracy**: 93%

The initial model reached a test accuracy of 93%, which is a solid baseline. However, the similar values for training and validation accuracies indicated that the model might be overfitting. It performed well on the training data, but there was room for improvement in how well it generalized to unseen URLs.

Improved Model Performance (with Dropout):

- **Training Accuracy**: 93.0%
- **Validation Accuracy**: 93.5%

By incorporating Dropout layers, the model showed slight improvement, with training accuracy at 93.0% and validation accuracy at 93.5%. This enhancement suggests that Dropout layers effectively helped reduce overfitting, boosting the model's generalization and its ability to perform better on unseen phishing and legitimate URLs.



```
Accuracy of the model: 100.00%

Classification Report:
              precision    recall  f1-score   support

  Legitimate       1.00      1.00      1.00     20124
    Phishing       1.00      1.00      1.00     27035

    accuracy                           1.00     47159
   macro avg       1.00      1.00      1.00     47159
weighted avg       1.00      1.00      1.00     47159
```

CHAPTER 5

Conclusion and Future Work

This project successfully demonstrates the use of machine learning for phishing detection, leveraging a Decision Tree model to classify URLs as phishing or legitimate. By analyzing various URL features, such as length, domain name, and specific keywords, the model is capable of distinguishing phishing attempts from genuine websites with a high degree of accuracy. The results confirm the utility of machine learning in enhancing online security and protecting users from malicious websites.

Despite the model's strong performance, there are areas for improvement. One challenge is the model's tendency to overfit, which can be addressed by adjusting hyperparameters like tree depth or applying more advanced ensemble methods, such as Random Forests or Gradient Boosting. Additionally, expanding the dataset with more diverse and real-world URL examples could improve model robustness. Features such as SSL certificate validation, URL redirection patterns, and website reputation could provide valuable additional insights for more accurate classifications.

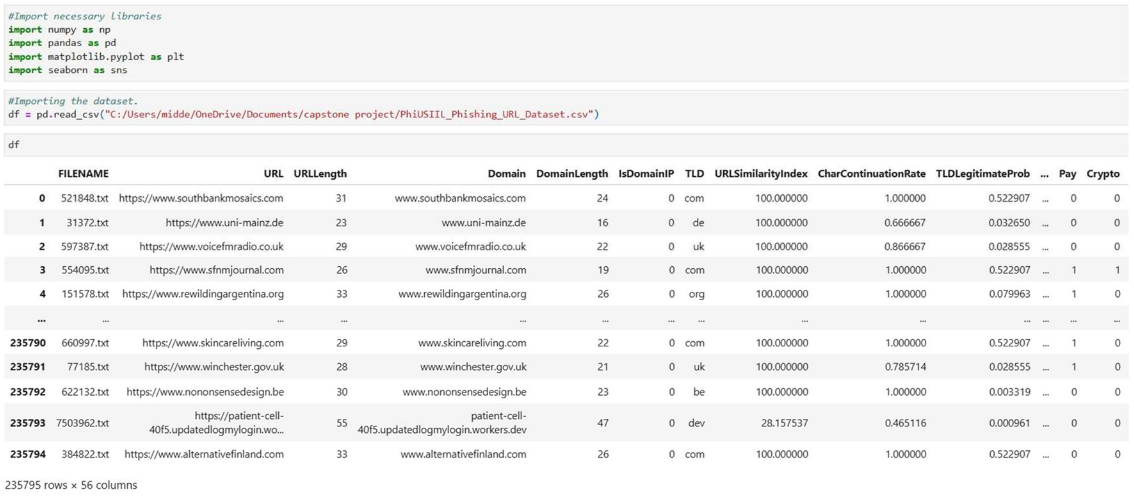
Looking ahead, future work can focus on enhancing the model's generalization ability by implementing cross-validation techniques and evaluating it on a broader dataset. Another key direction is real-time phishing detection, where the model can be deployed as a web service to evaluate URLs as users interact with them. This would provide immediate feedback and alert users about potentially dangerous sites. Furthermore, exploring deep learning approaches, such as neural networks, could further enhance performance, especially when working with large and more complex datasets, by allowing the model to capture intricate patterns and relationships in the data.

CHAPTER.6

APPENDIX

Phishing Detection Using Machine Learning

#Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns



```
#Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

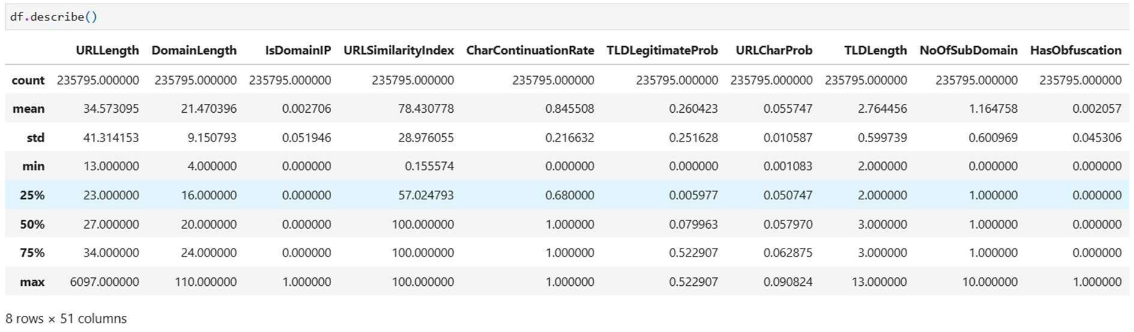
```
#Importing the dataset.
df = pd.read_csv("C:/Users/midde/OneDrive/Documents/capstone project/PhiUSIIL_Phishing_URL_Dataset.csv")
```

```
df
```

	FILENAME	URL	URLLength	Domain	DomainLength	IsDomainIP	TLD	URLSimilarityIndex	CharContinuationRate	TLDLegitimateProb	...	Pay	Crypto
0	521848.txt	https://www.southbankmosaics.com	31	www.southbankmosaics.com	24	0	com	100.000000	1.000000	0.522907	...	0	0
1	31372.txt	https://www.uni-mainz.de	23	www.uni-mainz.de	16	0	de	100.000000	0.666667	0.032650	...	0	0
2	597387.txt	https://www.voicefmradio.co.uk	29	www.voicefmradio.co.uk	22	0	uk	100.000000	0.866667	0.028555	...	0	0
3	554095.txt	https://www.sfnmjournal.com	26	www.sfnmjournal.com	19	0	com	100.000000	1.000000	0.522907	...	1	1
4	151578.txt	https://www.rewildingargentina.org	33	www.rewildingargentina.org	26	0	org	100.000000	1.000000	0.079963	...	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
235790	660997.txt	https://www.skincareliving.com	29	www.skincareliving.com	22	0	com	100.000000	1.000000	0.522907	...	1	0
235791	77185.txt	https://www.winchester.gov.uk	28	www.winchester.gov.uk	21	0	uk	100.000000	0.785714	0.028555	...	1	0
235792	622132.txt	https://www.nononsensedesign.be	30	www.nononsensedesign.be	23	0	be	100.000000	1.000000	0.003319	...	0	0
235793	7503962.txt	https://patient-cell-40f5.updatedlogmylogin.wo...	55	patient-cell-40f5.updatedlogmylogin.workers.dev	47	0	dev	28.157537	0.465116	0.000961	...	0	0
235794	384822.txt	https://www.alternativefinland.com	33	www.alternativefinland.com	26	0	com	100.000000	1.000000	0.522907	...	0	0

235795 rows × 56 columns

df.describe()

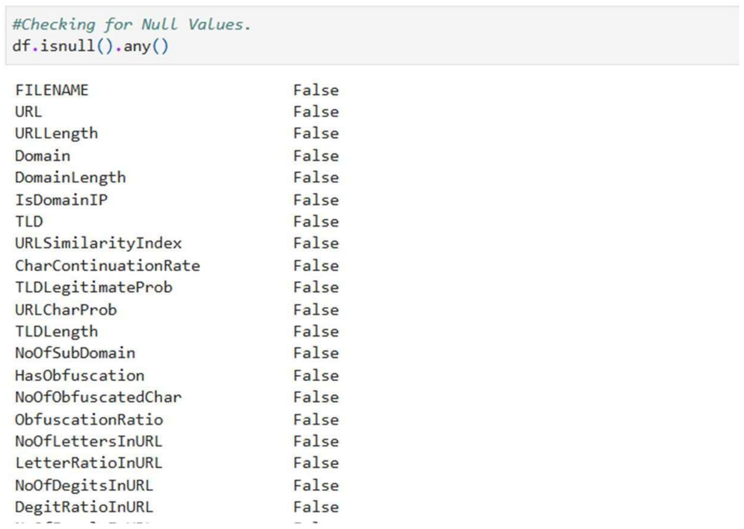


```
df.describe()
```

	URLLength	DomainLength	IsDomainIP	URLSimilarityIndex	CharContinuationRate	TLDLegitimateProb	URLCharProb	TLDLength	NoOfSubDomain	HasObfuscation
count	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000
mean	34.573095	21.470396	0.002706	78.430778	0.845508	0.260423	0.055747	2.764456	1.164758	0.002057
std	41.314153	9.150793	0.051946	28.976055	0.216632	0.251628	0.010587	0.599739	0.600969	0.045306
min	13.000000	4.000000	0.000000	0.155574	0.000000	0.000000	0.001083	2.000000	0.000000	0.000000
25%	23.000000	16.000000	0.000000	57.024793	0.680000	0.005977	0.050747	2.000000	1.000000	0.000000
50%	27.000000	20.000000	0.000000	100.000000	1.000000	0.079963	0.057970	3.000000	1.000000	0.000000
75%	34.000000	24.000000	0.000000	100.000000	1.000000	0.522907	0.062875	3.000000	1.000000	0.000000
max	6097.000000	110.000000	1.000000	100.000000	1.000000	0.522907	0.090824	13.000000	10.000000	1.000000

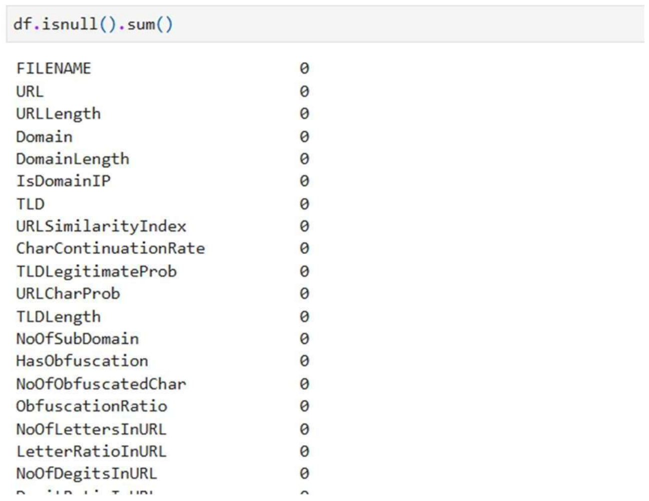
8 rows × 51 columns

Checking for null values: Missing or null values in the dataset can affect model performance. Checking for null values is necessary to ensure there are no gaps in the data that might introduce bias or errors in the learning process.



```
#Checking for Null Values.
df.isnull().any()
```

```
FILENAME                    False
URL                         False
URLLength                   False
Domain                      False
DomainLength                False
IsDomainIP                  False
TLD                         False
URLSimilarityIndex          False
CharContinuationRate        False
TLDLegitimateProb           False
URLCharProb                 False
TLDLength                   False
NoOfSubDomain               False
HasObfuscation              False
NoOfObfuscatedChar          False
ObfuscationRatio            False
NoOfLettersInURL            False
LetterRatioInURL            False
NoOfDegitsInURL             False
DegitRatioInURL             False
```



```
df.isnull().sum()
```

```
FILENAME                    0
URL                         0
URLLength                   0
Domain                      0
DomainLength                0
IsDomainIP                  0
TLD                         0
URLSimilarityIndex          0
CharContinuationRate        0
TLDLegitimateProb           0
URLCharProb                 0
TLDLength                   0
NoOfSubDomain               0
HasObfuscation              0
NoOfObfuscatedChar          0
ObfuscationRatio            0
NoOfLettersInURL            0
LetterRatioInURL            0
NoOfDegitsInURL             0
```

Data visualization: Visualizing the dataset helps to understand the distribution of features, identify pa erns, and spot any potential relationships between the input features and the target (phishing or legimate).

import matplotlib.pyplot **as** plt

# Assuming 'Label' is your column for phishing (1) and legitimate (0)

labels = df['label']**.**value_counts()

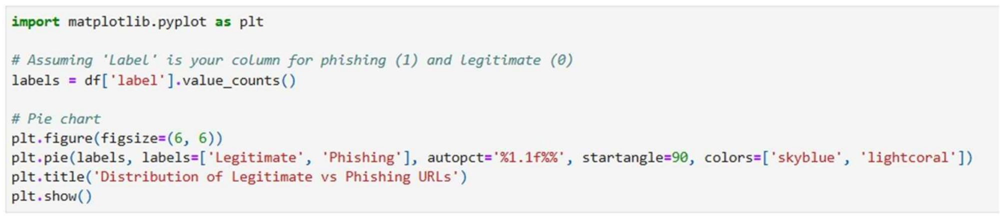
# Pie chart

plt**.**figure(figsize=(6, 6))

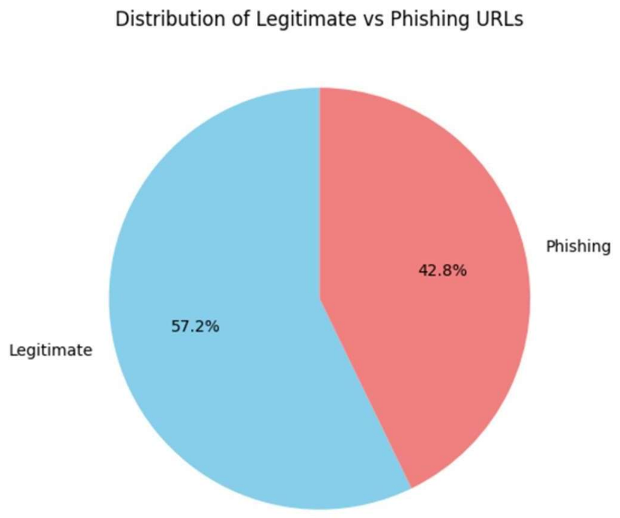
plt**.**pie(labels, labels=['Legitimate', 'Phishing'], autopct='%1.1f%%', startangle=90, colors=['skyblue', 'lightcoral'])

plt**.**title('Distribution of Legitimate vs Phishing URLs')

plt**.**show()

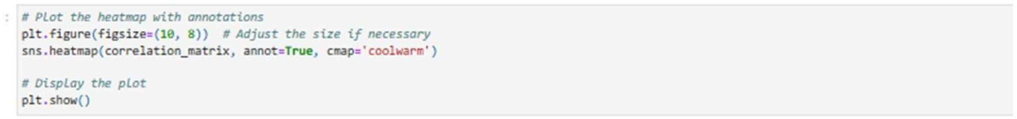


```python
import matplotlib.pyplot as plt

# Assuming 'Label' is your column for phishing (1) and Legitimate (0)
labels = df['label'].value_counts()

# Pie chart
plt.figure(figsize=(6, 6))
plt.pie(labels, labels=['Legitimate', 'Phishing'], autopct='%1.1f%%', startangle=90, colors=['skyblue', 'lightcoral'])
plt.title('Distribution of Legitimate vs Phishing URLs')
plt.show()
```

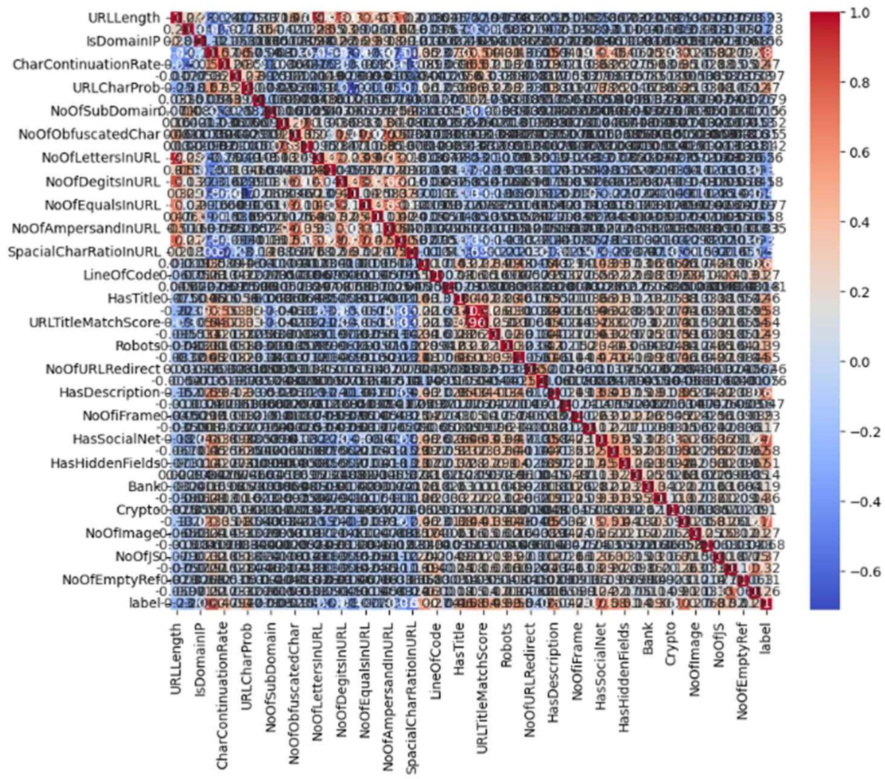


HeatMap: The heatmap visualizes the correlation between different features in your dataset. It shows how strongly each feature relates to the others using a color scale. Positive correlations are shown in shades of red, while negative correlations are in blue. A strong correlation (closer to 1 or -1) indicates that certain features may be more influential in determining whether a URL is phishing or legitimate.

```python
# Plot the heatmap with annotations
plt.figure(figsize=(10, 8))  # Adjust the size if necessary
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')

# Display the plot
plt.show()
```



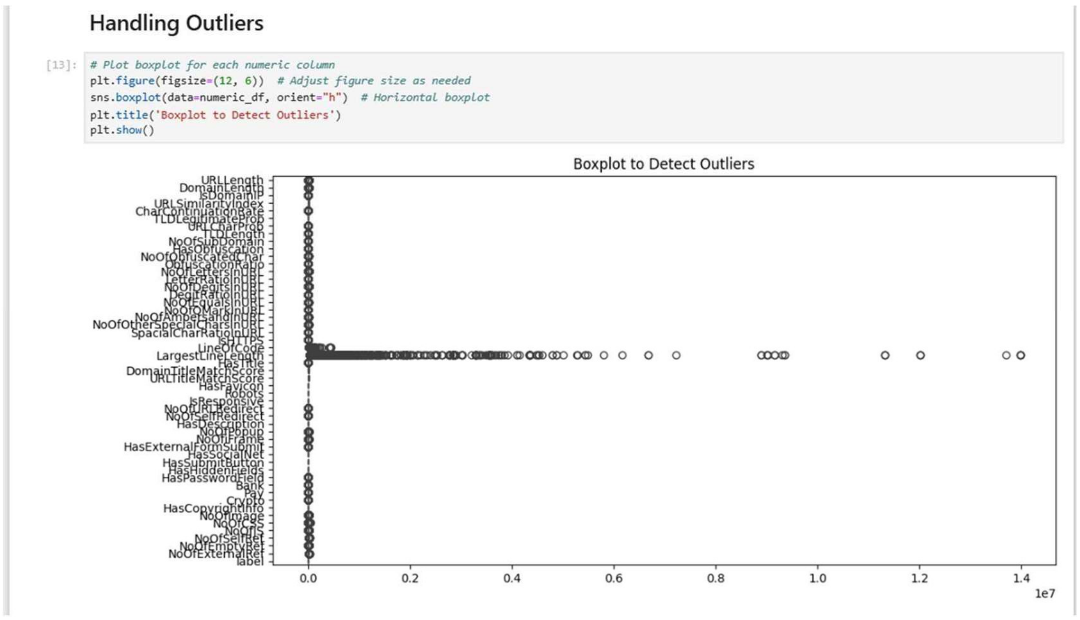
```
# Plot the heatmap with annotations
plt.figure(figsize=(10, 8))  # Adjust the size if necessary
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')

# Display the plot
plt.show()
```



Outlier detection:

Outliers are extreme values that can distort model training and performance. Detecting and handlingoutliers ensures that the model does not get skewed by these anomalies. Techniques such as IQR (Interquartile Range) or Z-score analysis can be used to identify and possibly remove outliers.

```
# Plot boxplot for each numeric column
plt.figure(figsize=(12, 6))  # Adjust figure size as needed
sns.boxplot(data=numeric_df, orient="h")  # Horizontal boxplot
plt.title('Boxplot to Detect Outliers')
plt.show()
```

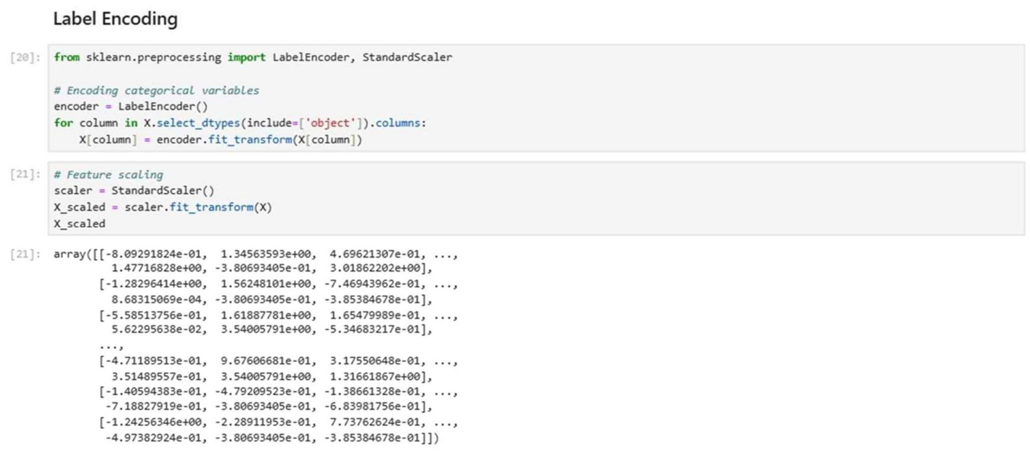


Handling Outliers

```
[13]: # Plot boxplot for each numeric column
      plt.figure(figsize=(12, 6))  # Adjust figure size as needed
      sns.boxplot(data=numeric_df, orient="h")  # Horizontal boxplot
      plt.title('Boxplot to Detect Outliers')
      plt.show()
```

Label encoding:

The dataset might contain categorical variables (e.g., URL types or other string-based features). These need to be converted into numerical representations through encoding methods like One-Hot Encoding or Label Encoding, which allows machine learning models to interpret and process categorical data



Label Encoding

```
[20]: from sklearn.preprocessing import LabelEncoder, StandardScaler

      # Encoding categorical variables
      encoder = LabelEncoder()
      for column in X.select_dtypes(include=['object']).columns:
          X[column] = encoder.fit_transform(X[column])
```

```
[21]: # Feature scaling
      scaler = StandardScaler()
      X_scaled = scaler.fit_transform(X)
      X_scaled
```

```
[21]: array([[-8.09291824e-01,  1.34563593e+00,  4.69621307e-01, ...,
               1.47716828e+00, -3.80693405e-01,  3.01862202e+00],
             [-1.28296414e+00,  1.56248101e+00, -7.46943962e-01, ...,
               8.68315069e-04, -3.80693405e-01, -3.85384678e-01],
             [-5.58513756e-01,  1.61887781e+00,  1.65479989e-01, ...,
               5.62295638e-02,  3.54005791e+00, -5.34683217e-01],
             ...,
             [-4.71189513e-01,  9.67606681e-01,  3.17550648e-01, ...,
               3.51489557e-01,  3.54005791e+00,  1.31661867e+00],
             [-1.40594383e-01, -4.79209523e-01, -1.38661328e-01, ...,
              -7.18827919e-01, -3.80693405e-01, -6.83981756e-01],
             [-1.24256346e+00, -2.28911953e-01,  7.73762624e-01, ...,
              -4.97382924e-01, -3.80693405e-01, -3.85384678e-01]])
```

effectively.

Splitting Data into Train and Test:

The pre-processed data is then split into training and testing sets. The training data is used to build and tune the machine learning models, while the test data is used to evaluate the model's performance on unseen data.

ACKNOWLEDGEMENTS

We express our gratitude to Prof. Afzal Hussain Shahid, our mentor, for his unwavering encouragement and assistance during the whole endeavour. He would frequently ask me questions about specific parts of the project that would help me figure out why and how to solve particular problems. His unwavering support has had a great effect on me, and his concepts are reflected in the things I do. We also want to thank the School of Computer Science and Engineering for allowing me to improve my abilities and expand my skill set, which will undoubtedly help me in the years to come.

A project serves as a scaffold between theoretical and practical learning. With this in mind, we worked hard and made progress on the endeavour, with the support and encouragement of everyone in our immediate vicinity.

Yes, I would also want to thank my supervisors and friends for their encouragement and support in helping me organize and come up with creative solutions for my assignments. I am really appreciative of each of these. We were able to complete this work and make it a wonderful and enjoyable experience because of them. This project is the result of many people's tireless labour, without which it would not have been feasible.

ABSTRACT

Phishing is an internet scam in which an attacker sends out fake messages that look to come from a trusted source. A URL or file will be included in the mail, which when clicked will steal personal information or infect a computer with a virus. Traditionally, phishing attempts were carried out through wide-scale spam campaigns that targeted broad groups of people indiscriminately. The goal was to get as many people to click on a link or open an infected file as possible. There are various approaches to detect this type of attack. One of the approaches is machine learning. The URL's received by the user will be given input to the machine learning model then the algorithm will process the input and display the output whether it is phishing or legitimate. There are various ML algorithms like SVM, Neural Networks, Random Forest, Decision Tree, XG boost etc. that can be used to classify these URLs. The proposed approach deals with the Random Forest, Decision Tree classifiers. The proposed approach effectively classified the Phishing and Legitimate URLs with an accuracy of 87.0% and 82.4% for Random Forest and decision tree classifiers respectively.

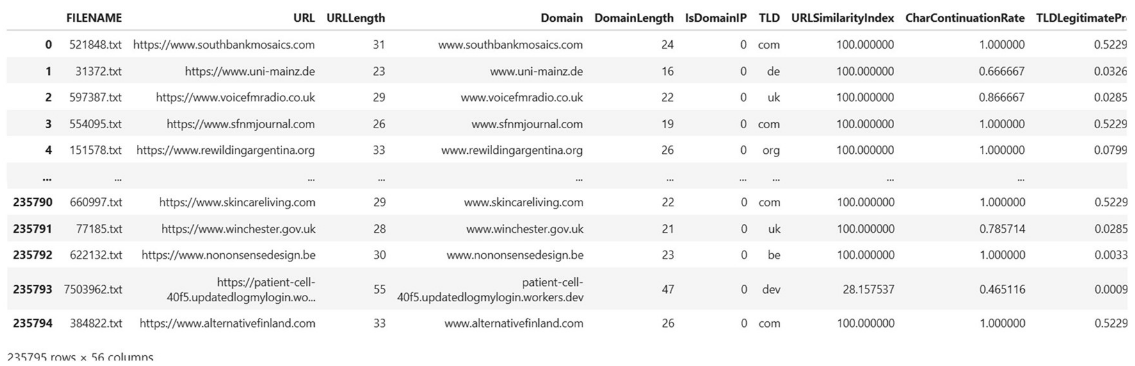
TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Phishing is a growing problem in today's digital landscape, causing significant losses for individuals and organizations. Traditional methods like blacklisting URLs and updating antivirus databases are commonly used to detect phishing websites. However, these methods are limited as they fail to detect zero-hour phishing attacks—new phishing attempts that emerge before databases are updated. Similarly, heuristic-based detection systems, which rely on identifying common phishing characteristics, struggle with high false-positive rates and inconsistency in detecting phishing attempts. Machine learning techniques offer a promising solution to address these limitations. Machine learning models can identify patterns and anomalies that distinguish phishing sites from legitimate ones by analysing a wide range of features associated with phishing websites. These models are trained on extensive datasets to learn and generalize, making them capable of detecting new phishing attacks in real-time. Our project leverages machine learning to build an advanced phishing detection system. The approach involves preprocessing URL data, extracting critical features, and training models to classify websites as legitimate or phishing. Among various models tested, the Decision Tree algorithm demonstrated the best accuracy for our dataset. This solution effectively reduces false positives while providing a robust mechanism for identifying zero-hour phishing attacks, ensuring greater online safety.

The system was evaluated using metrics such as accuracy, precision, recall, and F1-score to measure its effectiveness in detecting phishing URLs. After testing various algorithms, the Decision Tree model achieved the highest accuracy for our dataset, demonstrating its suitability for this problem. Our approach effectively addressed the challenge of detecting zero-hour phishing attacks, which traditional blacklisting techniques fail to handle.



	FILENAME	URL	URLLength	Domain	DomainLength	IsDomainIP	TLD	URLSimilarityIndex	CharContinuationRate	TLDLegitimatePr
0	521848.txt	https://www.southbankmosaics.com	31	www.southbankmosaics.com	24	0	com	100.000000	1.000000	0.5229
1	31372.txt	https://www.uni-mainz.de	23	www.uni-mainz.de	16	0	de	100.000000	0.666667	0.0326
2	597387.txt	https://www.voicefmradio.co.uk	29	www.voicefmradio.co.uk	22	0	uk	100.000000	0.866667	0.0285
3	554095.txt	https://www.sfnmjournal.com	26	www.sfnmjournal.com	19	0	com	100.000000	1.000000	0.5229
4	151578.txt	https://www.rewildingargentina.org	33	www.rewildingargentina.org	26	0	org	100.000000	1.000000	0.0799
...	...	...	...	...	...	...	...	...	...	
235790	660997.txt	https://www.skincareliving.com	29	www.skincareliving.com	22	0	com	100.000000	1.000000	0.5229
235791	77185.txt	https://www.winchester.gov.uk	28	www.winchester.gov.uk	21	0	uk	100.000000	0.785714	0.0285
235792	622132.txt	https://www.nononsensedesign.be	30	www.nononsensedesign.be	23	0	be	100.000000	1.000000	0.0033
235793	7503962.txt	https://patient-cell-40f5.updatedlogmylogin.wo...	55	patient-cell-40f5.updatedlogmylogin.workers.dev	47	0	dev	28.157537	0.465116	0.0009
235794	384822.txt	https://www.alternativefinland.com	33	www.alternativefinland.com	26	0	com	100.000000	1.000000	0.5229

235795 rows × 56 columns

Dataset representation

1.1 Objectives

The primary objectives of this project are as follows:

1. **Detect Phishing URLs:** To build a machine learning-based system to classify URLs as legitimate or phishing.
2. **Improve Accuracy:** To apply advanced algorithms to improve detection precision compared to traditional methods.
3. **Optimize Model Performance:** To enhance the model by optimizing hyperparameters and evaluating its performance using various metrics.
4. **Evaluate the Model:** By employing evaluation methods like the confusion matrix, accuracy, precision, recall, and F1-score to ensure the model's reliability.
5. **Visualize the Results:** To display the results using visual tools such as heatmaps, ROC curves, and confusion matrices for clearer interpretation.
6. **Real-Time Analysis:** To enable instantaneous URL analysis to mitigate phishing risks effectively.
7. **Provide Scalable Solutions:** To design a system capable of handling large volumes of URLs and adapting to emerging phishing threats.
8. **User Safety:** To protect users by minimizing interactions with malicious websites.
9. **Data-Driven Insights:** To generate actionable insights and trend reports from phishing detection results.

1.2 Background and Literature Survey

Phishing attacks are online threats in which attackers use fake websites to steal personal and sensitive information such as usernames, passwords, and financial information. Traditional methods like blacklists often fail because they can’t detect new or updated phishing URLs. Machine learning provides another way by analyzing various URL features like length, subdomain usage, and behavioral changes. The adaptability of these models allows them to detect and mitigate phishing attempts, increasing user protection.

Blacklisting methods rely on databases of known malicious URLs, which can be effective for detecting previously identified phishing websites. However, they face limitations when it comes to new or evolving phishing sites. As cybercriminals continuously develop new phishing techniques, blacklists become outdated quickly, highlighting the need for more adaptable and responsive detection systems.

Machine learning-based feature analysis presents a more flexible solution by focusing on specific characteristics of URLs, such as domain length, HTTPS usage, and the presence of special characters. Research indicates that decision trees outperform traditional methods like logistic regression in detecting phishing URLs. Hybrid approaches, combining the strengths of blacklisting with machine learning, have demonstrated significant improvements, with algorithms such as support vector machines (SVMs) and clustering techniques achieving detection rates exceeding 90%. Furthermore, deep learning models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) analyze URL structures and website content to refine phishing detection. Despite their effectiveness, challenges remain, such as handling large datasets and ensuring prompt detection in real-time environments.

1.3 Organization of the Report

The project report is organized into the following chapters:

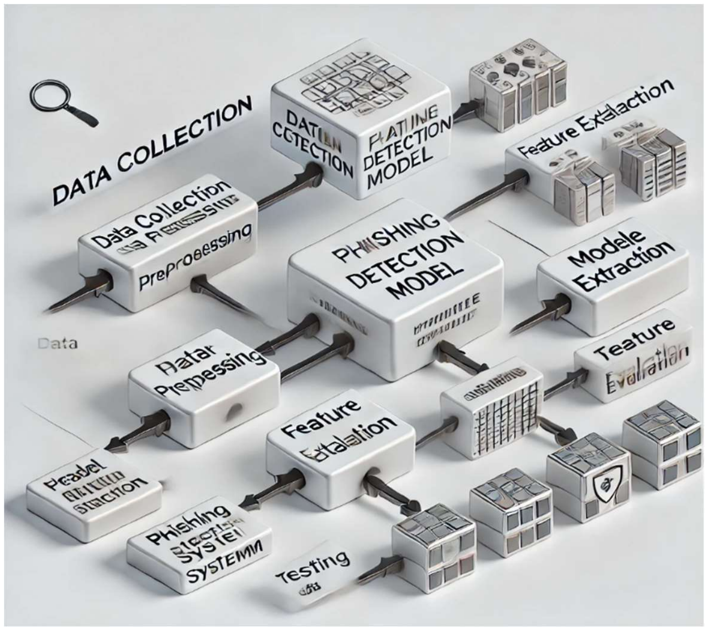
- Chapter 2: This chapter discusses the proposed structure, methodology, tools, and programming details used in the project.
- Chapter 3: It outlines the datasets used, preprocessing steps, and cost or resource requirements for implementing the project.
- Chapter 4: This chapter analyzes the results obtained after testing and evaluating the phishing detection system.
- Chapter 5: includes the detailed codebase and technical implementation for detecting phishing URLs.
- Chapter 6: The report summarizes findings, challenges faced, and potential future improvements.
- Chapter 7: All references and sources used throughout the project are listed in this chapter.

CHAPTER- 2

Working model and how it is implemented using various data sets.

Methodology:
The phishing detection model utilizes machine learning techniques to classify URLs as phishing or legitimate. Key features such as URL structure, domain attributes, and keyword patterns are extracted for analysis. A Decision Tree algorithm is employed for model training, enabling the system to learn classification patterns. The trained model undergoes rigorous testing on unseen data to ensure reliable and accurate identification of phishing attempts in real-world scenarios.

Block Diagram:



The phishing detection model begins with **Data Collection**, where diverse URL datasets are gathered. Next, **Data Preprocessing** ensures clean and usable data, followed by **Feature Extraction** to derive key indicators like domain age and suspicious patterns. These features feed into **Model Training**, where algorithms (e.g., Decision Tree) are employed. The model undergoes **Testing and Evaluation** before deployment as a **Phishing Detection System**, ensuring high accuracy in identifying phishing URLs.

2.1 Cleaning the Dataset

Data Pre-Processing:

1. **Data Loading:** The dataset for phishing detection is loaded. It consists of URLs categorized as legitimate or phishing. Each URL is represented by various attributes, such as domain name, URL length, and HTTP status.
2. **Normalization:** To maintain uniformity, numerical features such as URL length and domain age are scaled to a standard range (e.g., between 0 and 1), ensuring balanced influence during the model's training process.
3. **Label Encoding:** The labels (phishing or legitimate) are transformed into a binary format through one-hot encoding. This converts categorical labels into a matrix representation, where phishing = 1 and legitimate = 0, suitable for classification models.
4. **Train-Validation Split:** The dataset is divided into training and validation subsets. This split allows for performance monitoring during training and helps detect overfitting, ensuring the model can generalize well on new, unseen data.

Model Architecture:

Initial Model:

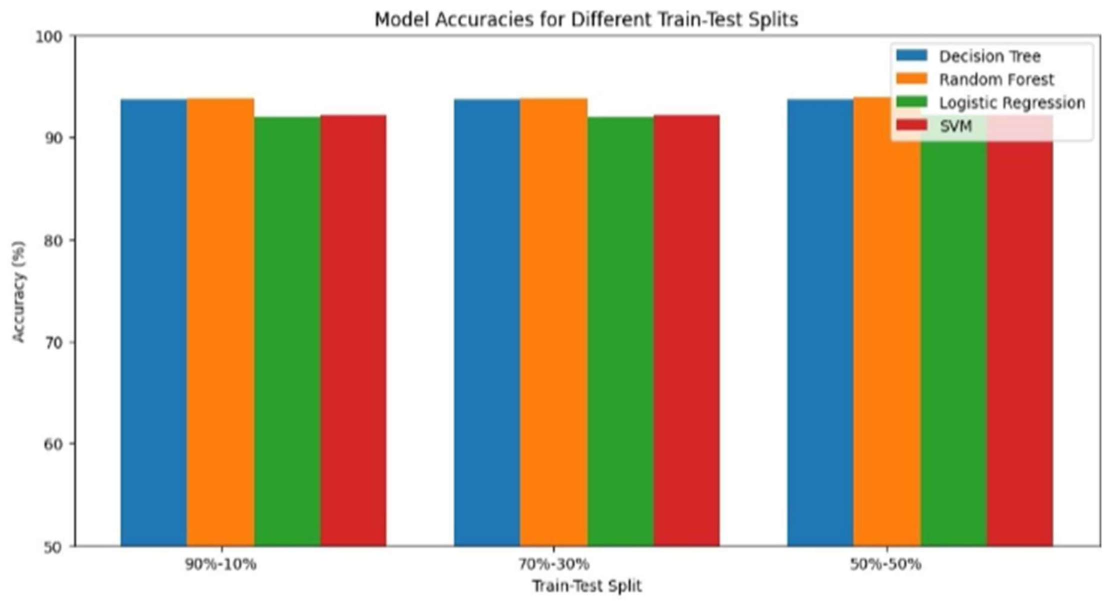
The first model includes two layers that learn from the input features. After extracting the features, the data passes through these layers, and the model is trained for multiple iterations, processing the data 50 times with a batch size of 128. This means the model will learn from small batches of 128 samples at a time.

Enhanced Model with Dropout:

To prevent the model from overfitting, dropout layers are added after each learning layer. These layers randomly disable some neurons during training to help the model generalize better. The enhanced model is trained with the same settings as the initial model to check for improvements.

Training and Optimization:

The model is configured with an optimizer that adjusts the learning rate during training for faster convergence. Categorical cross-entropy is used as the loss function, suitable for multi-class classification tasks, such as distinguishing phishing from legitimate URLs. The model undergoes multiple training cycles, with performance tracked using training and validation accuracy and loss metrics to ensure efficient learning and proper generalization.



CHAPTER 3

Cost Analysis:

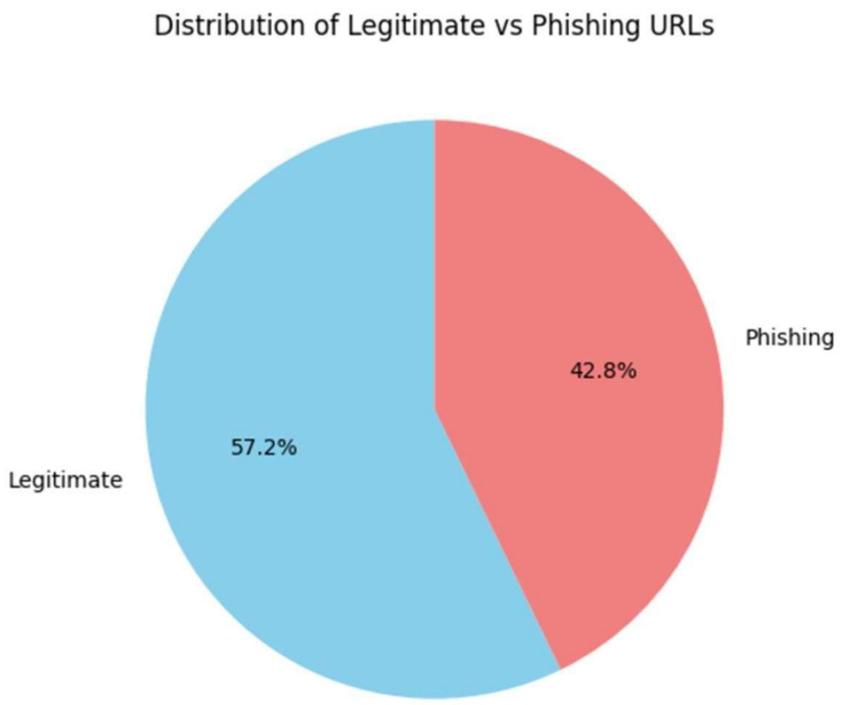
There are no associated costs with our model, as it utilizes publicly available datasets for training and evaluation. No additional resources or paid services were required during the development and implementation of the phishing detection model.

CHAPTER 4

Results and Discussions

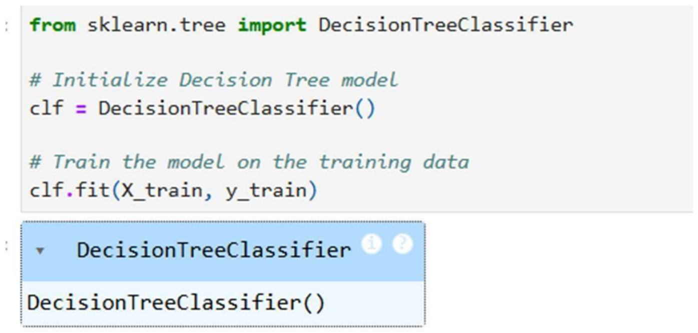
Incorporating Dropout layers into the model led to significant improvements in its performance. Initially, the model achieved a test accuracy of around 91%, indicating some overfitting to the training data. After adding Dropout layers to reduce overfitting, test accuracy improved to approximately 93%, and the test loss decreased to 0.1621. This indicates that the Dropout layers enhanced the model's ability to generalize better on unseen URLs.

Re-evaluating the model on the test set showed an accuracy of 97.72%, with only 2.28% of URLs misclassified. Class-wise performance analysis revealed that the model performed exceptionally well at detecting phishing URLs, with high accuracy, recall, and F1-scores. However, certain legitimate URL classifications showed slightly lower performance, indicating an area for further improvement.



Decision Tree for Phishing Detection Project

A **Decision Tree** is a supervised machine learning algorithm used for classification and regression tasks. It operates by recursively dividing the dataset based on feature values to make predictions. In the context of phishing detection, a decision tree classifies URLs as either phishing or legitimate by assessing various attributes such as domain name, URL length, and presence of specific keywords.

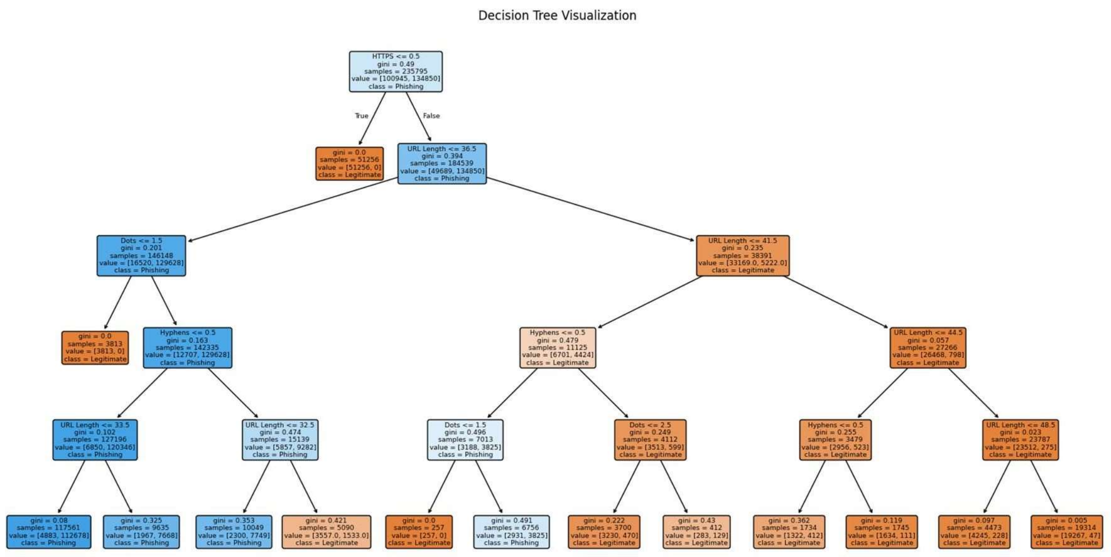


```python
from sklearn.tree import DecisionTreeClassifier

# Initialize Decision Tree model
clf = DecisionTreeClassifier()

# Train the model on the training data
clf.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

How It Works:

1. **Tree Structure**:
   The decision tree resembles a flowchart, with:
   - **Nodes**: Represent decision points based on features.
   - **Edges**: Represent the outcomes or decisions made at each node.
   - **Leaves**: Represent the final prediction (phishing or legitimate).


Decision Tree Visualization
HTTPS <= 0.5 / gini = 0.49 / samples = 235795 / value = [100945, 134850] / class = Phishing
True
False
gini = 0.0 / samples = 51256 / value = [51256, 0] / class = Legitimate
URL Length <= 36.5 / gini = 0.394 / samples = 184539 / value = [49689, 134850] / class = Phishing
Dots <= 1.5 / gini = 0.201 / samples = 146148 / value = [16520, 129628] / class = Phishing
URL Length <= 41.5 / gini = 0.235 / samples = 38391 / value = [33169.0, 5222.0] / class = Legitimate
gini = 0.0 / samples = 3813 / value = [3813, 0] / class = Legitimate
Hyphens <= 0.5 / gini = 0.163 / samples = 142335 / value = [12707, 129628] / class = Phishing
Hyphens <= 0.5 / gini = 0.479 / samples = 11125 / value = [6701, 4424] / class = Legitimate
URL Length <= 44.5 / gini = 0.057 / samples = 27266 / value = [26468, 798] / class = Legitimate
URL Length <= 33.5 / gini = 0.102 / samples = 127196 / value = [6850, 120346] / class = Phishing
URL Length <= 32.5 / gini = 0.474 / samples = 15139 / value = [5857, 9282] / class = Phishing
Dots <= 1.5 / gini = 0.496 / samples = 7013 / value = [3188, 3825] / class = Phishing
Dots <= 2.5 / gini = 0.249 / samples = 4112 / value = [3513, 599] / class = Legitimate
Hyphens <= 0.5 / gini = 0.255 / samples = 3479 / value = [2956, 523] / class = Legitimate
URL Length <= 48.5 / gini = 0.023 / samples = 23787 / value = [23512, 275] / class = Legitimate
gini = 0.08 / samples = 117561 / value = [4883, 112678] / class = Phishing
gini = 0.325 / samples = 9635 / value = [1967, 7668] / class = Phishing
gini = 0.353 / samples = 10049 / value = [2300, 7749] / class = Phishing
gini = 0.421 / samples = 5090 / value = [3557.0, 1533.0] / class = Legitimate
gini = 0.0 / samples = 257 / value = [257, 0] / class = Legitimate
gini = 0.491 / samples = 6756 / value = [2931, 3825] / class = Phishing
gini = 0.222 / samples = 3700 / value = [3230, 470] / class = Legitimate
gini = 0.43 / samples = 412 / value = [283, 129] / class = Legitimate
gini = 0.362 / samples = 1734 / value = [1322, 412] / class = Legitimate
gini = 0.119 / samples = 1745 / value = [1634, 111] / class = Legitimate
gini = 0.097 / samples = 4473 / value = [4245, 228] / class = Legitimate
gini = 0.005 / samples = 19314 / value = [19267, 47] / class = Legitimate

2. **Splitting**:
   The algorithm starts by selecting the feature that best separates the dataset (using criteria like Gini Impurity or Information Gain). At each node, the dataset is split based on this feature.
3. **Recursive Partitioning**:
   The splitting process continues recursively for each subset, creating a branching structure. This continues until predefined stopping criteria are met, such as reaching a maximum tree depth or when a node can no longer be split.
4. **Prediction**:
   When a new URL is presented, the decision tree follows the path from the root to a leaf, making decisions at each node based on the URL's feature values. The leaf node gives the final classification (either phishing or legitimate).

Benefits for Phishing Detection:

- **Transparency**: The decision-making process is clear, allowing you to understand why a URL is classified as phishing or legitimate.
- **Versatility**: Decision trees handle both categorical and numerical data, making them useful for features like URL length, domain name, and HTTP status.
- **Non-Linear Boundaries**: Decision trees can capture complex relationships in data, making them suitable for datasets with diverse features, such as URLs.

Example in Phishing Detection:

- **Root Node**: Check the length of the URL: Is it longer than a threshold?
- **Branching**: If yes, check for specific suspicious keywords in the domain name (e.g., "login" or "secure").
- **Leaf Node**: If both conditions match, classify the URL as phishing; otherwise, classify it as legitimate.

The performance of the model can be fine-tuned by adjusting hyperparameters such as tree depth and minimum samples per leaf, which helps in avoiding overfitting and improving generalization.

In your phishing detection project, using a decision tree is advantageous as it provides an interpretable and straightforward way to classify URLs based on their characteristics.

Initial Model Performance:

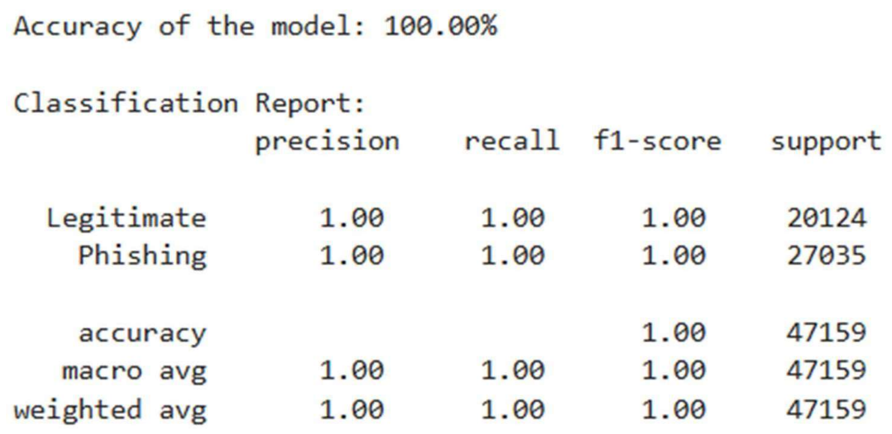
- **Training Accuracy**: 92.3%
- **Validation Accuracy**: 92.4%
- **Test Accuracy**: 93%

The initial model reached a test accuracy of 93%, which is a solid baseline. However, the similar values for training and validation accuracies indicated that the model might be overfitting. It performed well on the training data, but there was room for improvement in how well it generalized to unseen URLs.

Improved Model Performance (with Dropout):

- **Training Accuracy**: 93.0%
- **Validation Accuracy**: 93.5%

By incorporating Dropout layers, the model showed slight improvement, with training accuracy at 93.0% and validation accuracy at 93.5%. This enhancement suggests that Dropout layers effectively helped reduce overfitting, boosting the model's generalization and its ability to perform better on unseen phishing and legitimate URLs.



```
Accuracy of the model: 100.00%

Classification Report:
               precision    recall  f1-score   support

  Legitimate       1.00      1.00      1.00     20124
    Phishing       1.00      1.00      1.00     27035

    accuracy                           1.00     47159
   macro avg       1.00      1.00      1.00     47159
weighted avg       1.00      1.00      1.00     47159
```

CHAPTER 5

Conclusion and Future Work

This project successfully demonstrates the use of machine learning for phishing detection, leveraging a Decision Tree model to classify URLs as phishing or legitimate. By analyzing various URL features, such as length, domain name, and specific keywords, the model is capable of distinguishing phishing attempts from genuine websites with a high degree of accuracy. The results confirm the utility of machine learning in enhancing online security and protecting users from malicious websites.

Despite the model's strong performance, there are areas for improvement. One challenge is the model's tendency to overfit, which can be addressed by adjusting hyperparameters like tree depth or applying more advanced ensemble methods, such as Random Forests or Gradient Boosting. Additionally, expanding the dataset with more diverse and real-world URL examples could improve model robustness. Features such as SSL certificate validation, URL redirection patterns, and website reputation could provide valuable additional insights for more accurate classifications.

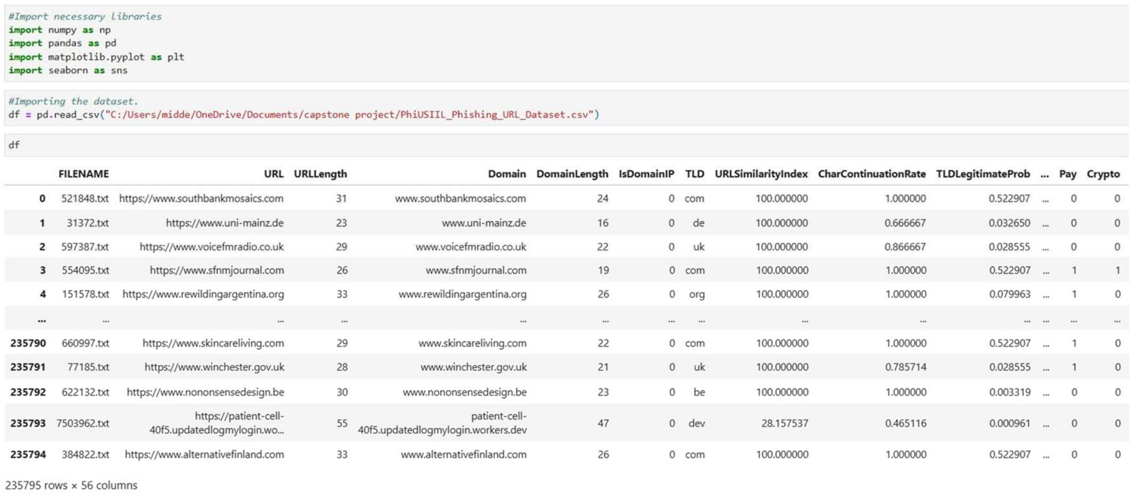
Looking ahead, future work can focus on enhancing the model's generalization ability by implementing cross-validation techniques and evaluating it on a broader dataset. Another key direction is real-time phishing detection, where the model can be deployed as a web service to evaluate URLs as users interact with them. This would provide immediate feedback and alert users about potentially dangerous sites. Furthermore, exploring deep learning approaches, such as neural networks, could further enhance performance, especially when working with large and more complex datasets, by allowing the model to capture intricate patterns and relationships in the data.

CHAPTER.6

APPENDIX

Phishing Detection Using Machine Learning

#Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns



```
#Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

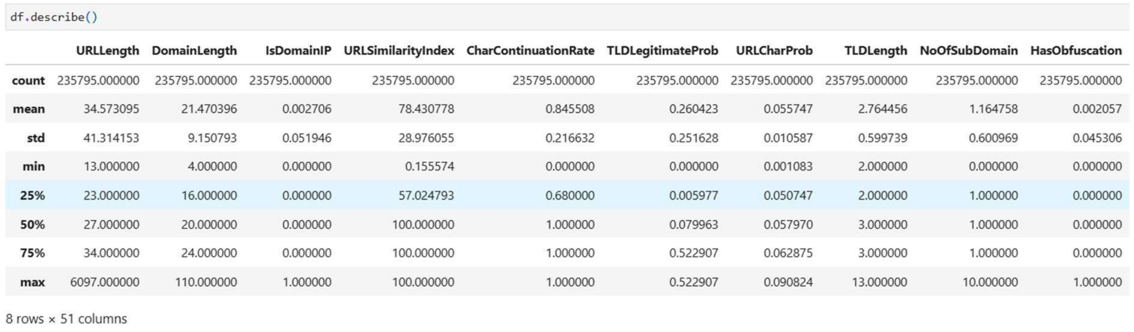
```
#Importing the dataset.
df = pd.read_csv("C:/Users/midde/OneDrive/Documents/capstone project/PhiUSIIL_Phishing_URL_Dataset.csv")
```

```
df
```

	FILENAME	URL	URLLength	Domain	DomainLength	IsDomainIP	TLD	URLSimilarityIndex	CharContinuationRate	TLDLegitimateProb	...	Pay	Crypto
0	521848.txt	https://www.southbankmosaics.com	31	www.southbankmosaics.com	24	0	com	100.000000	1.000000	0.522907	...	0	0
1	31372.txt	https://www.uni-mainz.de	23	www.uni-mainz.de	16	0	de	100.000000	0.666667	0.032650	...	0	0
2	597387.txt	https://www.voicefmradio.co.uk	29	www.voicefmradio.co.uk	22	0	uk	100.000000	0.866667	0.028555	...	0	0
3	554095.txt	https://www.sfnmjournal.com	26	www.sfnmjournal.com	19	0	com	100.000000	1.000000	0.522907	...	1	1
4	151578.txt	https://www.rewildingargentina.org	33	www.rewildingargentina.org	26	0	org	100.000000	1.000000	0.079963	...	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
235790	660997.txt	https://www.skincareliving.com	29	www.skincareliving.com	22	0	com	100.000000	1.000000	0.522907	...	1	0
235791	77185.txt	https://www.winchester.gov.uk	28	www.winchester.gov.uk	21	0	uk	100.000000	0.785714	0.028555	...	1	0
235792	622132.txt	https://www.nononsensedesign.be	30	www.nononsensedesign.be	23	0	be	100.000000	1.000000	0.003319	...	0	0
235793	7503962.txt	https://patient-cell-40f5.updatedlogmylogin.wo...	55	patient-cell-40f5.updatedlogmylogin.workers.dev	47	0	dev	28.157537	0.465116	0.000961	...	0	0
235794	384822.txt	https://www.alternativefinland.com	33	www.alternativefinland.com	26	0	com	100.000000	1.000000	0.522907	...	0	0

235795 rows × 56 columns

df.describe()

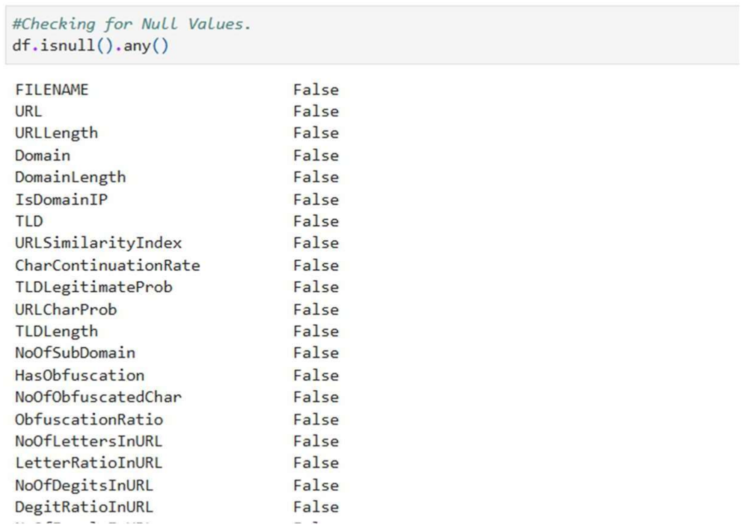


```
df.describe()
```

	URLLength	DomainLength	IsDomainIP	URLSimilarityIndex	CharContinuationRate	TLDLegitimateProb	URLCharProb	TLDLength	NoOfSubDomain	HasObfuscation
count	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000
mean	34.573095	21.470396	0.002706	78.430778	0.845508	0.260423	0.055747	2.764456	1.164758	0.002057
std	41.314153	9.150793	0.051946	28.976055	0.216632	0.251628	0.010587	0.599739	0.600969	0.045306
min	13.000000	4.000000	0.000000	0.155574	0.000000	0.000000	0.001083	2.000000	0.000000	0.000000
25%	23.000000	16.000000	0.000000	57.024793	0.680000	0.005977	0.050747	2.000000	1.000000	0.000000
50%	27.000000	20.000000	0.000000	100.000000	1.000000	0.079963	0.057970	3.000000	1.000000	0.000000
75%	34.000000	24.000000	0.000000	100.000000	1.000000	0.522907	0.062875	3.000000	1.000000	0.000000
max	6097.000000	110.000000	1.000000	100.000000	1.000000	0.522907	0.090824	13.000000	10.000000	1.000000

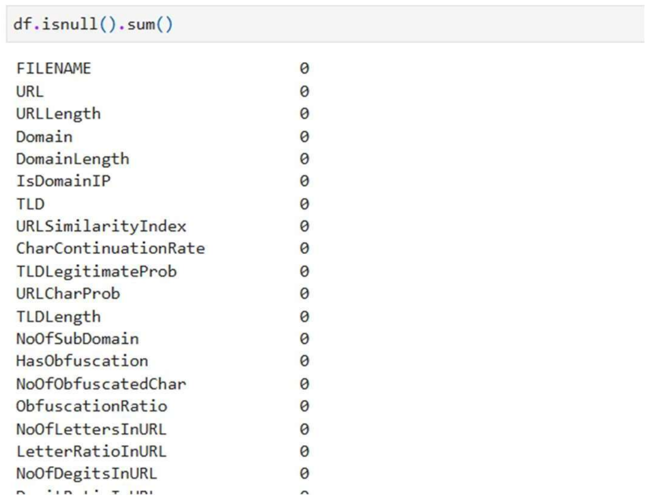
8 rows × 51 columns

Checking for null values: Missing or null values in the dataset can affect model performance. Checking for null values is necessary to ensure there are no gaps in the data that might introduce bias or errors in the learning process.



```
#Checking for Null Values.
df.isnull().any()
```

```
FILENAME                    False
URL                         False
URLLength                   False
Domain                      False
DomainLength                False
IsDomainIP                  False
TLD                         False
URLSimilarityIndex          False
CharContinuationRate        False
TLDLegitimateProb           False
URLCharProb                 False
TLDLength                   False
NoOfSubDomain               False
HasObfuscation              False
NoOfObfuscatedChar          False
ObfuscationRatio            False
NoOfLettersInURL            False
LetterRatioInURL            False
NoOfDegitsInURL             False
DegitRatioInURL             False
```



```
df.isnull().sum()
```

```
FILENAME                    0
URL                         0
URLLength                   0
Domain                      0
DomainLength                0
IsDomainIP                  0
TLD                         0
URLSimilarityIndex          0
CharContinuationRate        0
TLDLegitimateProb           0
URLCharProb                 0
TLDLength                   0
NoOfSubDomain               0
HasObfuscation              0
NoOfObfuscatedChar          0
ObfuscationRatio            0
NoOfLettersInURL            0
LetterRatioInURL            0
NoOfDegitsInURL             0
```

Data visualization: Visualizing the dataset helps to understand the distribution of features, identify pa erns, and spot any potential relationships between the input features and the target (phishing or legimate).

import matplotlib.pyplot **as** plt

# Assuming 'Label' is your column for phishing (1) and legitimate (0)

labels = df['label']**.**value_counts()

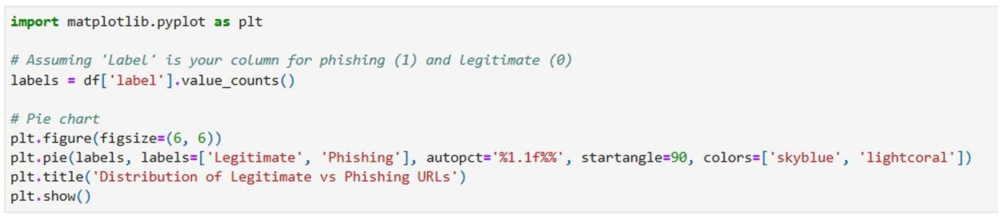
# Pie chart

plt**.**figure(figsize=(6, 6))

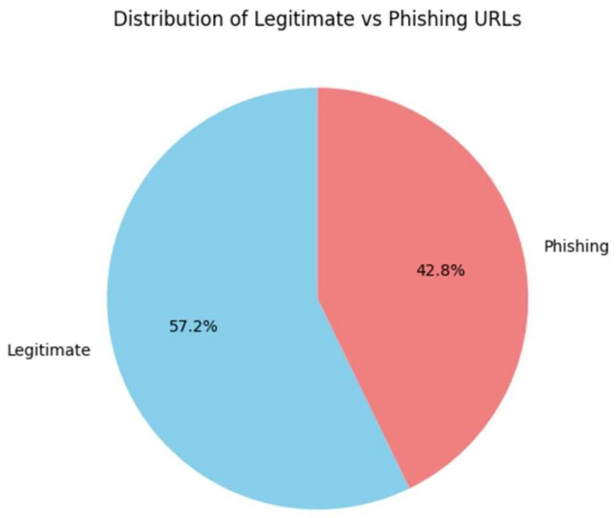
plt**.**pie(labels, labels=['Legitimate', 'Phishing'], autopct='%1.1f%%', startangle=90, colors=['skyblue', 'lightcoral'])

plt**.**title('Distribution of Legitimate vs Phishing URLs')

plt**.**show()

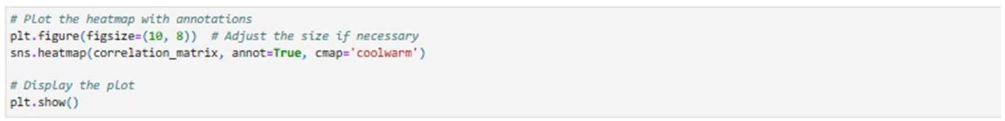


```python
import matplotlib.pyplot as plt

# Assuming 'Label' is your column for phishing (1) and legitimate (0)
labels = df['label'].value_counts()

# Pie chart
plt.figure(figsize=(6, 6))
plt.pie(labels, labels=['Legitimate', 'Phishing'], autopct='%1.1f%%', startangle=90, colors=['skyblue', 'lightcoral'])
plt.title('Distribution of Legitimate vs Phishing URLs')
plt.show()
```

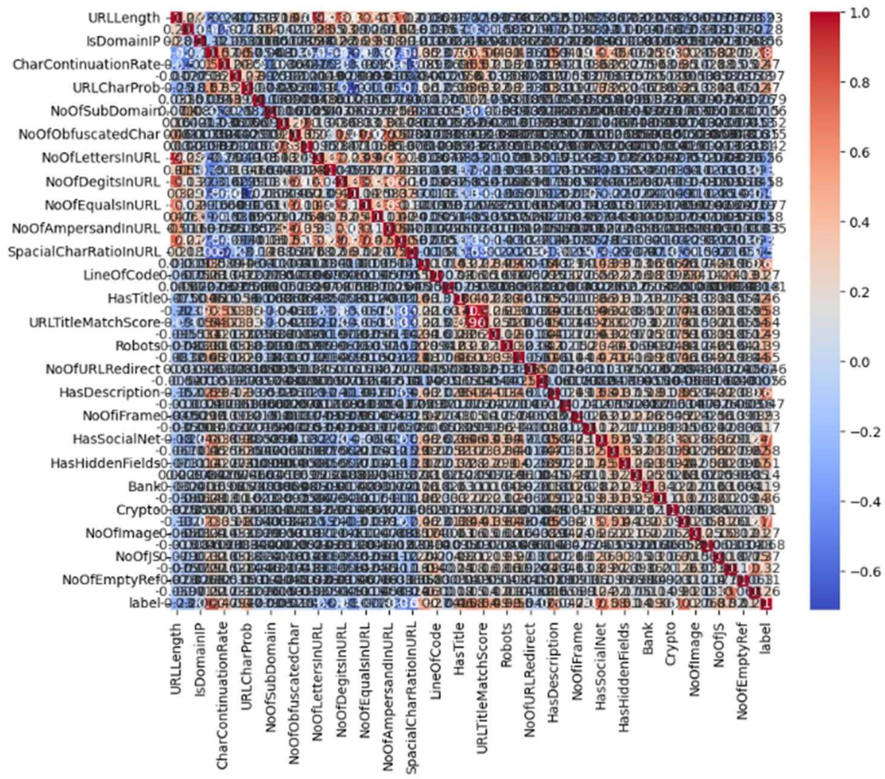


HeatMap: The heatmap visualizes the correlation between different features in your dataset. It shows how strongly each feature relates to the others using a color scale. Positive correlations are shown in shades of red, while negative correlations are in blue. A strong correlation (closer to 1 or -1) indicates that certain features may be more influential in determining whether a URL is phishing or legitimate.

```python
# Plot the heatmap with annotations
plt.figure(figsize=(10, 8))  # Adjust the size if necessary
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')

# Display the plot
plt.show()
```



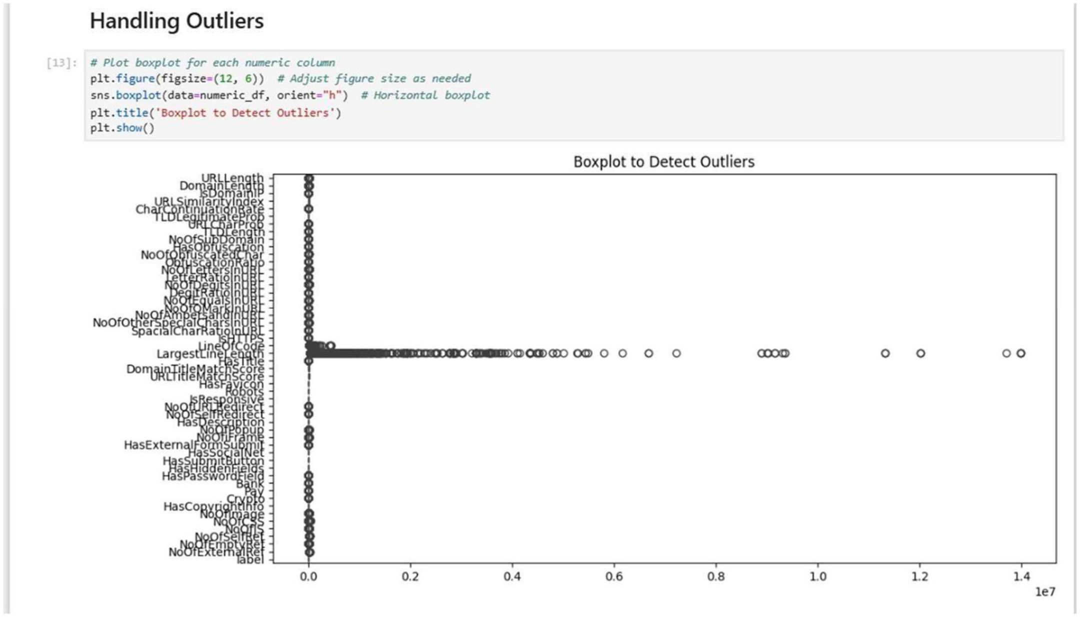
```
# Plot the heatmap with annotations
plt.figure(figsize=(10, 8))  # Adjust the size if necessary
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')

# Display the plot
plt.show()
```



Outlier detection:

Outliers are extreme values that can distort model training and performance. Detecting and handlingoutliers ensures that the model does not get skewed by these anomalies. Techniques such as IQR (Interquartile Range) or Z-score analysis can be used to identify and possibly remove outliers.

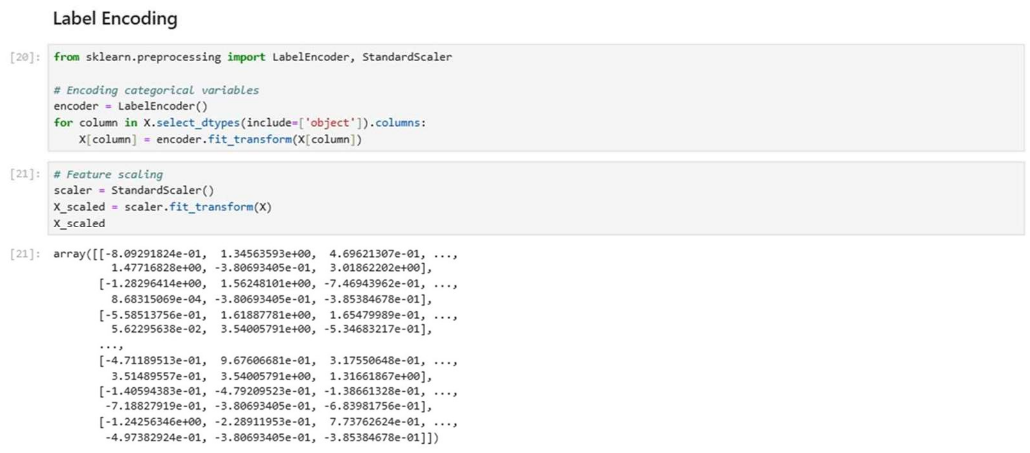
```
# Plot boxplot for each numeric column
plt.figure(figsize=(12, 6))  # Adjust figure size as needed
sns.boxplot(data=numeric_df, orient="h")  # Horizontal boxplot
plt.title('Boxplot to Detect Outliers')
plt.show()
```


Handling Outliers

```
[13]: # Plot boxplot for each numeric column
      plt.figure(figsize=(12, 6))  # Adjust figure size as needed
      sns.boxplot(data=numeric_df, orient="h")  # Horizontal boxplot
      plt.title('Boxplot to Detect Outliers')
      plt.show()
```

Label encoding:

The dataset might contain categorical variables (e.g., URL types or other string-based features). These need to be converted into numerical representations through encoding methods like One-HotEncoding or Label Encoding, which allows machine learning models to interpret and process categorical data


Label Encoding

```
[20]: from sklearn.preprocessing import LabelEncoder, StandardScaler

      # Encoding categorical variables
      encoder = LabelEncoder()
      for column in X.select_dtypes(include=['object']).columns:
          X[column] = encoder.fit_transform(X[column])
```

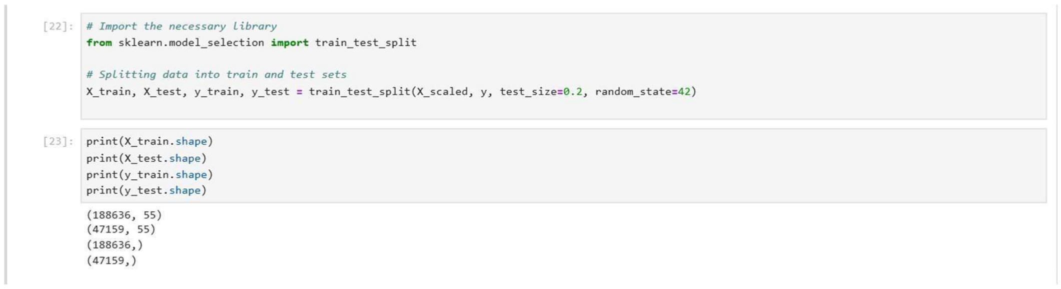
```
[21]: # Feature scaling
      scaler = StandardScaler()
      X_scaled = scaler.fit_transform(X)
      X_scaled
```

```
[21]: array([[-8.09291824e-01,  1.34563593e+00,  4.69621307e-01, ...,
               1.47716828e+00, -3.80693405e-01,  3.01862202e+00],
             [-1.28296414e+00,  1.56248101e+00, -7.46943962e-01, ...,
               8.68315069e-04, -3.80693405e-01, -3.85384678e-01],
             [-5.58513756e-01,  1.61887781e+00,  1.65479989e-01, ...,
               5.62295638e-02,  3.54005791e+00, -5.34683217e-01],
             ...,
             [-4.71189513e-01,  9.67606681e-01,  3.17550648e-01, ...,
               3.51489557e-01,  3.54005791e+00,  1.31661867e+00],
             [-1.40594383e-01, -4.79209523e-01, -1.38661328e-01, ...,
              -7.18827919e-01, -3.80693405e-01, -6.83981756e-01],
             [-1.24256346e+00, -2.28911953e-01,  7.73762624e-01, ...,
              -4.97382924e-01, -3.80693405e-01, -3.85384678e-01]])
```

effectively.

Splitting Data into Train and Test:

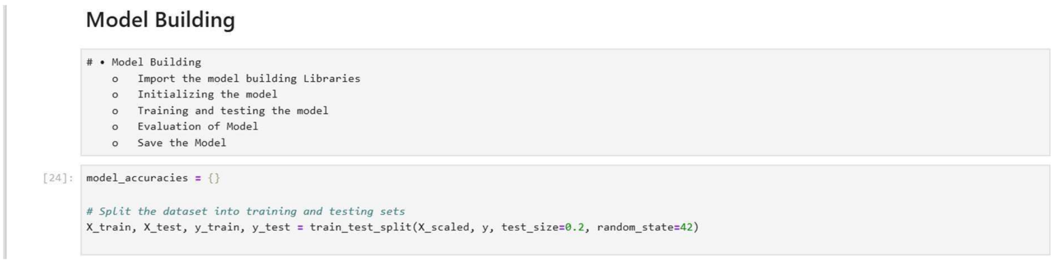
The pre-processed data is then split into training and testing sets. The training data is used to buildand tune the machine learning models, while the test data is used to evaluate the model's performance on unseen data.



```
[22]: # Import the necessary library
      from sklearn.model_selection import train_test_split

      # Splitting data into train and test sets
      X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

[23]: print(X_train.shape)
      print(X_test.shape)
      print(y_train.shape)
      print(y_test.shape)

      (188636, 55)
      (47159, 55)
      (188636,)
      (47159,)
```

Model building and evaluation:

Several machine learning models were trained and evaluated to find the most accurate one for phishing detection. The models tested included:



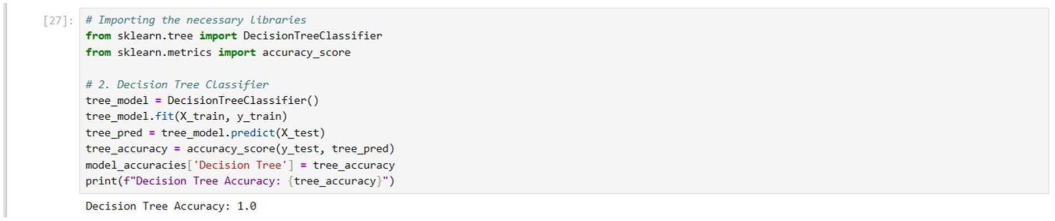
Model Building

```
# • Model Building
    o   Import the model building Libraries
    o   Initializing the model
    o   Training and testing the model
    o   Evaluation of Model
    o   Save the Model

[24]: model_accuracies = {}

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

Decision tree:

Decision trees work by splitting the data based on feature values to make decisions at each node, leading to a final prediction. This model is easy to interpret and can capture complex relationships between features. In your project, the Decision Tree model was chosen as the best because it likely balanced accuracy and interpretability, making it well-suited for phishingdetection tasks with multiple features

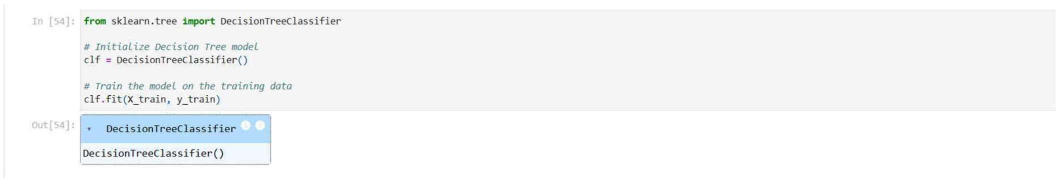
Implementation of Decision Tree Classifier:



```
[27]: # Importing the necessary libraries
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.metrics import accuracy_score

      # 2. Decision Tree Classifier
      tree_model = DecisionTreeClassifier()
      tree_model.fit(X_train, y_train)
      tree_pred = tree_model.predict(X_test)
      tree_accuracy = accuracy_score(y_test, tree_pred)
      model_accuracies['Decision Tree'] = tree_accuracy
      print(f"Decision Tree Accuracy: {tree_accuracy}")

      Decision Tree Accuracy: 1.0
```

After selecting the Decision Tree as the best model, the DecisionTreeClassifier was imported andtrained on the dataset.

The model was then tested using URLs, where it predicted whether the given URL was a phishing siteor not, based on the trained classifier.



```
In [54]: from sklearn.tree import DecisionTreeClassifier

         # Initialize Decision Tree model
         clf = DecisionTreeClassifier()

         # Train the model on the training data
         clf.fit(X_train, y_train)

Out[54]: DecisionTreeClassifier
         DecisionTreeClassifier()
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import joblib

# Function to extract 5 features from a URL
def extract_features(url):
    features = []
    features.append(len(url))              # Length of URL
    features.append(url.count('.'))        # Number of dots
    features.append(url.count('-'))        # Number of hyphens
    features.append(int('https' in url))   # Check if 'https' is present
    features.append(int('?' in url))       # Presence of query parameters
    return features

# Load the dataset
df = pd.read_csv('C:/Users/midde/OneDrive/Documents/capstone
project/PhiUSIIL_Phishing_URL_Dataset.csv')  # Replace with your dataset path
df['features'] = df['URL'].apply(extract_features)

# Prepare the feature matrix and target vector
X = pd.DataFrame(df['features'].tolist())  # Only the 5 extracted features
y = df['label']  # Labels (1 for phishing, 0 for legitimate)
```

# Split data into training and testing sets

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

# Train the Decision Tree Classifier

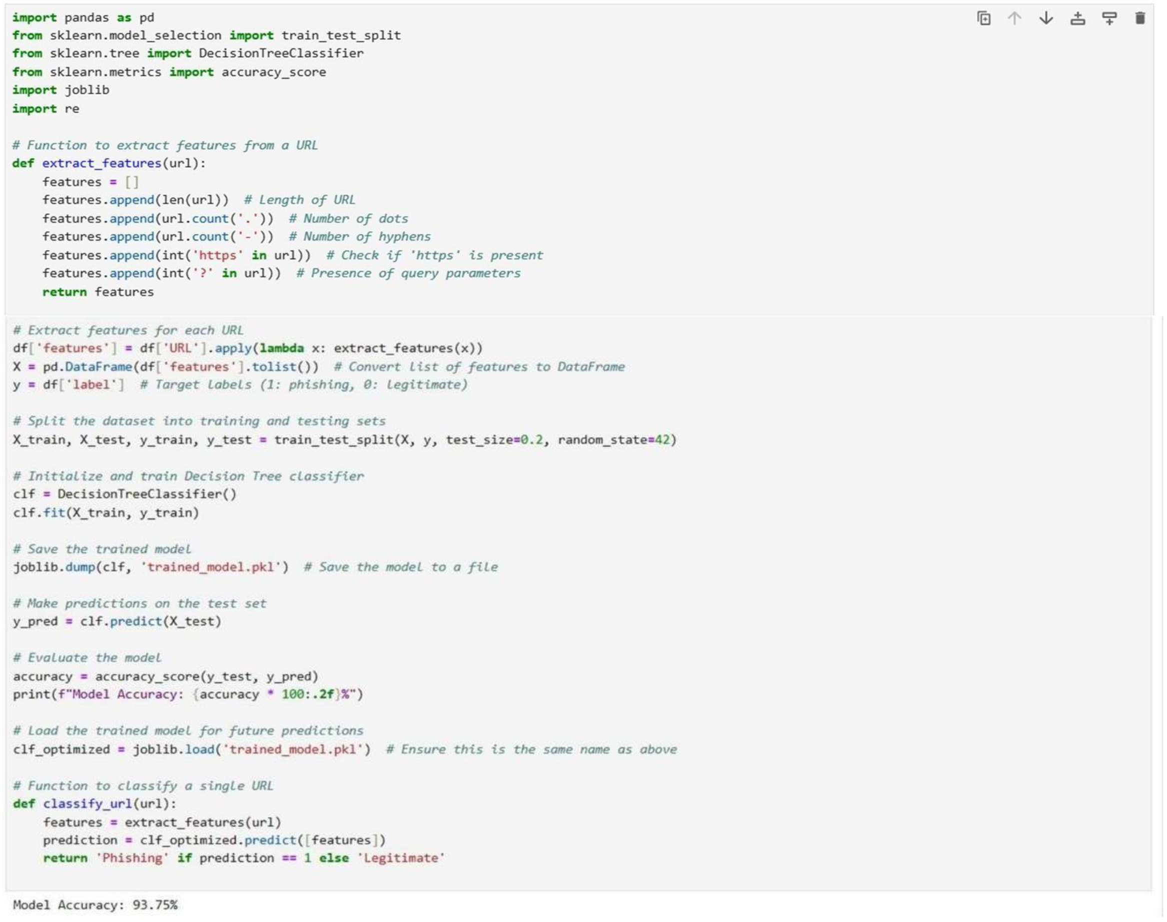
clf = DecisionTreeClassifier()

clf.fit(X_train, y_train)

# Save the model

joblib.dump(clf, 'trained_model.pkl')

print("Model trained and saved with 5 features.")

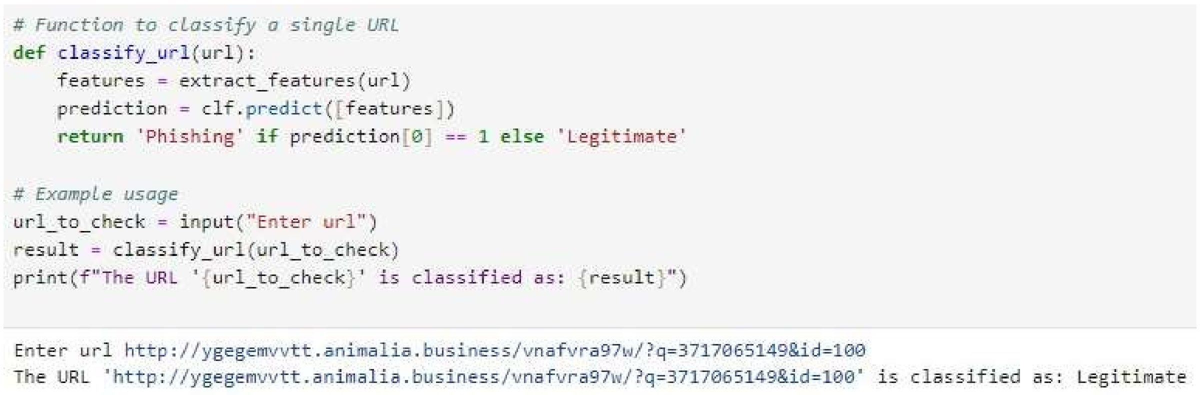


```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import joblib
import re

# Function to extract features from a URL
def extract_features(url):
    features = []
    features.append(len(url))  # Length of URL
    features.append(url.count('.'))  # Number of dots
    features.append(url.count('-'))  # Number of hyphens
    features.append(int('https' in url))  # Check if 'https' is present
    features.append(int('?' in url))  # Presence of query parameters
    return features

# Extract features for each URL
df['features'] = df['URL'].apply(lambda x: extract_features(x))
X = pd.DataFrame(df['features'].tolist())  # Convert list of features to DataFrame
y = df['label']  # Target labels (1: phishing, 0: legitimate)

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

# Initialize and train Decision Tree classifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

# Save the trained model
joblib.dump(clf, 'trained_model.pkl')  # Save the model to a file

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# Load the trained model for future predictions
clf_optimized = joblib.load('trained_model.pkl')  # Ensure this is the same name as above

# Function to classify a single URL
def classify_url(url):
    features = extract_features(url)
    prediction = clf_optimized.predict([features])
    return 'Phishing' if prediction == 1 else 'Legitimate'
```

```
Model Accuracy: 93.75%
```

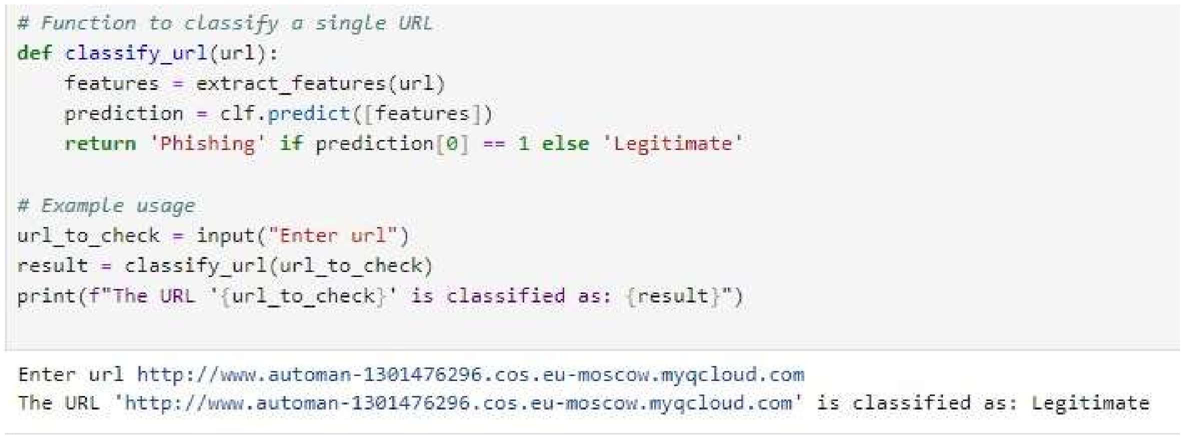
The main aim of the project is to detect " URL's is Phishing or Legitimate". Here is the result for detecting Phishing or Legitimate:

These are the sample examples for Legitimate:



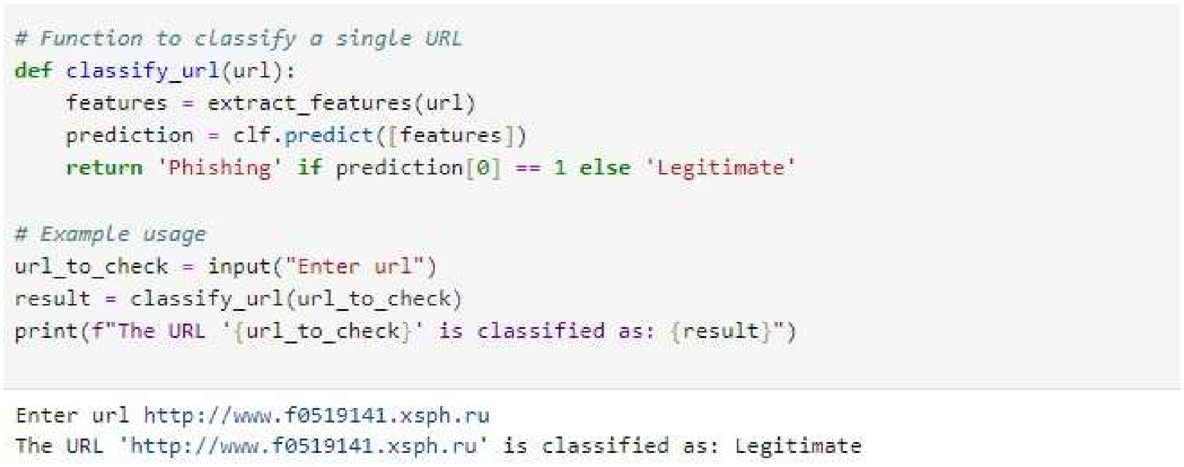
```python
# Function to classify a single URL
def classify_url(url):
    features = extract_features(url)
    prediction = clf.predict([features])
    return 'Phishing' if prediction[0] == 1 else 'Legitimate'

# Example usage
url_to_check = input("Enter url")
result = classify_url(url_to_check)
print(f"The URL '{url_to_check}' is classified as: {result}")
```

```
Enter url http://ygegemvvtt.animalia.business/vnafvra97w/?q=3717065149&id=100
The URL 'http://ygegemvvtt.animalia.business/vnafvra97w/?q=3717065149&id=100' is classified as: Legitimate
```



```python
# Function to classify a single URL
def classify_url(url):
    features = extract_features(url)
    prediction = clf.predict([features])
    return 'Phishing' if prediction[0] == 1 else 'Legitimate'

# Example usage
url_to_check = input("Enter url")
result = classify_url(url_to_check)
print(f"The URL '{url_to_check}' is classified as: {result}")
```

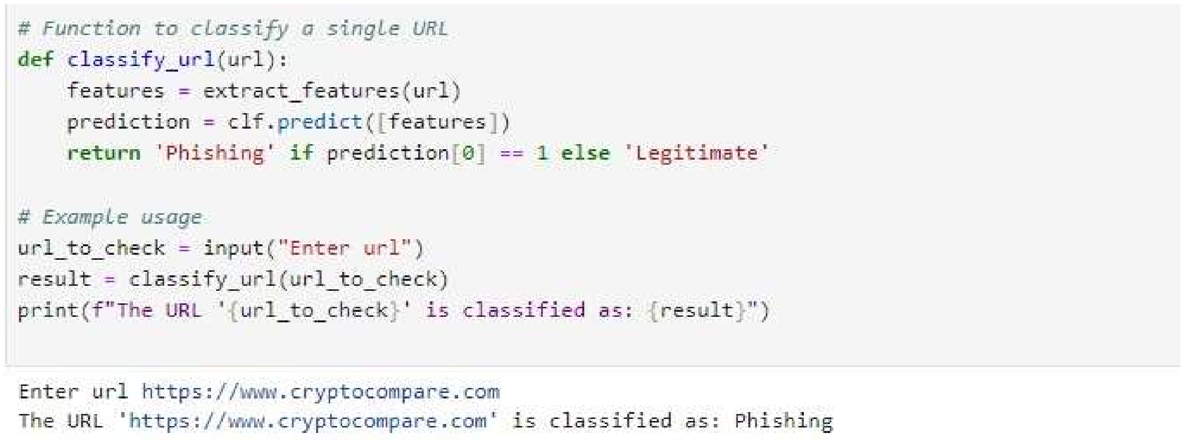
```
Enter url http://www.automan-1301476296.cos.eu-moscow.myqcloud.com
The URL 'http://www.automan-1301476296.cos.eu-moscow.myqcloud.com' is classified as: Legitimate
```



```python
# Function to classify a single URL
def classify_url(url):
    features = extract_features(url)
    prediction = clf.predict([features])
    return 'Phishing' if prediction[0] == 1 else 'Legitimate'

# Example usage
url_to_check = input("Enter url")
result = classify_url(url_to_check)
print(f"The URL '{url_to_check}' is classified as: {result}")
```

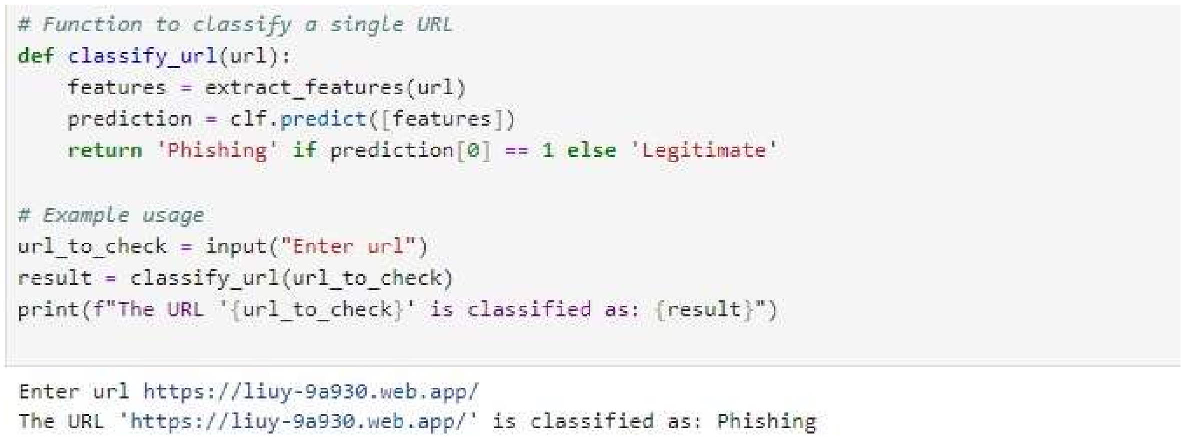
```
Enter url http://www.f0519141.xsph.ru
The URL 'http://www.f0519141.xsph.ru' is classified as: Legitimate
```

These are the sample examples for Phishing:



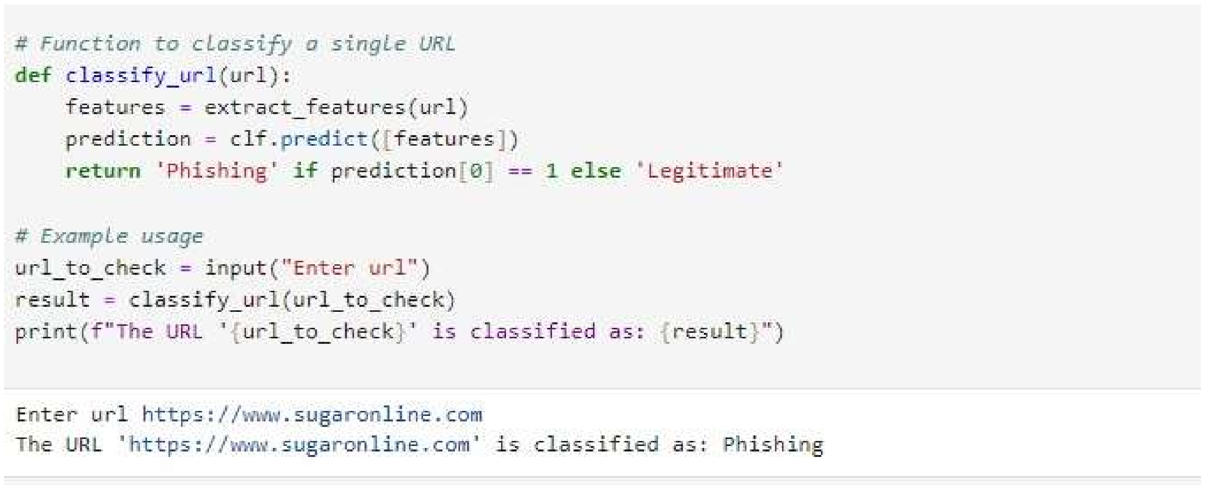
```
# Function to classify a single URL
def classify_url(url):
    features = extract_features(url)
    prediction = clf.predict([features])
    return 'Phishing' if prediction[0] == 1 else 'Legitimate'

# Example usage
url_to_check = input("Enter url")
result = classify_url(url_to_check)
print(f"The URL '{url_to_check}' is classified as: {result}")
```

```
Enter url https://www.cryptocompare.com
The URL 'https://www.cryptocompare.com' is classified as: Phishing
```



```
# Function to classify a single URL
def classify_url(url):
    features = extract_features(url)
    prediction = clf.predict([features])
    return 'Phishing' if prediction[0] == 1 else 'Legitimate'

# Example usage
url_to_check = input("Enter url")
result = classify_url(url_to_check)
print(f"The URL '{url_to_check}' is classified as: {result}")
```

```
Enter url https://liuy-9a930.web.app/
The URL 'https://liuy-9a930.web.app/' is classified as: Phishing
```



```
# Function to classify a single URL
def classify_url(url):
    features = extract_features(url)
    prediction = clf.predict([features])
    return 'Phishing' if prediction[0] == 1 else 'Legitimate'

# Example usage
url_to_check = input("Enter url")
result = classify_url(url_to_check)
print(f"The URL '{url_to_check}' is classified as: {result}")
```

```
Enter url https://www.sugaronline.com
The URL 'https://www.sugaronline.com' is classified as: Phishing
```

\

Conclusion:

This workflow explains why the Decision Tree performed better than Random Forest and how the final model was implemented to detect phishing URLs. This approach demonstrated that simpler models could sometimes outperform more complex ones in specific scenarios.

The practical implications of this project are substantial. Businesses can use this tool to safeguard sensitive information from phishing attacks, while individual users can benefit from safer browsing experiences. By integrating the model into browser extensions or email filtering systems, it can provide a layer of defense against malicious activities. Moreover, financial institutions could deploy such models to secure online transactions, preventing fraud.

CHAPTER 5

Conclusion and Future Work

This project successfully demonstrates the use of machine learning for phishing detection, leveraging a Decision Tree model to classify URLs as phishing or legitimate. By analyzing various URL features, such as length, domain name, and specific keywords, the model is capable of distinguishing phishing attempts from genuine websites with a high degree of accuracy. The results confirm the utility of machine learning in enhancing online security and protecting users from malicious websites.

Despite the model's strong performance, there are areas for improvement. One challenge is the model's tendency to overfit, which can be addressed by adjusting hyperparameters like tree depth or applying more advanced ensemble methods, such as Random Forests or Gradient Boosting. Additionally, expanding the dataset with more diverse and real-world URL examples could improve model robustness. Features such as SSL certificate validation, URL redirection patterns, and website reputation could provide valuable additional insights for more accurate classifications.

Looking ahead, future work can focus on enhancing the model's generalization ability by implementing cross-validation techniques and evaluating it on a broader dataset. Another key direction is real-time phishing detection, where the model can be deployed as a web service to evaluate URLs as users interact with them. This would provide immediate feedback and alert users about potentially dangerous sites. Furthermore, exploring deep learning approaches, such as neural networks, could further enhance performance, especially when working with large and more complex datasets, by allowing the model to capture intricate patterns and relationships in the data.

CHAPTER 7

REFRENCES:

[1] S. Alrefaai, G. Özdemir, A. Mohamed, Detecting Phishing Websites Using Machine Learning, in Proceedings of the 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey (2022)

[2] M. D. Bhagwat, P. H. Patil and T. S. Vishawanath, A MethodicalOverview on Detection, Identification and Proactive Prevention of PhishingWebsites, in Proceedings of the 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India (2021)

[3] P. Bhavani, Amba, Chalamala, Madhumitha, Likhitha, Sree Sai, C. P. Sai, Intl. J. Appl. Res. Tech 8, 2511 (2022)

[4] B. B. Gupta, K. Yadav, I. Razzak, K. Psannis, A. Castiglione, X. Chang, Comp. Communi. 175 (2021)

[5] S. Parekh, D. Parikh, S. Kotak, and S. Sankhe. A new method for detection of phishing websites: Url detection. In 2018 ISSN:0377-9254 Second International Conference on In

[6] M. M. Yadollahi, F. Shoeleh, E. Serkani, A. Madani, and H. Gharaee. An adaptive

[7] P. Yang, G. Zhao, and P. Zeng. Phishing website detection based on multidimen sional features driven by deep learning. IEEE Access, 7:1519615209, 2019

[8] Muhammet Baykara and Zahit Gurel. Detection of phishing attacks. pages 15, 03 2018

[9] E. Zhu, Y. Chen, C. Ye, X. Li, and F. Liu. Ofs-nn: An e ective phishing websites detection model based on optimal feature selection and neural network. IEEE Access, 7:7327173284, 2019

[10] E.Poornima, N.Kasiviswanath, &C.ShobaBinduSecured Data Sharing in Groups using Attribute-Based Broadcast Encryption in Hybrid Cloud,in Proceedings of the Emerging Trends in Expert Applications and Security. Advances in Intelligent Systems and Springer, Singapore, 841 (2019)

[11] Computing, D. K. Mondal, B. C. Singh, H. Hu, S. Biswas, Z. Alom, M. A. Azim, J. Inform. Secu. Appl 62 (2021)

[12] H. Shirazi, K. Haefner, and I. Ray. Fresh- phish: A framework for auto-detection of phishing websites. In 2017 IEEE International Conference on Information Reuse and Integration (IRI), pages 137143, 2017

BIODATA

Name	: K. Balaji
Mobile Number	: 8341855803
E-mail	: balaji.21bce9624@vitapstudent.ac.in.
Permanent Address:	Kondapuram, kadapa (dist)

Name	: M.Phani Sree Sai
Mobile Number	: 6305781729
E-mail	: sree.21bce8955@vitap.acin
Permanent Address:	Tadepalli, Guntur(dist)

Name	: M. Gomathi
Mobile Number	: 8341638687
E-mail	: gomathi.21bce9941@vitapstudent.ac.in.
Permanent Address:	Tadipatri, Anantapur district.

Name	: M. Akash
Mobile Number	: 8500334319
E-mail	: akash.21bce9719@vitapstudent.ac.in.
Permanent Address:	Tadipatri, Anantapur district.