SCTR's Pune Institute of Computer Technology
Dhankawadi, Pune

AN INTERNSHIP REPORT ON

Malicious URL Detection

SUBMITTED BY

Name: Harsh Lambe

Class: TE04

Roll no: 31437

Under the guidance of

Dr. A. R. Deshpande

DEPARTMENT OF COMPUTER ENGINEERING

ACADEMIC YEAR 2023-24

DEPARTMENT OF COMPUTER ENGINEERING

SCTR's Pune Institute of Computer TechnologyDhankawadi, Pune Maharashtra 411043

CERTIFICATE

This is to certify that the SPPU Curriculum-based internship report entitled.

Submitted by
Harsh Sanjay Lambe
T190054365

has satisfactorily completed the curriculum-based internship under the guidance of Dr. *A. R. Deshpande towards* the partial fulfillment of third year Computer EngineeringSemester VI, Academic Year 2023-24 of Savitribai Phule Pune University.

Dr. A.R. Deshpande
Internship Guide
PICT, Pune

Dr. G. V. Kale
Head
Department of Computer Engineering
PICT, Pune

Place:
Date:

ACKNOWLEDGEMENT

It gives me great pleasure in presenting the internship report on Project: Malicious URL Detection.

First, I would like to take this opportunity to thank my internship guide Dr.A.R. Deshpande for giving me all the help and guidance needed. I am grateful for her kind support and valuable suggestions that proved to be beneficial in the overall completion of this internship.

I am thankful to our Head of the Computer Engineering Department, Dr. G.V. Kale, for her indispensable support and suggestions throughout the internship work.

I would also genuinely like to express my gratitude to the Department Internship Coordinator, Prof. P.P. Joshi, for her constant guidance and support and for the timely resolution of the doubts related to the internship process.

Contents

List of Figures

1 Title

Malicious URL Detection

2 Introduction

In today's digital landscape, the prevalence of cyber threats poses significant challenges to online security. Among these threats, malicious URLs stand out as a potent weapon in the arsenal of cybercriminals, capable of luring unsuspecting users into traps ranging from phishing scams to malware infections. Detecting and neutralizing these malicious URLs swiftly is crucial to safeguarding individuals, organizations, and entire networks against cyber attacks.

In response to this pressing need, this project endeavors to develop an advanced system for the automatic detection of malicious URLs using Python. By harnessing the power of machine learning algorithms, we aim to construct a robust solution capable of swiftly and accurately identifying malicious URLs, thereby mitigating potential risks and fortifying cyber defenses.

This introduction sets the stage by highlighting the significance of the issue and outlining the project's objective to tackle it using Python and machine learning techniques.
This report delves into the technical aspects of Malicious URl Detection , providing insights gained through my internship experience. It explores the various data schemas employed to represent students, teachers, classes, and subjects. The report details the functionalities for creating, reading, updating, and deleting data within these schemas. I'll further explore how Malicious URL Detection facilitates communication channels between teachers and students, enabling them to exchange messages and collaborate on assignments. Additionally, the report sheds light on the mechanisms for managing attendance, recording marks, and generating automated reports based on this data. This comprehensive overview equips readers with a thorough understanding of Edu_sync's capabilities and its potential to revolutionize educational management practices.

3 Problem Statement

The proliferation of malicious URLs poses a significant threat to cybersecurity, jeopardizing the integrity and safety of online systems and users' personal information. Traditional methods of URL classification often fall short in accurately identifying and flagging malicious links, leaving individuals and organizations vulnerable to cyber attacks.

The primary challenge lies in developing a proactive and effective solution for the rapid detection of malicious URLs. Existing approaches typically rely on manual analysis or simplistic rule-based systems, which are prone to false positives and may struggle to keep pace with the evolving tactics of cybercriminals.

4 Objectives and Scope

The primary objective of this project is to develop a comprehensive system using Python and machine learning techniques to automatically identify malicious URLs. This involves several key steps:

Data Collection and Preparation: Gathering a diverse dataset containing both benign and malicious URLs. The dataset needs to be sufficiently large and representative to ensure effective model training.

Feature Extraction: Extracting relevant features from the URLs that can help distinguish between benign and malicious links. These features may include domain reputation, URL length, presence of suspicious characters, and other attributes indicative of malicious intent.

Model Development: Implementing machine learning algorithms, such as decision trees, random forests, or neural networks, to develop a classification model. The model will be trained on the extracted features to learn patterns associated with malicious URLs.

. The scope of this project encompasses a multifaceted approach to the development of a robust system for the automatic detection of malicious URLs using Python and machine learning techniques. Key aspects within the scope include:

Firstly, the project will focus on collecting a diverse and comprehensive dataset of URLs, meticulously curated to encompass a wide range of benign and malicious links. This dataset will serve as the foundation for training and testing the detection model, ensuring its efficacy across various real-world scenarios.

Secondly, extensive feature extraction will be conducted to capture pertinent characteristics from the URLs. These features will include but are not limited to domain reputation, URL structure, presence of suspicious keywords, and other indicators indicative of malicious intent.

Thirdly, leveraging the power of Python programming language, the project will implement and fine-tune machine learning algorithms for classification tasks. Various techniques such as decision trees, random forests, and neural networks will be explored to develop a robust and accurate detection model.

5 Methodology Details

The methodology for this project comprises several key stages aimed at developing an efficient system for detecting malicious URLs. Initially, a diverse dataset of URLs will be gathered and meticulously preprocessed to ensure quality and balance between benign and malicious samples. Subsequently, features will be extracted from the URLs, encompassing domain-based attributes, URL characteristics, and content-based indicators of malicious intent. Feature selection techniques will then be employed to identify the most informative features for model training.

In the model development phase, various machine learning algorithms, including decision trees, random forests, gradient boosting machines, and neural networks, will be implemented and fine-tuned using hyperparameter optimization techniques. Models will be evaluated rigorously using standard performance metrics and cross-validation to ensure robustness and generalization.

Additionally, model interpretability will be prioritized through techniques such as feature importance analysis, SHAP values, and LIME to elucidate the reasoning behind predictions. The deployment and integration of the trained model will involve developing user-friendly interfaces or APIs for seamless integration into existing systems, coupled with monitoring mechanisms to track performance and adapt to evolving threats.

Throughout the project, meticulous documentation will be maintained, detailing the methodology, data collection procedures, feature engineering strategies, model development, and evaluation results. Clear and concise reporting will encapsulate the project's findings, insights, and recommendations for future enhancements, aiming to contribute to strengthened cybersecurity measures in combating malicious URLs.

6 Modern Engineering Tools Used

Integrated Development Environment (IDE):

PyCharm, Visual Studio Code, or Jupyter Notebooks: These IDEs provide a comprehensive environment for Python development, offering features such as code editing, debugging, and version control integration.

Version Control System (VCS):

Git: Git can be used for version control, allowing collaborative development and tracking changes to the codebase. Platforms like GitHub or GitLab can host the project repository and facilitate collaboration among team members.

Machine Learning Libraries:

Scikit-learn, TensorFlow, or PyTorch: These libraries offer various machine learning algorithms and tools for model development, training, and evaluation.

Data Analysis and Visualization Tools:

Pandas, NumPy, and Matplotlib/Seaborn: These libraries are essential for data manipulation, analysis, and visualization, enabling exploration of the dataset and extraction of relevant features.

Model Evaluation and Metrics:

Scikit-learn provides functions for evaluating models using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

Containerization:

Docker: Docker can be used to containerize the application, ensuring consistency and portability across different environments.

Continuous Integration/Continuous Deployment (CI/CD):

Jenkins, Travis CI, or GitHub Actions: These CI/CD tools can automate the build, test, and deployment processes, ensuring that changes are integrated smoothly and reliably.

Documentation:

Sphinx or MkDocs: These tools can generate documentation from code comments and markdown files, ensuring that project documentation remains up-to-date and accessible.

Project Management:

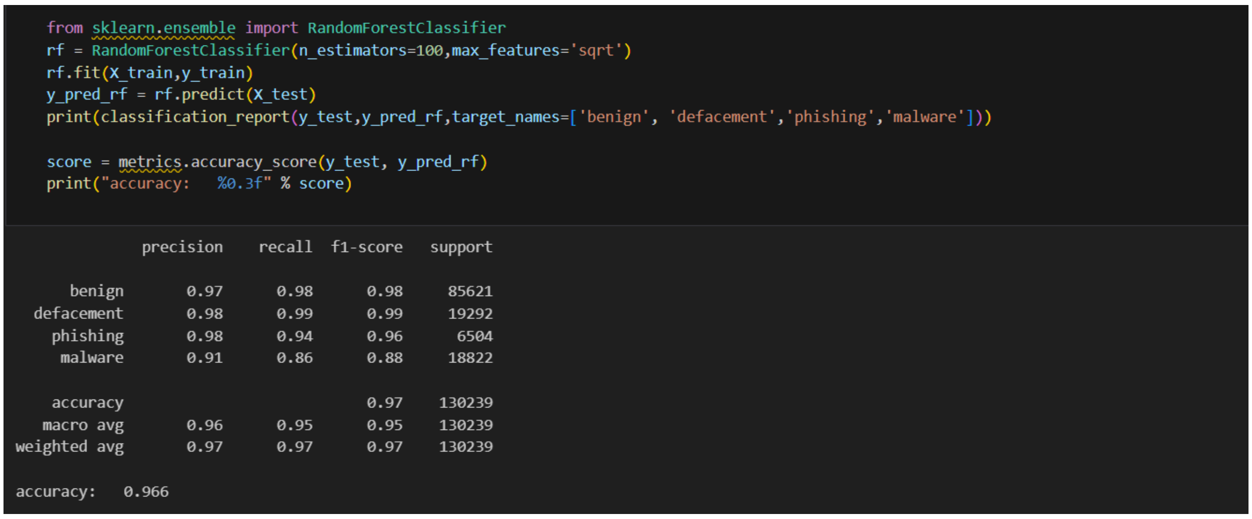
Jira, Trello, or Asana: Project management tools can help organize tasks, track progress, and facilitate communication among team members.

Collaboration and Communication:

Slack, Microsoft Teams, or Discord: These communication platforms enable team collaboration, discussions, and knowledge sharing.

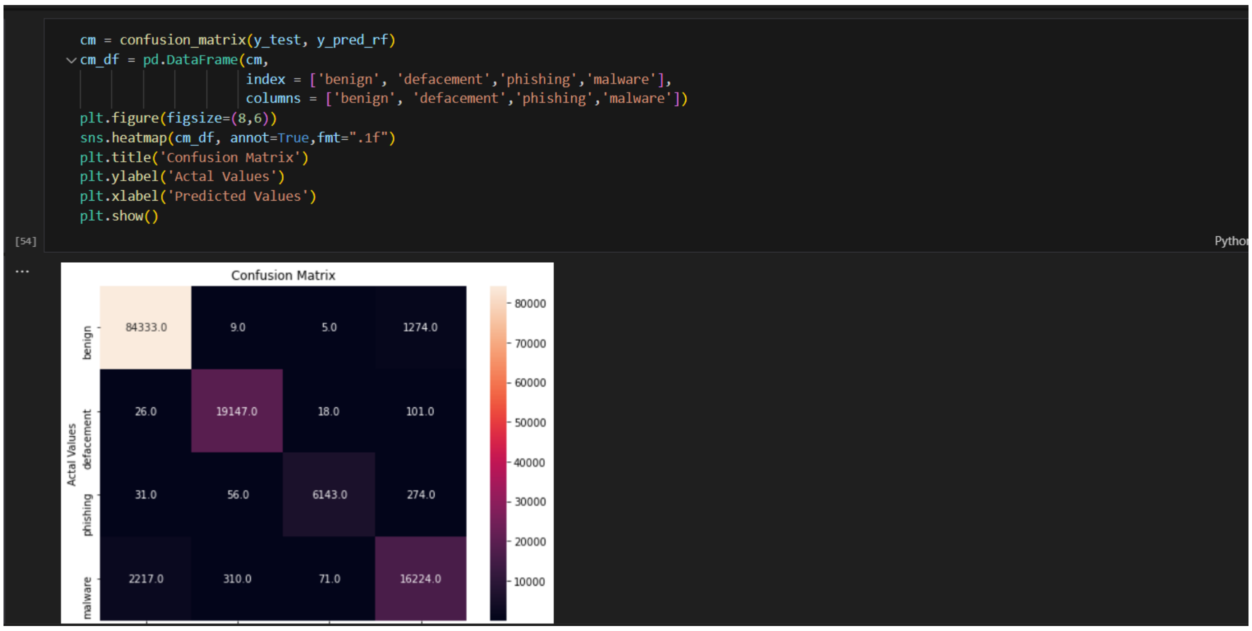
.

7 Outcome/ results of internship work (screenshots of workdone)

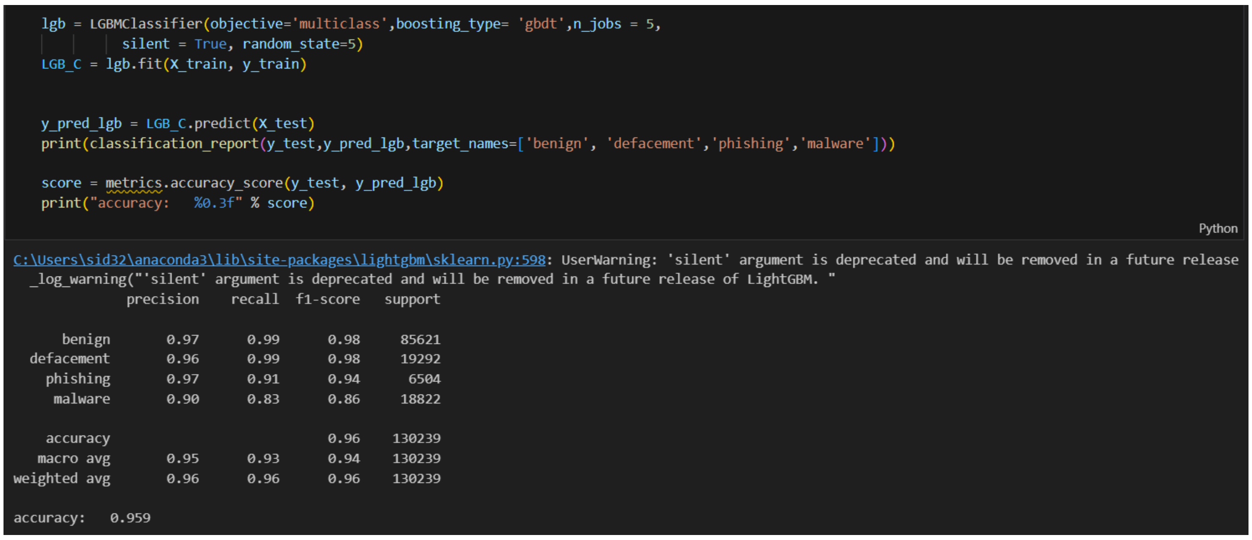


```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100,max_features='sqrt')
rf.fit(X_train,y_train)
y_pred_rf = rf.predict(X_test)
print(classification_report(y_test,y_pred_rf,target_names=['benign', 'defacement','phishing','malware']))

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

```
              precision    recall  f1-score   support

      benign       0.97      0.98      0.98     85621
  defacement       0.98      0.99      0.99     19292
    phishing       0.98      0.94      0.96      6504
     malware       0.91      0.86      0.88     18822

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

accuracy:   0.966
```



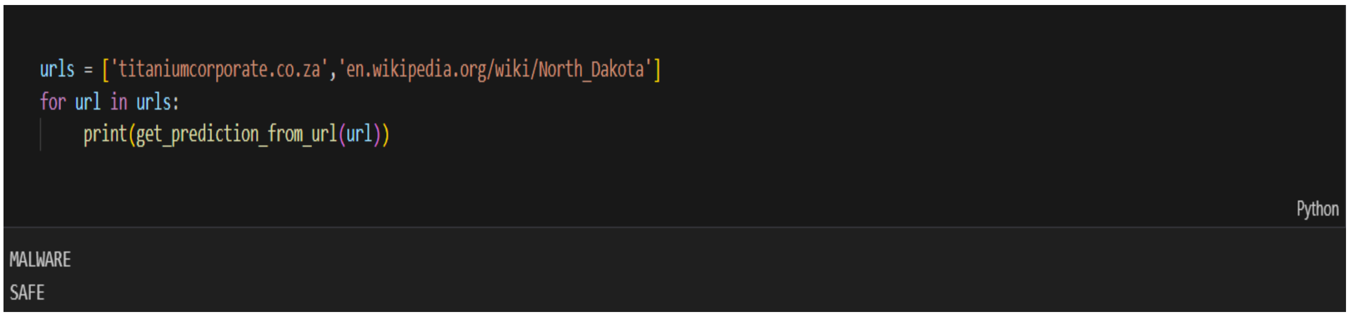
```
cm = confusion_matrix(y_test, y_pred_rf)
cm_df = pd.DataFrame(cm,
                     index = ['benign', 'defacement','phishing','malware'],
                     columns = ['benign', 'defacement','phishing','malware'])
plt.figure(figsize=(8,6))
sns.heatmap(cm_df, annot=True,fmt=".1f")
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
plt.show()
```



```
lgb = LGBMClassifier(objective='multiclass',boosting_type= 'gbdt',n_jobs = 5,
          silent = True, random_state=5)
LGB_C = lgb.fit(X_train, y_train)


y_pred_lgb = LGB_C.predict(X_test)
print(classification_report(y_test,y_pred_lgb,target_names=['benign', 'defacement','phishing','malware']))

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

Python

```
C:\Users\sid32\anaconda3\lib\site-packages\lightgbm\sklearn.py:598: UserWarning: 'silent' argument is deprecated and will be removed in a future release
  _log_warning("'silent' argument is deprecated and will be removed in a future release of LightGBM. "
              precision    recall  f1-score   support

      benign       0.97      0.99      0.98     85621
  defacement       0.96      0.99      0.98     19292
    phishing       0.97      0.91      0.94      6504
     malware       0.90      0.83      0.86     18822

    accuracy                           0.96    130239
   macro avg       0.95      0.93      0.94    130239
weighted avg       0.96      0.96      0.96    130239

accuracy:   0.959
```



```
urls = ['titaniumcorporate.co.za','en.wikipedia.org/wiki/North_Dakota']
for url in urls:
    print(get_prediction_from_url(url))
```

Python

```
MALWARE
SAFE
```

8 Conclusion

In conclusion, the development of a system for the automatic detection of malicious URLs using Python and machine learning techniques holds immense significance in bolstering cybersecurity measures and safeguarding users against online threats. Throughout this project, we have embarked on a journey to address this critical challenge by leveraging modern engineering tools and methodologies.

We began by meticulously collecting and preprocessing a diverse dataset of URLs, followed by extracting relevant features and selecting informative attributes indicative of malicious intent. Subsequently, we developed and fine-tuned machine learning models using state-of-the-art algorithms, ensuring robust performance and generalization.