High Level Design (HLD)

PHISHING DOMAIN DETECTION

Document Version: 0.1

Last Revised Date: 11-May-2023

Parvej alam Ansari

Document Version Control:

Version	Date	Author	Description
0.1	09-05-2023	Parvej alam	Introduction Problem Statement
0.1	10-05-2023	Parvej alam	Design Flow
0.1	11-05-2023	Parvej alam	Performance Evaluation

Contents

1. Introduction

1.1. About High-Level Design Document

The purpose of this HLD document is to feature the required details of the project and supply the outline of the machine learning model and also the written code. This additionally provides the necessary description on how the complete project has been designed end-to-end.

1.2. Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

2. Description

2.1. Problem Perspective

Phishing detection refers to the process of identifyng and preventing fraudulent attempts.It involves using various methods, such as machine learning algorithms or rule based systems, to analyse the charateristics of a message or website and determine whether it is legitimate or phishing.

2.2. Problem Statement

The goal of this project is to create classification models that come up with predictions for whether the URL provided is phishing or non-phishing. The best performing model shall be selected.

2.3. Proposed Solution

The proposed solution is to create User Interface (UI) which is user friendly. It will take required information from the user side as inputs and provides the best possible estimation interms of legitimate or phidhing as outputs using machine learning model which is classifier in nature.

2.4. Solution Improvements

In this Project, Classification solution shall be implemented using non-hyper-tuned machine learning model i.e. This model shall be capable of predicting if the URL is phishing or non-phishing.

2.5. Technical Requirements

First of all, no special hardware needed for implementation of this application, the user should have an interactive device that has access to the web and should have the fundamental understanding of providing the inputs. In the backend, the server should run all the package that's needed for up and running for this application.

2.6. Data Requirements

The information set or Dataset is accessible on the Mendeley phishing website dataset, UCI Repo. or Kaggle or Github in .csv as well as in .xlsx format. Furthermore, This dataset satisfies the real time need of the user which is required for prediction of heating and cooling load as the main theme of the project is to induce the expertise of real time issues.

2.7. Tools Used

- Python 3.8 is employed because the programming language and frameworks like numpy,pandas, sklearn, streamlit, Fast API, MongoDB database and other required modules for building the model.
- PyCharm is employed as IDE for product development and Jupyter notebook is used for analysis purpose.
- For Visualizations Matplotlib and Seaborn are used.
- For information assortment prophetess info is getting used like Pandas Profiling etc.
- Back-end development is completed by Fast API
- Front end development is completed by Streamlit.
- The whole solution is dockerised which makes it platform independent.
- GitHub is employed for Version Management.
- Docker hub and AWS (Elastic Beanstalk (EB) etc.) is employed for project deployment.

2.8. Constraints

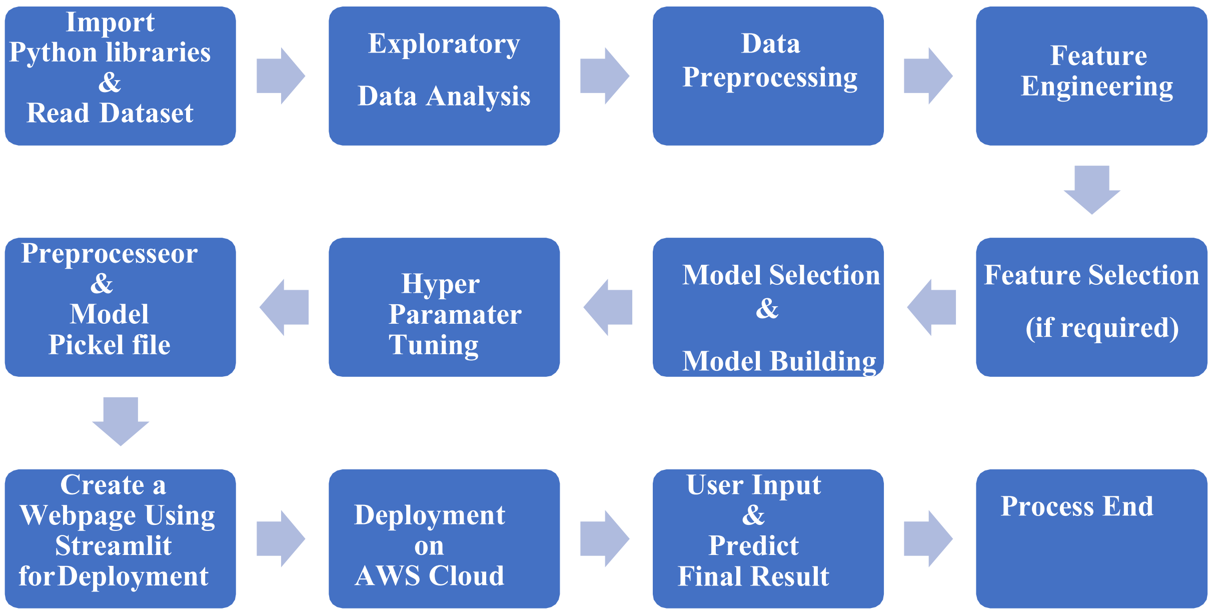
The Classification should be done by considering user inputs in such a way that whether the user is techinal or non-technical shall not matter. Further, answer (prediction) should be user friendly, as automatic as possible and also the user should not be needed to understand any of the operating behind the application.

2.9. Assumptions

The aim of this project is to classify the output interms of phishing URL or not for new data as per the need of the user. Designer of this project has assumed that this application shuld be flexible in nature to handle wide range of data from the user side.

3. Design Flow

3.1. Modelling and Deployment Process


Import Python libraries & Read Dataset
Exploratory Data Analysis
Data Preprocessing
Feature Engineering
Feature Selection (if required)
Model Selection & Model Building
Hyper Paramater Tuning
Preprocesseor & Model Pickel file
Create a Webpage Using Streamlit forDeployment
Deployment on AWS Cloud
User Input & Predict Final Result
Process End

3.2. Logging

Each modlue is would be logged as per the need by applying date-time format. Apart from that the concept of timestamp shall be also utilised so that we can have mutiple versions of the log generation as per the change. Logging will help to faster the execution of the code as well as it will also aid to know the problem in any part of the module of entire application.

3.3. Exception Handling

Whenever slip is occurred, the reason including modulename and its line number should logged in log file so the developer will easily understand and rectify the error or exception which causes the unrest in the code flow.

4. Performance Evaluation

4.1. Reusability

Elements of the code should be written in modular passion so each component of the code becomes independent from each other and hence easy to reuse and even scale the any module without any problem.

4.2. Application Compatibility

The various components of this project should be using python as an interface among them, Each componenet will have its own task to perform and it is task of Python to ensure proper information transfer.

4.3. Resource Utilization

Whenever any task is performed, it will possibly use all the process power available at that point of time till that function is finished.

4.4. Deployment

The model/application/project shall be deployed on AWS via Docker hub.

5. Conclusion

This application will help to classify whether input URL is malacious or genuine for user to surf on.