**High-Level Design Documentation (HLD)**



**Project On:**

**Title: Phishing Domain Detection**

**By: - Dharavath Ramdas**

**(iNeuron.ai)**

**Date: - 12/03/2023**

**Content**

1. **Abstract**
2. **Introduction**
   1. **Why this High-Level Document (HLD)?**
   2. **Scope**
3. **Description**
   1. **Problem Perspective**
   2. **Problem Statement**
   3. **Proposed Solution**
   4. **Solution Improvement**
4. **Requirements.**
   1. **Hardware**
   2. **Tools / Software requirements**
5. **Data requirements**
6. **Constraints**
7. **Assumptions**
8. **Design flow**
9. **Logging & Error Handling**
10. **Performance Evaluation**
    1. **Reusability**
    2. **Application compatibility**
    3. **Resource utilization**
11. **Deployment**
12. **Conclusion**
13. **Abstract.**

Phishing is a type of fraud in which an attacker impersonates a reputable company or person in order to get sensitive information such as login credentials or account information via email or other communication channels. Phishing is popular among attackers because it is easier to persuade someone to click a malicious link that appears to be authentic than it is to break through a computer's protection measures.

Phishing stands for a fraudulent process, where an attacker tries to obtain sensitive information from the victim. Usually, these kinds of attacks are done via emails, text messages, or websites. Phishing websites, which are nowadays in a considerable rise, have the same look as legitimate sites. However, their backend is designed to collect sensitive information that is inputted by the victim. Discovering and detecting phishing websites has recently also gained the machine learning community’s attention, which has built the models and performed classifications of phishing websites. This paper presents two dataset variations that consist of 58,645 and 88,647 websites labeled as legitimate or phishing and allow the researchers to train their classification models, build phishing detection systems, and mining association rules.

The mail goal is to predict whether the domains are real or malicious Features in the dataset that are used for the prediction of url include 'directory\_length', 'time\_domain\_activation', 'length\_url','file\_length', 'qty\_slash\_url', 'qty\_plus\_directory', 'domain\_length', 'qty\_vowels\_domain', 'qty\_asterisk\_directory', 'qty\_hyphen\_directory',

'qty\_dot\_domain', 'qty\_underline\_directory', 'qty\_percent\_directory',

'qty\_dot\_url', 'qty\_hyphen\_url', 'qty\_hyphen\_file', 'qty\_hyphen\_domain',

'params\_length', 'qty\_underline\_url', 'qty\_tld\_url', 'qty\_plus\_params',

'qty\_percent\_url', 'qty\_equal\_params', 'qty\_dot\_params', 'qty\_percent\_params', 'qty\_underline\_params','phishing'.

For prediction, xgboost classifier has been chosen as a final algorithm to create the model we trained the system and achieved an accuracy of 93%.

1. **Introduction.**
   1. **Why this High level Design Documentation?**

The purpose of High-Level Documentation is to (HLD) is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at the high level.

The HLD will:

* Present all of the design aspects and define them in detail.
* Describe the user interface being implemented.
* Describe the hardware and software interfaces.
* Describe the performance requirements.
* Include design features and the architecture of the project.
* List and Describe the non-functional attributes like:
  + Security
  + Reliability
  + Maintainability
  + Portability
  + Reusability
  + Application compatibility
  + Resource utilization
  + Serviceability
  1. **Scope.**

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

1. **Description.**
   1. **Problem Perspective.**

The Phishing Domain Detection is a hyper-tuned machine learning classification model which helps to determine the whether the domains are real or malicious.

* 1. **Problem Statement.**

Phishing is a type of fraud in which an attacker impersonates a reputable company or person in order to get sensitive information such as login credentials or account information via email or other communication channels. Phishing is popular among attackers because it is easier to persuade someone to click a malicious link that appears to be authentic than it is to break through a computer's protection measures.

The mail goal is to predict whether the domains are real or malicious.

Approach: The classical machine learning tasks like Data Exploration, Data Cleaning, Feature Engineering, Model Building and Model Testing. Try out different machine learning algorithms that’s best fit for the above case.

For Feature Engineering show:-

1. URL-Based Features

2. Domain-Based Features

3. Page-Based Features

4. Content-Based Features

* 1. **Proposed Solution.**

The solution proposed is to take the required batch file to predict the result. A pipeline has been created to get the prediction for the new dataset.

* 1. **Solution Improvements.**

The system can be made more futuristic by performing more hyper-tuning methods so that the prediction can be more accurately predictive. The project code has been designed in such a way that whenever new data will come, the model will go under training and if there will be an improvement in the model then the new model will be used for prediction.

1. **Requirements**

**4.1Hardware Requirements:-**

A working computer to code with an active internet connection.

**4.2 Tools / Software Requirements:-**

* Python version used for this project 3.8 (This may get updated and some features might not be available in new version. )
* Python libraries such as NumPy, pandas, flask, Jinja2, HTML, matplotlib, PyMongo, seaborn and scikit-learn (Used for implementation of machine learning algorithms.)
* Jupyter Notebook & Visual studio code is used as an IDE for writing the code.
* Github is used as the version control system
* AWS is used for deployment using docker image.
* Apache Airflow has been used to monitor the ML model.
* AZURE has been used to create and deploy web app



1. **Data Requirements.**

Whenever we are working on any project the data is completely dependent on the requirement of the problem statement. For this project, the problem statement was to create a hyper tuned classification machine learning model which can predict the determine the whether the domains are real or malicious on various parameters.

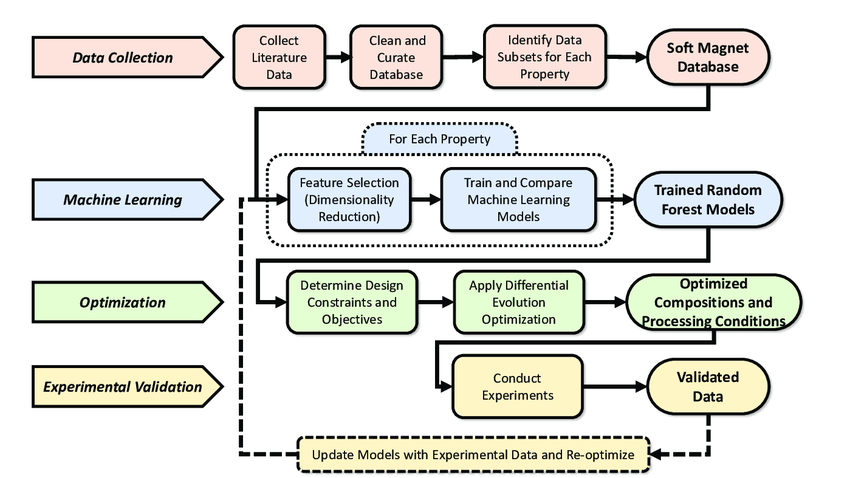
1. **Constraints**

The Apache Airflow application should be user-friendly so that without knowing any technical information he should be able to use our predictive system. A flask web app has been created and it should be user-friendly.

1. **Assumptions**

The main objective of the project is to utilize the data which is provided by the user and to predict the insurance premium. The Apache Airflow application should be accessible from every system which is connected to the internet to predict the result on new dataset. A flask web app has been created as well and it should be accessible to every user who is connected to the internet.

1. **Design Flow.**



The above flow chart represents the flow of any machine learning model which needs to be created .

1. **Logging & Error Handling.**

Each step is logged within the system that runs internally, it basically shows us the data time of each process which is done with our system. It provides us with logging information for end to end web applications.

The logging which we have done in the above process helps us to handle the error because the error is being logged in several log files so that the developer can rectify it.

1. **Performance Evaluation.** 
   1. **Reusability**

The elements of the code is written in such a way that it can be changed and easily written again without changing or creating an entirely different code from scratch. Just the slight changed in the code structure need to be adjusted.

* 1. **Application Compatibility.**

The elements of the project are written in python, it acts as the interface between the machine-learning model and the user. The Apache Airflow application can run on any system with a network connection. Also, the flask web app can be accessed using any system which is connected to the internet.

* 1. **Resource Utilization.**

Once the task is assigned to the model doubtlessly it will use all the resources which are allocated until the task is finished.

1. **Deployment.**

This model is deployed on AWS Ec2 instances. The following are the steps to deploy the model on the AWS platform:

* Create an AWS account
* Create an ECR
* Create S3 bucket
* Create an EC2 instance
* Edit security group
* Connect to an EC2 instance
* Install Docker
* Add the runner in the GitHub
* Add all the secret keys in the GitHub
* In the GitHub actions, run the continuous delivery and deployment workflow once after starting the runner in the ec2 instance
* Start the instance & locate the Docker run.sh file for to initiate the “Runner” to pick the job.
* Use Apache airflow to monitor the model and perform batch prediction
* A web app has been created and deployed using flask in azure

1. **Conclusion.**

We have successfully built end-to-end ML projects using machine learning that can help predict the medical expenses of the users based on various conditions. This type of system can help users to get a whether the domain is real or fake. Along with end to end projects, a flask webapp has been created as well to get the result based on certain inputs like url and dataset.