



URL Genie – Malicious URL Detection with NN and Genetic Algorithms

A project overview of a GitHub repository designed for advanced malicious URL detection.

Presented by :

Nishita Katepallewar - UCE2023542

Vardayini Deshmukh - UCE2023519

Jagruti Disle - UCE2023522

Introduction & Problem Statement

In today's interconnected digital landscape, malicious URLs represent a significant and persistent threat. These deceptive links are often vectors for **phishing attacks**, **malware distribution**, and **domain abuse**, leading to severe security breaches, data theft, and financial losses for individuals and organizations alike.

Effective and rapid detection of malicious URLs is paramount for maintaining robust cybersecurity defenses. This project directly addresses this critical need by developing a sophisticated model capable of classifying URLs or domains as either safe or malicious.

The repository, "Malicious URL Detection Model NN optimized by Genetic Algorithms," aims to provide an intelligent, adaptable solution to this evolving problem.



URL Genie

Malicious URL Detection Model made using Python

This program utilizes a Multilayer Perceptron Neural Network model with optimized hyper-parameters using genetic algorithms to perform malicious URL detection

Enter URL to scan

<https://www.google.com>

Classify URL

Classifying URL: <https://www.google.com>

✅ SAFE with 1.21% malicious confidence

[Explore GitHub Repositor](#)

Repository Structure & Key Components

The URL Genie project is meticulously organized to facilitate research, development, and deployment. Understanding its structure is key to navigating and contributing to the system.

Research_Notebooks/

Contains Jupyter notebooks for data exploration, feature engineering, and initial model prototyping. These notebooks provide a transparent view into the analytical process and iterative model development.

webapp/

Houses all files necessary for the Streamlit web application, enabling users to interact with the trained model through a simple user interface. Includes app.py and requirements.txt for dependencies.

genetic_algorithm_implementation.py

This script contains the core logic for applying genetic algorithms to optimize the neural network's parameters or structure. It drives the intelligent search for improved model performance.

model_generation.py

Responsible for training the base Neural Network model and orchestrating its interaction with the genetic algorithm for optimization. It handles data preparation and model persistence.

All dependencies are managed via requirements.txt within the webapp directory, ensuring a reproducible environment.

Dataset & Feature Extraction



The foundation of any robust machine learning model is high-quality data. This project leverages a comprehensive dataset of both malicious and benign URLs, sourced from platforms like Kaggle, to train and validate the detection model.

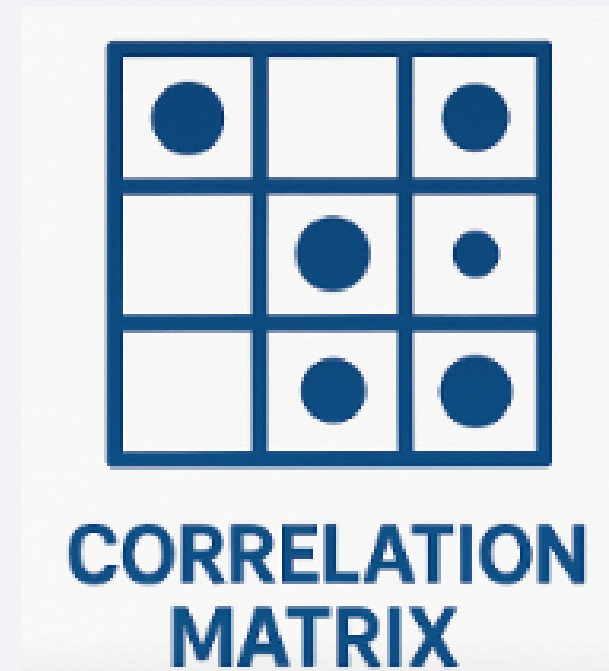
The Feature_Extraction_Notebook.ipynb plays a crucial role in transforming raw URLs into a numerical representation suitable for machine learning. It extracts pertinent information, such as:

- **URL Length:** Malicious URLs often employ length obfuscation.
- **Presence of Numbers/Special Characters:** Indicators of generated or deceptive URLs.
- **Domain Age:** Newly registered domains are frequently associated with malicious activities.
- **Subdomain Count:** An unusually high number can be suspicious.

Rigorous data cleaning steps are also applied, including removing unnecessary columns and addressing class imbalance using techniques like SMOTE (Synthetic Minority Over-sampling Technique) to prevent bias towards the majority class.

Data Visualization & Analysis

Understanding the characteristics of the dataset is vital for developing an effective model. The Data_Visualization_Notebook.ipynb provides critical insights through various visual representations.



Feature Distributions

Visualizing the distribution of features helps identify patterns. For instance, histograms often reveal that malicious URLs tend to be significantly longer than benign ones, or contain a higher frequency of certain special characters.



Correlation Matrices

A correlation matrix heatmap illuminates relationships between different extracted features. High correlations can indicate redundant features, while distinct patterns between feature sets of malicious and benign URLs are prime candidates for classification.



Class Imbalance

Visualizing class distribution highlights potential imbalances (e.g., significantly more benign URLs than malicious). This insight guides the application of techniques like SMOTE during preprocessing to ensure the model doesn't get biased.

These analyses confirmed key observations: malicious URLs often exhibit distinct structural characteristics, such as increased length and the strategic placement of unique characters, which are crucial indicators for the detection model.

Model Architecture – Neural Network

The core classification engine of URL Genie is a sophisticated Multilayer Perceptron (MLP) Neural Network, implemented and managed through the `model_generation.py` script.

Base Model

A Multilayer Perceptron (MLP) was selected for its proven ability to learn complex, non-linear relationships within the extracted URL features, making it well-suited for classification tasks.

Data Split

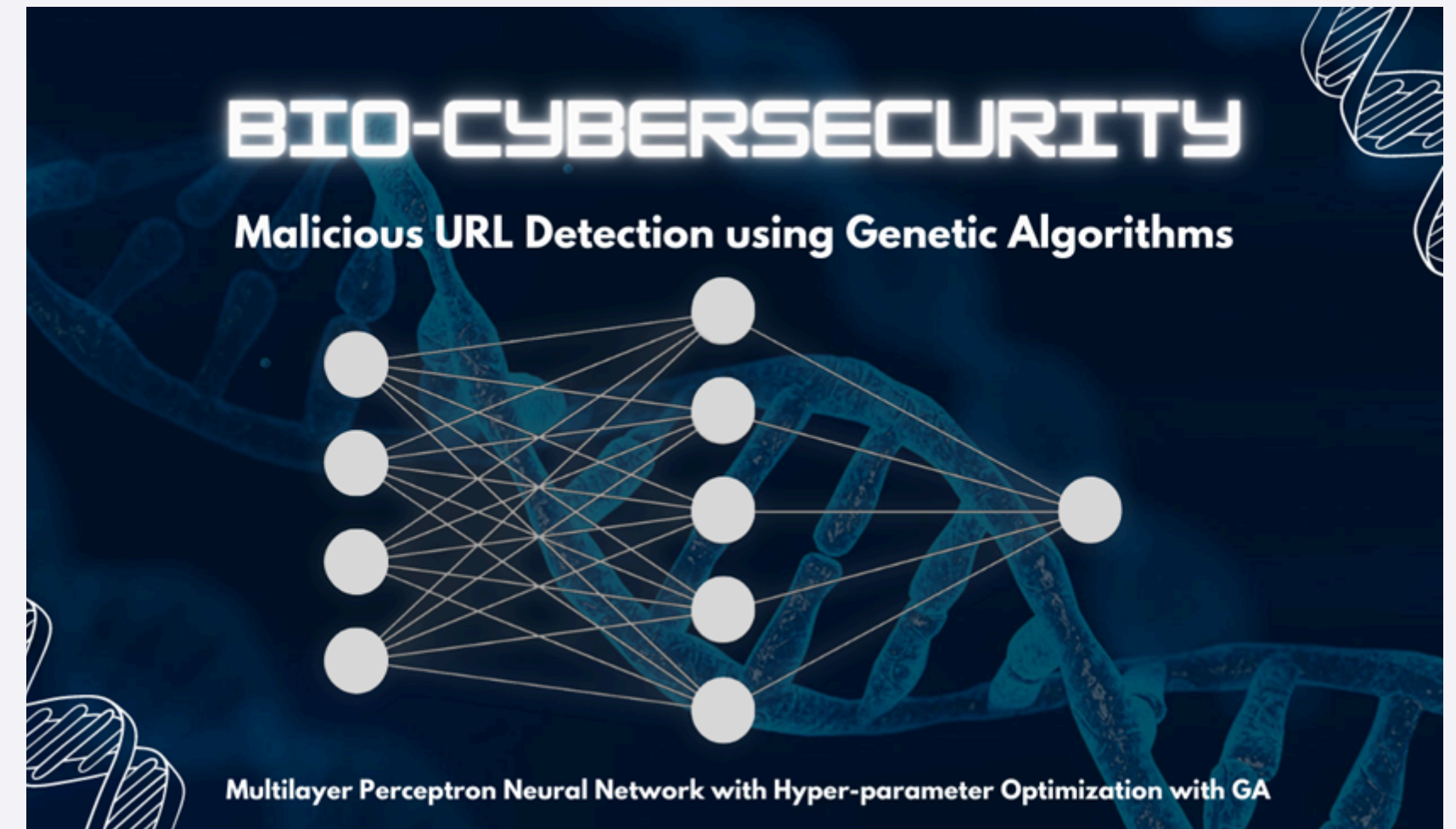
The dataset is rigorously divided into an 80:20 ratio for training and testing, ensuring that the model's performance is evaluated on unseen data to prevent overfitting.

Optimizer & Loss

Adam optimizer is employed for efficient gradient descent, while **Binary Cross Entropy** serves as the loss function, ideal for binary classification tasks (safe vs. malicious).

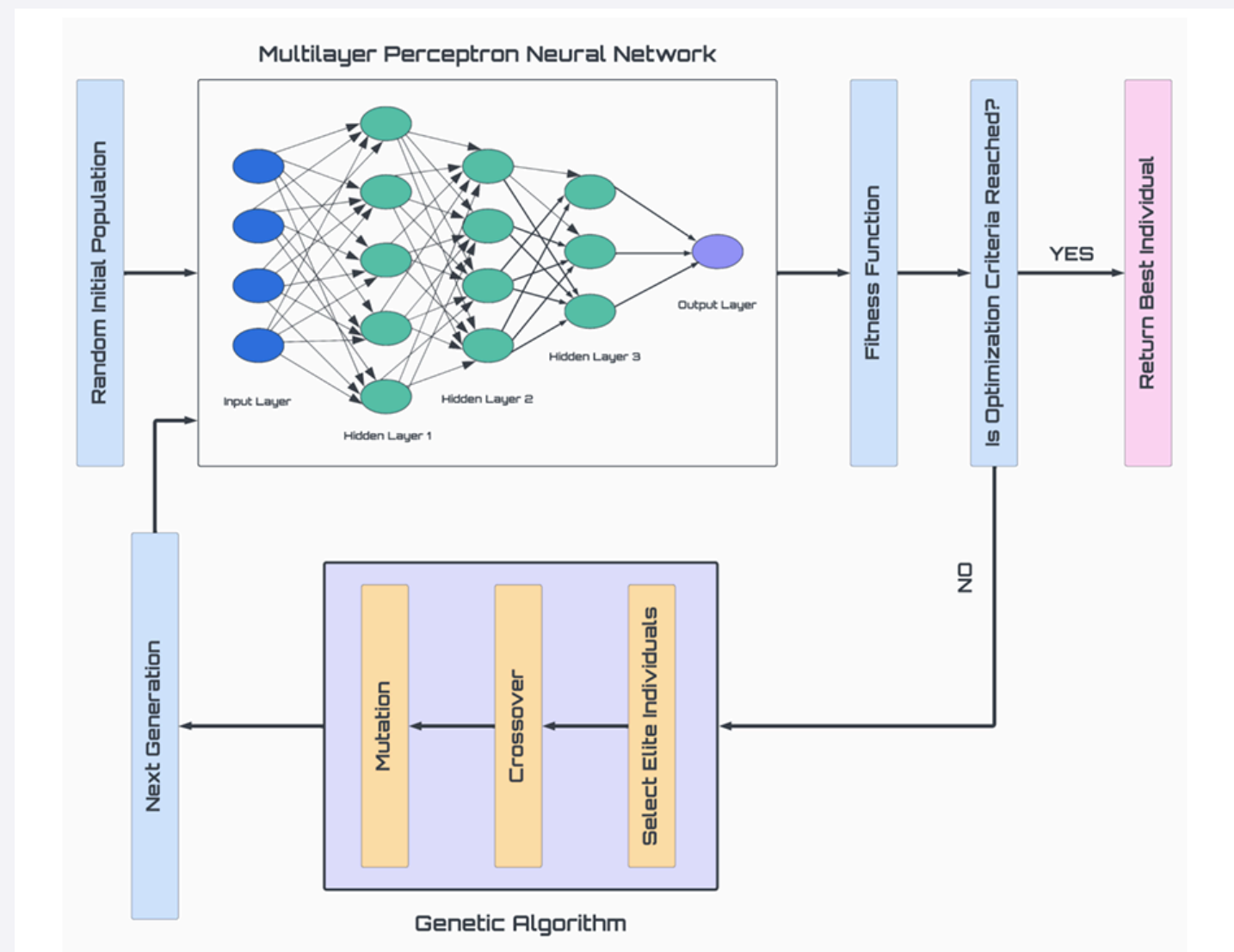
Training Epochs & Batch Size

The model is trained over **10 epochs** with a **batch size of 256**, balancing computational efficiency with effective learning.



Genetic Algorithm Optimization

To push the boundaries of the neural network's performance, URL Genie integrates a Genetic Algorithm (GA) for intelligent optimization.



What is a Genetic Algorithm (GA)?

Inspired by natural selection, a GA is an adaptive heuristic search algorithm. It evolves a "population" of potential solutions, where each "individual" represents a set of hyperparameters or structural configurations for the neural network. Through processes of **selection**, **crossover**, and **mutation**, the GA iteratively improves the population over generations, converging towards optimal solutions.

In this project, each iteration of the neural network model is passed through the GA via `genetic_algorithm_implementation.py`. The "fitness" of each model is rigorously measured by its **accuracy** on the validation or test set. Only the best-performing models are selected for reproduction, ensuring a continuous improvement cycle.

The primary benefit of this approach is that the GA can explore a vast and complex hyperparameter and model-structure space more broadly and efficiently than traditional manual tuning methods, ultimately yielding a superior, optimized model saved in .h5 format.

Web Application / Deployment

The URL Genie project culminates in an accessible web application, allowing users to easily interact with the sophisticated detection model.

1

Webapp Folder

The webapp/ directory contains all necessary components, including the app.py Streamlit application, designed for a seamless user experience.

2

Easy Local Setup

To run locally, simply execute streamlit run webapp/app.py. The application will then be accessible in your browser at localhost:8501.

3

User Interaction

Users are presented with an intuitive interface where they can input any URL or domain. The system then processes the input and provides an immediate classification: **SAFE** or **MALICIOUS**.

Demonstration Examples from README:

- `https://www.google.com` -> **SAFE**
- `http://stcdxmt.bigger1.in/klxtv/apps/uk/` -> **MALICIOUS**

The Streamlit application can be deployed in various environments, from local machines to cloud platforms, or even integrated as a microservice within larger cybersecurity systems, offering flexible deployment options.

Results, Evaluation & Insights

The thorough evaluation of URL Genie reveals promising performance and identifies key areas for continuous improvement.



Classification Performance

The model achieves **high accuracy** (typically >90% as observed in notebooks) on the test set, demonstrating its effectiveness. Key metrics like **precision** and **recall** are balanced, minimizing both false positives (legitimate URLs flagged as malicious) and false negatives (malicious URLs missed).



Feature Importance

Analysis indicates that URL length, presence of specific special characters, and domain age are among the **most influential features** for distinguishing malicious URLs. The genetic algorithm significantly **improved performance** by fine-tuning the NN's ability to leverage these indicators.



Limitations & Challenges

Current limitations include dependency on dataset size, potential for **generalization issues** with novel attack patterns, susceptibility to **adversarial attacks**, and the dynamic nature of malicious domains (feature drift).

Future Work & Enhancements:

- Implement **real-time streaming detection** for immediate threat response.
- Explore more complex models like **CNNs/RNNs** for deeper URL string analysis.
- Integrate with **DNS logs** for richer contextual information.
- Develop **continuous learning** mechanisms to adapt to new threats.

	precision	recall	f1-score	support
0	0.95	0.99	0.97	85621
1	0.90	0.97	0.93	19292
2	0.93	0.79	0.85	6504
3	0.88	0.73	0.80	18822
accuracy			0.94	130239
macro avg	0.92	0.87	0.89	130239
weighted avg	0.93	0.94	0.93	130239
accuracy:	0.936			

Conclusion & Next Steps

The URL Genie project demonstrates a robust and adaptable solution for malicious URL detection, combining the power of neural networks with genetic algorithm optimization.

[Thank you.](#) [GitHub Repository](#)

→ Feasibility of NN for URL Classification

Our findings confirm that neural networks are highly effective for URL classification, capable of discerning subtle patterns indicative of malicious intent.

→ Power of Genetic Algorithms

Genetic algorithms prove to be an invaluable tool for optimizing model hyperparameters and architecture, pushing performance beyond manual tuning.

→ Accessibility via Streamlit

The Streamlit web application makes this advanced detection tool accessible, enabling rapid testing and potential integration into security workflows.

