Phishing Domain Detection: Low-Level Design (LLD)

---

1. Overview

The Low-Level Design of the Phishing Domain Detection system provides a detailed explanation of the system's modules, their functionalities, data flow, and interactions. This design focuses on the internal workings of the backend, REST API, machine learning model, and frontend components.

---

2. Modules and Functionalities

2.1 Machine Learning Module

- **Input:** URL provided by the user.
- **Process:**
  - Extract features (lexical and host-based).
  - Pass the features to the pre-trained machine learning model.
  - Generate a probability score (0-1) for phishing detection.
- **Output:** A prediction (`phishing` or `safe`) and confidence score.

2.2 Backend Module (REST API)

- **Input:** JSON payload containing the URL (`{"url": "example.com"}`).
- **Process:**
  - Validate the input format.
  - Preprocess the URL (cleaning, feature extraction).
  - Pass the extracted features to the ML model.
  - Return the result in JSON format.
- **Output:** JSON response with the result and probability score (e.g., `{"result": "phishing", "confidence": 0.87}`).

2.3 Frontend Module (Web Interface)

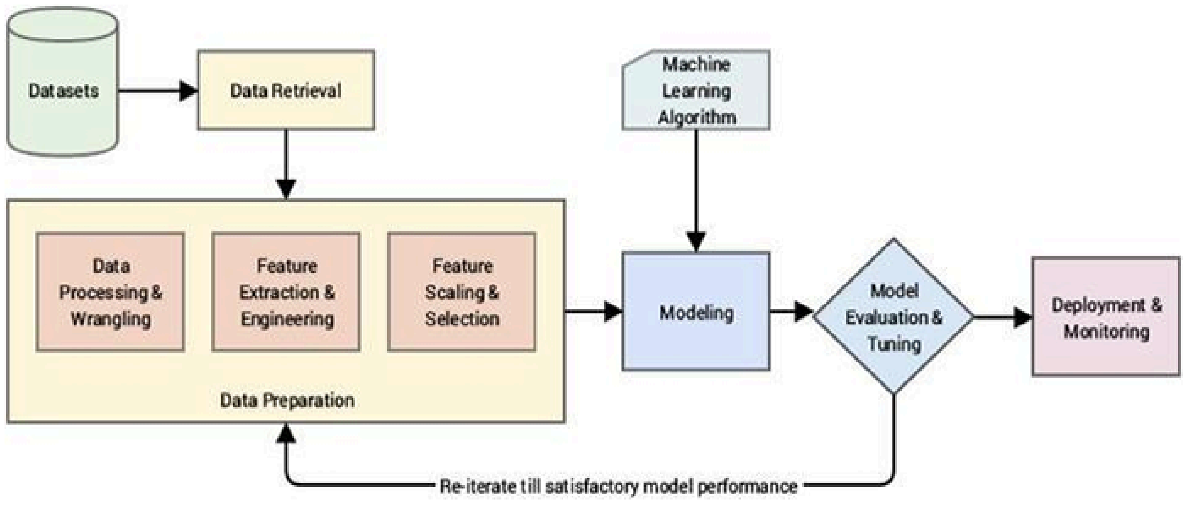
- **Input:** User enters the domain name in the input field.
- **Process:**
  - Send the domain to the REST API using an HTTP POST request.
  - Display the API's response in a user-friendly manner.
- **Output:** The result (`phishing` or `safe`) along with the confidence score.

---

3. Data Flow

Uday Patil

1. **User Input:** The user provides a URL through the web interface.
2. **Frontend to Backend:** The input is sent to the REST API as a JSON payload.
3. **Backend Processing:**
   - Validate the input.
   - Extract relevant features.
   - Pass the features to the ML model.
4. **Model Prediction:** The ML model predicts whether the domain is phishing or safe.
5. **Backend to Frontend:** The result and confidence score are sent back to the frontend.
6. **Result Display:** The frontend displays the result to the user.


Datasets
Data Retrieval
Machine Learning Algorithm
Data Processing & Wrangling
Feature Extraction & Engineering
Feature Scaling & Selection
Data Preparation
Modeling
Model Evaluation & Tuning
Deployment & Monitoring
Re-iterate till satisfactory model performance

---

4. Component Interactions

4.1 REST API Endpoints

- **POST /predict**

Request Body:
```
{

  "url": "example.com"

}
```

   ○

Response Body:
```
{

  "result": "phishing",
```

Uday

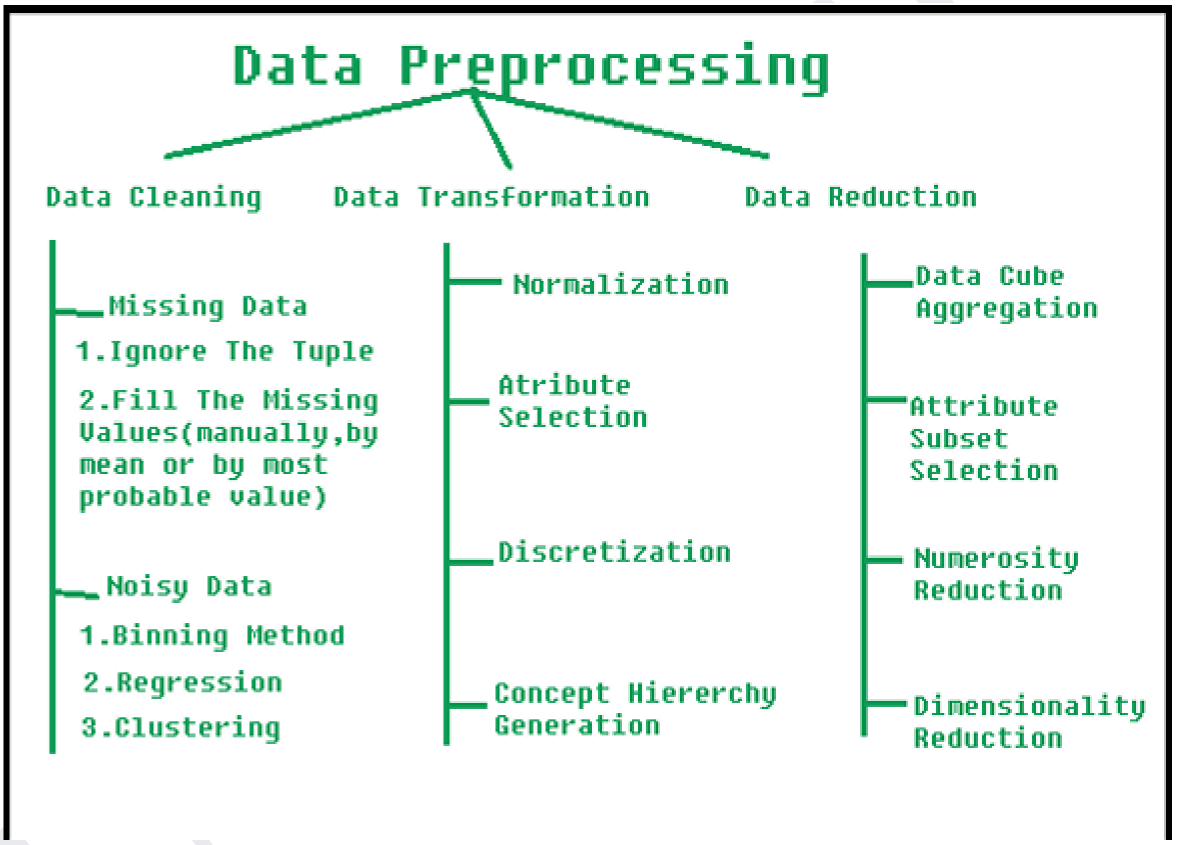
"confidence": 0.87

}

○

4.2 Feature Extraction Module

- Extracts:
  - Lexical Features (e.g., URL length, special characters).
  - Host-Based Features (e.g., domain age, WHOIS details).
- Methods: Implemented as Python functions or classes.



4.3 Frontend Interaction

- **Framework:** ReactJS
- **API Calls:** Axios or Fetch API for HTTP requests.
- **Result Rendering:** Dynamically update UI with the prediction result.

---

5. Data Structures

5.1 Input Data Format

- **URL:** String input provided by the user.
  Example: `https://example.com`

5.2 Output Data Format

- **Prediction Result:** String (`phishing` or `safe`).
- **Confidence Score:** Float (range 0-1).

5.3 Feature Extraction Output

Dictionary of extracted features:

```
{

   "url_length": 18,

   "has_https": True,

   "digit_count": 5,

   "special_char_count": 2,

   "domain_age": 120

}
```

Uday Patil

---

6. Algorithms

6.1 Input Validation

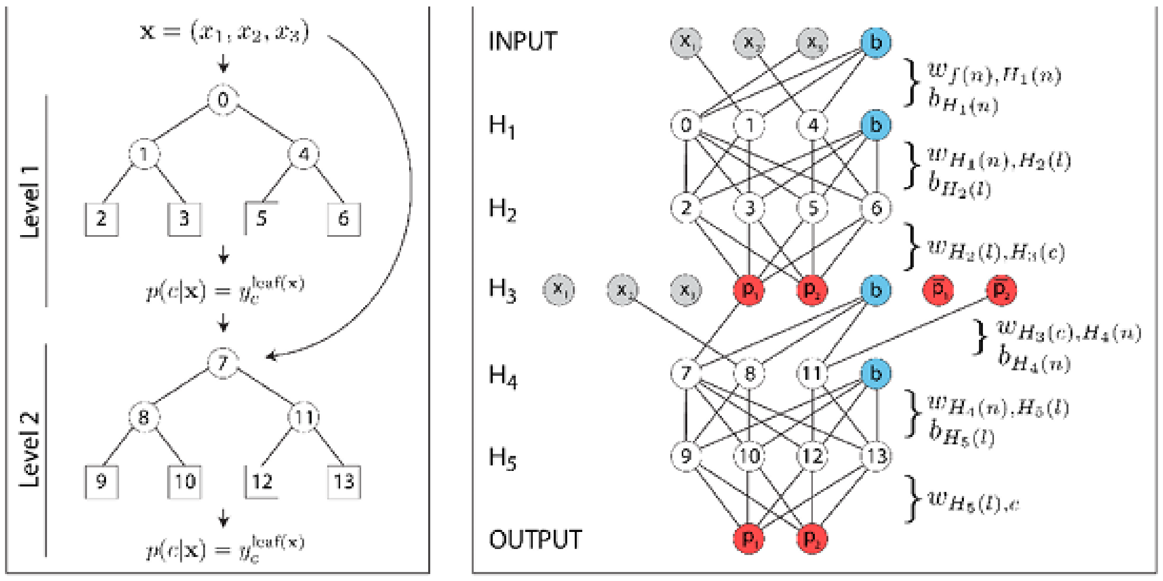
1. Check if the input URL is valid using regex.
2. Ensure the URL format includes a domain name.

6.2 Feature Extraction

- Parse the URL to extract features.
  - Count characters, digits, and special symbols.
  - Check for HTTPS.
  - Query WHOIS data for domain age.

6.3 Model Prediction

- Convert extracted features into a feature vector.
- Pass the vector to the pre-trained model.
- Use the model's output (probability score) to classify the URL.



$\mathbf{x} = (x_1, x_2, x_3)$
Level 1
Level 2
$p(c|\mathbf{x}) = y_c^{\text{leaf}(\mathbf{x})}$
INPUT
H_1
H_2
H_3
H_4
H_5
OUTPUT
$w_{f(n),H_1(n)}$ $b_{H_1(n)}$
$w_{H_1(n),H_2(l)}$ $b_{H_2(l)}$
$w_{H_2(l),H_3(c)}$
$w_{H_3(c),H_4(n)}$ $b_{H_4(n)}$
$w_{H_4(n),H_5(l)}$ $b_{H_5(l)}$
$w_{H_5(l),c}$

(a) (b)

6.4 Result Formatting

- If probability > 0.5: classify as phishing.
- Else: classify as safe.
- Include the confidence score.

---

7. Error Handling

Frontend Errors:

- Display an error message if the input is invalid (e.g., "Please enter a valid domain").
- Show a "Service Unavailable" message if the backend is down.

Backend Errors:

- Handle invalid JSON payloads gracefully (HTTP 400).
- Log server errors (HTTP 500) for debugging.
- Timeout handling for long API calls.

API Response Codes:

- **200:** Success
- **400:** Bad Request
- **500:** Internal Server Error

Uday Pat

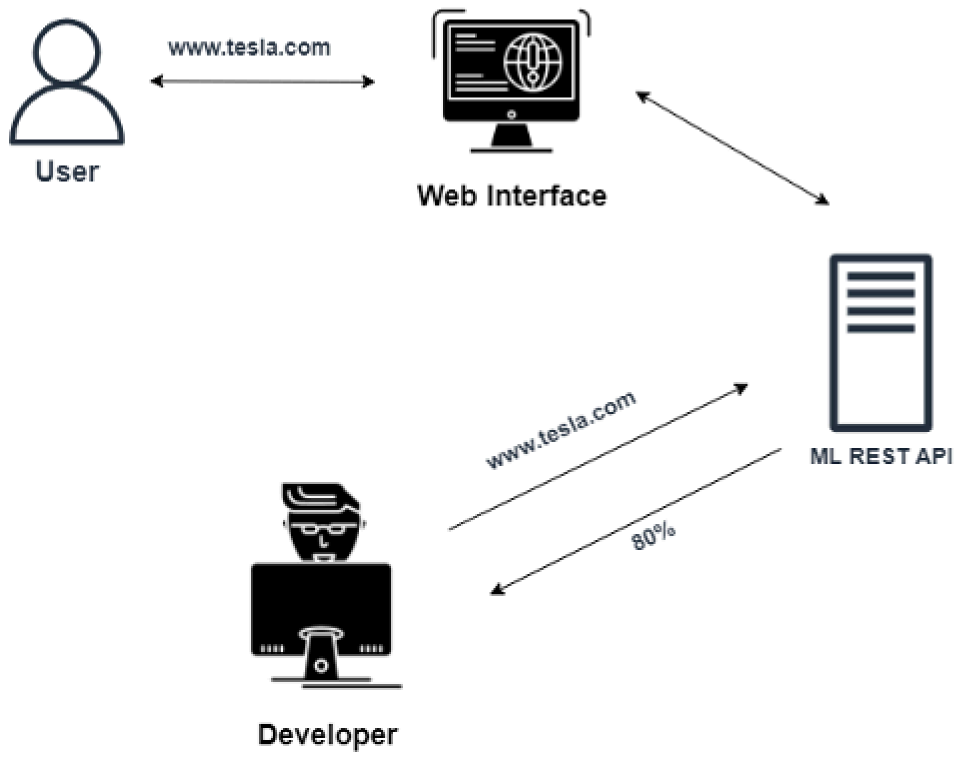
8. Security Measures

- **Input Sanitization:** Prevent SQL injection and malicious payloads.
- **HTTPS Communication:** Encrypt all frontend-backend communication.
- **Rate Limiting:** Protect the API from DDoS attacks.

---

9. Future Scope

- Integrate real-time WHOIS lookups for more accurate host-based features.
- Add multi-language support for the web interface.
- Implement user authentication for sensitive data usage.


www.tesla.com
User
Web Interface
ML REST API
www.tesla.com
80%
Developer