# 📘 Project Observation Report – Phishing Website Detection using PhiKitA Dataset

## 👨‍💻 Project Title

Detection of Phishing Websites using Machine Learning, Deep Learning, and Graph-Based Approaches

## 📌 Dataset Used

- PhiKitA Dataset: A phishing website detection dataset containing:

- website: URLs (as strings)

- status: Labels (phishing or legitimate)

- page\_rank: Popularity of the site

## 🔰 Phase 1: Machine Learning (Random Forest Classifier)

\*\*Goal\*\*: Build a baseline phishing detection model using traditional ML with engineered features from URLs.

\*\*Steps Taken\*\*:

1. Label Encoding: Converted status into binary (1 = phishing, 0 = legitimate).

2. Feature Engineering:

- url\_length, special\_chars, suspicious\_keywords, page\_rank

3. Feature Scaling using StandardScaler.

4. Trained a RandomForestClassifier.

5. Achieved ~85% accuracy.

\*\*Observations\*\*:

- Easy to set up.

- Good baseline performance.

- Limited by manual features.

## 🤖 Phase 2: Deep Learning – ABS-CNN (Basic)

\*\*Goal\*\*: Use deep learning to automatically learn patterns from numerical features.

\*\*Steps Taken\*\*:

1. Used the same features as Phase 1.

2. Reshaped features to (samples, 4, 1).

3. Built a basic CNN with attention (Dense + Multiply).

4. Achieved ~83% accuracy.

\*\*Observations\*\*:

- Better feature interaction.

- Simple attention added interpretability.

- Architecture was shallow.

## 🤖 Phase 3: Advanced Deep Learning – Full ABS-CNN (Time-Distributed Attention)

\*\*Goal\*\*: Implement a more paper-accurate ABS-CNN with advanced attention.

\*\*Steps Taken\*\*:

1. Used a 2-layer Conv1D stack.

2. Applied TimeDistributed attention + Softmax.

3. Multiplied attention weights with conv output.

4. Improved architecture and interpretability.

\*\*Observations\*\*:

- Attention per time step.

- Closely matches ABS-CNN from paper.

- Ready for visualization and deployment.

## 🔮 Upcoming Phase: Graph-Based Learning (Planned)

\*\*Goal\*\*: Use DOM graph representations of HTML for detection.

\*\*Plan\*\*:

1. Scrape/download HTML for each URL.

2. Parse using BeautifulSoup.

3. Convert to graph using networkx.

4. Train GNN model (GCN or GAT) with PyTorch Geometric.

\*\*Notes\*\*:

- HTML not in dataset, must be scraped.

- Offers highest potential accuracy (~92.5%).

## 📊 Feature Summary

| Feature | Description |

|---------|-------------|

| url\_length | Total characters in URL |

| special\_chars | Count of /, =, &, ? |

| suspicious\_keywords | Count of phishing-related words |

| page\_rank | Predefined site rank |

| has\_https | (Planned) Presence of HTTPS |

| num\_digits | (Planned) Digits in URL |

| num\_dots | (Planned) Dots in domain name |

## ✅ Overall Learnings

- ML models are fast to deploy with decent results.

- DL models (ABS-CNN) allow deeper feature learning.

- Graph-based methods offer the highest accuracy if HTML is available.

- Progressing from simple to complex models provided valuable insights.