COSC 435 Project: Phishing Website Detection Using Machine Learning

Project Overview:

This project aims to develop a machine learning-based classifier to identify phishing websites based on various web features. Phishing is a common cyber threat where malicious actors try to trick users into providing sensitive information by mimicking legitimate websites. The project involves preprocessing and extracting features from website URLs and web content, then training and testing various machine learning models to classify websites as legitimate or phishing.

Objective:

- Build a classifier that can accurately detect phishing websites using machine learning algorithms.
- Apply both basic and advanced ML techniques.
- Evaluate the robustness of the models against adversarial examples (e.g., slightly altered URLs).

Dataset:

1. **Phishing Corpus from UCI**
    - A well-established dataset from UCI Machine Learning Repository, containing features extracted from phishing and legitimate websites.

Project Phases and Guidelines:

Phase 1: Data Preprocessing and Feature Engineering

- **Tasks**:
    - Load and understand the dataset.
    - Clean the dataset by handling missing or erroneous values.
    - Engineer features from URLs and web content, such as:
        - URL length
        - Presence of special characters (e.g., @, -, ., ?)
        - Certificate information (if available)
        - Domain name-based features (e.g., domain age)
    - Create additional features based on heuristics to improve phishing detection.
- **Deliverable**: A clean, well-processed dataset with feature explanations and transformations documented in a Jupyter Notebook.

Phase 2: Basic Machine Learning Models

- **Tasks**:
    - Implement basic classification models such as Logistic Regression and k-Nearest Neighbors (k-NN).
    - Train these models on the processed dataset.
    - Evaluate the models using standard metrics such as accuracy, precision, recall, F1-score, and confusion matrix.
    - Perform cross-validation to assess model generalizability.

- **Deliverable**: A report detailing model implementation, performance metrics, and cross-validation results, along with the corresponding code in a Jupyter Notebook.

Phase 3: Advanced Machine Learning Models

- **Tasks**:
    - Implement advanced machine learning models, including:
        - **XGBoost**: A powerful ensemble method using gradient boosting.
        - **Support Vector Machine (SVM)**: For finding the optimal boundary between classes.
        - **Neural Networks**: A simple feed-forward neural network architecture for classification.
        - **Choice of Advanced Neural Network** (e.g., CNN, RNN, Transformer Based, etc.)
    - Compare the performance of advanced models with basic models.
    - Optimize hyperparameters using grid search or random search.
    - Use additional evaluation metrics like AUC-ROC curves for model performance.
- **Deliverable**: A comprehensive report comparing the performance of basic vs. advanced models, optimization steps, and final results in terms of model accuracy and robustness, along with the Jupyter Notebook containing the code.

Phase 4: Robustness and Adversarial Example Analysis

- **Tasks**:
    - Explore model robustness against adversarial attacks by modifying the dataset with small perturbations, such as:
        - Changing a few characters in phishing URLs.
        - Altering the case of characters or adding special symbols.
    - Implement basic adversarial training techniques to improve model resilience.
    - Evaluate the impact of these adversarial examples on the trained models.
- **Deliverable**: A report documenting the process and findings from adversarial testing, along with any modifications that improved robustness. The report should contain detailed evaluations of how different models respond to adversarial examples.

Team Collaboration Guidelines:

- **Regular Meetings**: The team should meet at least once a week to discuss progress, roadblocks, and findings. Meetings should ensure each phase's progress aligns with the overall timeline.
- **Code Sharing**: Use GitHub or GitLab for code sharing and version control. Ensure that each team member pushes their code regularly and that it is properly documented.
- **Documentation**: Each student is responsible for documenting their work thoroughly, including assumptions, decisions, and key insights. The documentation will be consolidated into a final report.
- **Cross-Collaboration**: Although each student has individual tasks, it’s important to review and provide feedback on other phases to ensure consistency across all stages.

Deliverables:

1. **Preprocessing Report and Clean Dataset**:
    - A documented Jupyter Notebook detailing data preprocessing steps and feature engineering.
    - A clean dataset ready for model training.
2. **Basic Model Implementation**:
    - Jupyter Notebook with Logistic Regression and k-NN implementations.
    - Performance metrics and a comparison report.
3. **Advanced Model Implementation**:
    - Jupyter Notebook with XGBoost, SVM, and 2 Neural Network models.
    - Report comparing basic and advanced models, including hyperparameter optimization and performance analysis.
4. **Adversarial Example Report**:
    - Jupyter Notebook showing adversarial attack implementation.
    - A report on the robustness of the models against these attacks and potential adversarial training strategies.
5. **Final Consolidated Report**:
    - A comprehensive final report summarizing all phases of the project, including key findings, results, and future improvement suggestions.
    - Include visualizations (confusion matrices, AUC-ROC curves, etc.) and all relevant performance metrics.
    - Ensure the final report is well-structured, concise, and follows a scientific paper format (introduction, methods, results, discussion).
6. **Presentation**:
    - Each team member will present their respective sections, and the entire team will collaborate on a 15-minute presentation of the overall project results.
    - Include key visuals, diagrams, and performance insights.

Evaluation Criteria:

- **Correctness and Completeness**: Are all tasks completed as per the guidelines? Is the code well-written and functional?
- **Model Performance**: How well do the models perform? Is there evidence of hyperparameter tuning and optimization?
- **Robustness**: How robust are the models against adversarial examples? Have the students tested and documented this thoroughly?
- **Team Collaboration**: Was the work well-distributed, and did each student contribute effectively? Are the roles clearly defined and executed?
- **Documentation and Reporting**: Is the documentation clear and detailed? Is the final report well-written, with appropriate analyses and visualizations?

Teams:

This work is to be done in teams of 3-4 members. No use of Generative AI or LLMs allowed. There must be a record of equal work division and it should be verifiable by the instructor at the time of the evaluation.

Due date: last day of classes (December 6, 2024 at 23:59)