Phishing Email Detection Project Documentation

1. Introduction

Objective

The primary objective of this project is to develop an efficient system to classify phishing emails from legitimate emails using machine learning (ML) and deep learning (DL) models. The goal is to build a robust pipeline that can accurately detect phishing emails based on textual content, helping users safeguard their information from malicious actors.

Importance

Phishing attacks are one of the most common threats to cybersecurity, often leading to data breaches and financial loss. Automating the detection of phishing emails helps reduce human error and provides an additional layer of protection.

Scope

This project uses a variety of ML and DL techniques to classify emails. A dataset containing email content is preprocessed, transformed, and fed into different models for classification. The evaluation focuses on performance metrics such as accuracy, F1-score, precision, and recall.

2. Dataset Description and Preprocessing Steps

Dataset Description

- The dataset contains a single feature: **Email Text** (the raw content of emails).
- The target variable, **Email Type**, indicates whether the email is legitimate (0) or phishing (1).

Preprocessing Steps

1. **Text Cleaning**:
    - **URLs Removal**: URLs in email content were removed as they are often irrelevant to classification and could bias models.
    - **Punctuation Removal**: Special characters and punctuations were stripped from the text.
    - **Lowercasing**: All text was converted to lowercase for consistency.
    - **Whitespace Removal**: Extra spaces were removed.
2. **Tokenization**:
    - Each email text was split into tokens (individual words) using a tokenizer from TensorFlow.
3. **Padding**:
    - Since models require inputs of uniform length, shorter email texts were padded, and longer texts were truncated to a maximum length of 150 tokens.
4. **Label Encoding**:
    - The target variable (Email Type) was encoded into numerical values (0 and 1) using LabelEncoder from sklearn.

3. Algorithms Used in Preprocessing

1. **Tokenization**:
    - The Tokenizer from TensorFlow was utilized to convert text data into sequences of integers, where each unique word is represented by a unique integer.
2. **Padding**:
    - Sequences were padded using pad_sequences to ensure uniformity, crucial for neural network models like LSTM and CNN.
3. **Embedding Layer**:
    - For deep learning models, an embedding layer was used to map integers (word indices) into dense vectors of fixed size.

4. Feature Selection Methodologies

Feature selection was minimal in this project as the dataset had a single textual feature, **Email Text**. The decision to set the maximum sequence length to 150 tokens was based on:

- Empirical observations of typical email lengths.
- Trade-offs between computational efficiency and the ability to capture meaningful information.

5. General Description of Models Used

Machine Learning Models

1. **Logistic Regression**:
    - A linear model used for binary classification. It estimates the probability of an input belonging to a specific class.
2. **Random Forest**:
    - An ensemble of decision trees that improves classification accuracy by averaging predictions across multiple trees.
3. **Decision Tree**:
    - A simple, interpretable model that splits data hierarchically based on feature values.
4. **K-Nearest Neighbors (KNN)**:
    - A non-parametric algorithm that classifies based on the majority label of nearest neighbors.
5. **Support Vector Machine (SVM)**:
    - A model that finds the optimal hyperplane for separating classes using kernel functions.
6. **Multi-Layer Perceptron (MLP)**:
    - A feedforward artificial neural network with hidden layers that capture complex patterns in data.

Deep Learning Models

1. **LSTM**:
    - Long Short-Term Memory networks are a type of recurrent neural network (RNN) designed to handle sequential data with long-term dependencies.
2. **CNN**:
    - Convolutional Neural Networks are typically used for spatial data but can capture local patterns in text.
3. **GRU**:
    - A simplified version of LSTM with fewer parameters, making it computationally more efficient while retaining performance.

6. Transformation Techniques for LSTM, GRU, CNN, and DRL

1. **LSTM**:
    - Input sequences were transformed using an embedding layer to dense vectors.
    - Sequential data was processed using LSTM layers, capturing temporal dependencies.
    - A final dense layer with a sigmoid activation was used for binary classification.
2. **GRU**:
    - GRU layers were used in place of LSTM layers to reduce the number of parameters while retaining sequential learning capabilities.
3. **CNN**:
    - Input sequences were passed through convolutional layers to identify local patterns.
    - Global Max Pooling was applied to reduce dimensions, followed by dense layers for classification.
4. **DRL** (if applicable):
    - Involves agent-environment interaction; however, its application in this project was not detailed.

Conclusions

- **Key Insight**: Deep learning models were more effective at capturing patterns in textual data.
- **Challenges**: Longer training times for deep learning models.
- **Recommendation**: LSTM models are suitable for phishing detection tasks, balancing accuracy and computational cost.

Future Work:

- Use larger datasets for better generalization.
- Explore pre-trained language models (e.g., BERT, GPT) for advanced text representations.
- Investigate hybrid approaches combining ML and DL techniques.