Phishing Message Case-Study

Introduction:
The rapid evolution of blockchain technology and the rise of decentralized applications (dApps) have given birth to the Web3 ecosystem. While this technology promises greater control and ownership over digital assets, it has also introduced new security challenges. One of the most significant concerns is the prevalence of phishing attacks targeting Web3 users with the intent of obtaining their seed phrases and subsequently draining their wallets for fraudulent activities.

Problem Description:
Web3 phishing attacks are a critical issue in the modern blockchain landscape. These attacks involve the use of deceptive and fraudulent tactics to manipulate users into revealing their sensitive information, primarily their seed phrases. Seed phrases, also known as mnemonic phrases or recovery phrases, are sets of words that act as a cryptographic key to access and control users' wallets and assets within the Web3 ecosystem.

The core problem lies in the fact that users often lack awareness and knowledge about the security practices required to protect their seed phrases. Phishing attackers exploit this vulnerability by deploying convincing phishing messages through various communication channels, including email, social media, messaging apps, and even fake dApps. These messages typically prompt users to click on malicious links, enter their seed phrases on counterfeit websites, or share their confidential information.

Task:
For this task, you are required to build an end-to-end training pipeline to train a phishing message detection model using a web3 phishing dataset, with a **Large Language Model**.

The end-to-end training pipeline must be implemented with Python scripts and not in a Jupyter notebook.

Lastly, package the model as a Flask application and provide a simple UI that allows users to interact with it. The output should be either binary or the probability score of the message being a phishing message.

Evaluation:
Your submission should demonstrate proper programming conventions. In particular, your work should demonstrate:

1. Proper choice of machine learning / deep learning algorithms
2. Proper clean coding practices
3. Reusability and reproducibility

If you find it helpful, you may describe your thought process for this task in a README.md file as well.

Deliverables:

- Working code for the end-to-end training pipeline using Python scripts.
- Working code for loading the trained model and performing inference using Python scripts.
- Evaluate your trained model and document the metric results in a README.
- Working code for the Flask application.
- README with instructions on how to execute the training pipeline.
- README on how to run the packaged Flask application.
- A requirements.txt file for all dependencies.
- UI can be as simple as an input box and a run button. It should display the response from the model after hitting the run button with some inputs. You can use simple HTML and Javascript or a frontend framework that you are familiar with.
- Dockerfile(s) or docker-compose.yaml file to start the frontend and backend application.

Please ensure that all files needed in the Dockerfile(s) are also included. We will build and run the docker image.