Fraud_Email_MNB.ipynb — Detailed Overview

Detailed Breakdown of Functions and Classes

1. EmailFraudDetector Class

Purpose:
This PyQt5-based class creates a graphical user interface (GUI) for the fraud email detection system. It uses a pre-trained Multinomial Naive Bayes model and a vectorizer to process and classify email data.

Methods:

- `__init__(self, model, vectorizer)`
  Description:
  Constructor that initializes the EmailFraudDetector object with a given model and vectorizer, then calls `init_ui` to set up the GUI.
  Parameters:
  - `model`: Pre-trained Multinomial Naive Bayes model.
  - `vectorizer`: A fitted `CountVectorizer` instance used to transform email text into feature vectors.
- `init_ui(self)`
  Description:
  Initializes the GUI components. This method sets up the main window title, icon, dimensions, and layout using PyQt5 widgets.
  Features:
  - Sets window title and icon.
  - Defines the layout and minimum window size.
  - Configures additional UI elements (buttons, text areas, etc.) for user interaction.

2. Other Components

While the primary focus is on the `EmailFraudDetector` class, the notebook also includes the following key components:

a. Data Processing and Feature Extraction

- **Regular Expression Functions:**
  Used for cleaning and pre-processing email text data before feature extraction.
- **CountVectorizer:**
  Converts email text data into a matrix of token counts, which serves as input features for the Multinomial Naive Bayes classifier.

b. Model Training and Evaluation

- **Multinomial Naive Bayes Model:**
  The notebook trains the Naive Bayes classifier using the processed features, then evaluates its performance on email data using accuracy metrics from scikit-learn.

c. PDF Processing Utilities

- **Libraries such as PyPDF2 and pdfplumber:**
  These libraries are imported to facilitate reading and processing PDF files, possibly to extract email content for analysis.

d. GUI Components Using PyQt5

- **PyQt5 Widgets:**
  A modern GUI is implemented with several widgets including:
  - **QApplication, QMainWindow:** Base classes for building the application window.
  - **QVBoxLayout, QHBoxLayout, QWidget:** Layout managers and container widgets for organizing the UI.
  - **QLabel, QTextEdit, QPushButton:** Basic UI elements for displaying text and interacting with the application.
  - **File Dialogs and Status Bar:** Allowing users to select files and view application status.
  - **Matplotlib Integration:** Embedding plots in the GUI using FigureCanvas.