**Email Classifier.py**

**## The Prompt##**

Create a Python program using pandas, numpy, email-utils and other necessary libraries called email\_classifier.py that preprocesses and standardizes email data with the following specifications:

The program should:

1. Accept CSV and Excel files as input through a command-line prompt

2. Handle multiple file encodings (utf-8, latin1, iso-8859-1, cp1252)

3. Clean and standardize email data with these key functions:

- Extract sender/recipient names and email addresses

- Extract sender domains from email addresses

- Normalize dates to a standard format

- Clean text by removing invalid characters and normalizing Unicode

- Truncate email body content to 1000 characters

- Replace empty or invalid values with 'NULL'

4. Include junk detection that:

- Identifies rows with too many NULL values

- Detects specific junk patterns like "AAAfAAAAA", "AAAAfA?s", etc.

- Checks for repeated characters that might indicate encoding errors

- Preserves rows with valid email addresses even if other fields are problematic

5. Use these key data structures:

- COLUMN\_MAPPING dictionary to standardize column names

- OUTPUT\_COLUMNS list for consistent column ordering

- EmailDataProcessor class with methods: load\_file, extract\_email\_components, parse\_email\_date, is\_junk\_row, process\_data, generate\_report, and save\_report

6. Generate output files:

- Processed CSV with standardized data

- Text report with processing statistics including:

\* Total rows initially loaded

\* Rows removed as junk

\* NULL value counts and percentages

\* List of output columns

\* Sample of the processed data (first 5 rows)

- Use timestamp in filenames (format: YYYYMMDD\_HHMMSS)

7. Include helper functions:

- clean\_text: remove invalid characters and normalize Unicode

- extract\_domain: extract domain from email address

- is\_junk\_text: detect meaningless junk data

8. Main execution flow:

- Load file with error handling

- Process data with junk removal

- Generate and display report

- Prompt user for confirmation before saving files

- Save processed data and report with timestamped filenames

**Domain Analyzer.py**

**##The Prompt##**

Create a Python program called domain\_analyzer.py that analyzes malicious email patterns with the following features:

1. Load and process email CSV data with columns like sender\_email, sender\_domain, subject, and body

2. Handle file loading with multiple encoding options (utf-8, latin1, iso-8859-1, cp1252)

3. Analyze sender domains to show top sources and calculate domain entropy

4. Analyze email addresses to show most common senders and detect username patterns

5. Detect action-oriented language in subject lines (like "urgent", "action required", etc.)

6. Use scikit-learn for text analysis:

- TF-IDF vectorization with max\_features=1000 and English stop words

- K-means clustering (default 4 clusters) to group similar emails

- PCA for dimensionality reduction and visualization

7. Generate a clean visualization (email\_clusters.png) showing:

- Email clusters in a 2D plot using Principal Components

- Clear labels "Principal Component 1" and "Principal Component 2"

- Annotations showing top 3 domains for each cluster

- Different colors for each cluster

- No explanatory text box at the bottom

8. Create a comprehensive text report (email\_analysis\_report.txt) with findings

9. Include username pattern analysis (presence of numbers, underscores, average length)

10. Handle error cases gracefully with sample data generation if the file doesn't exist

The final program should be about 644 lines of code with all visualization and reporting capabilities.