Instruction Document:

* **How did I approach the Solution?**

In my approach to solving the problem, I started by focusing on the textual data provided in various files. My goal was to analyse this data to extract meaningful insights about its content. I began by writing a script to process each text file, which involved reading the content and calculating several key metrics. These metrics included sentiment scores, readability scores, and various linguistic features like average sentence length and word complexity.

For the sentiment analysis, I used a tool to determine whether the text had a positive or negative tone, which helps in understanding the overall sentiment of the content. I also assessed readability to ensure that the text is accessible to the intended audience by evaluating factors like sentence length and the complexity of words used.

After processing the data, I compiled the results into a structured format and saved them into an Excel file. This file includes all the metrics calculated for each text file, making reviewing and interpreting the findings easier.

Throughout the process, I focused on ensuring that the analysis was thorough and that the results would be useful for making data-driven decisions. My approach aimed to provide clear insights that could help in evaluating and improving the content based on the analysed metrics.

* **How to run the above jyupter.py file to generate output?**

**Running a Python Script in Jupyter Notebook or Jupyter Lab**

1. Open Jupyter Notebook/Lab:

Launch Jupyter Notebook or Jupyter Lab from your Anaconda Navigator or by running `jupyter notebook` or `jupyter lab` from your command line.

2. Navigate to the Directory:

Use the file browser in Jupyter Notebook/Lab to navigate to the directory where your `jupyter.py` file is located.

3.Create a New Notebook:

Create a new notebook by clicking on the “New” button and selecting “Python 3” from the dropdown menu.

4. Run the Python Script:

In a code cell in your notebook, you can use the following command to run your Python script:

**!python jupyter.py**

This command uses the `!` operator to run shell commands directly from a Jupyter Notebook cell.

5. Check the Output:

After running the command, check the output directly in the notebook cell or in the output file specified in your script.

**Handling Errors:**

If you encounter errors such as `PermissionError` or `FileNotFoundError`, make sure:

- The file paths in your script are correct and accessible.

- You have the necessary permissions to read/write files in the specified directories.

- All required files and dependencies are present and correctly referenced in your script.

* **All Dependencies Required.**

Including Dependencies in Your Script

1. Add Imports:

At the beginning of your `jupyter.py` script, make sure to import all the necessary libraries. For example, if your script uses `pandas`, `numpy`, and `openpyxl`, include these imports:

**import pandas as pd**

**import numpy as np**

**import openpyxl**

2. Handling File Paths:

If your script depends on specific file paths, ensure that the paths are correctly set and accessible. You might also include error handling for file operations.

3. Include Installation Instructions:

To ensure all dependencies are installed, you can create a `requirements.txt` file listing all necessary libraries. Here’s an example of what the `requirements.txt` might look like:

**pandas**

**numpy**

**openpyxl**

You can generate this file by running:

**pip freeze > requirements.txt**

**By: Prachi Gangwar**