Data Preparation and Cleaning (cleaning.ipynb)

The following steps were taken to clean and process the provided product data from the Max Mara, Net-A-Porter, and Luisa Via Roma websites.

1. Importing Necessary Libraries:

The code starts by importing the necessary libraries to handle data manipulation and cleaning, such as:

- **pandas**: For data frame operations.
- **numpy**: For handling numerical operations.
- **re**: For regex operations to extract specific product IDs from URLs.
- **rapidfuzz**: For approximate string matching to clean up product names.
- **nltk**: For natural language processing tasks, like stop word removal.

2. Loading the Data:

We load the three provided datasets (in `.jl` format) using `pd.read_json()`. This function is used to load the data into pandas DataFrames:

- `maxmara_df`
- `netaporter_df`
- `luisaviaroma_df`

Each DataFrame is then checked for basic information like column types and missing values.

3. Data Cleaning Functions:

- **clean_price()**: This function cleans the price data by handling missing or empty price values and returning them as `NaN` if necessary, or converting them into float values.
- **expand_dataframe()**: This function flattens and normalizes the nested data structure of the original DataFrames. The colors are expanded into individual rows, allowing each product to appear multiple times if it has multiple colors. It extracts the following attributes from the raw data:
    - `url`, `product_id`, `main_title`, `description`, `color`, `current_price`, `previous_price`, `sizes`, etc.

4. Product ID Extraction:

The function `product_id_extractor()` uses regular expressions to extract the product IDs from the URLs in the three datasets. Different patterns are defined for each source (Max Mara, Net-A-Porter, and Luisa Via Roma) to correctly identify the product ID from the URL structure.

5. Processing Each Dataset:

For each of the three datasets (Max Mara, Net-A-Porter, and Luisa Via Roma), the following steps are performed:

- **URL extraction**: Extract the correct URL for each product.
- **Product ID**: Extract the product ID using the regex function `product_id_extractor()`.
- **Main title and description**: Clean and convert text to lowercase, stripping extra spaces.
- **Material and brand**: Extract material and brand information from the nested 'details' column.
- **Price Handling**: Fill missing `previous_price` values with `current_price` if necessary, and create a `discounted` column to indicate whether a product is on discount.

6. Brand Filtering:

After processing, we filter out any brands from the datasets that don't match the Max Mara brand. The brands found in the Net-A-Porter and Luisa Via Roma datasets are then filtered and kept only if they also appear in the Max Mara dataset.

7. Saving Cleaned Data:

Finally, the cleaned datasets are saved into CSV files for further analysis:

- `maxmara_cleaned.csv`
- `netaporter_cleaned.csv`
- `luisaviaroma_cleaned.csv`

AI-Powered Classification (AI.ipynb)

This section of the project applies **OpenAI's GPT model** to standardize product information, such as **color** and **material**, by converting these attributes into commonly recognized or more famous names. The goal is to ensure consistency across product details for better analysis and reporting.

1. Libraries and Data Loading:

The following libraries were used:

- **pandas** and **numpy**: For data manipulation and handling.
- **matplotlib** and **seaborn**: For potential visualization (not directly used in this snippet, but could be useful later).
- **openai**: For calling OpenAI's API to generate classifications based on product descriptions.

- **ast**: For safely evaluating the response from OpenAI's API (converting it to a Python data structure).

The datasets `maxmara_cleaned.csv`, `netaporter_cleaned.csv`, and `luisaviaroma_cleaned.csv` were loaded into pandas DataFrames.

2. OpenAI API Integration:

An **API key** is set for OpenAI to interact with the GPT-3.5 model. The function `classify_product()` is designed to:

- **Send a request to OpenAI's API**: The product details (title, description, material, color) are included in a prompt that asks the model to return standardized categories for each product.
- **Sanitize the response**: The raw response from OpenAI is cleaned (stripped of unnecessary code block characters and extra spaces) before parsing it into a usable format.

3. Classifying Products in Chunks:

To avoid API limits, the data is split into smaller chunks (50 rows at a time) for processing:

- **Net-A-Porter and Luisa Via Roma** datasets are processed in batches using `classify_product()`.
- The results from each chunk are stored in `netaporter_list` and `luisaviaroma_list`, respectively.

4. Updating the DataFrames:

Once the classification is complete, a new DataFrame is created from the results:

- The `Index` column is used to ensure that the new classification data aligns with the original products.
- The classified data (standardized category, color, and material) is joined back to the original dataset.

5. Saving the Results:

The updated datasets, now with standardized classifications, are saved into CSV files:

- `netaporter_AI.csv`
- `luisaviaroma_AI.csv`

These files contain the additional columns for the standardized **Category**, **Color**, and **Material**, making the data more consistent and ready for further analysis.

Data Analysis (Analysis.ipynb)

This part of the project focuses on **data analysis** of product data across the three platforms: **Max Mara**, **Net-A-Porter**, and **Luisa Via Roma**. The analysis includes **price comparison**, **discount analysis**, **material distribution**, and **top product categories**. Here's an overview of each analysis section and the visualizations generated:

---

1. Price Comparison Across Platforms

- The **average price** for each platform (Max Mara, Net-A-Porter, and Luisa Via Roma) is calculated and compared.
- **Price differences** between **Net-A-Porter** and **Luisa Via Roma** relative to **Max Mara** are computed and printed as percentages.
- A **bar plot** is created to visualize the average prices across the three platforms. The plot is saved as `average_price_comparison.png`.

Insights:

- This comparison helps in understanding the price positioning of **Max Mara** products across the three platforms.

---

2. Discount Comparison Across Platforms

- The **discount rate** for each platform is calculated based on the difference between the **previous price** and **current price**.
- The **average discount rate** for each platform is computed.
- A **bar plot** is created to visualize the average discount rates across the platforms, saved as `average_discount_comparison.png`.

Insights:

- This analysis shows how aggressive the platforms are in offering discounts, potentially indicating pricing strategies or seasonal trends.

---

3. Material Distribution Across Platforms

- The distribution of materials used in the products listed on **Luisa Via Roma** and **Net-A-Porter** is analyzed.

- The top 4 most common materials are highlighted, and any remaining materials are grouped as **"Others"**.
- **Pie charts** are created for both platforms to visualize the material distribution. These are saved as `luisaviaroma_Material_distribution.png` and `netaporter_Material_distribution.png`.

Insights:

- The material distribution gives an understanding of the type of products offered (e.g., fabrics, blends), and helps identify any dominant trends in the materials used by these platforms.

---

4. Top Product Categories on Net-a-Porter vs. Luisa Via Roma

- The top 5 product **categories** by frequency are analyzed for both **Net-a-Porter** and **Luisa Via Roma**.
- **Bar plots** are generated to visualize the most common categories on each platform. The plot is saved as `top_categories.png`.

Insights:

- This analysis highlights the **category distribution** and helps identify if certain categories (e.g., dresses, shoes) dominate the platforms or if the platforms cater to different types of products.

Summary of Results

The analysis compares key aspects of the product data, including **pricing**, **discounts**, **materials**, and **categories** across **Max Mara**, **Net-A-Porter**, and **Luisa Via Roma**. The visualizations provide insights into the similarities and differences between the platforms, helping businesses and stakeholders make informed decisions regarding pricing strategies, inventory management, and product offerings.

Explanation of Streamlit Code (main.py)

The Streamlit app is designed to visualize and interact with the cleaned product data from three platforms: Max Mara, Net-A-Porter, and Luisa Via Roma. The app allows users to select and view datasets, perform basic statistical analysis, and explore different types of product analysis through a simple and intuitive interface.

Key Sections of the Code:

1. **Loading Data**: The `load_data()` function is used to load CSV files into pandas DataFrames. The `@st.cache_data` decorator ensures that the data is cached for better performance, preventing reloading every time the app is refreshed.
2. **Preloading Datasets**: A dictionary `file_paths` contains the paths to the CSV files for the three platforms. These files are loaded into a dictionary `dfs` with platform names as keys and DataFrames as values.
3. **Dataset Selection**: In the sidebar, users can select which dataset they want to view. The selected dataset is displayed in the main area of the app.
4. **Basic Statistics**: The `describe()` function is used to generate basic statistics for the selected dataset. This includes summary statistics such as mean, count, standard deviation, and unique values for each column.
5. **Analysis Options**: A selectbox allows users to choose from various analysis options, including:
    - Summary of product availability and pricing
    - Most popular colors
    - Discount analysis
    - Size availability analysis
6. Depending on the user's selection, the corresponding analysis is performed and displayed:
    - **Product Availability and Pricing**: Displays summary statistics for the current price, previous price, and discount status.
    - **Most Popular Colors**: Displays a bar chart of color counts in the dataset.
    - **Discount Analysis**: Calculates and displays the percentage of discounted items in the dataset.
    - **Size Availability Analysis**: Displays a bar chart showing the count of different available sizes.

Note on Incompletion:

Due to time constraints, the Streamlit app is not fully completed. Although the core functionality and the majority of analysis options have been implemented, some sections (such as the raw data display and additional analyses) remain unfinished. Further refinement and expansion of the app are possible with more time.