Methodology:

The scraper.py script scrapes eBay's Global Tech Deals page by:

- Setup: Configures a headless Chrome browser with a random user-agent.
- Scrolling: Scrolls the page to load all products.
- Data Scraping: Extracts product details like title, price, and shipping.
- Saving: Saves the scraped data into a CSV file.
- Execution: Calls the scraping function, saves the data, and handles errors.
- Termination: Closes the browser after completion.

The process dynamically loads and saves all product information for analysis.

The scraper.py script was added to the GitHub workflow actions and configured to run every 3 hours through the scrape.yml file. This setup automates the process of scraping eBay's Global Tech Deals page at regular intervals, ensuring that the product data is updated and saved periodically without manual intervention.

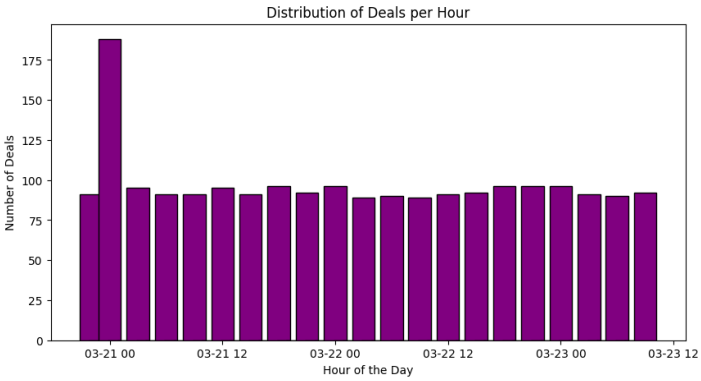
The clean_data.py script cleans the scrapped data in the ebay_tech_deals.cav by:

- Load Data: Load the CSV file with all columns as strings.
- Clean Numeric Columns: Remove unwanted characters from price and original_price, then convert them to numbers.
- Handle Missing Values: Fill missing original_price with price and set missing shipping info to "Shipping info unavailable".
- Convert to Numeric: Convert price and original_price to numbers, handling errors.
- Calculate Discount: Create a new column for the discount percentage.
- Remove Invalid Rows: Drop rows with missing essential data.
- Save Cleaned Data: Save the cleaned data to a new CSV file.

After cleaning the data, I started visualizing it in the EDA.ipynb

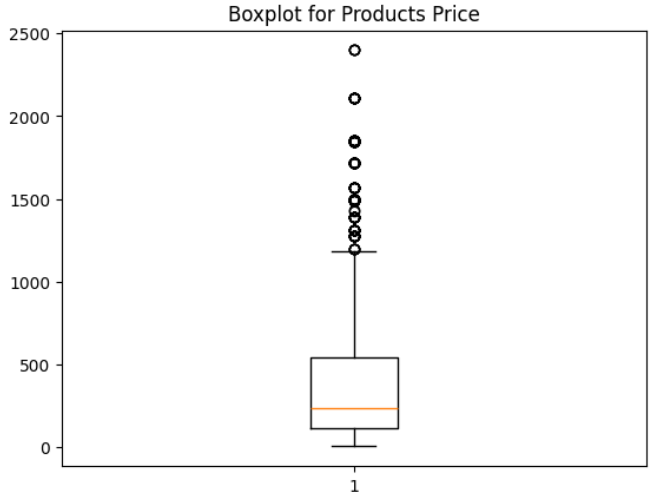
s

Key Findings from the EDA



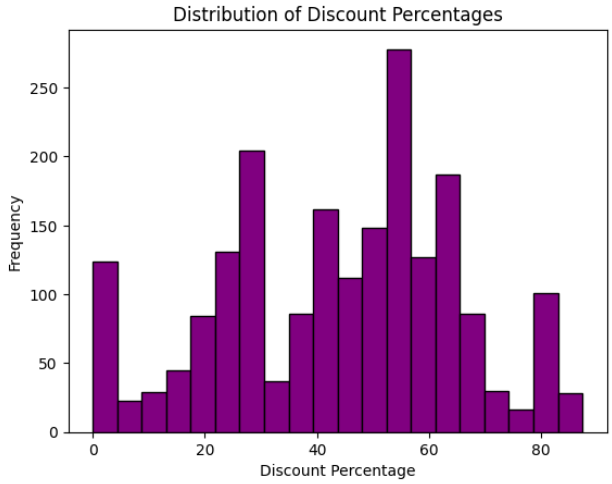
Deal activity fluctuates throughout the day, with peaks around 03-21 12 and 03-22 12, and lows around 03-21 00 and 03-23 00, indicating varying deal volumes by time.

Most products are priced low (0-500 range), with fewer high-priced items, suggesting affordability is common.



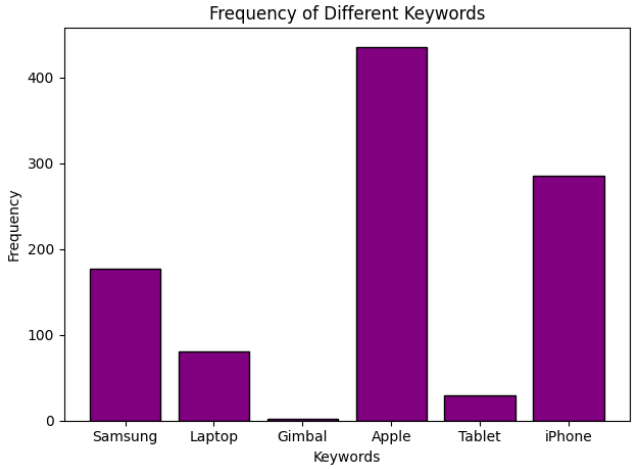
The boxplot shows most product prices are low, with a median around 500, and few high-priced outliers.

Current prices are generally lower than original prices, indicating widespread discounts, with larger discounts on higher-priced items.

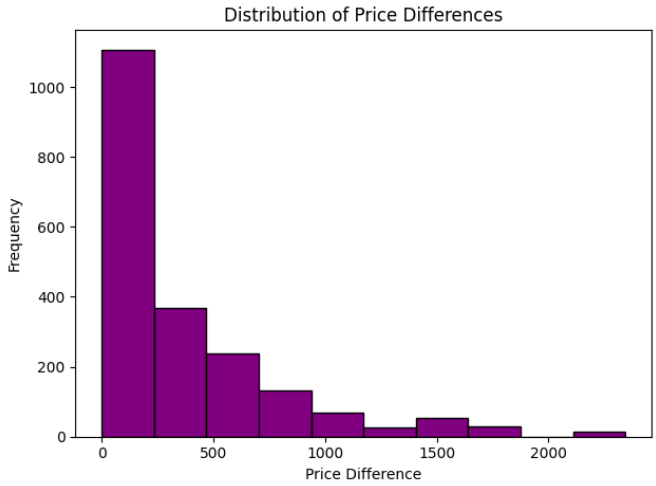


Most discounts are small (0-20%), with (50-59%) being the most frequent and larger discounts (60-80%) being less common.

Free shipping is the most frequent option, with few products lacking shipping info.



"Laptop" and "iPhone" are the most frequent keywords, indicating higher interest compared to less common items like "Gimbal" and "Tablet."



Most price differences are small (0-500), with larger differences being rare, suggesting minor price variations are common.

Challenges Faces:

During the analysis, I faced a few syntax issues, such as needing to convert counts from np.int64 to regular Python integers (int(count)) for clearer output. Additionally, to plot the keyword counts, I converted the keyword_counts dictionary into a pandas Series, which made it easier to use for plotting with matplotlib.

These issues were resolved by thoroughly reading the documentation for both pandas and matplotlib, which provided detailed explanations and examples on how to handle data types and plot more efficiently.