E-Commerce Data Analysis Dashboard

Yar Moradpour and Amila Mesic

San Jose State University

CS-22B Sec 01 - Python Data Analysis

December 3, 2023

Table of Contents

Background:

When visualizing this project, we were inspired by the growing importance of using data to make smart decisions in online selling. Since the demand for online markets is getting bigger and bigger, in a sense it's going to be more practical for businesses to utilize a simple and user-friendly tool to help them understand things such as how much their selling item is most in demand within customers. This dashboard is envisioned to be a helpful guide for e-commerce businesses and an interactive way for them to visualize important product data. Our mission was to create an analytical dashboard that is user-friendly and easy to navigate. The dashboard utilizes data from Walmart's e-commerce services through an API from a site called Axesso. The data includes information such as item name, item price, seller name, rating, and number of reviews. The Walmart API we chose to use collects realtime up to date data from the Walmart website. By cleaning and visualizing this data through an interactive dashboard, we aim to gain valuable insights into the performance of our products and sellers. With this tool, businesses can now confidently plan their path forward, knowing exactly how they are doing in the digital marketplace. This deep understanding not only helps them handle challenges but also tends to grab a hold of new opportunities, leading to remarkable growth to success.

Problem:

In the expansive realm of online commerce spanning the web, a wealth of valuable information is routinely gathered. We were faced with the challenge of finding useful ecommerce data, cleaning it to create a usable dataframe, creating descriptive visualizations, and creating a dashboard from scratch. The information we portray on the dashboard is one of the most important parts of completing this project in the highest quality. Key questions include

identifying top-performing sellers, understanding the correlation between item prices and customer ratings, and visualizing trends in item prices over time. To do so we used Python libraries like Pandas, Numpy, Matplotlib, Seaborn for data manipulation and visualization. To complete the dashboard creation process we used Axesso API to access necessary data and multiple python functions to display the correct information on the dashboard. The development environment we decided on was Google Colab and VS Code for seamless collaboration. The ultimate goal of the project was to create an interactive dashboard that presents useful insights for companies in the e-commerce industry. Our data utilizes information from Walmart's website, but our code can ultimately be applied to any e-commerce data set.

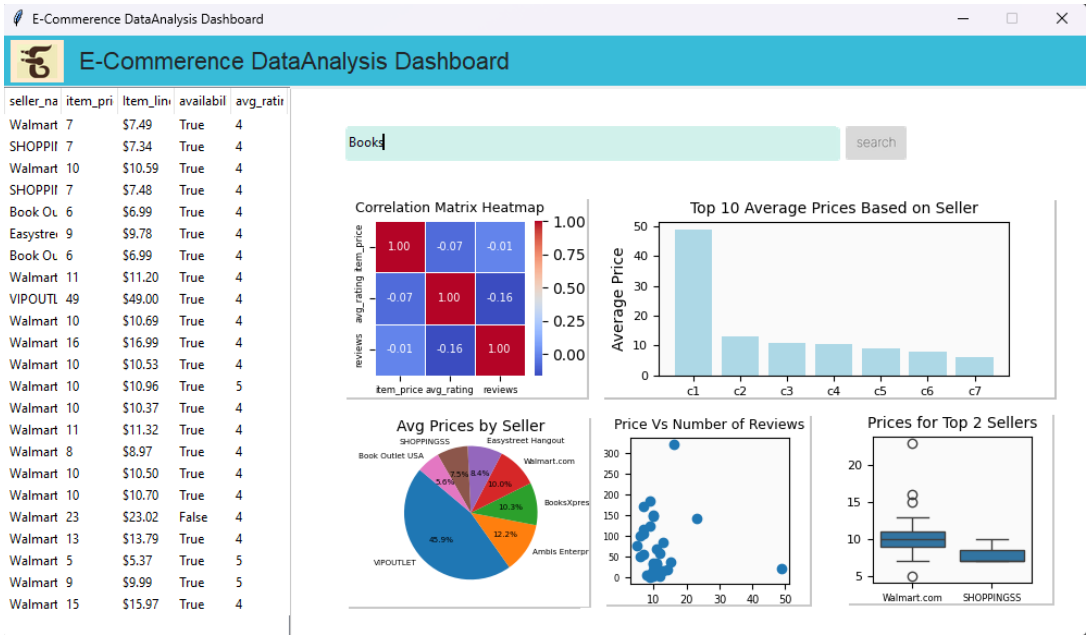
Organization:

First, we had to receive the API key to access the updated data everytime we ran our code. Once we gained access to the data we had to parse through data and understand the structure of the data to extract the information we thought was most important. This step allowed us to create a simple data frame that was easy to work with, giving us free range to develop the charts and graphs that would be incorporated into the dashboard. The next step was creating the charts and graphs. To start, we developed a correlation matrix, not including the availability column because the data type is boolean which throws off our findings, using the .heatmap function in seaborn. Then using the **.plot** function, we created a bar chart that only illustrated the top 10 average prices based on the Seller name. We mimicked the same step to create the box plot, but using the matplotlib **pie** function. To understand if there was any significant coordination between the price of a product and the number of online reviews the product got we created a scatter plot using the seaborn **relplot** function. Finally, to illustrate the distribution of

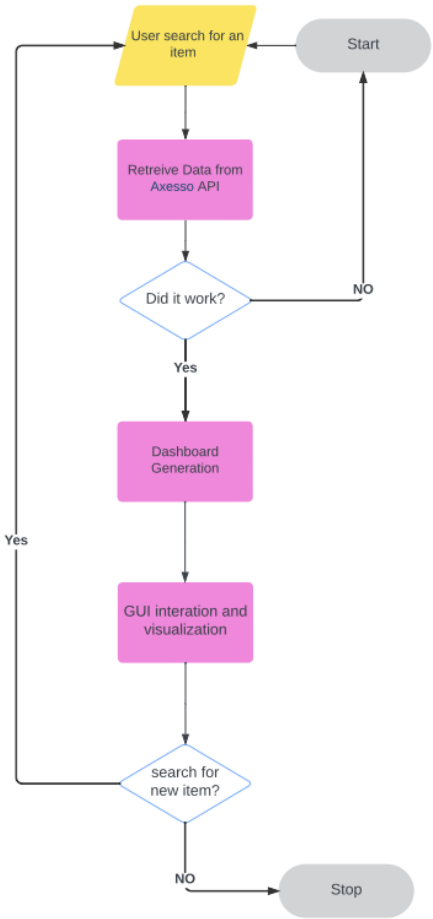
prices for the top 2 sellers we created a box plot using the seaborn **boxplot** function. After finalizing the plots we moved onto creating the dashboard.

To begin, we created a function called **print_entry_content()** which allows us to have a search box that creates a new data frame based on the searched keyword. By searching a keyword the function goes to the API and retrieves the data that matches. The function also runs two other methods known by **create_matplotlib_plots()** and **display_dataframe**. The first method being **create_matplotlib_plots** tends to recreate the plots based on the updated data frame generated based on the new search result and the second method **display_dataframe** tends to update the table located on the left of the dashboard to the new dataframe. Next, we created 6 widgets to map out the areas we wanted our plots and data table to be displayed. Everytime a new keyword is searched the functions will update the necessary information to develop new graphs and place them into the designated widgets. The analysis of the visualizations are subject to change due to the ever changing data based on real time updates and keyword searches. In that case the dashboard would need to be interpreted by an individual that has experience with data visualization. In the end, our efforts came together in the creation of an interactive dashboard that shines a spotlight on crucial performance indicators within our e-commerce data, all neatly organized by sellers. This dynamic dashboard provides a user-friendly interface for all. Analysis

Program Design:



Program Flow chart:



Code Instructions:

Step 1: set up project dependencies:

Use terminal to navigate to the root directory of the project before proceeding any further

Step 2: Install necessary Libraries

When inside the root directory execute "pip install -r requirements.txt" command to install necessary dependencies for the program to properly function.

Step3: finally running the dashboard!

Once all the dependencies and libraries are installed run "python gui.py" command and the program should be functional within three seconds!