Samantha Cohen
Final Project Report

An Analysis of the Relationship Between Product Review Star Ratings and Comments

I. Introduction:

The goal of my project is to examine the relationship between comment reviews and the star ratings that individuals write about products after they buy them. My inspiration for this project stems from my interest in learning more about ways in which sentiment analysis can be used to answer investigative questions. I often read comment reviews on products before buying them. This gave me the idea to use Sentiment Analysis to investigate the relationship between users' star ratings and their comments. The goal of my project is to answer this question: What is the relationship between electronic product review star ratings and comment rating reviews?

For this project, I used Datumbox Sentiment Analysis API. I will use a csv file that shows product review data from Amazon and Best Buy. I will run sentiment analysis on the text comment reviews from this csv file. I will then compare the sentiment analysis score from Datumbox (positive, negative, or neutral) to the star rating that the consumer gave the product (this information is located in the csv file with the product reviews). The star rating that each user gave the product is between 1-5. I will than compare the relationship between the Datumbox sentiment analysis for that comment compared to the star rating.

In the folder SI330-FinalProject-Samcoh, my code for this project is located in the file called SI330-finalproject-samcoh. The cache file that stores the Datumbox Sentiment Analysis API responses is cache_analysis_2.json. The CSV used is DatafinitiElectronicsProductData.csv.

I. Data Sources:

Data Source #1:

- ***Name:*** DatafinitiElectronicsProductData
- ***Size***: 7300 Rows of data
- ***Location***: https://www.kaggle.com/datafiniti/amazon-and-best-buy-electronics/home
- ***Format***: CSV file
- ***Access Method***: Downloadable CSV file

Data Source #2:

- ***Name***: Datumbox Sentiment Analysis API
- ***Size***: will determine if 7300 reviews are positive, negative, or neutral (call API 7300 times)
- ***Location***: http://www.datumbox.com/machine-learning-api/
- ***Format***: returns the results as a JSON object
- ***Access Method***: HTTP requests

II. Data Processing Steps:

There are four main parts of this project. The first step of this project is to import modules needed to process the data: pandas, requests, json, numpy, and altair.

Part One*:* Put the CSV file into a Pandas DataFrame and Filter/Manipulate the Columns

1. Open the csv file with electronic product reviews, called a DatafinitiElectronicsProductData.csv. Create a Pandas DataFrame called df.
2. Create a new Pandas DataFrame, called new_df, with just the columns you want. This step filters out the unnecessary columns. The new columns: id, name, brand, dateAdded, reviews.username, reviews.rating, reviews.text, and sourceURLs.
3. To clean up the new Pandas DataFrame further, I will rename the columns in new_df: id, product, brand, review_date_added, username, review_rating, review_text,and sourceURLs.

Part Two*:* Get Sentiment Analysis Data for the review_text column in the DataFrame using the

Datumbox API, and store responses into a cache file.

1. Get a token from the Datumbox Sentiment Analysis API.
2. Cache the Sentiment Analysis Data in a json file called cache_analysis_2.json. This file stores a dictionary. The keys are indexes from the Pandas DataFrame for the text comment. The values are the responses from the Datumbox API for each comment.
   a. Assign CACHE_FNAME to "cache_analysis_2.json"
   b. Try opening the cache file and load its contents. If you can open the file assign contents to CACHE_DICTION
   c. If the cache file does not exist create an empty dictionary called CACHE_DICTION
3. Function make_request_using_cache: purpose is to get response data from the cache file if it is there. If the response data is not in cache, call Datumbox API and add response to cache file.
   a. Parameters: baseurl, params, and the index of the text comment (review_text) in the Pandas DataFrame
   b. Returns: the response from the Datumbox API (either from cache file or call the API)
4. Function get_datumbox_caching: purpose of this function is to create the params dictionary for the request. Once creating the params dictionary, the function will call make_request_using_cache with baseurl, params, and index.
   a. Parameters: review_text comment from the pandas Dataframe, and the index of this review_text comment.
   b. Returns: the response from the Datumbox API
5. Function api_call: purpose is to stop calling the Datumbox API if it returns an error. Will call get_datumbox_caching function.
   a. Parameters: a dictionary. The keys are the index from the pandas DataFrame, and the values are the review_text associated with that index.
   b. Returns: if runs successfully return the string "ran". If it does not run successfully, it will return the error dictionary from Datumbox API.
6. Datumbox API has a limit of 1000 calls per/day. Create functions to get the sentiment analysis from the last review_text left off at.
   a. Function last_index_gotten: purpose is to see the last review_text left off at.
      i. Parameters: none
      ii. Returns: last index in the cache file. If cache file is empty return the index of 0
   b. Function get_1000_rows: purpose is to get the next 1000 review_text. Will form a dictionary (keys = index, values = review_text). Call api_call with dictionary.
      i. Parameters: an integer that indicates the last index left off at from the cache file.
      ii. Returns: none. Prints ran, if the api_call ran successfully, or error.
7. Call last_index_gotten() and assign response to the variable index. Call get_1000_rows with index. Run line of code everyday until see 'json file has all the sentiment analysis data'

Part Three: Combine data: Once all sentiment analysis data is in cache, put it in the DataFrame.

1. Open the cache file and get the data. Assign contents to the variable responses.
2. Use responses to create a list of all responses for each index (in order). Assign the list to variable evaluations.
3. Create a new column in the DataFrame with Datumbox responses (assigned to evaluations)

Part Four: Create Visualizations to use for analysis

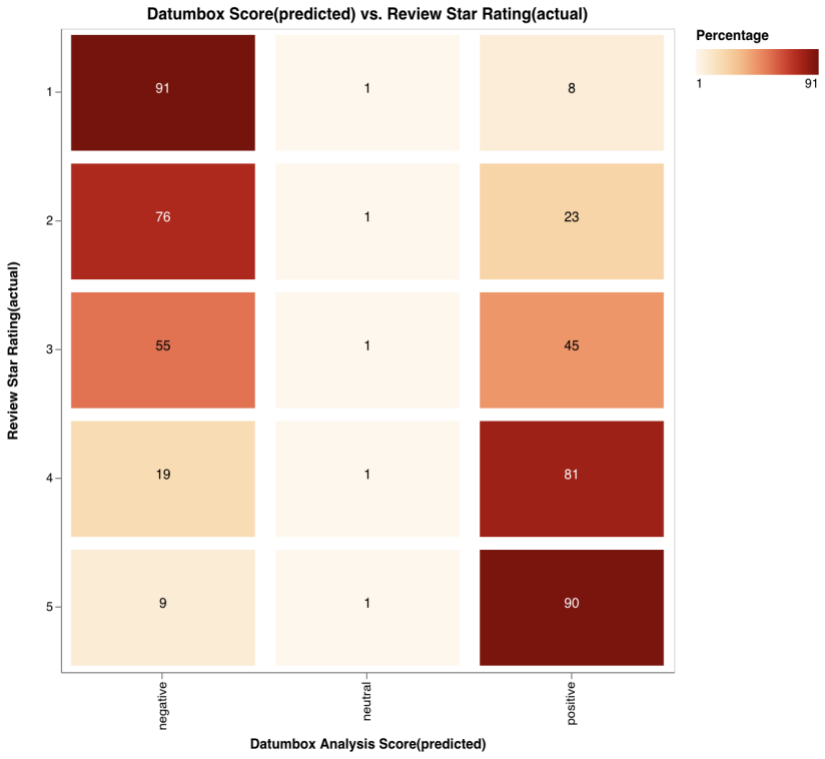
1. Create a new pandas DataFrame called graphing. Use new_df and groupby "review_rating" and "Datumbox_Score. Include a count for each row. This column should be called counts.

2. Create a heatmap where you normalize by review_rating (row)
    a. Create a new column in graphing that shows sum of the counts for each group in review_rating. The new column is called Sum_of_Counts in DataFrame Graphing
        i. Get sum of counts for each group in review_rating. Group graphing DataFrame by review_rating and get a sum of counts for each group. Assign result to sums
        ii. Create a dictionary using pandas with sums. Assign the result to the variable d
        iii. Create a new column in graphing called Sum_of_Counts. Initially, make the values in this column the same values in review_rating.
        iv. Replace values in Sum_of_Counts with the sum of counts in dictionary, called d. Sum of counts for each rating are nested: d[“counts”][insert int rating score (1-5)]
    b. Create a new column in graphing that shows a percentage for each row. This percentage is calculated for each row in each review_rating group. This percentage is calculated by doing (counts for that row)/(sum of counts for that review_rating group)
        i. Assign variable percentages to the array counts/Sum_of_Counts
        ii. Create an empty list called percentages_rounded. Loop through array percentages and round each decimal 3 decimal places, multiply by 100 (to get a percentage), and append result to the list percentages_rounded.
        iii. Create a new column in graphing called Percentage with percentages_rounded
    c. Create the first heatmap visualization using Altair
        i. Create one chart that is the heatmap and assign it to the variable heatmap
            1. X axis is the Datumbox_Score from the pandas DataFrame graphing
            2. Y axis is review_rating from the pandas DataFrame graphing
            3. Use the Percentage column from the pandas DataFrame graphing for the color. The color shown is based on how high or low the percentage is (the higher the percentage the darker the color—highest is dark red and lowest is light orange)
        ii. Create a chart to show text percentage on the heatmap. Assign it to the variable text.
            1. X axis is the Datumbox_Score from the pandas DataFrame graphing
            2. Y axis is review_rating from the pandas DataFrame graphing
            3. The color of the text is changed depending on the Percentage column from the pandas DataFrame graphing so it is easily readable
                a. If percentage is over 70% color text is white, else the color is black
        iii. Combine these two charts so the percentage is shown (heatmap + text).
3. Create a heatmap where you normalize by Datumbox_Score (columns)
    a. Create new column in graphing that shows sum of the counts for each group in Datumbox_Score. This column should be called Sum_of_Counts_Datumbox_Score.
        i. Get the sum of the counts for each group in Datumbox_Score. Group graphing by Datumbox_Score and get a sum of counts for each group. Assign result to sums_2
        ii. Create a dictionary using pandas with sums_2. Assign the result to the variable d2
        iii. Create a new column in graphing called Sum_of_Counts_Datumbox_Score. Initially, make the values in this column the same values in Datumbox_Score.
        iv. Use .replace to replace the values in Sum_of_Counts_Datumbox_Score. Replace these values with the true sum of counts located in the dictionary created above, called d2. The sum of counts for each Datumbox_Score are nested: d[“counts”][insert Datumbox Score str (positive, negative, or neutral)

b. Create a new column in graphing that shows the percentage for each row. This percentage is calculated for each row in each Datumbox_Score group. [Calculated by doing (counts for that row)/(sum of counts for Datumbox_Score group)]
   i. Assign variable percentages to the array counts/Sum_of_Counts_Datumbox_Score
   ii. Create an empty list called percentages_rounded. Loop through array percentages and round each decimal 3 decimal places, multiply by 100 (to get a percentage) and append to the list percentages_rounded.
   iii. Create a new column in graphing called Percentage_Datumbox_Score with all the percentages in percentages_rounded
c. Create second heatmap visualization using Altair
   i. Create a visualization that is the heatmap and assign it to the variable heatmap
      1. X axis is the Datumbox_Score from the pandas DataFrame graphing
      2. Y axis is review_rating from the pandas DataFrame graphing
      3. Use the Percentage_Datumbox_Score column from the pandas DataFrame for the color. The color shown on the heatmap is based on how high or low the percentage is (the higher the percentage the darker the color—highest is dark red and the lowest is a super light orange)
   ii. Create another visualization to show the text percentage on top of the heatmap, and assign it to the variable text
      1. X axis is the Datumbox_Score from the pandas DataFrame graphing
      2. Y axis is review_rating from the pandas DataFrame graphing
      3. The color of the text is changed depending on Percentage_Datumbox_Score column from the pandas DataFrame graphing so it is easily readable
         a. If percentage is over 30% the text color is white, else the color is black
   iii. Combine these two charts (heatmap + text).

III. Results:

The first visualization I created, shown to the right, normalizes by review_rating. I created this heatmap using altair. I used the graphing DataFrame, as described in the data processing section, to create this heatmap. The x-axis is the Datumbox_Score (represents the Datumbox Analysis Score: positive, neutral, or negative) . The y-axis is the review_rating (represents the Review Star Rating: 1-5). The color represents the Percentage score. The darker the color red the higher the percentage. In the data processing section I explain how to make this chart in detail.

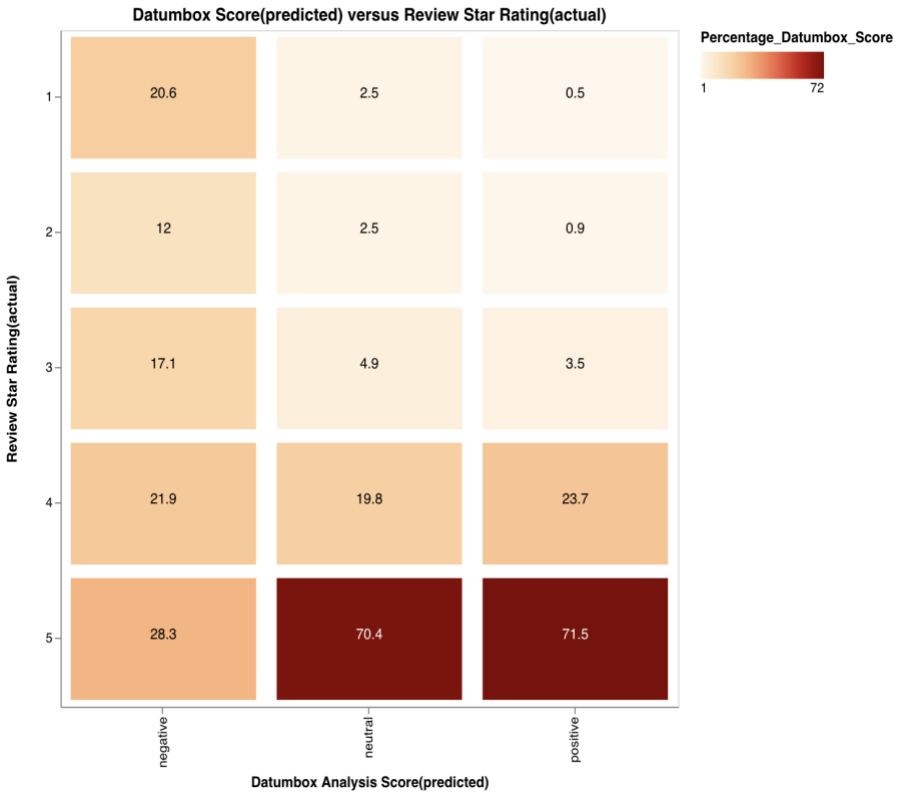


This heatmap shows the strong relationship that exists between Review Star Rating and Datumbox Analysis Score. When given a review rating score of 1,2,4, and 5 there is a high likelihood

an individual can predict the sentiment analysis score. For example, when a review star rating is 1 there is a 91% chance that the sentiment analysis score will be negative. Although, when a rating score is 3, there is a lesser of chance of predicting an accurate Datumbox Analysis Score. This could be because 3 is a relatively neutral score, and Datumbox rarely categorizes a comment as neutral. Datumbox will push to categorize the comment as negative or positive.

Overall, the outcome of this chart is what I expected. I thought there would be a relatively strong relationship between review star ratings and Datumbox analysis score for the comment. This chart shows that there is an even stronger relationship than what I predicted.

The second visualization I created (shown below) normalizes by Datumbox_Score. I used the graphing DataFrame, as described above to create this heatmap. The x-axis is the Datumbox_Score (represents Datumbox Analysis Score: positive, neutral, or negative) . The y-axis is review_rating (represents the Review Star Rating: 1-5). The color represents the Datumbox Score Percentage. The darker the color red the higher the percentage. In the data processing section I explain in detail how I created this chart.



When given a sentiment score that is positive or neutral there is a high likelihood that you can predict the Review Star Rating. For example, when a Datumbox Score is positive there is 71.5% chance that the review star rating is 5. Although, when the sentiment analysis score is negative it is hard to predict the review star rating. This is because the data is skewed due to the fact that there are less comments with a sentiment score of negative.

IV. Discussion and Conclusion:

If I had more time I would extend this project by investigating other Sentiment Analysis APIs. It would be interesting to see if other APIs give similar responses for comments. My expanded investigative question would be: How does the relationship between review star ratings and sentiment analysis scores change when using other APIs?

I have learned a lot from this project: ways in which sentiment analysis can be used to answer investigative questions, strategies for handling API limitations, and gained experience with analyzing and creating relevant visualizations. When starting this project I knew I wanted to develop an investigative question that involved using sentiment analysis. By pushing myself to think about ways in which I could apply sentiment analysis to different investigative questions I learned about the uses of sentiment analysis.

Datumbox had a limit of 1000 calls per day. I was able to work around this limitation by writing code that cached results, and called the API with the last comment I left off with whenever I ran the program. Fetching the data was one of the most time consuming processes of this whole project.

Finally, coming up with and producing the visualization for this project has expanded my skills in data analysis. Producing a chart that best represents your data helps one see patterns and relationships that is otherwise hard to identify with just a table. I have also gained experience using altair. I feel much more confident about choosing effective visualizations and producing them because of this project.