Tech Bootcamp Project Report
MSBA Team 7
Chris Chou, Zackery Goldberg, Pandora Shou, Tiffany Zhao

Amazon Web Services offer access to datasets that contain review information about different types of products. For this project, we use Python to transfer data between a web server, SQL, and R; then, we use SQL to clean and analyze the data; finally, we use R to conduct exploratory data analysis.

Cleaning the Data

After uploading the data to the SQL server, we cleaned the data step by step before further analysis. The first step is text processing. When we looked at the review_body column, there were a lot of unnecessary symbols, such as ':)', ',', 'and', 'or', etc. We removed these characters to reduce the data size so that text mining will be easier in the future.

Next, we standardized the data. We noticed that the headline column and review_body column are in mixed uppercase and lowercase, which is a problem if we want to filter distinct values. Therefore, we changed the data in these two columns into lowercase.

Lastly, we removed redundant attributes. After exploratory data analysis, we found only one value in the marketplace column -- "US". A column that contains one class doesn't add any value to our prediction. Therefore, we dropped the marketplace column.

Analyzing the Data

Next, we analyzed the cleaned data to dig for more insights. For summary, we have 2791929 rows and 15 columns in total after appending the pet. Then we used Distinct Count to count the unique values of products and customers, and found there exist 241117 unique products and 207944 customers. By calculating the unique combinations of products and customers, we found there are 241255 types of combinations, which is higher than both the product type and the number of customers, which indicates that there may be customers purchasing different types of products, which coordinates with real-life situations.

After analyzing the basic attributes of the data frame, we look into the statistics side.
By comparing and visualizing the star rating for all purchases and verified purchases, we concluded that the verified purchases have lower star rating compared to average. This is actually a warning for the customers, as it indicates potential marketing fraud which involves some unfair star rating for unverified purchases. These unverified purchases might have been

returned as soon as delivered or just from the partner's help to make the rating look better, which is suspicious and unreliable in either situation.

For better visualization, we connect SQL databases to R through the "RMySQL" package, and we fetch the data into a new data frame. Then we used the ggplot package to plot the distribution of star rating, the plot shows star rating aggregates at 5 points, which seems like an overall good comment.

However, in question 7, when we created a metric and excluded those unfair data, the average decreased a lot. This proves the validity of out metric and indicates the potential unfair and bias in star rating. We also aggregate time series data into monthly and yearly data in question 8 and question 10 relatively. The trend shows that there exists a seasonality. In the second quarter, the frequency is extremely high. While in the first and the last quarter, the frequency is reduced.

Based on unfair validation and data analysis, we concluded that the vine program can be retained, but it should be managed on the star rating and reviews function to avoid potential unfair or biased comments.

Overcoming the Challenges

Throughout the project, we encountered several technical difficulties. When we attempted to upload data to the SQL server using Python, our biggest challenge is cutting down the program running time. Initially, we picked the sql-connector package to connect to the SQL server, which allowed us to upload data one row at a time. This way, we wrote simpler code, but uploading the entire gift card dataset (148,310 rows) took 2 hours, not to mention that the gift card dataset is the smallest of all datasets! To optimize the algorithm, we found the to_sql function from the sqlalchemy package, which inserts data by bulks. Since sqlalchemy requires that we use pyodbc as the connection driver, we had to re-do all the code about connecting to the SQL server. However, our efforts paid off. We eventually cut down the python program running time to under 3 minutes. As the result, we are more flexible in test-running the python script, and we can manipulate the data using MySQL without worrying about accidentally making mistakes.

Learning from the Project and the Bootcamp

Looking back at the 8 days of the intense boot camp, we have not only improved our coding skills but also learned a lot about rules in the programming world.

First, always keep in mind what we want to accomplish. If we are trying to query and analyze the data, we can use SQL. If we are trying to conduct hypothesis tests and create visualizations, we can use R. If we are trying to build machine learning models or even use one language to do everything, python will be our best choice. By using the right language for the right task, we can

maximize each language's utility. What we aim for also influences our choice of the package/driver. Therefore, we should be clear about our goal before jumping into coding.

Also, coding is not done after a script can function properly. The understandability and speed of the script also impact the performance a lot. If we just pile everything together without formatting and commenting, when we go back to one particular step for debugging, even the coder cannot trace the code. The speed of running the script also matters. From the project perspective, cutting down running time allows more opportunities to make mistakes. The other team members have more time to finish their parts of the project. There is also less chance of program interruptions, which leads to fewer potential data corruptions. Therefore, even when the code can achieve the desired function, we should always aim to optimize the algorithm.