The Trio

Retail Therapy!

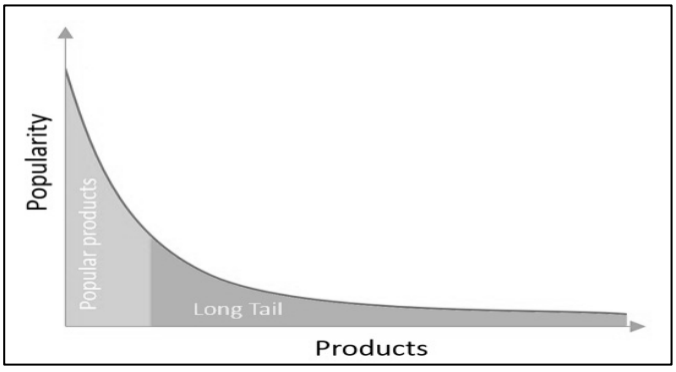
Market Basket Analysis- Recommendation System

Team Members:
Akansha Jain
Swapna Bandaru
Ranjani Anjur Venkatraman

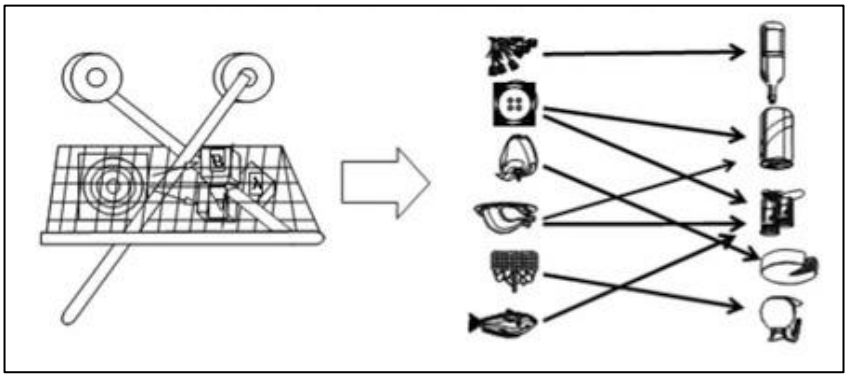
Purpose and Introduction

Recommendation systems have been widely popular in recent times. Netflix, Amazon, LinkedIn are a few examples of organizations that affluently use the recommendation models to provide the best of their services to the customers. The underlying engine collects information about people's habits to predict that if people buy commodity A and B, they are usually also interested in C. For example, while at Costco if a person buys bread and cereal, they are more likely to end up buying milk than someone who did not buy cereal. The response is made based on the customer's past purchasing behavior. Recommendation systems have advanced from being a novel idea used by only a handful, to an acquired aspect of everyday businesses. These systems are a crucial element of today's business world, as they enable organizations to efficiently target consumers with their products and services, thereby revolutionizing the e-commerce landscape. There are different techniques for providing recommendations, namely matrix factorization, collaborative filtering, and association rule-based methods. In our project, we are concentrating on using association rule-based recommendation approach targeting a consumer goods retail store.

In retail, the recommendation systems are used to perform market basket analysis explicitly created to promote sales. It is an unsupervised data mining and analysis technique that discovers the co-occurrence of relationships or associations among customer's purchase behavior. In layman terms, market basket analysis works by looking for combinations of items that frequently occur together in transactions. In retailing, most items are bought on impulse. Market basket analysis gives hints as to what a customer might have purchased if the idea had occurred to them. However, not all relationships are as apparent as the example of milk, bread, and cereal. The foresight of consumer behavior can help increase sales and give the retailer a significant edge over other competitors in the market. For example, there exists a well-known case of beer and diapers: the impossible correlation. Retailers can identify relationships among the products that people buy and utilize this data to design new products or pricing models to generate increased revenue. Market basket analysis can be used by retailers to cross-sell the products i.e. selling related or complementary products to an already existing customer. It can also be used by websites to recommend associated products that are frequently bought together. Typically, there are quite a number of popular items that users interact with a lot and others that they don't. These are the items that comprise of what is known as the "Long Tail". Recommendation systems work reasonably well on popular products, although that's probably not very interesting to users as they might already know about them. The products in the Long Tail are the most interesting ones, because they may not be considered by the user at all if the items were never recommended to them or if the thought of these items never occurred to them.



Association rule mining finds the association between data items, find patterns in relational database or any other information repository. It is simply the study of what goes with what.



Association rules follows if – then statement to uncover how items are related to each other. The major application is to find sales correlations in transactional data of retail industry. They are transaction based or event-based analysis, also called as market basket analysis or affinity analysis. Its widely used in "Recommendation system". By analysing the historical data, the rule generates items which are brought together by customers. The result is in the form of if this then that. These results can be used by clients for following marketing strategies:

- To change store layout according to trends
- Cross marketing of products
- To send customized email with add on sales
- Offers on products brought together
- To analyze customer behavior

An association rule is divided into two parts namely antecedent(if) and consequent(then). Antecedent is an item found in a database. Consequent is items brought in association with antecedent. Antecedent and consequent are disjoint.
For example:
Avocado -> Lime
Here avocado is antecedent, and Lime is consequent. If avocado is purchased, then lime is also bought together.
Three common ways to measure association are:

Support: Support is percentage of transaction that include an itemset. It gives the information on how popular an itemset is. An itemset can have multiple items. Support count gives information about frequency of transaction.
Support(X) = Frequency(X)/N
X is item-set and N is total number of transaction.

Confidence: Confidence is percentage of antecedent transaction that also have consequent item set. It is likelihood that consequent will be found in transaction with antecedent.
Confidence(A=>B) = P(A∩B) / P(A) = frequency(A, B) / frequency(A)
The above formula represents number of transactions that include both A and B divided by total transaction having A.

$$Support = \frac{frq(X,Y)}{N}$$

$$Rule: X \Rightarrow Y \rightarrow Confidence = \frac{frq(X,Y)}{frq(X)}$$

$$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$$

Support and confidence are used to identify the most important relationships among itemset in the database. It is set by min support and min confidence threshold. Closer to threshold the rule is more useful to clients. The rule which satisfies minimum support and minimum confidence the it is classified as strong rule. Frequent itemset are ones whose support is greater or equal to minimum support threshold.

Lift: Lift is ratio of confidence divided by benchmark confidence. Benchmark confidence is percentage of transaction with consequent to all transaction. The rule shows how effective it is in finding consequent.
Lift((A=>B)) = P(A∩B) / P(A) * P(B)
Gives the correlation between A and B, that mean it shows how one item-set(A) effects another item-set(B).
If the lift value is 1, then A and B are independent, and no rule can be derived from them.
If lift value is greater than 1, the A and B are dependent on one and other.
If lift value is less than 1 then the presence of one itemset have negative impact on the other one.
Always we should select the rule which has lift value greater than 1.

Apriori algorithm is an algorithm for finding frequent itemset for association rule learning over a transactional database. An algorithm is named apriori because it uses the prior information of frequent itemset properties. It is an iterative process where k frequent items are used to find k + 1 itemset. Below are two essential phases of the algorithm.

1. Frequent item-set generation-Apriori is a popular algorithm for extracting frequent item-set to apply in association rule. The apriori algorithm has been designed to work on the transactional database, such as purchases by customers of a store. An itemset is considered as "frequent" if it meets a user-specified support threshold.

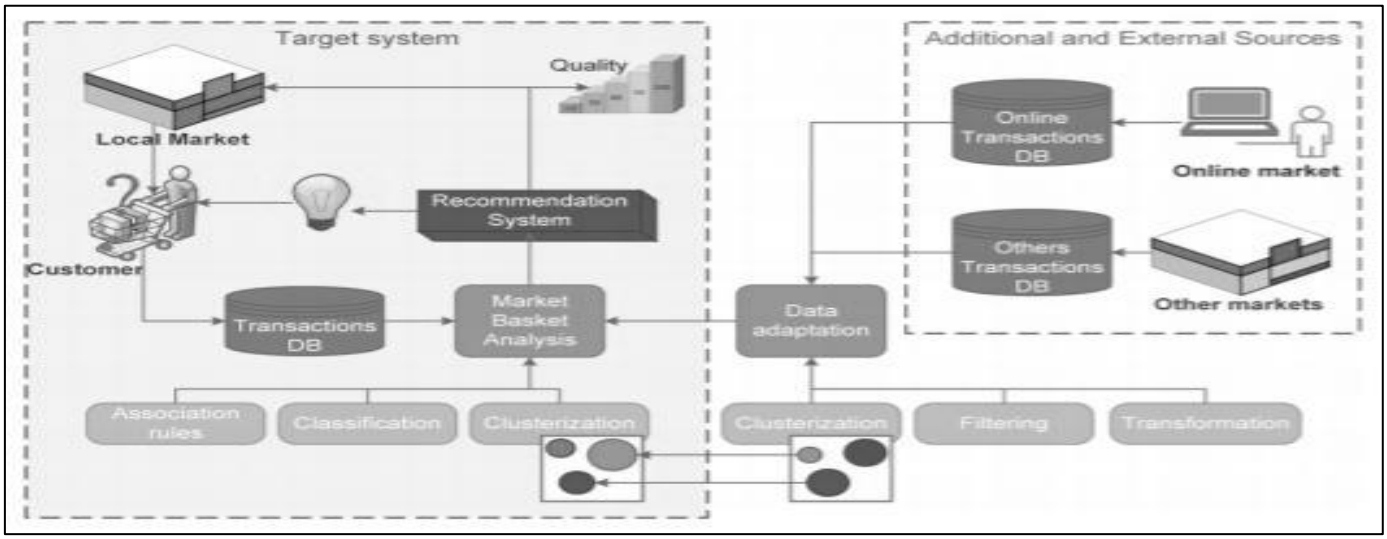
2. Rule generation-Should generate rules for frequent item-sets. Then calculate support and confidence. Remove the ones who fail minimum support and confidence threshold level.

A grocery ordering and delivery app intends to make it effortless to fill your refrigerator and pantry with your favorites and staples whenever you need them. The capability to identify which products the users will probably buy again, and automatically adding them to the cart through the predictions obtained and thereby render a seamless interface for doing so will enhance their user experience. Once the customer selects products through their app, personal shoppers review the order and do the in-store shopping and deliver it to the customer. The project aims to use the consumer's purchase data to generate models that will be able to predict the items a particular user will order again, will try a product for the first time ever, or add a particular product to their shopping cart next. Using these machine learning models, we will be able to predict the user's next order based on the previously purchased products. The obtained predictions can be sent out to users in the form of personalized communications giving them a subtle notification or reminder to order these products.

Data Sourcing

Our project aims at building a recommendation engine based on market basket analysis for an online retail store and suggest products to the customers similar to what they have put in their carts. The data for performing the analysis is downloaded from Instacart public release, "The Instacart Online Grocery Shopping Dataset." It consists of multiple sets of files that describe customers' purchases over time. The datasets comprise of grocery order records over 3 million from more than 200,000 Instacart customers. The data contains details about the sequence of products purchased in each order, day of the week and time of making the order and a relative measure of time between orders made by a customer. The datasets consist of the following files:

- Aisles: specifies the ids and names of aisles
- Departments: specifies the ids and names of departments
- Order_products: specifies which products were purchased in each order, their add-to-cart order, and if they are reordered
- Orders: specifies detailed order information
- Products: specifies the products information



POS Data?

The point of sale (POS) refers to the site at which a retail transaction happens. In simpler terms, it's the place where the goods are scanned, packed and paid for. As for POS data, it's collected directly from the retail POS system installed in stores, which is a combination of hardware and software. So, POS is the system that manages transactional data (OLTP) that is collected and later used by an OLAP system for data analysis and other business intelligence operations.

In order to perform a market basket analysis, a business should get the details about the Products, Orders, Departments from their online transactional processing (OLTP). Once the data is moved from OLTP to OLAP (Online Analytical Processing), data mining and machine learning algorithms can be used to find out patterns and relationships among the items in the dataset. Repeated data mining leads to the identification of associations and

correlations among items in huge datasets. With heavy amounts of data continuously being gathered and stored, several industries are becoming interested in mining patterns from their databases. The discovery of striking correlations among business transaction records can help in many decision-making processes for businesses such as catalog design, cross-selling, and consumer buying behavior analysis. Quite often the arrangement of items on shelves of a supermarket and online recommendations of some websites do not correlate with each other, which leads to unproductive use of the store's resources. In retail, the analysis of the transactions made is a significant part of understanding the consumer's behavior. This data allows stores to tailor promotional offers, individual preferences and helps in improving the quality of services. Therefore, the quality of a recommendation system can be enhanced by using external heterogeneous data sources. This could include data from online and offline markets inside one company or data from partner companies. Within a single market area, a variety of the products offered may be similar to each other, however, the association found among them may differ. Therefore, for the external data sources to be well integrated into the existing recommendation system, the structure and content of additional data sources must be analyzed, so that we use only the valuable parts of that data. Such integration will aid in improving the performance of the recommendation system. These external data sources may have a completely different format and may differ in their content. For the analysis of external data, it is required to convert all the data to a suitable single form. This includes analysis of data features, data clustering and data filtering.
To add more flavors, we created a csv file which includes various cultures around the world and we will map it to the products name in order to perform cultural analysis. By this we can understand various cultural products available in the shop and we can also learn about which cultural product is fast moving among the customers.

Sample Data

Number of Records included in our analysis from data extracted from Instacart and Kaggle. Our dataset contains the following information:

Orders: 3.5 Million order data with details related to order id, users, time of ordering, etc.

Products: Around 50 thousand varied products available for purchase spread across 3.5 million orders along with aisles and departments.

Users: There are about 200 thousand unique users.

Data Dictionary: Created by considering 67 world cultures and mapping them with related products.

A sample of our dataset is shown below:

Order Purchases:

1	order_id	user_id	eval_set	order_number	order_dow	order_hour_	days_since_prior_order
2	2539329	1	prior	1	2	8	
3	2398795	1	prior	2	3	7	15
4	473747	1	prior	3	3	12	21
5	2254736	1	prior	4	4	7	29
6	431534	1	prior	5	4	15	28
7	3367565	1	prior	6	2	7	19
8	550135	1	prior	7	1	9	20
9	3108588	1	prior	8	1	14	14
10	2295261	1	prior	9	1	16	0
11	2550362	1	prior	10	4	8	30
12	1187899	1	train	11	4	8	14
13	2168274	2	prior	1	2	11	
14	1501582	2	prior	2	5	10	10
15	1901567	2	prior	3	1	10	3
16	738281	2	prior	4	2	10	8
17	1673511	2	prior	5	3	11	8
18	1199898	2	prior	6	2	9	13
19	3194192	2	prior	7	2	12	14
20	788338	2	prior	8	1	15	27

Product Details:

product_id	product_name
1	Chocolate Sandwich Cookies
2	All-Seasons Salt
3	Robust Golden Unsweetened Oolong Tea
4	Smart Ones Classic Favorites Mini Rigatoni With Vodka Cream Sauce
5	Green Chile Anytime Sauce
6	Dry Nose Oil
7	Pure Coconut Water With Orange
8	Cut Russet Potatoes Steam N' Mash
9	Light Strawberry Blueberry Yogurt
10	Sparkling Orange Juice & Prickly Pear Beverage
11	Peach Mango Juice
12	Chocolate Fudge Layer Cake
13	Saline Nasal Mist
14	Fresh Scent Dishwasher Cleaner
15	Overnight Diapers Size 6
16	Mint Chocolate Flavored Syrup
17	Rendered Duck Fat
18	Pizza for One Suprema Frozen Pizza

Aisle details:

aisle_id	aisle
1	prepared soups salads
2	specialty cheeses
3	energy granola bars
4	instant foods
5	marinades meat preparation
6	other
7	packaged meat
8	bakery desserts
9	pasta sauce
10	kitchen supplies
11	cold flu allergy
12	fresh pasta
13	prepared meals
14	tofu meat alternatives
15	packaged seafood
16	fresh herbs
17	baking ingredients
18	bulk dried fruits vegetables
19	oils vinegars

Department Details:

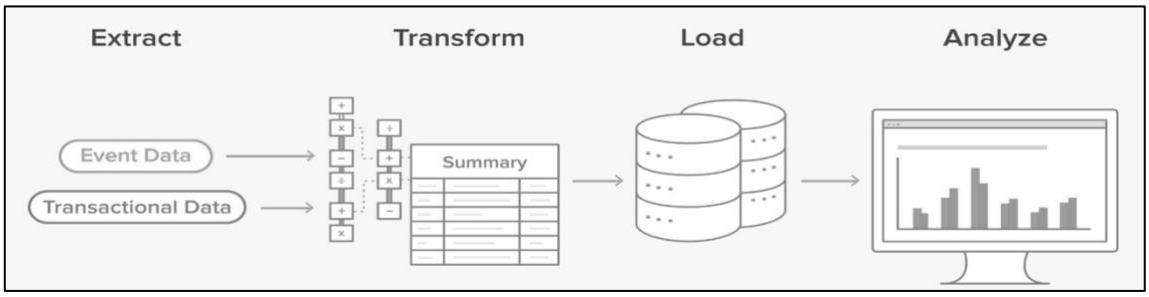
department_id	department
1	frozen
2	other
3	bakery
4	produce
5	alcohol
6	international
7	beverages
8	pets
9	dry goods pasta
10	bulk
11	personal care
12	meat seafood
13	pantry
14	breakfast
15	canned goods
16	dairy eggs
17	household
18	babies
19	snacks
20	deli

Data for creating dictionary to be used for Cultural analysis:

id	Culture_names
1	Afrikaans
2	Albanian
3	Arabic
4	Armenian
5	Azeri
6	Basque
7	Belarusian
8	Bulgarian
9	Catalan
10	Chinese
11	Croatian
12	Czech
13	Danish
14	Dhivehi
15	Dutch
16	English
17	Estonian
18	Faroese
19	Farsi

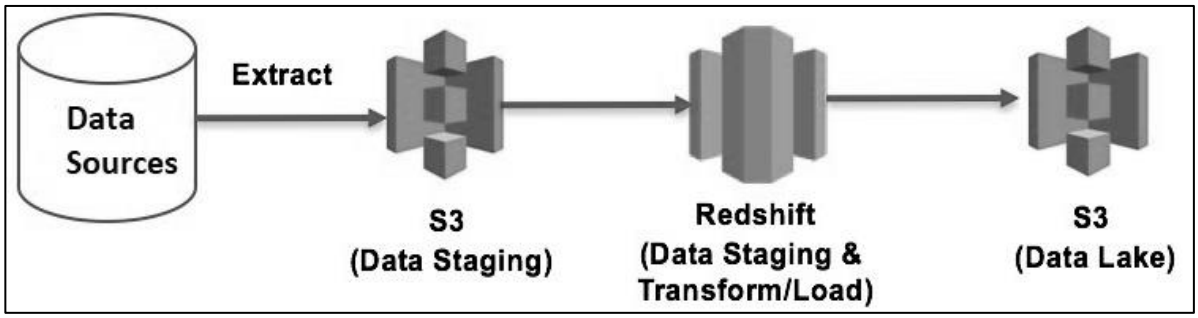
Data Cleaning & ETL

Considering the power that the consumers have today - they can decide a store's fate by picking to either buy its products or not, which makes it crucial for businesses to remain a step ahead. That is why retail data is so important. It provides information that is required to compete and compete well!
For retail stores, Data needs to be extracted from online transaction processing (OLTP) databases, more commonly known as transactional databases, and other data sources.



Many a times, data from non-OLTP systems is also included for analysis, such as text files, legacy systems, and spreadsheets; such data also requires extraction, transformation, and loading. Once the data is extracted from all the required and available sources, it is refined and transformed into a staging area according to the requirements. A staging area acts as intermediate storage for your data. Its basic objective is to serve as a platform where the data can be transformed into a format, which can then be matched to a target. They can also be referred to as staging tables, landing zones or staging areas. These transformations comprise of both data cleansing and standardization of the data for further analysis. Some of the common irregularities found in retail data are incorrectly formatted phone numbers,

redundant e-mail addresses or duplicate usernames and other information. The transformed data is later loaded into an online analytical processing (OLAP) database. Since the onset of Big Data as a phenomenon, the volume of data getting created is huge and so the retailers have started moving towards more optimized ways of data storage that is cloud. Amazon, Microsoft Azure and Google are some major cloud storage providers.



For the purpose of our project, we have considered Amazon's S3 for our data storage needs. The data from the retailers OLTP system is gathered and categorized as being closely related with product, orders and departments and hence segregated on the same basis.
The first step in this process is getting the data from different sources into Amazon S3. Amazon S3 provides a highly durable, inexpensive, and scalable storage platform that can be written to in parallel from many different sources at a very low cost. Amazon Redshift is used to transform and cleanse the data from the source format to go into the destination format. Once we have the transformed data, it is loaded into S3. Amazon Redshift can be integrated with Amazon S3, which allows parallel threads of throughput. Each transformation job loads formatted, cleaned data into Amazon S3. We use Amazon S3 here again because Amazon Redshift can load the data in parallel from Amazon S3, using multiple threads from each cluster node. Amazon S3 also provides a historical record and serves as the formatted source of truth between systems. Data on Amazon S3 can be consumed by other tools for analytics, if additional requirements are introduced over time.

Upon having the data in S3, we look out for errors and issues existing with our dataset. Since the data is segregated under the product, orders and department, we check the individual files for missing or invalid data. For doing so, we launch an AWS EC2(Amazon Elastic Compute Cloud) instance by selecting an Amazon Machine Image (AMI) which is a special type of virtual appliance that is used to create a virtual machine within EC2. It serves as the basic unit of deployment for services delivered using EC2. We selected the Amazon Linux 2 AMI(HVM) and a general purpose t2 micro instance type eligible for free tier usage. We created a role with full access to the instance and filled in the role details. Lastly, we configured the security settings including Custom TCP Rule and SSH and launch the EC2 instance after reviewing all the details. Once the instance is up and running, we use the SSH details to connect it with our systems. Once we have it ready, we installed Anaconda onto it which is then used to configure Jupyter Notebook.

For any sales or retail business, collection along with analysis of a large amount of transactional data is vital to the organization. However, processing the data in a strategic and structured manner is the key to add value to the business. Irregular and impure data

causes disruptions and may impede their effective utilization. Reliable information about consumers will guarantee that you are targeting the right people with apt products and personalized offers. It helps in converting first-time customers in addition to making a store ready for subsequent business, ensuring growth, year over year. Clean and valid data on consumer behavior support in generating insights on their personal preferences likes, and dislikes, which provides endless opportunities for businesses to up-sell their products. Data validation helps to ensure that the marketing and sales process operates on data that will help in better decision making. Naturally, the first thing we did towards data cleaning was to check for missing values in all the data sets. We used the isnull() function to check for null values in the data sets and also tried to get the percentage of the null values in each data set. To find the missing values, we first determined the total for each data set using the isnull() and sum() functions. This gave us the number of null values for each column in each data set. Then to get a better idea of how significant these null values are, we calculated the percentage of null values in each column for all the 6 data sets using the total and then dividing it by using the isnull() and count() functions. Then we made missing values tables for better visualization using the pd.concat function to put together the total and percentage of each dataset. Only the orders dataset has some null values in one of the columns and the percentage for those null values was around 7%. Since it is not a significant number and our dataset would not suffer much if we lose these records, we decided to drop the null values from the order dataset to make it clean. Some columns in the datasets have character types, and this will become a problem in XGboost algorithm, which controls only numeric vectors. Therefore, we recoded the categorical variables into factors. Hence, we created a new dataset for the products by merging aisles and departments along with the products. The new datasets are stored again after transformation in AWS S3.

Following all the above-mentioned steps, we now have our data set ready and moving forward we are going to deep dive into the exploratory and predictive analysis to understand and explore about market basket analysis. We plan to use unsupervised modeling techniques such as clustering and association to understand how products are recommended to customers.

Data Storage

Determining the most suitable data storing method is one among several aspects an organization must solve. It is necessary to understand that data is much more than mere information. Present day businesses depend heavily on their proprietary data assets to make critical business decisions, employing powerful algorithms to gather invaluable insights from the unstructured data they've obtained from their consumers and competitors. Various sources like business applications, digital media, database systems, etc. are driving companies to experience a tremendous growth of data. This veritable volume of data paired with the analytical capabilities available at present translates to the idea that organizations now have the potential to get a good understanding on their customers' behavior in the past, present as well as be able to predict what can happen in the future. Considering how essential data is to the success of any organization, it is plausible to suggest that data storage is a valuable investment for any business's long-term goals. Figuring out where to store this data can be an intimidating task. To make meaningful

decisions, companies need to contemplate the benefits of choosing from among different available solutions.

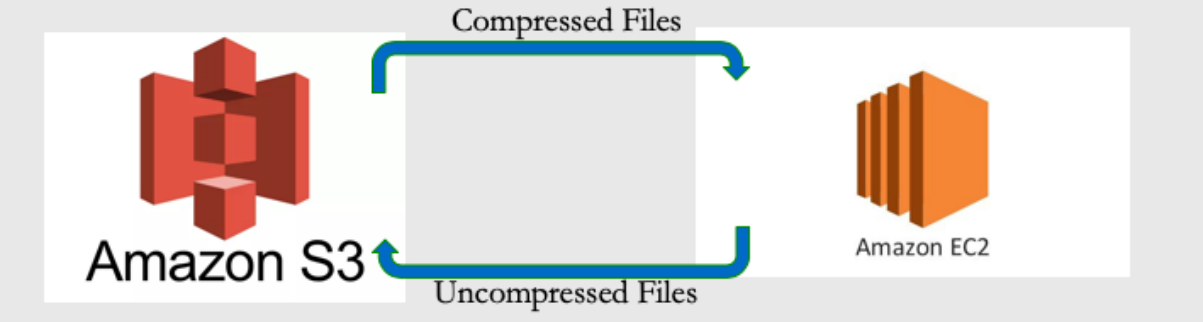
A typical type of data storage that can be thought of is databases. A database is simply an organized collection of data. A conventional database is inclusive of both the structure and design of a data environment, as well as the data itself. One of the most common types of databases is relational databases.
Relational databases store data in two-dimensional tables having distinct relationships among themselves. These databases are an efficient, simple, and effective way of storing data. However, they are traditionally more stringent and controlled having restricted or limited ability to translate complex data such as unstructured data. In today's world of ever-increasing data in the form of tweets, images, videos, etc. Relational databases cannot put up with this flood of information, as well as the variety of data types that occurred from this evolution. This eventually led to the rise of non-relational databases (NoSQL). NoSQL databases bypass the rigidity of SQL by replacing organized storage with more flexibility. Not only can NoSQL systems handle both structured and unstructured data, but they also use an ad-hoc approach for organizing the data as well as process large amounts of data with higher speed and versatility. Such data storage technologies survived almost consistent for decades, but over the last several years, a new series of data storage solutions are available that provide more cost-efficient, flexible, and scalable storage options. These are available in the form of Data Lake, Data Warehouse, Data Marts, Data Swamp, Data Cube, etc.

A data lake refers to a centralized repository that allows storing large scale structured and unstructured data. Data can be stored as-is, without having to structure, and run complex analytics to guide better decisions. A data lake stores the data in its raw format until the time it is required. It uses a flat architecture where a file may not be associated with other files in the lake. Each data element is associated with a unique identifier and metadata tags. When a business requirement emerges, relevant data can be mined from the lake to perform further analysis as per the requirements. Many cloud providers are providing storage options in the form of data lakes, namely Amazon S3, Azure Blob storage service, Google cloud platform cloud storage data lake. By using cloud infrastructures, data can be processed and analyzed securely and cost-effectively.

A data warehouse stores data in an ordered manner with everything archived and organized in a defined way. To develop a data warehouse, a significant amount of effort is made during the initial stages to examine data requirements and understand the business processes and needs. Decisions are made concerning what data needs to be included or excluded from the warehouse. Firstly, the use of the data is identified, beyond which it is loaded into the warehouse. Data from multiple online transaction processing (OLTP) applications and additional sources are obtained for business intelligence activities, decision support, and answer user queries. Data warehouses can serve organizations from both IT and business viewpoint. Separating the analytical and operational processes can improve the operational systems and allow business users to access and query-relevant data quicker from multiple sources. Additionally, data warehouses can provide enhanced data quality and consistency, thereby enhancing business intelligence. In terms of cloud providers, some examples of data warehouse are Amazon Redshift, Azure Data Warehouse.

A data mart is a business line-oriented database that is usually a partitioned section of an enterprise data warehouse. The subset of data contained in a data mart typically aligns with a particular business unit like sales, finance, or marketing. Data marts expedite business processes by allowing access to appropriate information in a data warehouse or operational data store in a short amount of time. Since a data mart only consists of the data relevant to a particular business area, it is a powerful way to obtain actionable insights quickly. In a market dominated by big data and analytics, data marts are one solution to efficiently transforming information into insights as data analysis requires easy-to-find and readily available data which data mart can immediately provide.

Data cubes are most often used by decision support system users. The data cube represents data along some measure of interest. It can be 2-dimensional, 3-dimensional, or more. Each dimension in the data cube represents some attribute in the database, and the cells represent the measure of interest. Often, queries are performed on the cube to retrieve support information on decisions.



We have considered the Amazon S3 Data Lake for our data storage requirements. Since the file size of our data is huge, we uploaded a zipped version of the file to S3. Amazon Simple Storage Service (Amazon S3) is an object storage service that allows high scalability, data availability, security, and performance. Customers of all scopes and businesses can use it to store and protect an enormous amount of data for a variety of purposes, including data analytics. Since its primary purpose is to store data, it lacks the capability to uncompress or unzip the compressed files uploaded to it. So, to uncompress our files, we instead launched an EC2 instance and extracted our zipped files there. Once the EC2 instance is launched and we put in our AWS CLI details, we use the commands required to unzip the files and move it back to our S3 data lake bucket.

Analysis

Data analysis refers to the techniques and processes utilized to enhance productivity and business gain. Data is extracted and classified to identify and interpret behavioral data and patterns. Data analysis or analytics is categorized into four categories: Descriptive, Diagnostic, Predictive, and Prescriptive Analytics.
With our retail store dataset, we have performed Descriptive, Predictive, and Prescriptive Descriptive.

Descriptive Analysis: Descriptive analysis or statistics does precisely what the name implies they “Describe” raw data and make it interpretable by humans. These analytics describes the past. Past refers to the time when an event occurred, be it one minute or one year ago. Descriptive analytics is useful since it allows us to determine trends and relationships from past behaviors and use them to understand the future outcomes may be influenced. Few examples of descriptive analytics are reports that present historical insights concerning the company’s finances, operations, sales, customers, etc. So, in simple terms descriptive analysis provides ‘insights into the past’. For our project, we have used seaborn, matplotlib libraries in Jupyter and Power BI to understand the trends and patterns.

Diagnostic analysis: For this type of analytics historical data points are measured against other data points to explain why something happened. Using diagnostic analytics, analysis can be drilled down to identify dependencies, trends, and patterns. Companies utilize diagnostic analytics to gain in-depth insights related to a specific problem. For doing so, a company should have comprehensive information at their end, else data collection for every issue individually may turn out to be very time-consuming.

Predictive Analysis: Predictive analytics holds the ability to “Predict” what might happen by understanding the future. It provides businesses and organization with actionable insights by analyzing the data. Predictive analytics present estimations about the likelihood of a future occurrence. It is essential to remember that no statistical algorithm can predict the future with 100% assurance. Businesses use these statistics to anticipate what may occur in the future. It is because predictive analytics is built on probabilities. For making predictions, the statistics take the data that you have and fill in the missing data with most reliable estimates. They consolidate historical data found in different systems like ERP, CRM, HR, and POS, to identify trends and patterns in the data and employ algorithms to capture links between various data sets. Firms use predictive analytics whenever they want to gaze into the future. Predictive analytics has a large number of implementations within an organization, for eg., it can be used to forecast customer behavior like churn predictions and purchasing patterns to recognizing trends in sales, estimate requirement for inputs from the inventory, supply chain, and other retail operations. Prevalent business uses include, understanding how sales might close at the end of the year or during black Friday, predicting what items customers will purchase together by applying market basket analysis. In layman terms, we can describe predictive analysis as ‘understanding the future’.

For the purpose of our project, we have used predictive analysis to check if the user's previously purchased product will be reordered in the next order. We used different algorithms like Logistic regression, Decision tree, XGBoost and Light GBM since our target variable, "reordered," is a categorical variable. Reordered tells us if a user has re-ordered a particular product that he has already purchased in the past. Both XGBoost and Light GBM algorithms are based on the concept of Boosting. Boosting is a sequential method which operates on the principle of ensembles. It combines sets of weak learners and delivers improved prediction accuracy. At any instance of time t, the model outcomes are weighed based on the results of the previous instance of time t-1. The outcomes predicted rightly are assigned a lower weight, and the ones misclassified are weighted higher.

Logistics regression is commonly used Machine Learning algorithm and is a useful regression method for solving the binary classification problem. It is simple to implement, and it can be used as a base for any binary classification problem. Its core concepts are valuable in deep learning. Logistic regression details and predicts the relationship between one dependent binary variable and independent variables. Then need to divide the columns into two types of variables like dependent(target variable) and independent variable(or feature variable). Logistic Regression() function is used to create Logistic Regression classifier object. Then we fit our model on the train set using fit() and perform prediction on the test set using predict().

Random forests generate decision trees on randomly selected data samples, gets a prediction from each tree, and selects the best solution using voting. It provides a good indicator of the feature importance. It can be used both for classification and regression. It is also the most adaptable and easy to use an algorithm. A forest is composed of trees. The more trees it has, the more robust a forest is. Random Forest gives high accuracy because of several decision trees participating in the process. Random forest uses Gini importance or means a decrease in impurity to calculate the significance of each feature. Random Forest() function is used to create Random Forest classifier object.

XGBoost is a powerful algorithm whose absolute strength comes from its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage. XGBoost is a decision-tree based ensemble algorithm which uses of gradient boosting framework. This algorithm is a great pick for use cases involving unstructured data like images, text, etc. and can easily beat other frameworks or algorithms. It can be used for regression, classification, ranking, and other user-defined prediction problems. It also supports AWS and Azure eco-systems. This algorithm applies the principle of boosting weak learners using gradient descent architecture and also improves over the initial GBM framework through system optimization and algorithmic enhancements. XGBoost optimizes standard GBM algorithms through parallelization, tree pruning, hardware optimization, regularization, sparsity awareness and cross validation. The algorithm is provided with a built-in cross-validation method at each iteration and takes away the need to explicitly program the search and to specify the exact number of boosting iterations required in a single run.This helps offer a methodical solution to combine the predictive power of multiple models. The end result is a single model which gives the aggregated output from several models.

Light GBM is a high-performance gradient boosting framework that is based on the decision tree algorithm and can be used for other machine learning tasks like ranking and classification.
As it is based on decision tree algorithms, it splits the tree leaf-wise trying to get the best fit. Therefore, while growing on the same leaf, the leaf-wise algorithm is able to reduce more residual errors than any level-wise algorithm and consequently results in much better accuracy which is not often achievable by other boosting algorithms. The algorithm follows a histogram-based approach where continuous feature values are bucketed into discrete bins which fasten the training process surprisingly fast, and there is no wonder why this algorithm is named as "Light" GBM. Since this algorithm buckets the continuous features, it results in lower memory usage. But sometimes, leaf-wise splits command an increase in

complexity and might result in overfitting. This can be subdued by using a parameter called max-depth, which specifies the depth to which splitting in the tree should occur. This algorithm is capable of working well with large-sized datasets along with a notable reduction in training time. Different parameters in this algorithm allow for parameter tuning that brings in various advantages like best fit(num_leaves, min_data_in_leaf, max_depth), faster speed(bagging_fraction, feature_fraction,
max_bin) and better accuracy(num_leaves, max_bin). LightGBM provides good accuracy with integer-encoded categorical features. Categorical features must be encoded as non-negative integers (int), ideally, within a contiguous range of integers starting from 0. Unlike LightGBM, XGBoost is not capable of working with categorical features by itself. It only allows for numerical values. This is the reason why different encodings like one-hot encoding, mean encoding, or label encoding have to be performed prior to providing categorical data to XGBoost.
Both XGBoost and Light GBM have an in-built mechanism that enables them to handle missing values.

Prescriptive Analysis: In recent times, the Retail sector has been moving away from predictive analytics and adopting the trend of prescriptive analytics. For most of the retailers, predictive analytics has now become a situation of 'been there, done that.' Retailers have been able to execute marketing campaigns successfully using the insights derived from data analytics, thereby effectively targeting consumers. There is more and more competition developing among these retailers to lay their hands-on prescriptive analytics, which analyzes data or content to answer the question "What should be done?" Prescriptive analytics provides the ability to suggest intelligent recommendations to buyers or consumers. Online shopping sites have taken personalization to a different level catering to each customer individually. Prescriptive analytics helps organizations draw explicit recommendations based on past history and plays a significant role in customer retention. It prescribes a number of different possible actions to a target problem and guides them towards a solution. In short, prescriptive analytics is all about providing advice attempting to quantify the impact of future decisions to recommend different possible outcomes before decision making itself. With prescriptive analytics, one can predict what will happen as well as why it will happen, providing recommendations about actions that will be taken based on the predictions. Prescriptive analytics uses a combination of techniques and tools which can be applied upon input from different data sets like historical and transactional data, real-time data streaming, etc. Prescriptive Analytics is best used when one needs to provide advice to users on what action to take. When implemented in the right way, this can have a great impact on business' strategic decision making and therefore, the effectiveness of actions taken by the business.

Recommender systems take a generous pool of available data and make the decision-making process easier by providing just a few targeted selections. A great example of a recommender system is Netflix's recommendation system for tv series and movie streaming. Instead of recommending an endless number of possible videos, the algorithm refines the pool of availability to a few titles based on Big Data that it collects so that the user can see what he might be interested in. Determining the right type of recommender system is as crucial as deciding to utilize one in the first place. There are multiple available options to choose from:

Collaborative Filtering
This type of recommender system references the data collected from other users to make proposals. The notion behind this type is that if some of the people have made similar choices in the past, for example, same products in the shopping cart, there is a high probability that they would buy the additional picks in future.

Content-Based Filtering
This type of recommender system, the machine learns about the user and creates a personalized profile to determine items that a specific user would like. For example, a site may utilize an algorithm that suggests items with similar names in its description to an item the user has earlier bought.

Demographic Based Filtering
In this type of system, demographic information of the user are used to make recommendations. The system would suggest items that have been picked by other users who have identical demographic info.

Utility-Based Filtering
The utility that the user gets from the product is the basis of this type of recommendation system. It can involve things like vendor ratings, availability, services, reliability and availability to make sure it suggests products effortlessly obtainable by users.

Knowledge-Based Filtering
This system recommends choices based on a user's recognized buying patterns and preferences. Since the system has the historical data of user's purchases, it makes recommendations based on what might be purchased by the user in the future. For example, a user always orders 5kg of flour in the first week of every month, so the system can use this data to start suggesting the same to him around the same time.

Hybrid Filtering
This type of system fuses two or more different recommendation techniques to build a more meticulous recommending system.

For our project, we have used two techniques to build the recommendation systems. The first one is the apriori algorithm that is based on association rules. The second one is the product bundle recommendation that uses collaborative filtering technique to make precise recommendations to the customers.

Apriori algorithm is an algorithm for finding frequent itemset for an association rule over a transactional database. An algorithm is named apriori because it uses the prior information of frequent itemset. It is an iterative process where k frequent items are used to find k + 1 itemset. Below are two essential phases of the algorithm.

1. Frequent item-set generation-Apriori is a popular algorithm for extracting frequent item-set to apply in association rule. The apriori algorithm has been designed to work on the

transactional database, such as purchases by customers of a store. An itemset is considered as "frequent" if it meets a user-specified support threshold.
2. Rule generation-Should generate rules for frequent item-sets. Then calculate support and confidence. Remove the ones who fail minimum support and confidence threshold level.

Product Bundle based recommendation is based on the bundles of products, as the name suggests. Frequently bought items for shopping is called product bundle. For example, when we add cereal, then we will also add milk and fruits along with it. We will use this product bundle information to provide a recommendation. That is when a customer adds a product; then we will suggest a list of products that can be bought together with it. To provide recommendation first, we need bigram and its frequency. A bigram is a series of two adjacent elements. In our project, it is a sequence of two products bought together. For instance, if a customer adds milk, diaper, and beer to the cart, then bigram is 'milk diaper' and 'diaper beer'. After capturing all the bigram, we will count the frequency of each bigram. In code, we have shown bigram in a nested dictionary format like {avocado: {lime: 7}}. In the first layer, the key is first product and value are another dictionary which contains the following product that is bought together with the first product, and its value is frequency. The second step is to recommend a product based on bigram frequency for which frequency of bigram is sorted in descending order. Say for example, {'avocado': {'lime': 7, 'onion':4, 'tomato': 4, 'cilantro': 2}}. In code, we have written a function wherein if we give product and number of recommendations needed(k), then function provides the result based on the descending order of frequency of bigram. Then, the customer can pick products from the recommendation list. There are two situations we have taken care of in our project. The first one is what happens when there are several bigrams with the same frequency. In this case, it solely depends upon the number of recommendations(k) from the above example if k is 3 then lime, onion, tomato, are recommended whereas if k is 2, then it randomly picks from either of two products which have the same frequency (tomato or onion). Secondly, we also thought about the size of k being too large. From the above example if k is 4, then all products are listed, but if k is 10, then we select the highest frequency product (lime) and provide a recommendation of that product to fill in other spaces.
The evaluation is carried on test data. Test data contains the actual recommended product. We have written a function to compare the recommended products with the predicted one. For a given order we are comparing the actual and predicted value, if a predicted product is present in actual order then we give 1 as score else it is 0. Mean test score is the sum of the score given for all orders divided by the count of order.

Visualization/Descriptive Analysis

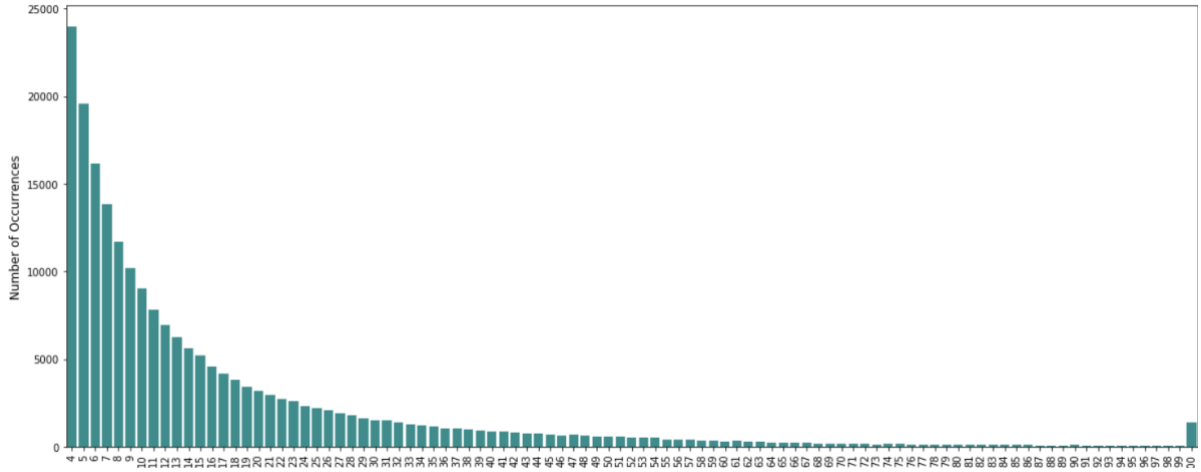
Data visualization presents static and interactive visuals of a specific context to help humans understand and to make sense of vast data. The data presented in a format highlights the hidden pattern, trends, and correlations, which may otherwise go unnoticed. Data visualization is often used to monetize data as a product.

There are various tools available for big data visualization. The major ones are Tableau, Google chart, and Power BI. Jupyter is an open source platform, which provides Big Data

analysis, visualization, and real-time collaboration on software development across more than a dozen programming languages. Tableau is one among the market leaders for the Big Data visualization, exceptional in delivering interactive visuals for the results derived from Big Data operations, deep learning algorithms, and multiple types of AI-driven apps. Google chart offers a plethora of visualization types, from easy pie charts and time series, all the way up to multi-dimensional interactive matrixes also, it is free. Power BI has native apps which can retrieve data from anywhere and alerts about changes. It can also use the publish to web feature that lets you add your visualizations directly to your blog or website. In our project, we have used both Power BI and various libraries like seaborn and matplotlib in Jupyter for descriptive analysis.

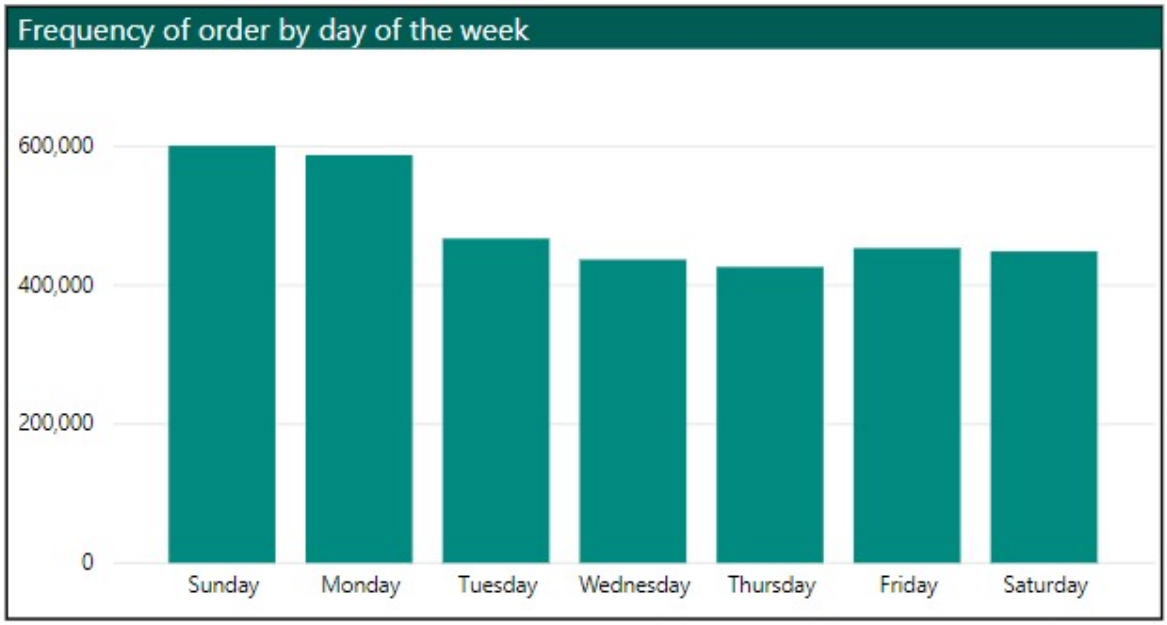
Below are few visualizations which helps in data exploration:

1) Bar graph showing order reorder counts

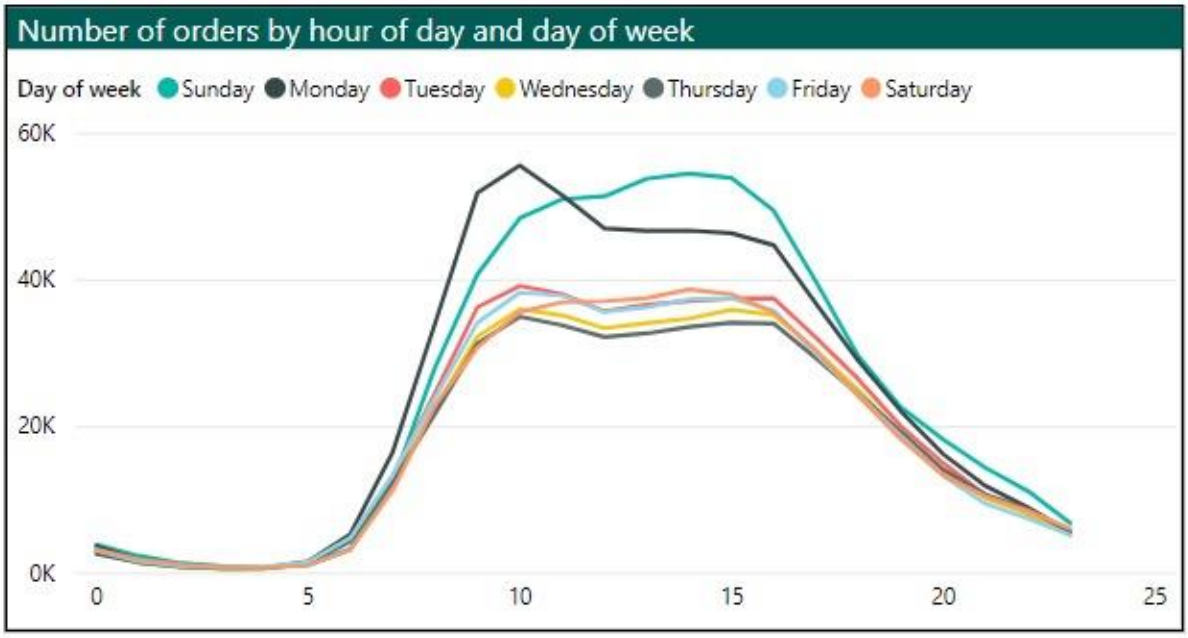


For the above graph, we considered unique users of the orders in order to find frequency of orders. We can see that for unique user the minimum number of previous orders is four and maximum is hundred.

2) Busiest day of week of the orders

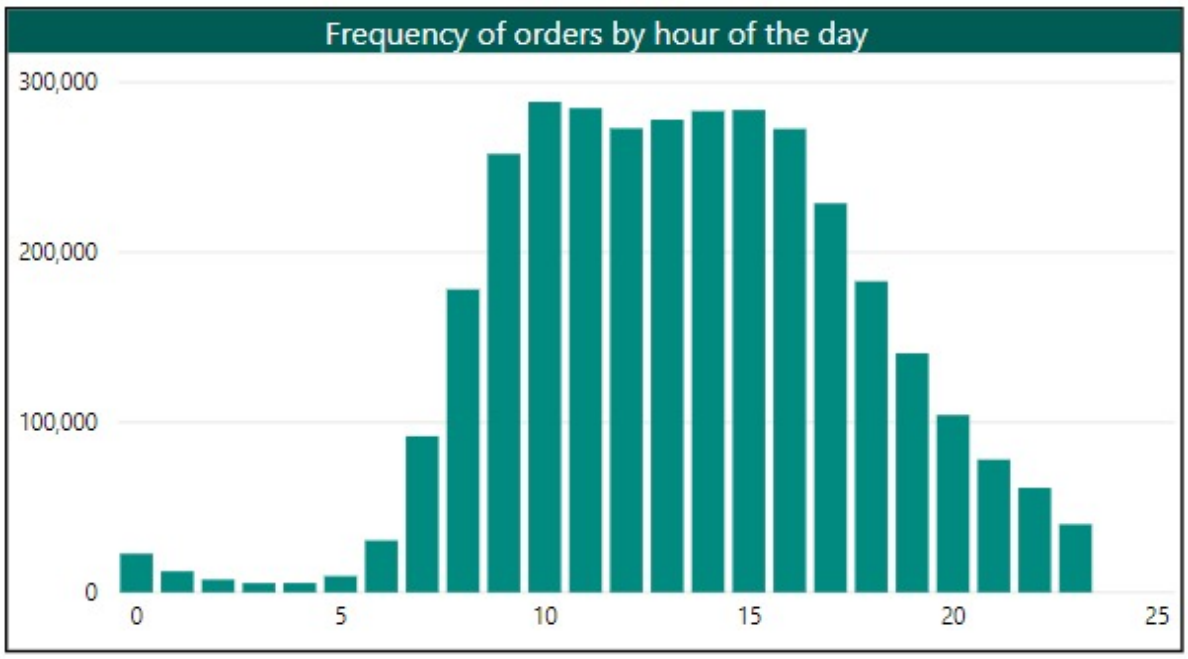


From above bar graph we can see that maximum order is placed during Monday and Tuesday.



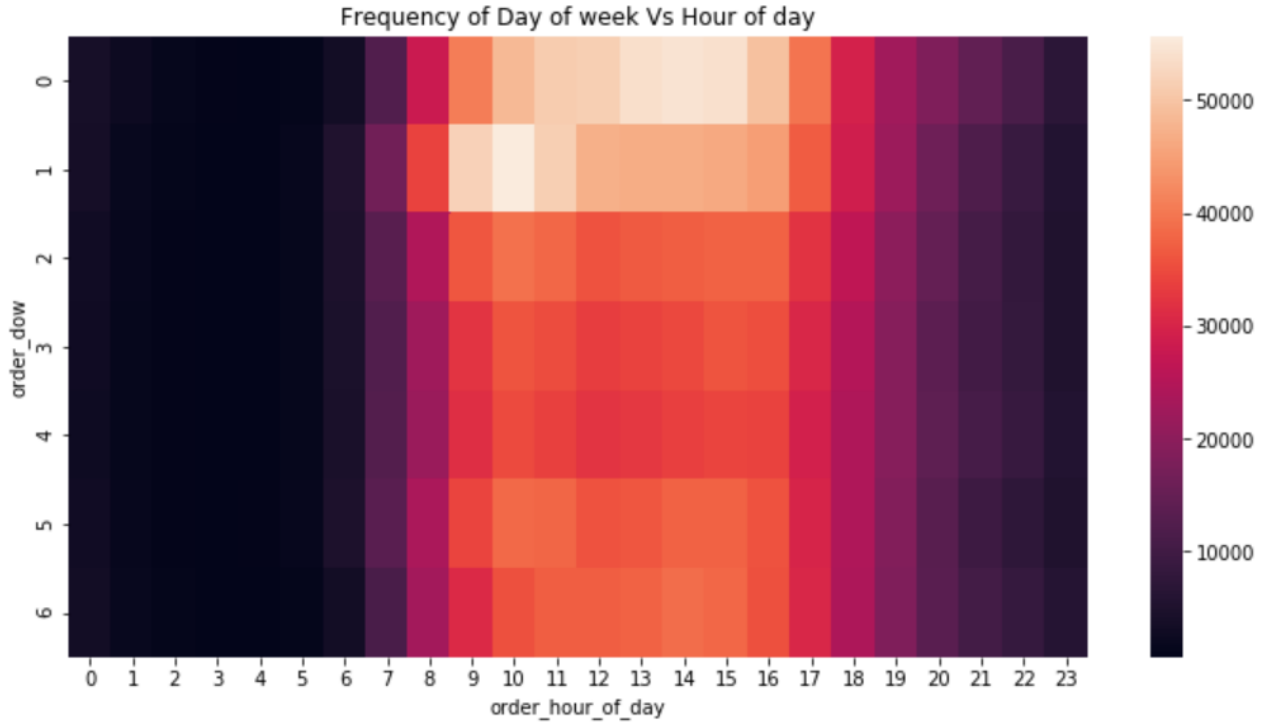
During weekend many people place order but considerably less than first two, and then ordering tends to go down gradually.

3) Busiest hour for an order



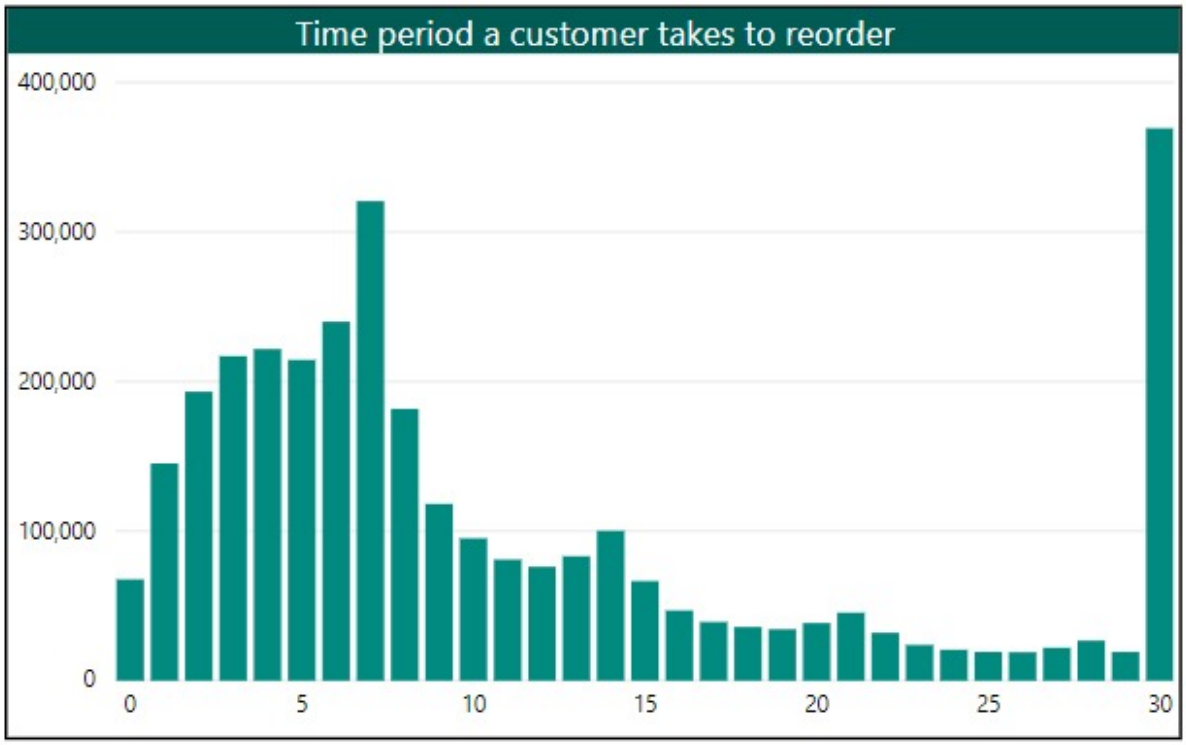
From the above graph we can infer what time in a day maximum orders are placed like morning, evening or afternoon. We see that orders are maximum from 10 am to 5 pm window and are minimum during early morning and late night which makes sense.

4) Heat Map



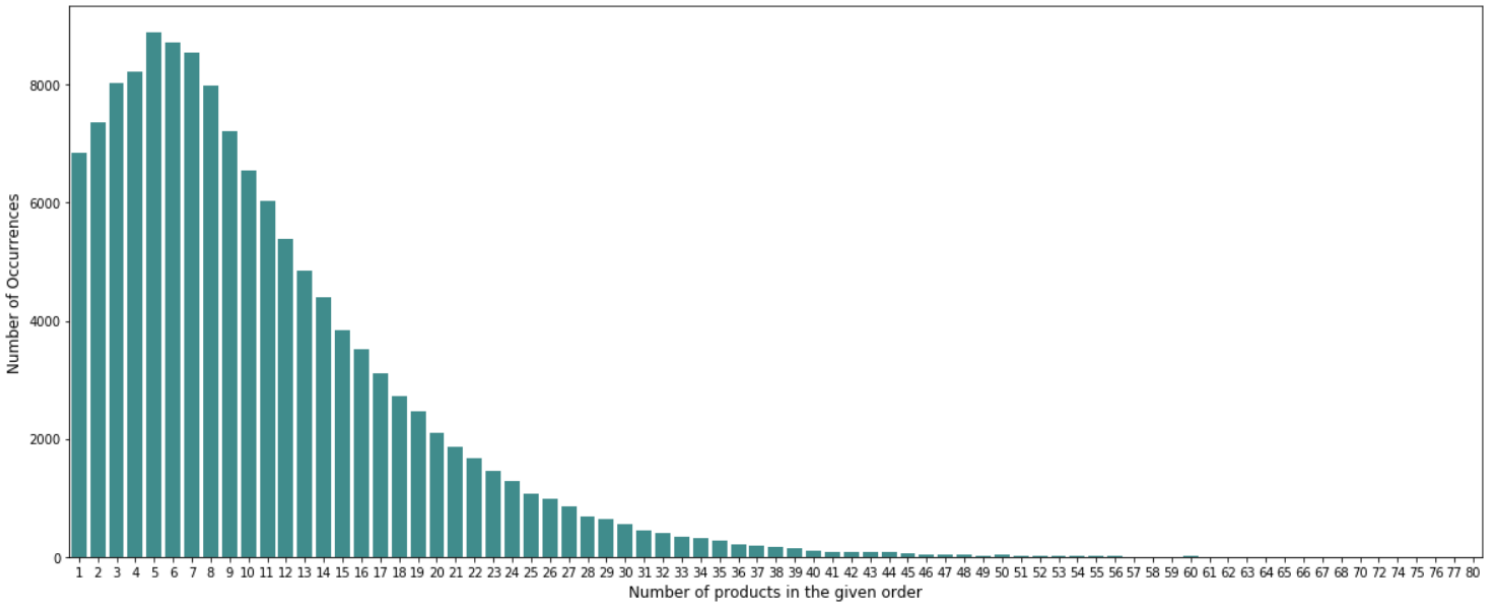
The above heat map represents the intersection of day of the week (y axis) and hour of the day (x axis). So, the peak time for ordering are around 10:00 AM - 4:00 PM on Monday and Tuesday.

5) Frequency distribution showing by which day order is reordered again



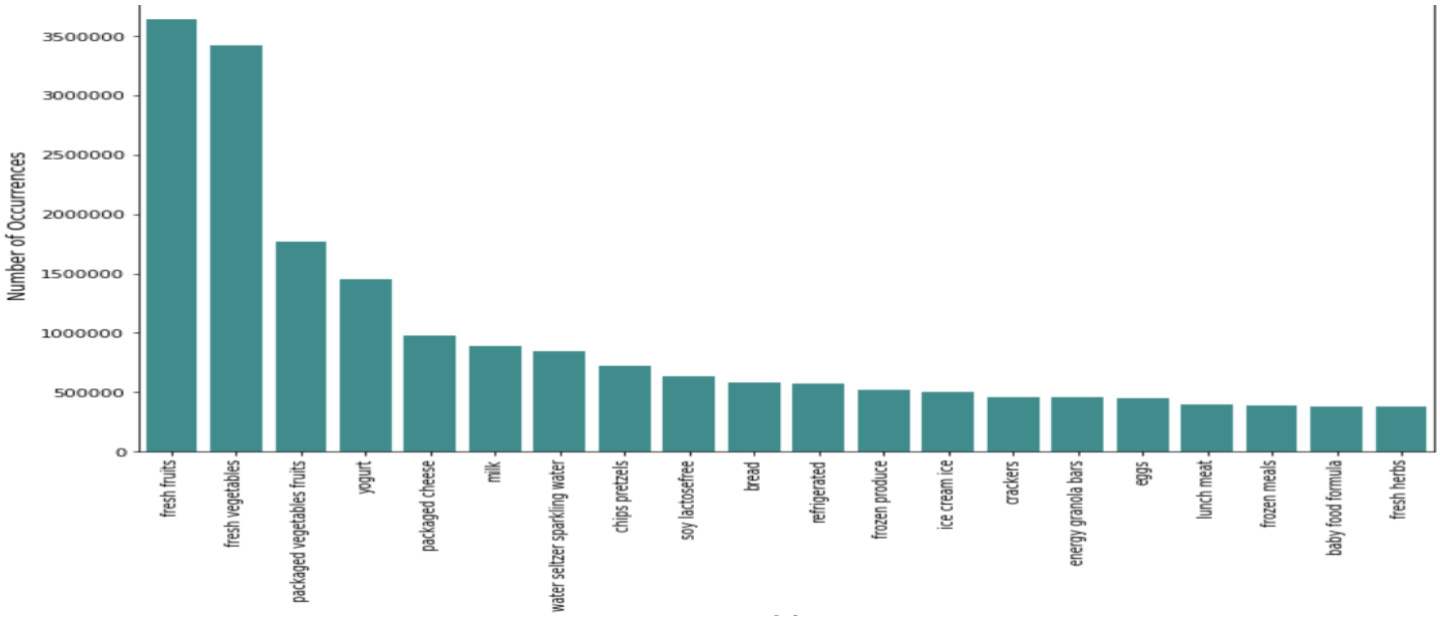
From above graph we can see that customer tend to reorder maximum on 30th day or 7th day which is obvious as people buy in weekly or monthly fashion. We can also see the pattern that after 15th reorder rate is minimum.

6) Bar graph showing number of products per order on an average



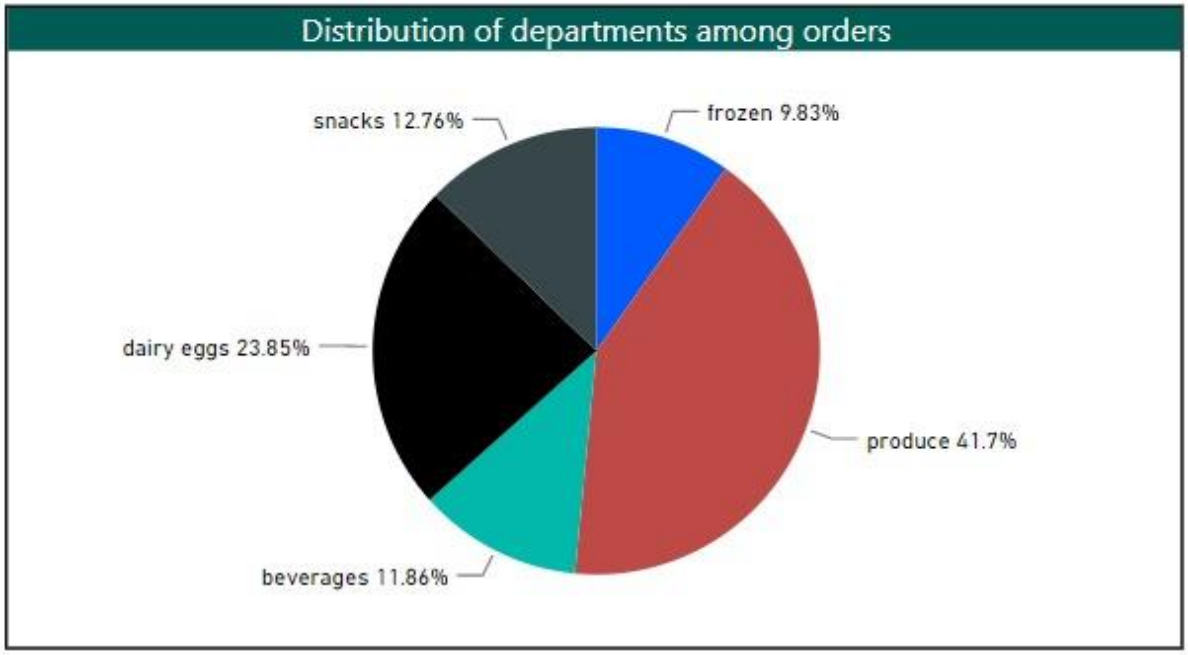
From above graph we can see number of products bought for each order. So, on average maximum of five products are bought for an order. We also see that number of occurrences are decreasing for the number of products greater than thirty.

7) Popular Aisle and its count of sale



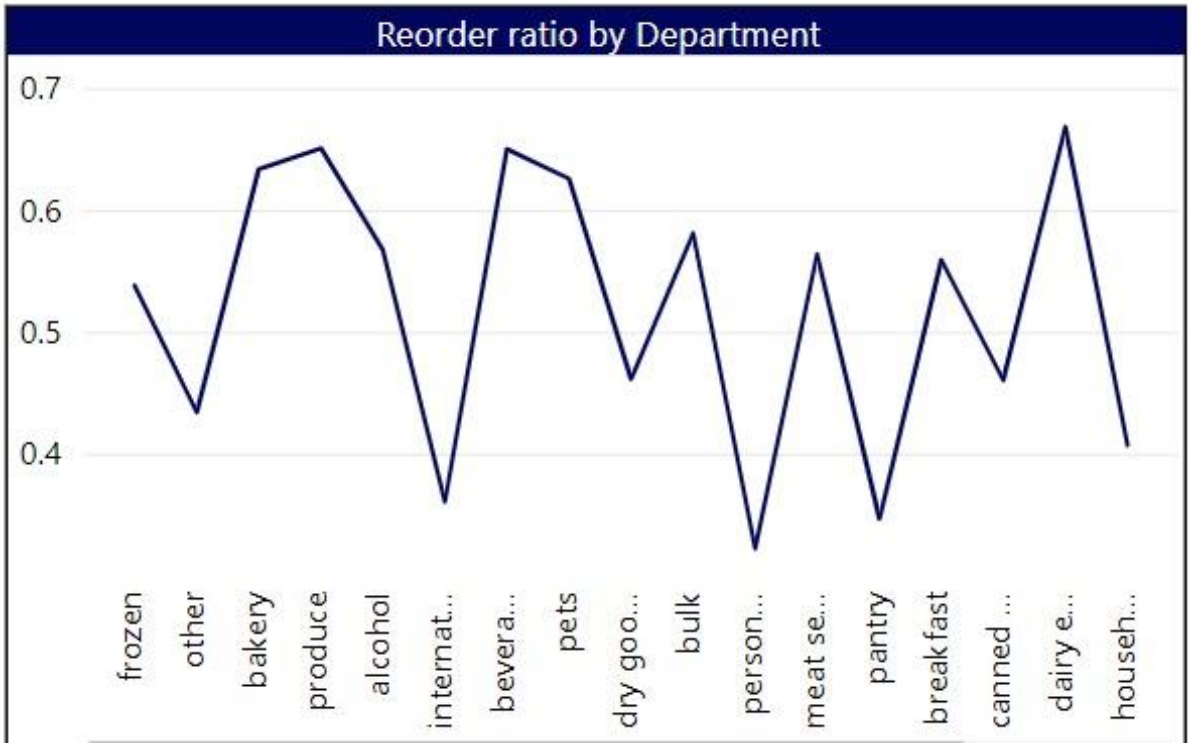
Clearly from above graph we can see that fresh fruits and vegetable are top grossing products and their aisle is most popular one. We can also see that meat, frozen and herbs are least popular aisle.

8) Pie chart showing Department wise sales



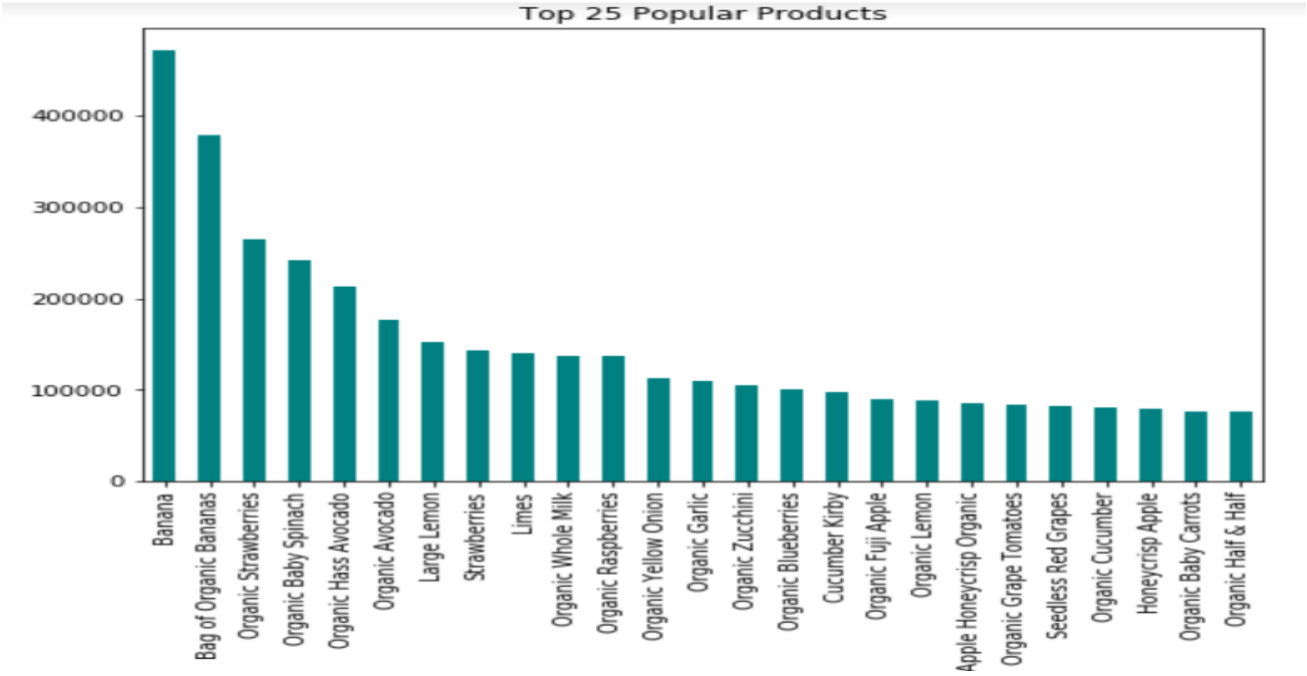
From above pie we can see that fresh produce is having maximum sales and frozen is least. It proves that people are prioritizing health over anything by choosing fruits and veggies and using less frozen food.

9) Line graph showing Department wise reorder ratio



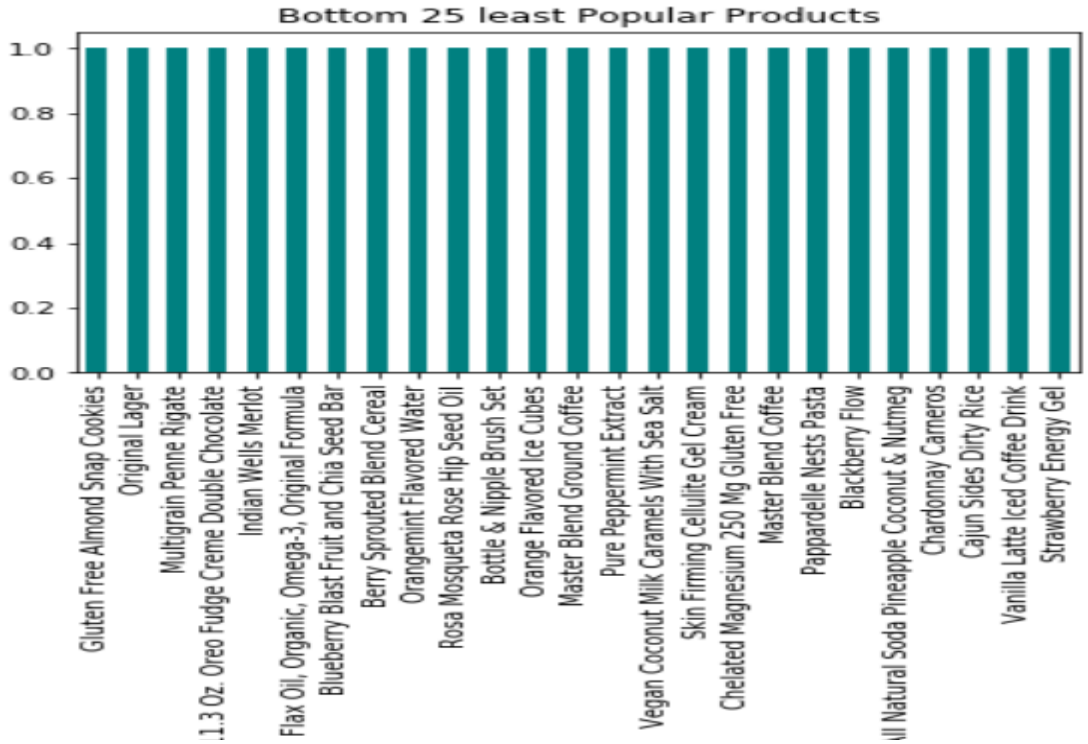
Above graph shows what are the chances that previously ordered products will be reordered again. We see that Dairy eggs have highest reorder ratio and personal care has the least.

10) Top 25 Popular products and their count



Above graph shows the top fast-moving products. We see that banana is the highest trending product of the store followed by remaining fresh produce.

11) Least popular products and their count

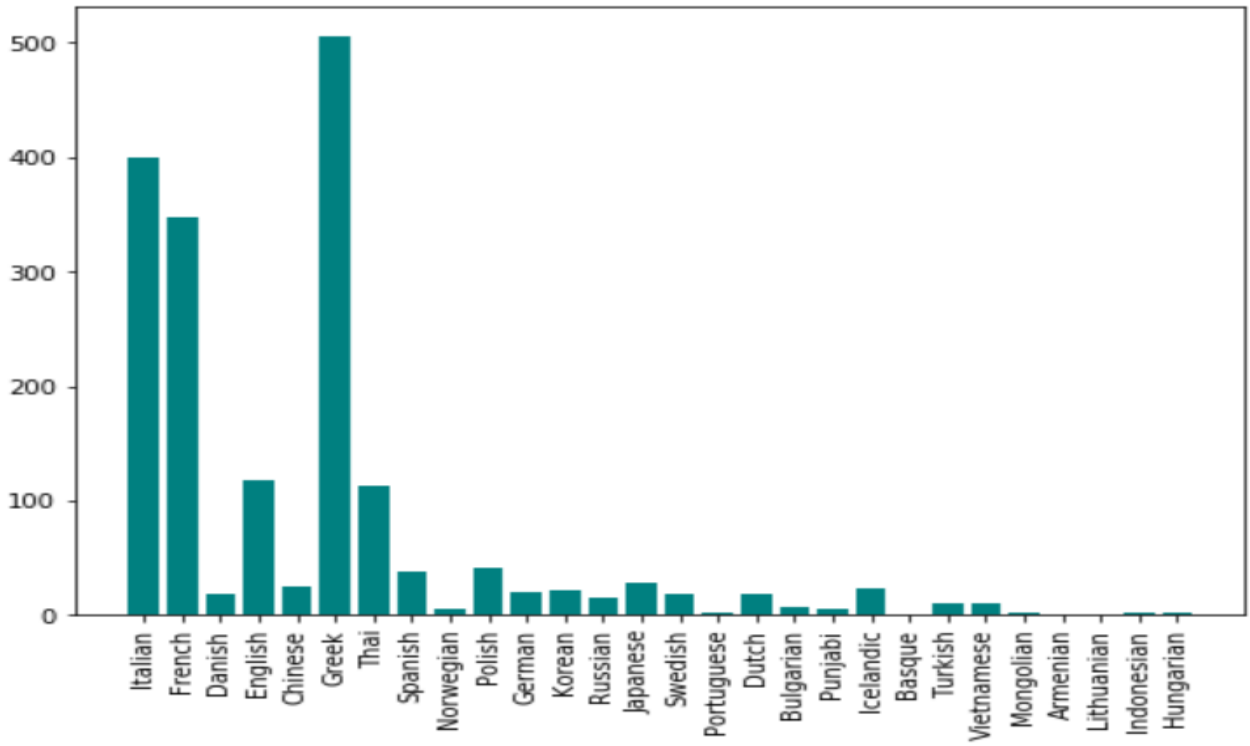


Above graph shows least moving products. All the above-mentioned products are sold only once.

12) Word cloud

Above word cloud shows list of items which are bought more times in store. The higher the size of the word, the more the product is being sold.

13) Cultural Analysis



In the above graph name of various cultures around the world are mapped with product name to check which cultural product are sold most in store. From above graph we can see that Greek and Italian are most sold products.

Results and Discussion

Predictive Analysis

As discussed above, we have used the four algorithms – Logistic Regression, Random Forest, XGBoost and LightGBM to predict the probability of an item being reordered by a customer in his next purchase. Dealing with such a large scale dataset, it was interesting to monitor the time taken by each algorithm to get trained. The below table shows the accuracy provided by each of the model in predicting the reordering probability as well as the approximate time taken.

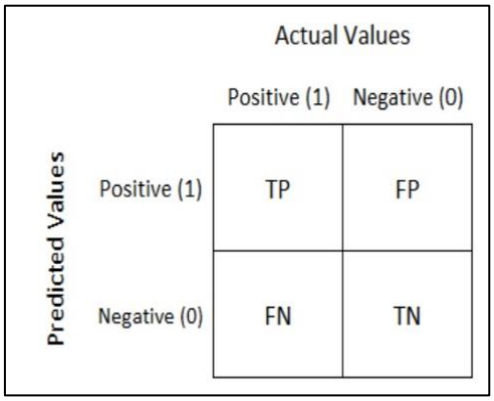
Models	Accuracy	Approximate time(in minutes)
Logistic Regression	90.19	35
Random Forest	90.41	40
XGBoost	90.43	30
LightGBM	90.46	15

From the above table, it is clear that our best performing models are XGBoost and LightGBM algorithms. The time taken by these reiterates the fact that these algorithms have a fast

training speed and higher efficiency. Beyond this point, we have considered out best two performing models for further analyzing the results obtained.

We have used Confusion matrix and AUC-ROC curve for measuring the effectiveness of our model.

Confusion Matrix: It is a performance measurement for any Machine Learning classification problem. The image shows a typical confusion matrix. A confusion matrix consists of four different combinations of predicted and actual values. In the context of our project, these are defined as:



True Positive (TP) – The cases in which we predicted that a product will be re-ordered, and it is actually re-ordered.
True Negative (TN) – The cases in which we predicted that the product will not be reordered, and it is actually not re-ordered.
False Positive (FP) – The cases in which we predicted that the product will be reordered, but it is actually not reordered.
False Negative (FN) – The cases in which we predicted that the product will not be reordered, but in fact the product has been re-ordered.

Getting these values is required for measuring recall, precision, specificity and accuracy.

Recall/Sensitivity: Out of all the actual reordered products, how many did we predict correctly as reordered.
Precision: Out of all the predicted reordered products, how many did we predict correctly as reordered.
Specificity: Out of all the actual non-reordered products, how many did we correctly identify as non-reordered.
Accuracy: Out of all the available products, how many did we correctly classify as reordered and non-reordered.

Below are the images of the confusion matrices obtained for both XGBoost and LightGBM models.

```
In [101]: print("Accuracy: %.2f%%" % (accuracy * 100.0))

          Accuracy: 90.43%

In [102]: from sklearn.metrics import confusion_matrix
          confusion_matrix = confusion_matrix(y_test, predictions)
          print(confusion_matrix)

          [[2504329   18193]
           [ 249534   24583]]
```

XGBoost: From the image we can see that out of all 2,796,639 products ordered across multiple orders, 2,504,329 products were correctly identified as reordered and 24583 products were correctly identified as not re-ordered. Also, 18193 products were incorrectly identified as reordered and 249534 were incorrectly identified as non-reordered products.

LightGBM: From the image we can see that out of all 2,796,639 products ordered across multiple orders, 2,501,066 products were correctly identified as reordered and 28978 products were correctly identified as not re-ordered. Also, 21456 products were incorrectly identified as reordered and 245139 were incorrectly identified as non-reordered products.

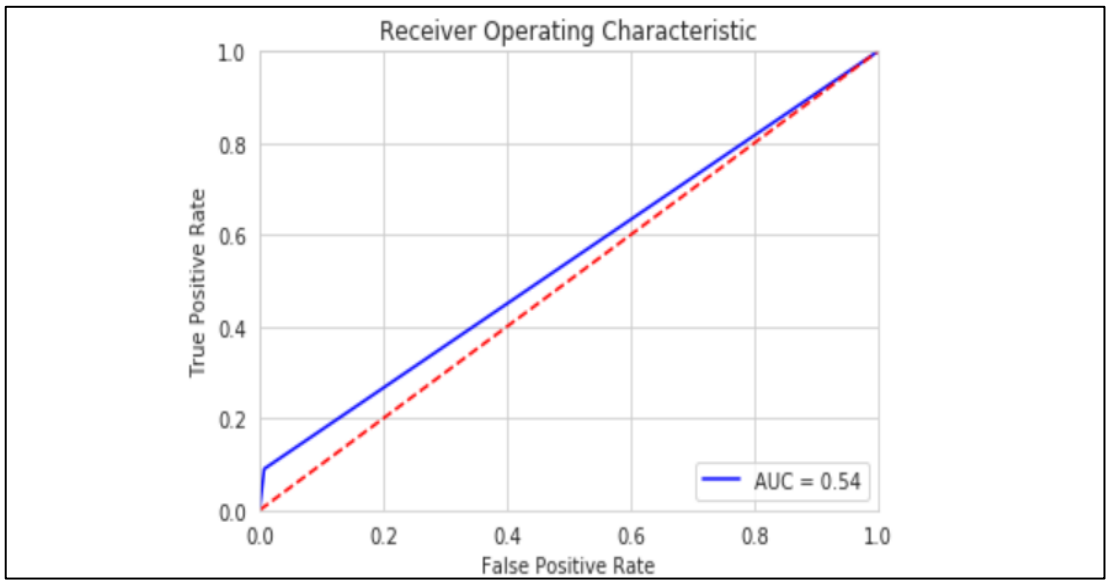
```
In [34]: #accuracy score of Light GBM
         print(accuracy_score(y_test, pred))

         0.9046730736430408

In [35]: from sklearn.metrics import confusion_matrix
         confusion_matrix = confusion_matrix(y_test , pred)
         print(confusion_matrix)

         [[2501066   21456]
          [ 245139   28978]]
```

	XGBoost	LightGBM
Recall/Sensitivity	0.909	0.917
Precision	0.992	0.991
Accuracy	0.9043	0.9046
Specificity	0.574	0.574

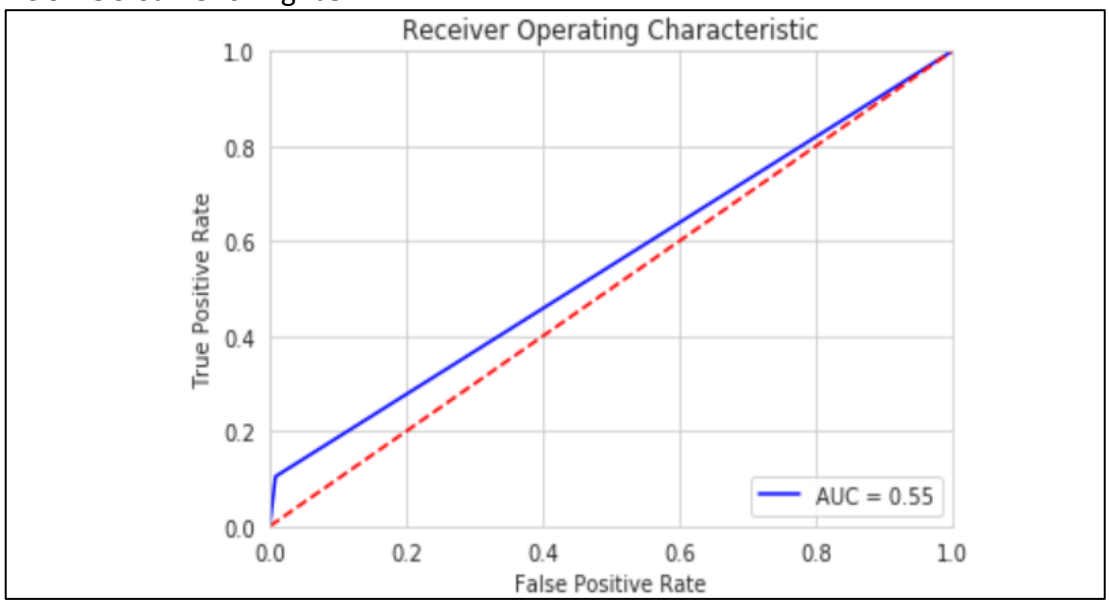
From the above table, we can see that the LightGBM gave better performance than XGBoost. As visible, the value of specificity i.e. correctly predicting non-reordered products among all the non-reordered products is pretty low. This means that the model is not very good at correctly identifying the non-reordered products and is marking them as reordered. Although the score is low, it does not impact our system much, as recommending a product incorrectly is still better than missing a valid product from the recommendation, which actually can impact a sale. Thus, for our model, precision plays a much important role and as shown in the results the value obtained for precision is quite high making it less likely to miss an important product in the recommended product list. Also, the accuracy of both our models is high enough to infer that most of the times it is classifying the ordered and non-reordered products correctly.

The **AUC-ROC curve** is one of the most important evaluation metrics for checking any classification model's performance. AUC tells how much the model is capable of distinguishing between classes which means that higher the AUC, better the model is at identifying whether the product will be reordered or not. In our case, the AUC we obtained is 0.54 and 0.55 for XGBoost and LightGBM respectively which means that our classifiers are currently doing a decent job. We can also see that LightGBM is able to distinguish better between the reordered and not-reordered products by a customer.

AUC-ROC Curve for XGBoost:



AUC-ROC Curve for LightGBM:



Prescriptive Analysis

We have used two approaches in our project to provide recommendations to the customers. These are apriori algorithm i.e. association rule based and product bundle frequency-based recommendations. We have explained the results from both the approaches below.

Apriori Algorithm – Association rule-based recommendation:

The below table shows the results of the apriori algorithm based recommendation system. We can see that Frequent item sets are built using Apriori algorithm. By utilizing the frequent set rules are created by Association rules.

Frequent items

```
In [30]: frequent_itemsets = apriori(basket_sets, min_support=0.0045, use_colnames=True)
```

```
In [31]: frequent_itemsets
```

Out[31]:

	support	itemsets
0	0.016088	(100% Raw Coconut Water)
1	0.025855	(100% Whole Wheat Bread)
2	0.015825	(2% Reduced Fat Milk)
3	0.035751	(Apple Honeycrisp Organic)

From the results below, we can infer that if the customer buys Lime Sparkling water, then Sparkling water grapefruit is also bought together. An antecedent is an item which is purchased first, and consequent is an item bought alongside antecedent. From the above output, let's take another example say, Limes is antecedent, and Bunched Cilantro is consequent. Association rule represented as {Limes}->{Bunched Cilantro}, which shows there is a strong relationship between the customers that purchased Limes also purchased Bunched Cilantro is the same transaction. Let's consider the first transaction to discuss support, confidence, and lift.

```
In [23]: rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
         rules.sort_values('lift',ascending = False)
```

Out[23]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
220	(Lime Sparkling Water)	(Sparkling Water Grapefruit)	0.019873	0.032463	0.005649	0.284239	8.755768
221	(Sparkling Water Grapefruit)	(Lime Sparkling Water)	0.032463	0.019873	0.005649	0.174005	8.755768
177	(Bunched Cilantro)	(Limes)	0.019438	0.060080	0.005346	0.275005	4.577326
176	(Limes)	(Bunched Cilantro)	0.060080	0.019438	0.005346	0.088974	4.577326
200	(Limes)	(Jalapeno Peppers)	0.060080	0.018213	0.004891	0.081413	4.469982
201	(Jalapeno Peppers)	(Limes)	0.018213	0.060080	0.004891	0.268556	4.469982
330	(Organic Ginger Root)	(Organic Garlic)	0.022160	0.046674	0.004603	0.207713	4.450248
429	(Bag of Organic Bananas, Organic Strawberries)	(Organic Raspberries)	0.026505	0.058418	0.005011	0.189067	3.236452
432	(Organic Raspberries)	(Bag of Organic Bananas, Organic Strawberries)	0.058418	0.026505	0.005011	0.085784	3.236452

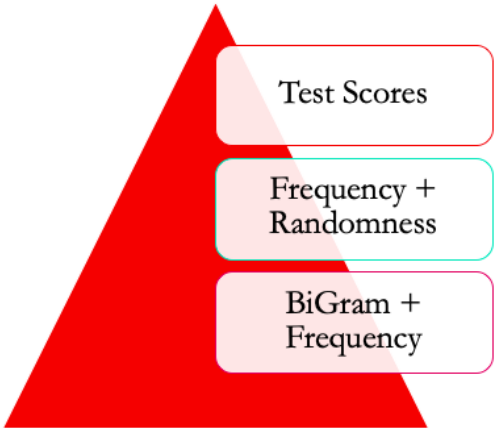
Support tells us how popular the item is, as measured by the proportion of transactions where an itemset appears. Antecedent support is 0.019 for Lime Sparkling water, which is the measure of the number of Lime Sparking water to the total number of transactions. Similarly, consequent support of 0.032 is nothing but a total number of Sparkling grape water divided by a total number of transactions. Support of 0.005 is the overall occurrence of both Lime Sparkling water and Sparking Water Grapefruit divided by a total number of transactions. We have set a min_support as 0.0045, so we consider only the transaction with support greater than a threshold which we set so that we can concentrate on those which has a significant impact on profit.

Confidence tells about the reliability of the rule. Confidence of .28 means that 28% of cases where Lime Sparkling water was purchased, the purchase also includes Sparkling grape water. It is a measure of support of {Lime Sparkling water, Sparkling water grapefruit} divided by the support of {Lime sparkling water}. We can also set the confident threshold and choose only those transactions that are above the min_confidence level. The closer the confidence is to the threshold, the rule is more useful. The rule which satisfies minimum support and minimum confidence, then it is classified as a strong rule. A frequent itemset is the one whose support is greater or equal to the minimum support threshold. One drawback that could be identified with the confidence value observed in our results is that it only accounts for how popular Lime sparkling water is, but not Sparking water grapefruit which may misinterpret the importance of Association rule, so we consider Lift in that case. Lift value of 8.75 is a ratio of confidence divided by benchmark confidence. Benchmark confidence is nothing but the support of consequent by a total number of transactions. Lift signifies how effective the association rule is in finding the consequent items. If the lift value is 1, then antecedent and consequent are independent, and no rule can be derived from them. If the value of lift is greater than 1, the antecedent and consequent are dependent on each other denoting a positive correlation. If the value is less then 1, the antecedent and consequent are negatively correlated, which means that increased probability of occurrence of antecedent decreases the probability of occurrence of the consequent. It ideal to select a rule which has a lift value greater than 1 which signifies a strong relationship between antecedent and consequent.

Product Bundle Recommendation

Shown in the table below is our recommendation result set for multiple products, for each of which we have recommended 7 products. The prediction and the credibility of our predictions are done based on the 3 rules mentioned in the below triangle. The first part includes the recommendation criteria created by using bi-grams and the calculating the frequency of items that are frequently bought together, in order to recommend those items for making purchases. The second part is where we measure our recommendation results

with output and provide a score based on the correct predictions.



We begin with deciding on the number of recommendations as 7 for each product. The recommendations are based on the Bigram + frequency combination. For instance, the first row of our result shows 7 products being recommended for Peach_Mango_juice, the result set shows the output from our algorithm directly if it is 7 or more. In case the recommended products for Peach_Mango_Juice is less than 7, it takes the first recommended product (i.e. the one with highest frequncy) and make recommendations for that. In this case it would first consider Honeycrisp_Apple to make further recommendations, followed by Organic_Avocado, Strawberries and so on. In case of a rare but possible scenario, when the model does not find frequent items for a product and is not able to get a product bundle for it, it gives out random products as recommendations.

Scoring Analysis on recommendation results:

The second part of our project focuses on recommending other products to a consumer when they order a product based on past purchases. The aforementioned is known as Product Bundle Recommendation and is written as a function to recommend products. We evaluate our results on the test dataset. For each order in the test dataset, beginning with the very first product, our algorithm can offer a list of recommendations for this particular product where the number of recommendations is equal to the total number of

Product	Rcommendation List(k = 7)
Peach_Mango_Juice	'Honeycrisp_Apple', 'Organic_Avocado', 'Strawberries', 'Organic_Strawberries', 'Cucumber_Kirby', 'Organic_Fuji_Apple', 'Organic_Whole_Milk'
Strawberries	Banana', 'Raspberries', 'Organic_Blueberries', 'Bag_of_Organic_Bananas', 'Seedless_Red_Grapes', 'Blueberries', 'Large_Lemon'
Banana	'Organic_Avocado', 'Organic_Strawberries', 'Organic_Fuji_Apple', 'Honeycrisp_Apple', 'Strawberries', 'Organic_Large_Extra_Fancy_Fuji_Apple', 'Organic_
Large_Lemon	Limes', 'Banana', 'Yellow_Onions', 'Organic_Italian_Parsley_Bunch', 'Organic_Baby_Spinach', 'Organic_Strawberries', 'Organic_Avocado'
Organic_Baby_Spinac	'Banana', 'Organic_Hass_Avocado', 'Organic_Grape_Tomatoes', 'Large_Lemon', 'Bag_of_Organic_Bananas', 'Organic_Baby_Arugula', 'Organic_Avocado'

products in the original order. Then, we check if any products purchased later in this order are present in the recommendation list. If they are present, it means that the recommendation provided for this product is accurate, and hence label this product as "rightly predicted."

```
scores = TestScore(test_data)
print("Mean Test Scores: ", numpy.mean(scores))
```

```
Mean Test Scores:  0.18298472779702069
```

Then, we iterate on all the products in this order to do the calculation mentioned above and compute the percentage of correct recommendations. Finally, we iterate on all the orders in the test dataset and calculate the average percentage of the right predictions. Our final score for the test data is around 0.18, which means that about 18% of the products in each order in the test dataset is from the recommendation list generated via our algorithm.

Key Takeaways

Amazon web services: AWS were put into practical use by our team for the first time while doing this project. We learnt about the various cloud-based service providers namely AWS, Azure and Google cloud platform and then selected AWS for building our models. We learned about a lot of services during the course of our project like Amazon S3, EC2, IAM, DynamoDB, Redshift, Amazon CloudWatch, Amazon Kinesis, Amazon Sagemaker, GPU etc. For running our project, although, we mainly used S3, IAM and EC2. We learned on connecting EC2 instance to Jupyter notebook by using Amazon CLI services. We even discovered how to decompress files in EC2 to overcome the decompressing capabilities of the amazon's storage data lake, S3.

XGBoost and Light GBM: We got to learn about new algorithms XGBoost and Light GBM and were also able to compare and see how each of them works for our model. We understood the basic functioning of both the tree-based algorithms. XGBoost follows level wise tree growth while Light GBM follows a leaf-wise tree growth, which causes the difference in timings required for executing models based on these two algorithms. Both of the algorithms can run over a GPU and have an in-build algorithm to handle missing values in the data, saving a significant amount of effort in terms of data preparation.

Power BI: During our course, we have done descriptive analysis and generated data visualizations using some libraries like matplotlib, seaborn, scikit-learn among others. A tool like Power BI was relatively new for us and this project provided us an opportunity to deep dive into the capabilities of this tool. In Power BI, we created measures that helped us create different reports using values of frequency occurrences of a particular product and so on. We also created conditional columns that helped us map numbers to category labels, for e.g., day of week. This mapping helps in providing clear and understandable visuals as we will be able to see the actual name of the day instead of viewing it as numbers. Another important feature of Power BI that we made use of was the slicers which enable us to filter

the reports based on our choice of departments, aisles, days of week, hour of day etc. Though not used in this project, we also learnt about interactions between different reports and how we have the ability to control these interactions based on our requirement. We also created a Power BI app from our workspace so that we have all our reports bundled together and we can share this app with end users for consumption.

Recommendation system: In the era where we are used to work with apps like Netflix, Facebook, Linkedin and Amazon, we had a basic understanding of "WHAT" a recommendation system is but this project pushed us far in the direction of actually understanding ''HOW" it is built. We researched deeply about Netflix recommendation system as our base and learned how they use Spark, Kafka and Cassandra in their architecture. We also read multiple papers explaining the algorithms or the logics used in building such systems. We learnt about the different types of filtering techniques as well as product bundling which is used to produce the recommendations.

Enhancements

Within the limited time frame, we had to restrict the scope of our project to the objectives discussed above. We had some more ideas in mind that we could have accomplished in case we had more time. We would definitely like to achieve these enhancements in the future. They are:

- Improve model efficiency and performance
- Web application and Mobile App
- Personalized recommendations
- GANS - introduce new functionality to the app where a user can click a picture of a product and a suitable machine learning can be used to recognize the product among several GANS generated and already existing images of products which could then be used for price comparison.

These are explained in detail below.

Although our model is giving pretty good result, if we had more time we could have worked towards improving it even further by working on hyperparameter tuning, feature engineering and regularization.

We could deploy this model to a web application where users can click on a product and get recommendations based on the product they picked. We were planning to develop this using PyCharm and Django framework. This could also be extended to a mobile application. We could also add images for the products as well as get their prices, and this will make the app more interactive to the end-users.

Another feature that can be added to enhance this model is personalized recommendations. Currently, our model recommends products to a customer based on the orders placed by different customers in the past (shopping behavior in general). This comes under the widget "Frequently bought together" or "People who bought this also bought

this." Since we have user information (user IDs) as well as the information about the orders placed by a particular user, we can also recommend products to a customer based on their individual purchase history. This comes under the widget "Recommended for you" and most of the times makes it easier for the customer to pick products on the go. We can also consider the purchase patterns of an individual customer, for example, the day of the week, the time of the day that they most probably place orders or the frequency of placing orders itself. Based on this information, we can provide notifications to the users, maybe reminding them that it's time to place an order again. This way, customers feel that they are given importance and taken care of. This feature also helps accelerate their work and therefore save time.

The next possible application is to use GANS in our project. When a user clicks the image of a product, we can identify the product irrespective of the angle or focus with which the image has been captured, from among the several GANS generated images as well as existing images, which can be further used to provide the user with the lowest prices of that product available across multiple brands and stores. This way, customers have the ability to choose the product with a pricing that is suitable for them.

References

https://www.datacamp.com/community/tutorials/market-basket-analysis-r
https://ieeexplore.ieee.org/document/7231468
https://ieeexplore.ieee.org/document/7855955
https://ieeexplore.ieee.org/document/5645371
https://en.wikipedia.org/wiki/Affinity_analysis
https://www.medium.com