Capstone Project

on

Amazon Product Review Analysis

(Prime Pantry Data Set)

Under Supervision of Dr. Amit Kumar

Advanced PGP in Data Science and Machine Learning

Submitted by:

Sania Inamdar

Subhranil Mondal

Tejaswini Dhangar

Yash Singh

Table of Contents

Chapter 1

Introduction

As online marketplaces have been popular during the past decades, the online sellers and merchants ask their purchasers to share their reviews about the products they have bought.

Everyday millions of reviews are generated all over the Internet about different products, services and places.

This has made the Internet the most important source of getting ideas and opinions about a product or a service. However, as the number of reviews available for a product grows, it is becoming more difficult for a potential consumer to make a good decision on whether to buy the product.

Different reviews about the same product on one hand and ambiguous reviews on the other hand makes customers more confused to get the right decision. Here the need for analyzing this contents seems crucial for all e-commerce businesses.

Sentiment analysis ,classification, clustering and time series analysis is a computational study which attempts to address this problem by extracting subjective information from the given texts in natural language, such as reviews and sentiments. Different approaches have used to tackle this problem from natural language processing, text analysis, computational linguistics, and biometrics.

In recent years, Machine learning methods have got popular in the semantic and review analysis for their simplicity and accuracy. Amazon is one of the e-commerce giants that people are using every day for online purchases where they can read thousands of reviews dropped by other customers about their desired products.

These reviews provide valuable opinions about a product such as its quality and recommendations which helps the purchasers to understand almost every detail of a product. This is not only beneficial for consumers but also helps sellers who are manufacturing their own products to understand the consumers and their needs better.

This project is considering the sentiment analysis problem for online reviews using supervised, unsupervised and time series analysis approaches to determine the overall semantic of customer reviews by classifying them into positive, neutral and negative sentiment. The data used in this study is a set of Prime Pantry product reviews from Amazon that is collected from Base Dataset (https://jmcauley.ucsd.edu/data/amazon/).

1.1 Objective

Thomas, a global market analyst, wishes to develop an automated system to analyze and monitor an enormous number of reviews. By monitoring the entire review history, he will analyze tone, language, keywords and trends over time to provide valuable insights that increase the success rate of existing and new products and marketing campaigns.

1.2 Scenarios Provided

Here we are provided with two scenarios:

1. Inventory Optimization and Demand Forecasting
2. Customer Retention and Sentiment Forecasting

Based on above two scenarios we have to work on any one of them and help Thomas build the required automated product review analyzing system.

1.3 Scenario chosen for further analyses

We are working on scenario two in our project i.e. Customer Retention and Sentiment Forecasting. In this we have to focus our analysis on Customer retention strategy through feedback analysis (customer classification and clustering as an outcome of analyzing the review text). Trend and seasonality analysis to predict how frequently a particular category of customer would shop in the future. (Time Series Analysis)
Time Series component: Trend, Seasonality analysis to predict how frequently this
the customer would buy new products.

Chapter 2

Data Preparation and Preprocessing

This section presents the method of the study. The programming environments will be described in the first part. How and where the data was gathered as well as the data preparation approach will be discussed in the second part. In the last part, the procedure of data preprocessing will be explained.

2.1 Programming Environment

Python is one of the most widely used programming language in data science and machine learning. Python has a huge libraries that can be used for solving various data related problems. Some of well-known libraries are Pandas, Numpy, Scikit-learn(Supervised and unsupervised models), Statsmodels(Time Series Analysis), Matplotlib, Seaborn and many more.

2.2 Data Fetching

The first step of any analysis is data fetching from respective data source. Here in our case data is fetched from Amazon product data. The data we have to download was **5-core** and **ratings only** data from the Base dataset (https://jmcauley.ucsd.edu/data/amazon/). These two file contains reviews and rating related columns. And for price, descriptions, sales-rank, brand info and co-purchasing links of different categories pf Amazon products we have to download **metadata.** We are going to work on **Prime Pantry** dataset from the Amazon product data.

Amazon Prime Pantry data consist of information of Prime members and non-Prime members in selected areas to buy nonperishable food items and household supplies in everyday package sizes.

Format of the downloaded file was one-review-per-line in JSON format. For analysis purpose we read the JSON file through command available in Python. The size of Prime pantry 5core that is (137788 ,12) and the size of meta Prime pantry (10813, 19). After merging both the dataset on asin we are getting the size of dataset is (137769, 30).

The following is example of JSON file:

{ "image": ["https://images-na.ssl-images-amazon.com/images/I/71eG75FTJJL._SY88.jpg"], "overall": 5.0, "vote": "2", "verified": True, "reviewTime": "01 1, 2018", "reviewerID": "AUI6WTTT0QZYS", "asin": "5120053084", "style": { "Size:": "Large", "Color:": "Charcoal" }, "reviewerName": "Abbey", "reviewText": "I now have 4 of the 5 available colors of this shirt... ", "summary": "Comfy, flattering, discreet--highly recommended!", "unixReviewTime": 1514764800 }

2.2.1 Reading data in Python:

```
[5]  #Importing Prime Pantry 5 core data
     pantry_df = pd.read_json('/content/Prime_Pantry_5.zip', orient='records', lines=True)
     pantry_df.shape
```

```
(137788, 12)
```

```
#Importing Prime Pantry meta data
pantry_meta_df = pd.read_json('meta_Prime_Pantry.zip', orient='records', lines=True)
pantry_meta_df.shape
```

```
(10813, 19)
```

2.2.2 Merging data

```
[8]  # Merging Prime Pantry 5core data with the meta data

     pantry=pd.merge(pantry_df,pantry_meta_df,on='asin')
     pantry.shape
```

```
(137769, 30)
```

COLUMNS	DESCRIPTION
OVERALL	Rating given to product
VERIFIED	User is verified or not
REVIEWTIME/DATE	Date of the review
REVIEWERID	Id of the user
ASIN	Product ID
REVIEWER NAME	Name of the reviewer
REVIEWER TEXT	Review text of reviewer
SUMMARY TEXT	Summary of review text
UNIXREVIEWTIME	Time of the review
VOTE	Helpful votes of the review
IMAGE	Images that users post after they have received the product
STYLE	A dictionary of the product metadata, e.g., "format" is "hardcover"
CATEGORY	List of categories the product belongs to
TECH1	The first technical detail table of the product
DESCRIPTION	Description of the product
FIT	
TITLE	Name of the product
ALSO BUY	Also buy product
TECH2	The second technical detail table of the product
BRAND	Brand name
FEATURE	Bullet-point format features of the product
RANK	Sales rank information
ALSO VIEW	Also view products
DETAILS	Product Details
MAIN CATEGORY	URL
SIMILAR ITEMS	Similar product table
DATE	Date of review
PRICE	Price in US dollars (at time of crawl)
IMAGEURL	URL of the product image
IMAGEURLHIGHRES	URL of the high resolution product image

Fig 2.1: Description of each product

2.3 Data Preparation

For preparing the desired data a simple code was written in python to remove the features which were irrelevant or entire column was having null values. Many features were removed using these code we were able to filter out features/columns which were irrelevant or entire column was having null values.

```
#checking values for each column
for i in list(pantry_bkp.columns):
  print('Column Name={}'.format(i),'\n',pantry_bkp[i].value_counts())
  print(30*'=====')
```

By analyzing each column we drop these columns

```
#Dropping non required columns
pantry_bkp.drop(['vote','image','style','tech1','tech2','category','details','main_cat','imageURL','imageURLHighRes','fit','similar_item'],axis=1,inplace=True)
```

- As vote,image,style,tech1,tech2,fit,similar_item column having majorily null value's and are not relevant so dropped it.
- category having blank [] so dropped it.
- main_cat,imageURL,imageURLHighRes having images url that are not important for analysis so dropped it.

During analyses of columns we find whitespaces and irrelevant text in price column. We clean the data and change it into desired format by using regular expression techniques.

```
pantry_bkp['price'].head(50)
```

```
0
1
2
3
4
5
6
7
8
```

```
#for finding string value in price column
pantry_bkp['price'][340:390]
```

```
340    .a-section,.amazon_yum_pantry.burj-body .burj ...
341    .a-section,.amazon_yum_pantry.burj-body .burj ...
342    .a-section,.amazon_yum_pantry.burj-body .burj ...
343    .a-section,.amazon_yum_pantry.burj-body .burj ...
344    .a-section,.amazon_yum_pantry.burj-body .burj ...
345    .a-section,.amazon_yum_pantry.burj-body .burj ...
346    .a-section,.amazon_yum_pantry.burj-body .burj ...
347    .a-section,.amazon_yum_pantry.burj-body .burj ...
```

After cleaning the price column we got some 16634 null values in column and as the data was category-wise so the prices were same for observations in continuation upto 1000-3000 observations. As dropping observation can lead to data loss. So we filled the null values with forward fill and backward fill command available in Python.

```
[ ]  pantry_bkp['price'].fillna(method='bfill',inplace=True)
```

```
[ ]  pantry_bkp['price'].fillna(method='ffill',inplace=True)
```

Here, we have completed with missing value treatment and all the steps required for data to be ready for further analysis.

In the next part we will do Exploratory Data Analysis (EDA) to see how the review text, ratings, price, brand, reviewerID, reviewer name columns etc. are relating with each other. So we can go for selection of columns for further analysis.

2.4 Data Preprocessing:

Text preprocessing involves transforming text into a clean and consistent format that can then be fed into a model for further analysis and learning. To preprocess your text simply means to bring your text into a form that is predictable and analyzable for your task.

Raw text data might contain unwanted or unimportant text due to which our results might not give efficient accuracy, and might make it hard to understand and analyze. So, proper pre-processing must be done on raw data.

2.4.1 Benefits of Data Preprocessing:

Preprocessing of Data is necessary because of the presence of unformatted real-world data (Raw Data). Mostly real-world data is composed of -

- Inaccurate data (missing data) - There are many reasons for missing data such as data is not continuously collected, a mistake in data entry, technical problems with biometrics, and much more, which requires proper Data Preparation.
- The presence of noisy data (erroneous data and outliers) - The reasons for the existence of noisy data could be a technological problem of gadget that gathers data, a human mistake during data entry and much more.

- Inconsistent data - The presence of inconsistencies are due to the reasons such that existence of duplication within data, human data entry, containing mistakes in codes or names, i.e., violation of data constraints and much more necessitate Data Preparation and analysis.

The preliminary step of the data preprocessing is to import all the required libraries.

Following are the libraries required for data preprocessing:

String: Python String module contains a single utility function. This function split the specified string into words using str.

Spacy: Spacy is a free open-source library for natural language processing (NLP) in Python. It features named entity recognition (NER), parts of speech (POS) tagging, words vectors and more.

Re: A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression.

STOP_WORDS: This library is used to remove stop words from the data.

Word Cloud: Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud.

```
#libraries required for data preprocessing
import string
import spacy
import re
nlp=spacy.load('en_core_web_sm')
from spacy.lang.en.stop_words import STOP_WORDS

import nltk
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')                                    #Multilingual Wordnet Data from OMW with newer Wordnet versions
from nltk.stem import WordNetLemmatizer

from wordcloud import WordCloud, STOPWORDS
import spacy
from collections import Counter

nlp=spacy.load("en_core_web_sm")
```

The next step is to remove non-word characters, extra spaces, remove all the numbers except those attached to a word, remove all the hyphens other than those between the words, remove punctuation in the strings and to remove the stop words.

```
#defining function for preprocessing
def preprocess(text,remove_digits=True):
    text = re.sub('\W+',' ', text)                                  #for replacing non-word characters
    text = re.sub('\s+',' ', text)                                  #for replacng extra spaces
    text = re.sub("(?<!\w)\d+", "", text)                           # Remove all numbers except those attached to a word
    text = re.sub("-(?!\w)|(?<!\w)-", "", text)                     # Remove all hyphens except between two words
    text=text.lower()
    nopunc=[char for char in text if char not in string.punctuation]      #for removing punctuation in the strings
    nopunc=''.join(nopunc)


    return [word for word in nopunc.split() if word.lower() not in STOP_WORDS] #for removing stop words
```

2.4.2 Stop Words:

Stop words are the most common words in any language that do not carry any meaning and are usually ignored by NLP. In NLP, stop words are typically removed from a text before it is processed for analysis. This is done to reduce the size of the text and to avoid irrelevant information.

The next process is Stemming and Lemmatization.

2.4.3 Stemming:

Stemming is a technique used to extract the base form of the words by removing affixes from them. It is just like cutting down the branches of a tree to its stems.

2.4.4 Lemmatization:

Lemmatization is a text normalization technique used in Natural Language Processing (NLP), that switches any kind of a word to its base root mode. Lemmatization is responsible for grouping different inflected forms of words into the root form, having the same meaning.

```
# Defining a function for lemitization
def lemmatize_verbs(words):

    lemmatizer = WordNetLemmatizer()
    lemmas = []
    for word in words:
        lemma = lemmatizer.lemmatize(word, pos='v')
        lemmas.append(lemma)
    return lemmas
def lemmatize(words):
    lemmas = lemmatize_verbs(words)
    return lemmas
```

2.4.5 Text Blob:

Text Blob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

```
#using textblob for sentiment analysis
from textblob import TextBlob
#function to calculate polarity
def getPolarity(review):
  return TextBlob(review).sentiment.polarity

# function to analyze the reviews
def analysis(score):
    if score <= 0:
        return 'Negative'
    elif score >=0.5:
        return 'Positive'
    else:
        return 'Neutral'
```

Using Text Blob for Sentiment Analysis, we calculate the polarity.

2.4.6 Polarity:

Sentiment polarity for an element defines the orientation of the expressed sentiment, i.e., it determines if the text expresses the positive, negative or neutral sentiment of the user about the entity in consideration. It is a float where score lesser than or equal to zero is termed 'Negative', Score greater than or equal to 0.5 is termed 'Positive', and score ranging between 0 to 0.5 is termed 'Neutral'.

In the next step, on the basis of polarity score, word labelling is done

[ ]

reviewerID	asin	reviewerName	unixReviewTime	description	title	also_buy	brand	feature	rank	also_view	price	review_text	clean_text	Polarity	Analysis
A31Y9ELLA1JUB0	B0000DIWNI	Her Royal Peepness Princess HoneyBunny Blayze	1443052800	[Saran Premium Wrap is an extra tough yet easy...	Saran Premium Plastic Wrap, 100 Sq Ft	[B01MY5FHT6, B000PYF8VM, B000SRMDFA, B07CX6LN8...	Saran	[]	[]	[B077QLSLRQ, B00JPKW1RQ, B000FE2IK6, B00XUJHJ9...	1.48	Pretty Good For plastic Wrap I purchased this ...	pretty good plastic wrap purchase saran premiu...	0.187037	Neutral
A2FYW9VZ0AMXKY	B0000DIWNI	Mary	1435017600	[Saran Premium Wrap is an extra tough yet easy...	Saran Premium Plastic Wrap, 100 Sq Ft	[B01MY5FHT6, B000PYF8VM, B000SRMDFA, B07CX6LN8...	Saran	[]	[]	[B077QLSLRQ, B00JPKW1RQ, B000FE2IK6, B00XUJHJ9...	1.48	The Best Plastic Wrap for your Cooking, Baking...	best plastic wrap cook bake food storage need ...	0.317857	Neutral
A1NE43T0OM6NNX	B0000DIWNI	Tulay C	1434153600	[Saran Premium Wrap is an extra tough yet easy...	Saran Premium Plastic Wrap, 100 Sq Ft	[B01MY5FHT6, B000PYF8VM, B000SRMDFA, B07CX6LN8...	Saran	[]	[]	[B077QLSLRQ, B00JPKW1RQ, B000FE2IK6, B00XUJHJ9...	1.48	Good and strong. Good wrap, keeping it in the ...	good strong good wrap keep fridge make easier ...	0.611111	Positive
AHTCPGK2CNPKU	B0000DIWNI	OmaShops	1433289600	[Saran Premium Wrap is an extra tough yet easy...	Saran Premium Plastic Wrap, 100 Sq Ft	[B01MY5FHT6, B000PYF8VM, B000SRMDFA, B07CX6LN8...	Saran	[]	[]	[B077QLSLRQ, B00JPKW1RQ, B000FE2IK6, B00XUJHJ9...	1.48	Doesn't cling as well to dishes as other brand...	doesn t cling dish brand tangle prefer saran w...	0.000000	Negative

After labelling, value counts of each sentiments are taken into consideration. Positive Reviews in the dataset are 64528, Negative reviews are 27181, and Neutral Reviews are 45824.

2.4.7 Word Cloud:

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud.

```
def wc(data,bgcolor,title):
    plt.figure(figsize = (20,20))
    wc = WordCloud(background_color = bgcolor, max_words = 1000,  max_font_size = 50)
    wc.generate(data)
    plt.imshow(wc)
    plt.axis('off')
```

2.4.7.1 Top Most Common Words:

Fig 2.2: Top Most Common Words

2.4.7.2 Top Most Positive Words:

Fig 2.3: Top Most Positive Words

2.4.7.3 Top Most Neutral Words:

Fig 2.4: Top Most Neutral Words

2.4.7.4 Top Most Negative Words:

No table of figures entries found.

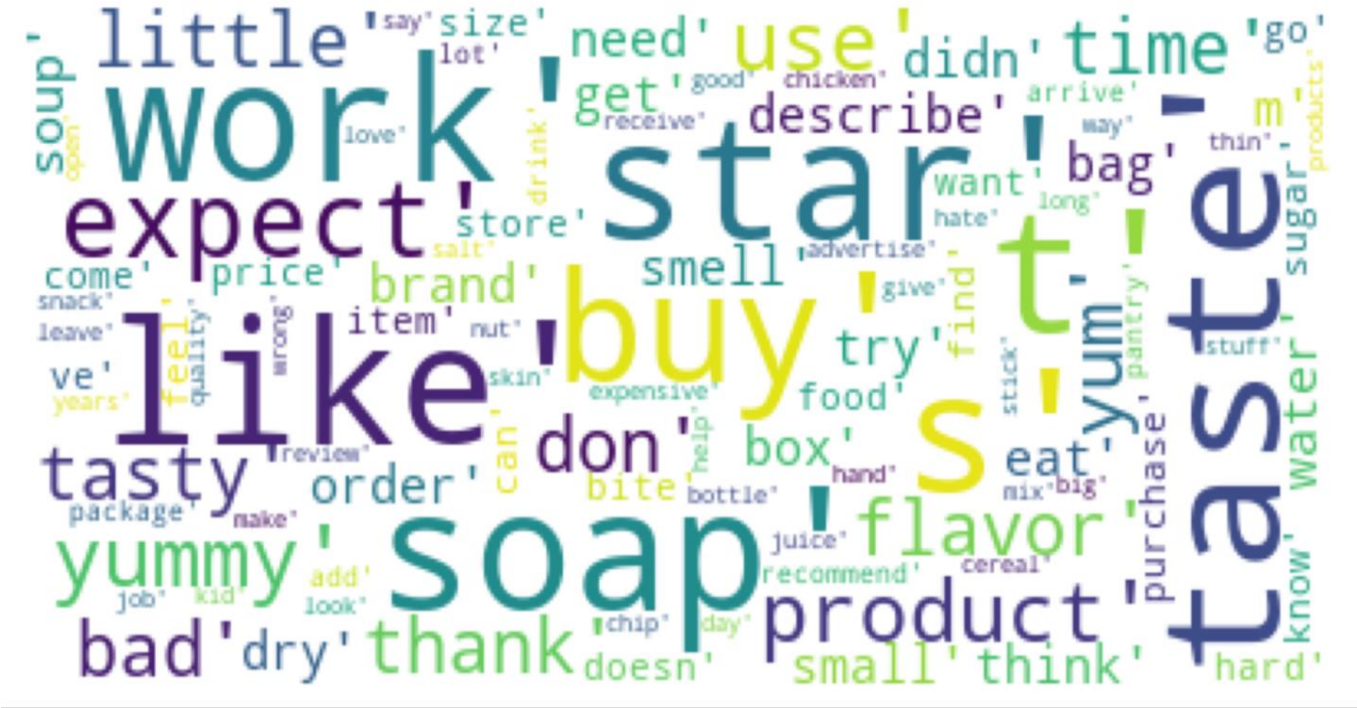


Fig 2.5: Top Most Negative Words

Chapter 3
Exploratory Data Analysis (EDA)

Exploratory Data Analysis is a process of examining or understanding the data and extracting insights or main characteristics of the data. EDA is generally classified into two methods, i.e. graphical analysis and non-graphical analysis.

EDA is very essential because it is a good practice to first understand the problem statement and the various relationships between the data features before getting your hands dirty.

3.1 Objectives:

1.Data Exploration

2. Identifying Outliers

3. Identifying Null values

4.Time wise spread of data

3.2 Data Exploration:

In our dataset we have total 30 columns. After preprocessing we come to know that there are 5 important features that we have to consider mainly to build the EDA.

1. Rating - Rating given to product
2. Verified - User is verified or not
3. Price - Price of products in US dollars
4. Review - Review text of reviewer
5. Analysis - Sentiment of customers toward products

3.2.1 Rating:

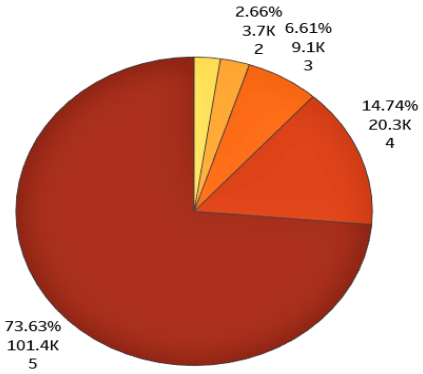


Fig 3.1: Ratings

So, we can conclude that majority of pantry products are liked by customers, as they give more 5 star rating.

3.2.2 Verified:

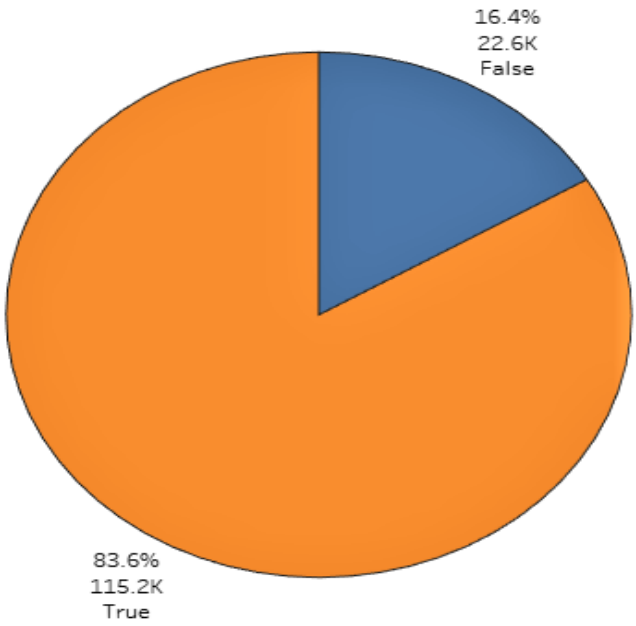


Fig 3.2 Verified Customer Percentage

In amazon prime pantry 115.2K customers are verified. So we can say that the reviews from the verified customers are appropriate.

3.2.3 Price:

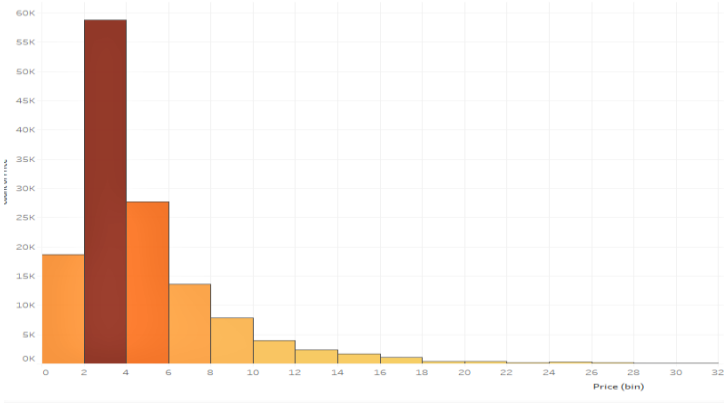


Fig 3.3: Price Distribution

Here we can see the maximum price of the product ranges in between $0 - $8. So we can say that maximum prime pantry products are affordable to all.

3.2.4 Analysis:

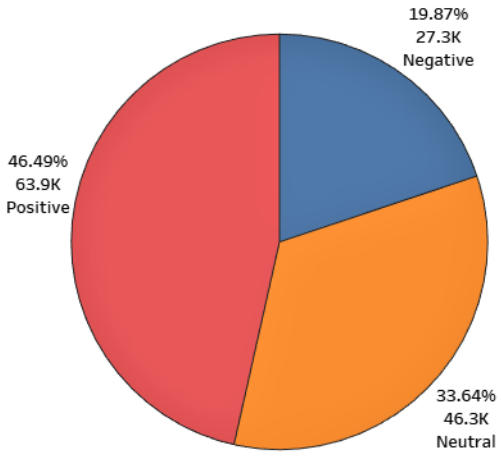


Fig 3.4: Sentiment Analysis Ratio

Here we can see 46.5% reviews are positive reviews and 19.87% reviews are negative reviews we got. So we can say that most of the customers are liking pantry products.

3.2.5 Rating wise Data Exploration:

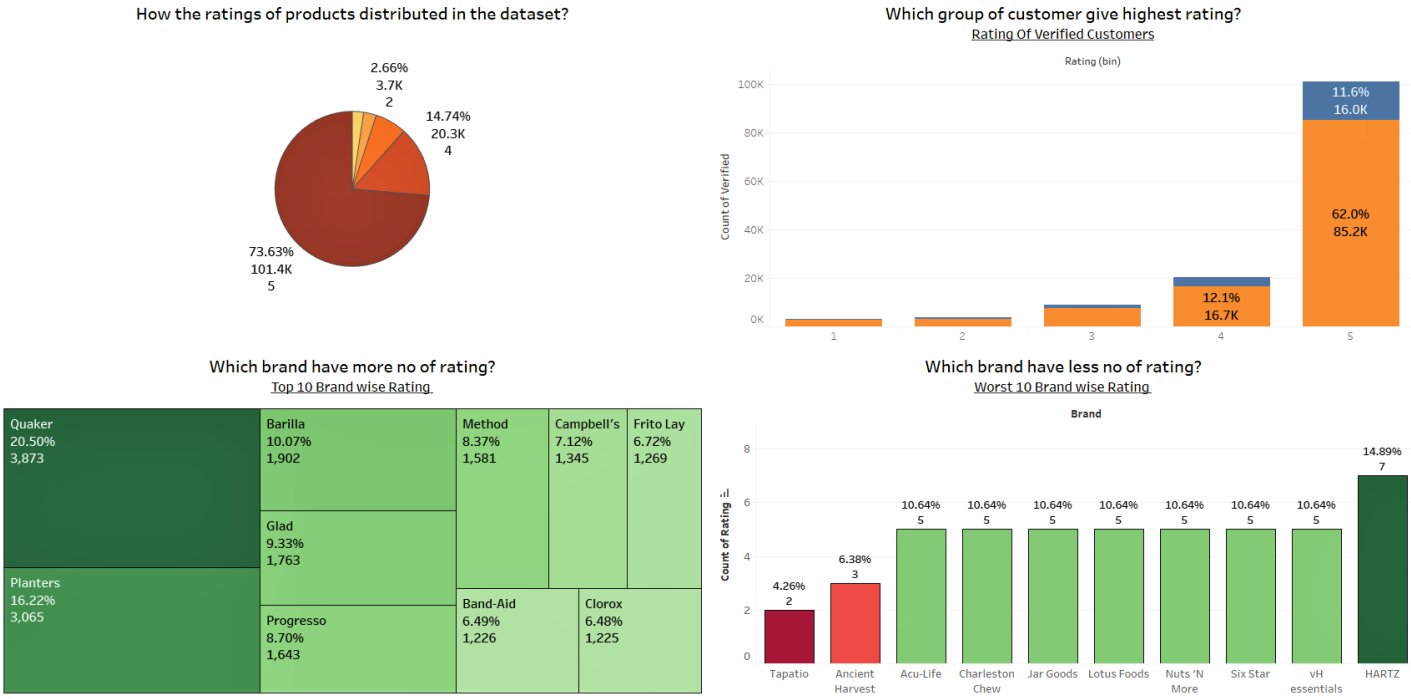


Fig 3.5: Rating Wise Data Exploration

From the above dashboard we can conclude,

- Verified customers are giving more 5 star rating than non-verified customers.
- The brand Quaker got highest no of rating than rest of the products.So we can say that Quaker is most popular.
- The brand Tapatio got lowest no of rating among all. So we can recommend Tapatio to improve the product qualities.

3.2.6 Price wise Data Exploration:

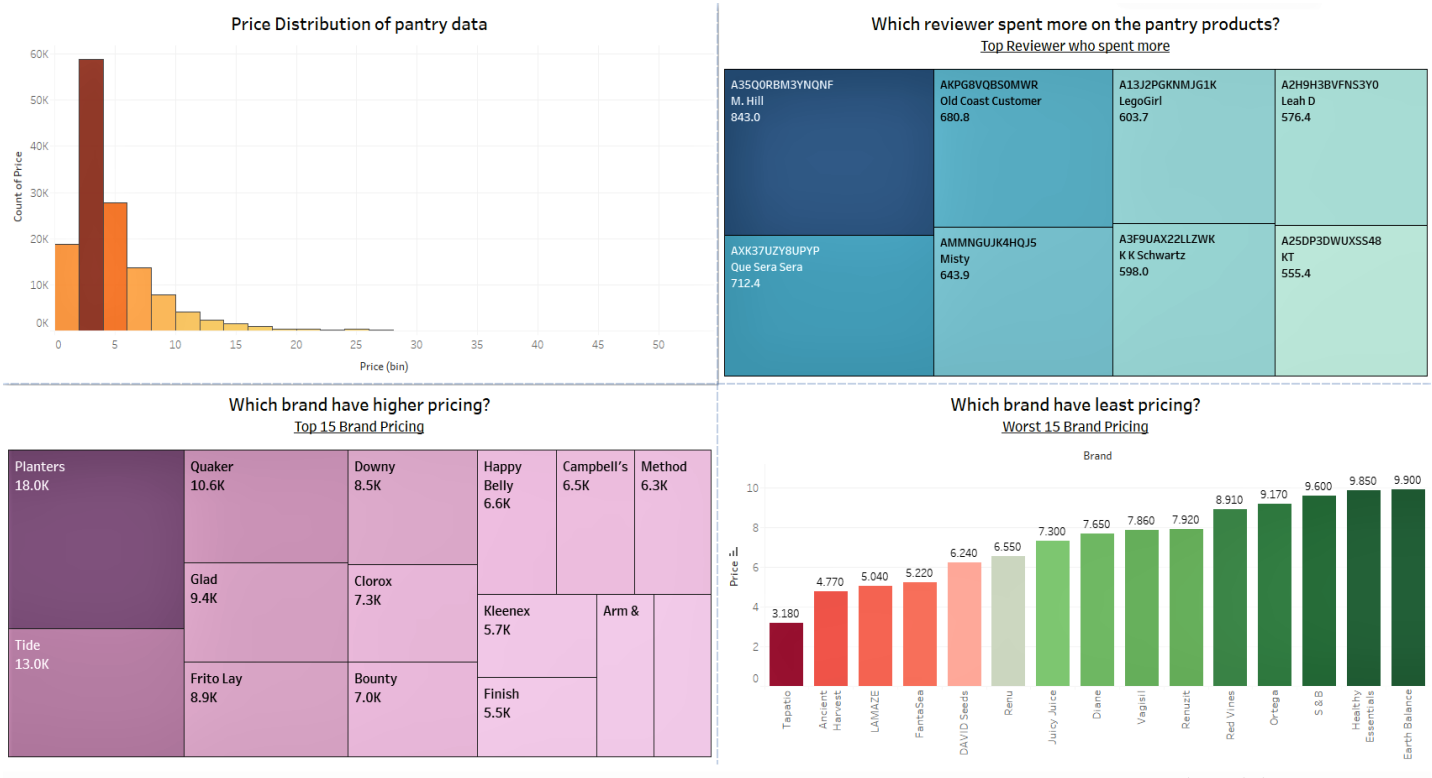


Fig 3.6: Price wise Data Exploration

From the above dashboard we can conclude,

- Reviewer M.Hill has spent $ 843 till now on pantry products. So we can say all his reviews are reliable as he purchased all kinds of pantry products.
- Planter have the highest pricing among all. So we can say that planter have more price distribution among all brands.
- Tapito has the least pricing among all. So we can say that Tapito have least price distribution among all brands.

3.2.6 Top 20 Brands based on Reviews and Total Sale:

This image is showing top 20 brands based on ratings, price and review text columns and also shows top 50 customer according to review count.

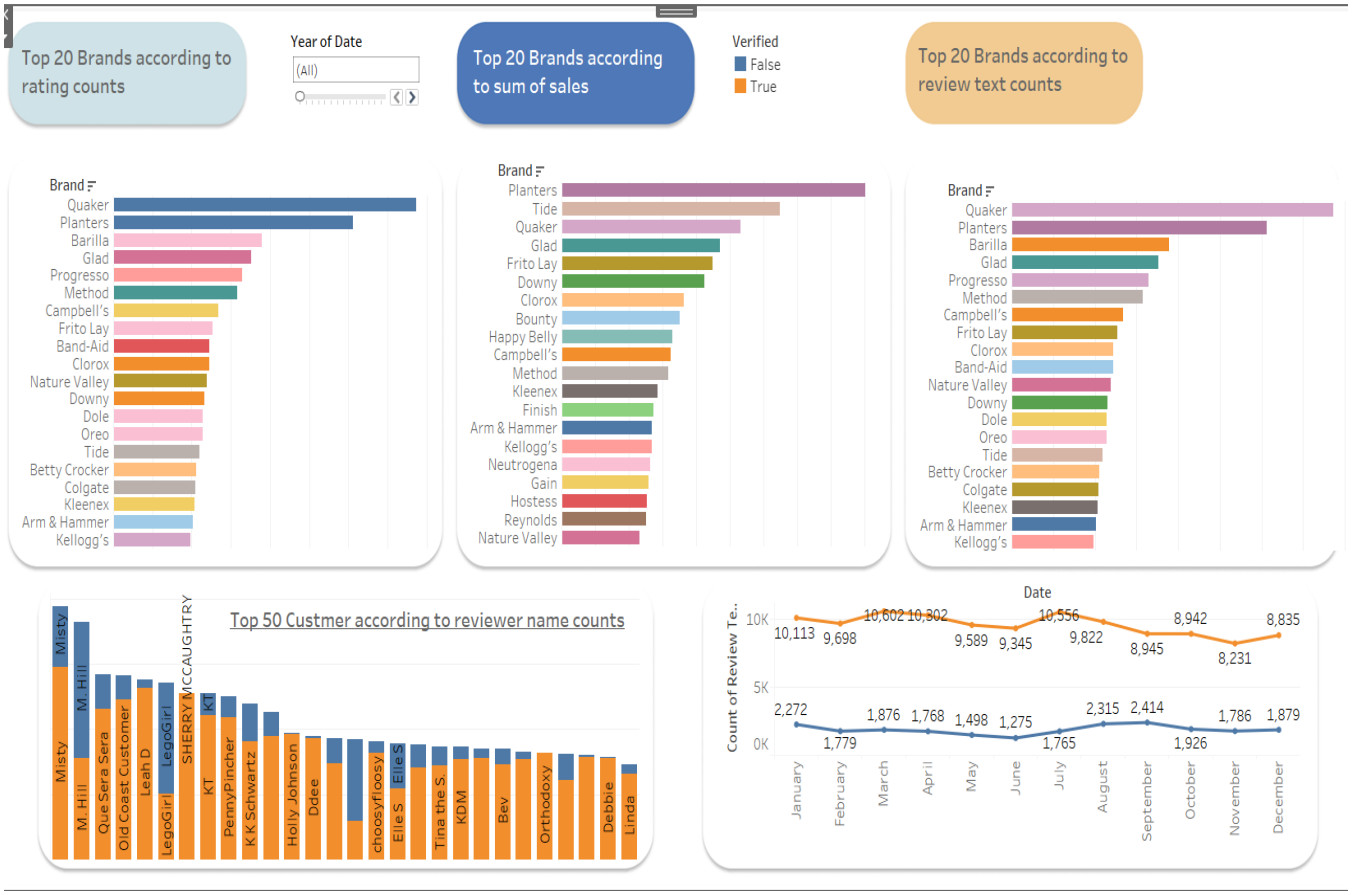


Fig 3.7: Top Brands based on Reviews and Total Sale

In the next part we have also done EDA for bottom 20 brands based on ratings, price and review text columns and also shows bottom 50 customer according to review count.

3.2.6 Bottom 20 Brands based on Reviews and Total Sale:

This dashboard also displays total percentage of ratings and the reviewer is verified or not.

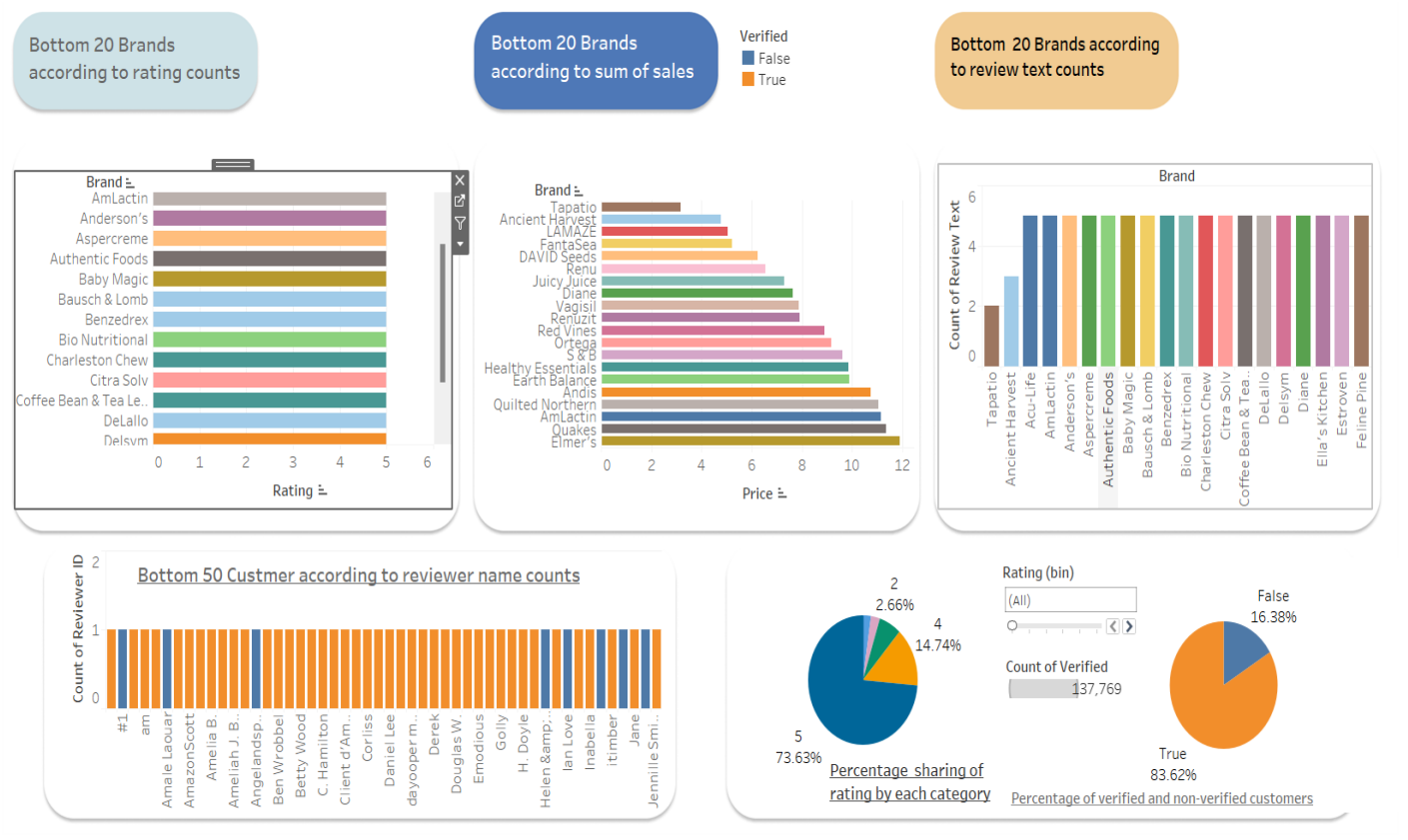


Fig 3.8: Bottom 20 Brands based n Reviews and Total Sales

3.2.7 Sentiment Wise Data Distribution:

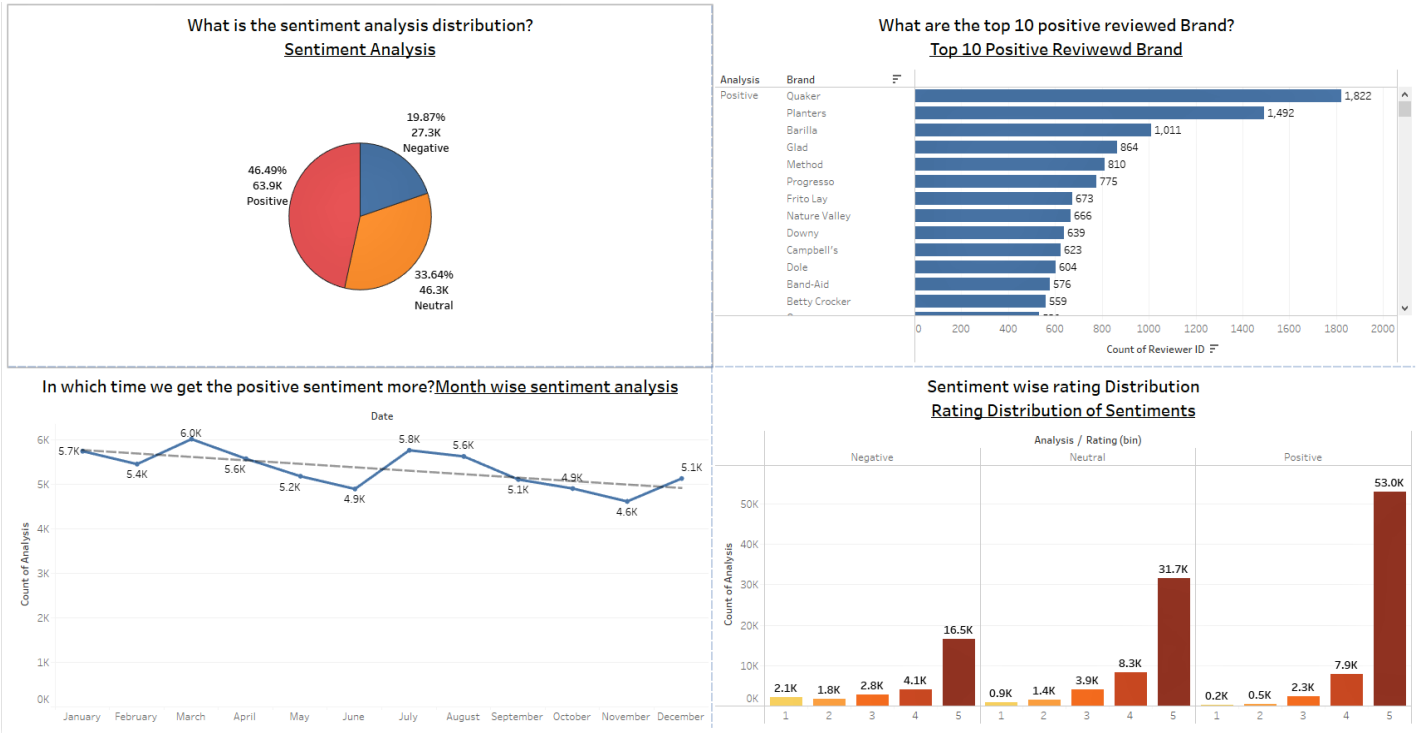


Fig 3.9: Sentiment Wise Data Distribution

From the above dashboard we can conclude,

- The brand Quaker got the most positive sentiment review among all. So the products from Quakers are pretty good among all.
- In March we've got the most positive sentiments among all. In April we've got the most negative sentiments among all.
- Positive sentiment products got most no of 5 star rating & less number of 1 star rating.

Amazon Prime Pantry

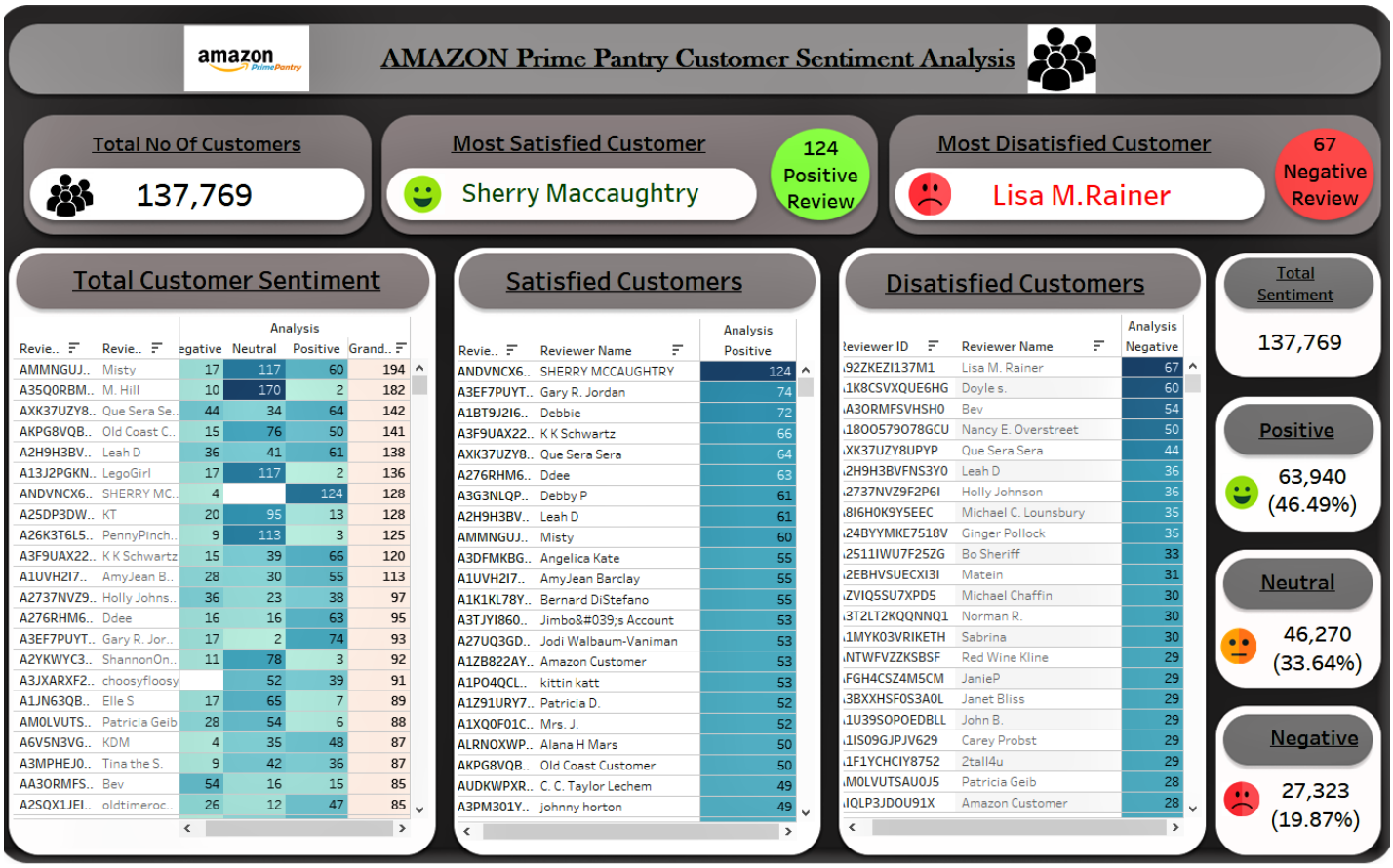
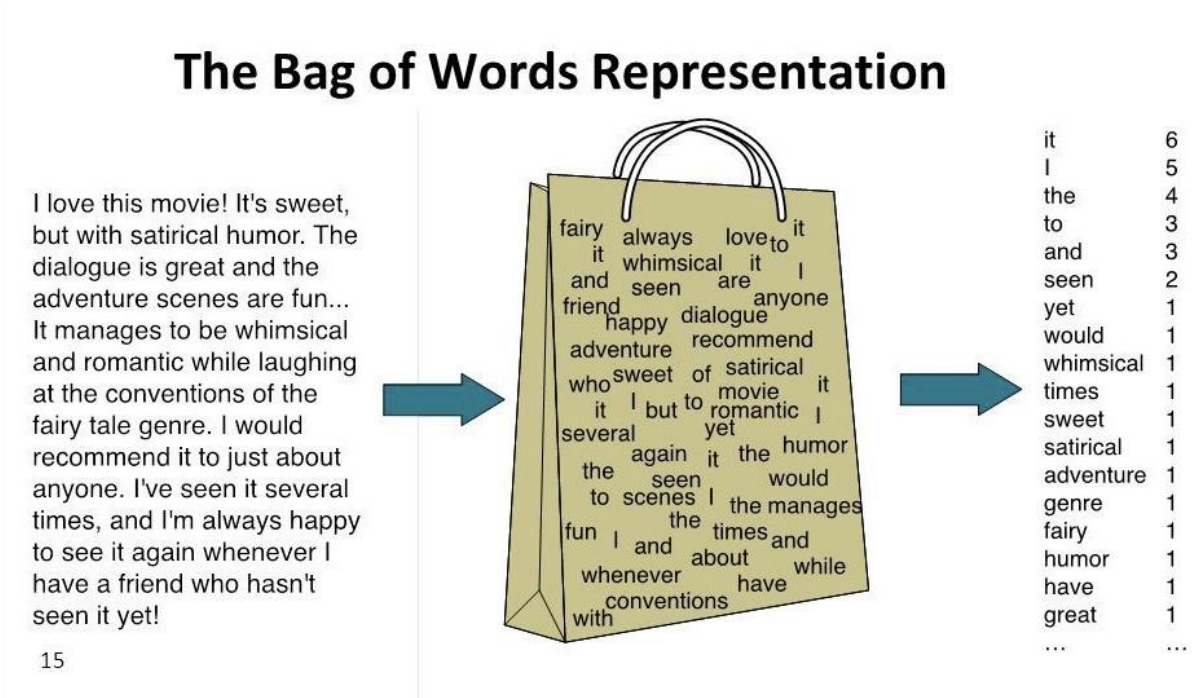


Fig 3.10: Customer Sentiment Analysis

From the above dashboard we can conclude that there are a total 137,769 customers present. There are 63,940 positive reviews, 46,270 neutral reviews and 27,323 negative reviews present in our dataset. If we consider the total sentiments of each customer then we find that the customer **Sherry Maccaughtry** is the most satisfied, as the customer gives more number (124) of positive reviews & only 4 negative reviews to the products. And **Lisa M.Raine**r is the most dissatisfied customer,as the customer gives more number(67) of negative reviews & only 17 neutral reviews to the product.

Chapter 4
Feature Extraction

4.1 Bag Of Words Method:



Machine learning algorithms work only with fixed-length vector of numbers rather than raw text, the input (in this case text data) need to be parsed. The method for transforming the texts into features is called the Bag of words model of text, which is a commonly used method of feature extraction. The approach works by creating different bags of words that occur in the training data set where each word is associated with a unique number. This number shows the occurrence of each word in the document. A simple illustration of the Bag of words model can be seen in figure below. The model is called a bag of words because the position of the words is in the document discarded.

4.2 Count Vectorization Method:

CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text, but it doesn't change the meaning of the text, it will consider the order of the text so that the context of the text doesn't change. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis). Let's take one example:

```
text = ['Hello my name is james, this is my python notebook']
```

	hello	is	james	my	name	notebook	python	this
0	1	1	1	1	1	0	0	0
1	0	1	0	1	0	1	1	1

I have 2 text inputs, what happens is that each input is preprocessed, tokenized, and represented as a sparse matrix. By default, Countvectorizer converts the text to lowercase and uses word-level tokenization.

4.3 TF-IDF Vectorization Method:

Term frequency-inverse document frequency is a text vectorizer that transforms the text into a usable vector. It combines 2 concepts, Term Frequency (TF) and Document Frequency (DF).The term frequency is the number of occurrences of a specific term in a document. Term frequency indicates how important a specific term in a document. Term frequency represents every text from the data as a matrix whose rows are the number of documents and columns are the number of distinct terms throughout all documents. Document frequency is the number of documents containing a specific term. Document frequency indicates how common the term is. Inverse document frequency (IDF) is the weight of a term, it aims to reduce the weight of a term if the term's occurrences are scattered throughout all the documents. IDF can be calculated as follow:

$$idf_i = \log(\frac{n}{df_i})$$

Where idf_i is the IDF score for term i, df_i is the number of documents containing term i, and n is the total number of documents. The higher the DF of a term, the lower the IDF for the term. When the number of DF is equal to n which means that the term appears in all documents, the IDF will be zero, since log(1) is zero, when in doubt just put this term in the stop words list because it doesn't provide much information. The TF-IDF score as the name suggests is just a multiplication of the term frequency matrix with its IDF, it can be calculated as follow:

$$w_{i,j} = tf_{i,j} \times idf_i$$

Where w_{ij} is TF-IDF score for term i in document j, tf_{ij} is term frequency for term i in document j, and idf_i is IDF score for term i.

Chapter 5
Classification Analysis

5.1 Supervised Machine Learning Models:

In Supervised Learning the process where the algorithm is learning from the training data can be seen as a teacher supervising the learning process of its students .The supervisor is somehow teaching the algorithm what conclusions it should come up with as an output. So, both input and the desired output data are provided. It is also required that the training data is already labeled. If the classifier gets more labeled data, the output will be more precise. The goal of this approach is that the algorithm can correctly predict the output for new input data. If the output were widely different from the expected result, the supervisor can guide the algorithm back to the right path. There are however some challenges involved when working with supervised. The supervised learning works fine as long as the labelled data is provided. This means that if the machine faces unseen data, it will either give wrong class label after classification or remove it because it has not ”learnt” how to label it.

In Our Dataset we used 4 Type of Classification Models:

1. Logistic Regression
2. Multinomial Naïve Bayes
3. Decision Tree Classifier
4. K Nearest Neighbor Classifier

5.2 Objectives:

1. Reading preprocessed data
2. Checking & filling the null values if there is present any
3. Defining Target & Non-Targeted columns & splitting our data into 70:30 ratio
4. Vectorize our text data to convert it in numerical form.
5. Model building
6. Model Evaluation
7. Model Comparison
8. Hyperparameter Tuning
9. Predicting the sentiments of some random data
10. Conclusion

5.2.1 Reading Preprocessed Data:

Here, In our dataset first we have to read our preprocessed data, then we define our Target Variable & Non-Targeted Variables & after defining it we have to proceed for train-test split to prepare our data to train the model. So, we split our data into 70:30 ratio here

```
## Reading the preprocessed data
####################################
pantry_bkp=pd.read_csv('clean_data_finall.zip',compression='zip')
#pantry_bkp.drop('Unnamed: 0',axis=1,inplace=True)
pantry_bkp.head()
```

	rating	verified	Date	reviewerID	asin	reviewerName	unixReviewTime	description	title	also_buy	brand	feature	rank	also_view	price	review_text	clean_text	Polarity	Analysis
0	4	True	2015-09-24	A31Y9ELLA1JUB0	B0000DIWNI	Her Royal Peepness Princess HoneyBunny Blayze	1443052800	['Saran Premium Wrap is an extra tough yet eas...	Saran Premium Plastic Wrap, 100 Sq Ft	['B01MY5FHT6', 'B000PYF8VM', 'B000SRMDFA', 'B0...	Saran	[]	[]	['B077QLSLRQ', 'B00JPKW1RQ', 'B000FE2IK6', 'B0...	1.48	Pretty Good For plastic Wrap I purchased this ...	pretty good plastic wrap purchase saran premiu...	0.187037	Neutral
1	5	True	2015-06-23	A2FYW9VZ0AMXKY	B0000DIWNI	Mary	1435017600	['Saran Premium Wrap is an extra tough yet eas...	Saran Premium Plastic Wrap, 100 Sq Ft	['B01MY5FHT6', 'B000PYF8VM', 'B000SRMDFA', 'B0...	Saran	[]	[]	['B077QLSLRQ', 'B00JPKW1RQ', 'B000FE2IK6', 'B0...	1.48	The Best Plastic Wrap for your Cooking, Baking...	best plastic wrap cook bake food storage need ...	0.317857	Neutral
2	5	True	2015-06-13	A1NE43T0OM6NNX	B0000DIWNI	Tulay C	1434153600	['Saran Premium Wrap is an extra tough yet eas...	Saran Premium Plastic Wrap, 100 Sq Ft	['B01MY5FHT6', 'B000PYF8VM', 'B000SRMDFA', 'B0...	Saran	[]	[]	['B077QLSLRQ', 'B00JPKW1RQ', 'B000FE2IK6', 'B0...	1.48	Good and strong. Good wrap, keeping it in the ...	good strong good wrap keep fridge make easier ...	0.611111	Positive
3	4	True	2015-06-03	AHTCPGK2CNPKU	B0000DIWNI	OmaShops	1433289600	['Saran Premium Wrap is an extra tough yet eas...	Saran Premium Plastic Wrap, 100 Sq Ft	['B01MY5FHT6', 'B000PYF8VM', 'B000SRMDFA', 'B0...	Saran	[]	[]	['B077QLSLRQ', 'B00JPKW1RQ', 'B000FE2IK6', 'B0...	1.48	Doesn't cling as well to dishes as other brand...	doesn t cling dish brand tangle prefer saran w...	0.000000	Negative
4	5	True	2015-04-20	A25SIBTMVXLB59	B0000DIWNI	Nitemanslim	1429488000	['Saran Premium Wrap is an extra tough yet eas...	Saran Premium Plastic Wrap, 100 Sq Ft	['B01MY5FHT6', 'B000PYF8VM', 'B000SRMDFA', 'B0...	Saran	[]	[]	['B077QLSLRQ', 'B00JPKW1RQ', 'B000FE2IK6', 'B0...	1.48	Five Stars Thanks	star thank	0.000000	Negative

5.2.2 Checking & filling the null values if there is present any:

```
# Checking the null values
pantry_bkp.isnull().sum()
```

```
rating          0
verified        0
Date            0
reviewerID      0
asin            0
reviewerName    0
unixReviewTime  0
description     0
title           0
also_buy        0
brand           0
feature         0
rank            0
also_view       0
price           0
review_text     0
clean_text      0
Polarity        0
Analysis        0
dtype: int64
```

This is how we check for the null value first, there is no null value present in our dataset, so we can proceed further, then we define our Target & non-targeted variables accordingly, then we split our data into train-test-split in 70:30 ratio.

5.2.3 Defining Target & Non-Targeted columns & splitting our data into 70:30 ratio:

```
x=pantry_bkp['clean_text']
y=pantry_bkp['Analysis']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
```

5.2.4 Word Embedding Techniques:

Here we can see that our non-targeted variable (clean_text) is in text format so 1st we have to convert our text data into numeric data, as machine learning models can only deal with numerical data. So to convert our text data into numerical data we've used 3 vectorization technique here.

1. Bag Of Word
2. CountVectorization
3. Tf-Idf Vectorization

5.2.5 Model Building:

So, In our Dataset we convert our text data into numeric data by all these vectorization technique in each models, and after that we build and evaluate our model accordingly.
And here we've used one-vs-rest classifier strategy to deal with the target column (Analysis) as it contains multiple class (positive, negative, neutral).

- **Logistic Regression Model:**

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

Sigmoid Function:

$$p = \frac{1}{1 + e^{-y}}$$

Apply Sigmoid function on linear regression:

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots.. + \beta_n X_n)}}$$

- **By Bag Of Word Vectorization:**

Function to build Logistic Regression Model by Bag Of Word Method:

```
# Self defining function to convert the data into vector form by bow model and classify and create model Logistic Regression

def model_ovr_lr_bow(x_train, x_test, y_train, y_test):
    global acc_lr_bow,f1_lr_bow
    # Text to vector transformation
    vector = CountVectorizer(min_df=1, ngram_range=(1,1))
    x_train = vector.fit_transform(x_train)
    x_test = vector.transform(x_test)

    #Classifing data of the four classes using a one vs. rest strategy with Logistic Regression

    lr = LogisticRegression()
    ovr = OneVsRestClassifier(lr)

    #fitting training data into the model & predicting
    t0 = time()

    ovr.fit(x_train, y_train)

    y_pred = ovr.predict(x_test)

    # Model Evaluation

    conf=confusion_matrix(y_test,y_pred)
    acc_lr_bow=accuracy_score(y_test,y_pred)
    f1_lr_bow=f1_score(y_test,y_pred,average='weighted')

    print('Time : ',time()-t0)
    print('Accuracy: ',acc_lr_bow)
    print(10*'==========')
    print('Confusion Matrix: \n',conf)
    print(10*'==========')
    print('Classification Report: \n',classification_report(y_test,y_pred))


    return y_test,y_pred,acc_lr_bow
```

- **By Count Vectorization Method :**

Function to build Logistic Regression Model by Count Vectorization Method:

```
# Self defining function to convert the data into vector form by countvectorizer and classify and create model by Logistic Regre

def model_ovr_lr_cv(x_train, x_test, y_train, y_test):
    global acc_lr_cv,f1_lr_cv
    # Text to vector transformation
    vector = CountVectorizer(ngram_range=(1,3))
    x_train = vector.fit_transform(x_train)
    x_test = vector.transform(x_test)

    #Classifing data of the four classes using a one vs. rest strategy with Logistic Regression

    lr = LogisticRegression()
    ovr = OneVsRestClassifier(lr)

    #fitting training data into the model & predicting
    t0 = time()

    ovr.fit(x_train, y_train)

    y_pred = ovr.predict(x_test)

    # Model Evaluation

    conf=confusion_matrix(y_test,y_pred)
    acc_lr_cv=accuracy_score(y_test,y_pred)
    f1_lr_cv=f1_score(y_test,y_pred,average='weighted')
    print('Time: ',time()-t0)
    print('Accuracy: ',acc_lr_cv)
    print(10*'==========')
    print('Confusion Matrix: \n',conf)
    print(10*'==========')
    print('Classification Report: \n',classification_report(y_test,y_pred))


    return y_test,y_pred,acc_lr_cv
```

- **By TF-IDF Vectorization Method :**

Function to build Logistic Regression Model by TF-IDF Vectorization Method:

```
# Self defining function to convert the data into vector form by tf idf vectorizer and classify and create model by Logistic regression

def model_ovr_lr_tf(x_train, x_test, y_train, y_test):
    global acc_lr_tf,f1_lr_tf
    # Text to vector transformation
    vector = TfidfVectorizer()
    x_train = vector.fit_transform(x_train)
    x_test = vector.transform(x_test)

    #Classifing data of the four classes using a one vs. rest strategy with Logistic Regression

    lr = LogisticRegression()
    ovr = OneVsRestClassifier(lr)

    #fitting training data into the model & predicting
    t0 = time()

    ovr.fit(x_train, y_train)

    y_pred = ovr.predict(x_test)

    # Model Evaluation

    conf=confusion_matrix(y_test,y_pred)
    acc_lr_tf=accuracy_score(y_test,y_pred)
    f1_lr_tf=f1_score(y_test,y_pred,average='weighted')
    print('Time :',time()-t0)
    print('Accuracy: ',acc_lr_tf)
    print(10*'==========')
    print('Confusion Matrix: \n',conf)
    print(10*'==========')
    print('Classification Report: \n',classification_report(y_test,y_pred))


    return y_test,y_pred,acc_lr_tf
```

- **Multinomial Naïve Bayes:**

Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.
Naive Bayes classifier is a collection of many algorithms where all the algorithms share one common principle, and that is each feature being classified is not related to any other feature. The presence or absence of a feature does not affect the presence or absence of the other feature.

- **By Bag Of Word Model:**

We have use the same self defined function that we used in Logistic Regression Model ,here we have change the model name to build our Multinomial Naïve Bayes Model by Bag Of Word Method.

- **By Count Vectorization Method:**

We have use the same self defined function that we used in Logistic Regression Model ,here we have change the model name to build our Multinomial Naïve Bayes Model by Count Vectorization Method.

- **By TF-IDF Vectorization Method:**

We have use the same self defined function that we used in Logistic Regression Model ,here we have change the model name to build our Multinomial Naïve Bayes Model by Bag Of Word Method.

- **Decision Tree Classifier:**

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.

The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

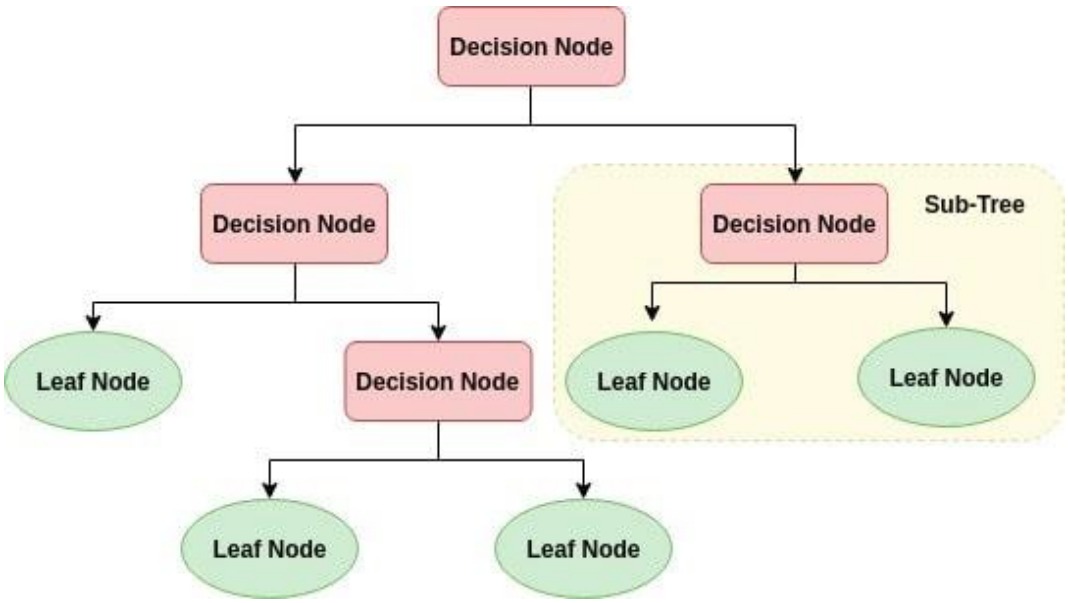


Fig 5.1: Decision Tree

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm.

The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

- **By Bag Of Word Model:**

We have use the same self defined function that we used in Logistic Regression Model ,here we have change the model name to build our Decision Tree Classifier Model by Bag Of Word Method.

- **By Count Vectorization Method:**

We have use the same self defined function that we used in Logistic Regression Model ,here we have change the model name to build our Decision Tree Classifier Model by Count Vectorization Method.

- **By TF-IDF Vectorization Method:**

We have use the same self defined function that we used in Logistic Regression Model ,here we have change the model name to build our Decision Tree Classifier Model by Bag Of Word Method.

- ## K Nearest Neighbour Classifier:

KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions. Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.

Despite its simplicity, nearest neighbors has been successful in a large number of classification and regression problems, including handwritten digits and satellite image scenes. Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular.

- **By Bag Of Word Model:**

We have use the same self defined function that we used in Logistic Regression Model ,here we have change the model name to build our KNN Classifier Model by Bag Of Word Method.

- **By Count Vectorization Method:**

We have use the same self defined function that we used in Logistic Regression Model ,here we have change the model name to build our KNN Classifier Model by Count Vectorization Method.

- **By TF-IDF Vectorization Method:**

We have use the same self defined function that we used in Logistic Regression Model ,here we have change the model name to build our KNN Classifier Model by Bag Of Word Method.

5.2.6 Model Evaluation:

To evaluate our Classification models we used 3 parameters

Accuracy Score - To measure the accuracy of the model, it will work well if our data is normally distributed

F1 Score – It is the harmonic mean of precision score & recall score, it will work well on the imbalanced data

Classification Report – It will summarize our whole model.

On the basis of Accuracy score & F1Score we've made a comparison chart ,given below:

```
# Creating a Dataframe to Summarize the Model Evaluation
############################################################
tbl=pd.DataFrame()
tbl['Model']=pd.Series(['Logistic Regression- BOW','Logistic Regression- CV','Logistic Regression- TFIDF',
                        'Multinomial NB- BOW','Multinomial NB- CV','Multinomial NB- TFIDF',
                        'Decision Tree- BOW','Decision Tree- CV','Decision Tree- TFIDF',
                        'KNN- BOW','KNN- CV','KNN- TFIDF'])
tbl['Accuracy']=pd.Series([acc_lr_bow,acc_lr_cv,acc_lr_tf,
                           acc_nb_bow,acc_nb_cv,acc_nb_tf,
                           acc_dt_bow,acc_dt_cv,acc_dt_tf,
                           acc_knn_bow,acc_knn_cv,acc_knn_tf])
tbl['F1']=pd.Series([f1_lr_bow,f1_lr_cv,f1_lr_tf,
                     f1_nb_bow,f1_nb_cv,f1_nb_tf,
                     f1_dt_bow,f1_dt_cv,f1_dt_tf,
                     f1_knn_bow,f1_knn_cv,f1_knn_tf])
tbl.set_index('Model')
```

Model	Accuracy	F1
Logistic Regression- BOW	0.938730	0.938424
Logistic Regression- CV	0.929226	0.928783
Logistic Regression- TFIDF	0.921322	0.921278
Multinomial NB- BOW	0.751382	0.736585
Multinomial NB- CV	0.794637	0.789675
Multinomial NB- TFIDF	0.765251	0.746678
Decision Tree- BOW	0.869120	0.866888
Decision Tree- CV	0.873097	0.871250
Decision Tree- TFIDF	0.865677	0.863494
KNN- BOW	0.720274	0.678415
KNN- CV	0.677844	0.618092
KNN- TFIDF	0.661914	0.641593

According to Accuracy & F1 score we can observe that in Logistic Regression by Bag Of Word method we are getting highest Accuracy & F1 score ,but as we know that Bag of Word method only count the occuracnce of each word present in the paragraph, it doesn't consider the order of the words. Because of that our Meaning/ Context of the sentence may get change. So bag of word method wasn't reliable to predict the sentiments.

So , Next we will go for the model building part again by only TF-IDF vectorization method with cross validation.

The result We are getting, given below:

```
 OneVsRestClassifier(estimator=LogisticRegression())
f1-Score(train): mean= (0.918), min=(0.914)) ,max= (0.922), stdev= (0.003)
f1-Score(test): 0.9176

 MultinomialNB()
f1-Score(train): mean= (0.750), min=(0.742)) ,max= (0.753), stdev= (0.004)
f1-Score(test): 0.7495

 DecisionTreeClassifier()
f1-Score(train): mean= (0.880), min=(0.870)) ,max= (0.885), stdev= (0.004)
f1-Score(test): 0.8807

 KNeighborsClassifier()
f1-Score(train): mean= (0.610), min=(0.591)) ,max= (0.648), stdev= (0.019)
f1-Score(test): 0.6106
```

Here, we are taking f1 score in cross validation technique to evaluate the model. Because if the data is imbalance then F1 score is reliable to evaluate the model.

The F1 score is calculated as the harmonic mean of the precision and recall scores, as shown below. It ranges from 0-100%, and a higher F1 score denotes a better quality classifier.

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

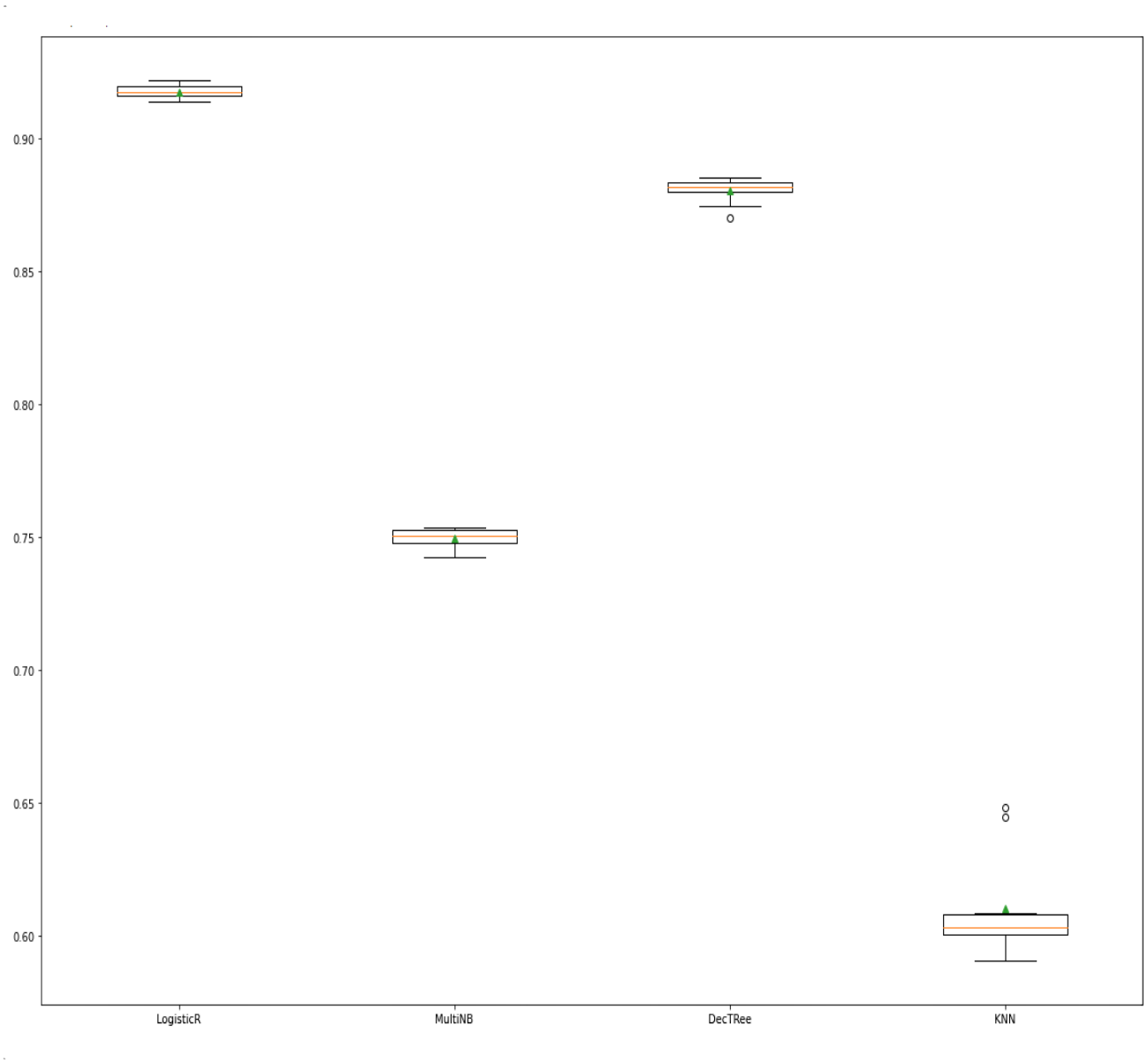


Fig 5.2: Comparison between the Classification Models

This is the visualization of F1 score distribution of each model. From the visuals we can say that the Logistic Regression model gets the higher F1 score (91.8%) among all the models.

According to our observation Multinomial NB model takes minimal time to evaluate. Decision tree taking much more time to evaluate. Knn model also taking more time to evaluate.

So, now we go for the Hyperparameter tuning of Logistic Regression model by TF-IDF vectorization method to improve the accuracy & reducing errors.

5.6.7 Hyperparameter Tuning:

As we got better accuracy & f1 score from Logistic Regression by TFIDF vectorization technique. So now we go for the hyperparameter tuning of this model to get better accuracy & precision for better model.

Here we are applying Grid SearchCV method to get the best hyperparameter for our model.

```
[ ] lr = LogisticRegression()
    # Penalty Type
    penalty = ['l1', 'l2'] #l2 penalty- Ridge model, l1 penalty- Lasso model, It will used to reduce the error & increasing the accuracy of model

    # use logarithimically spaced c values
    c= np.logspace(0, 4, 10) #Inverse of regularization(l1,l2) strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.
    # Applying Grid Search CV
    grid_model_lr_tf = GridSearchCV(lr,
                            param_grid = {'C':c, 'penalty': penalty})
```

```
    vector = TfidfVectorizer()
    x_train = vector.fit_transform(x_train)
    x_test = vector.transform(x_test)
    #Classifing data of the four classes using a one vs. rest strategy with Logistic Regression


    #fitting training data into the model & predicting

    grid_model_lr_tf.fit(x_train, y_train)

    #y_pred = grid_model_lr_bow.predict(x_test)
```

```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': array([1.00000000e+00, 2.78255940e+00, 7.74263683e+00, 2.15443469e+01,
       5.99484250e+01, 1.66810054e+02, 4.64158883e+02, 1.29154967e+03,
       3.59381366e+03, 1.00000000e+04]),
                         'penalty': ['l1', 'l2']})
```

```
[ ] # Getting the best parameters by Grid Search CV
    ############################################################
    grid_model_lr_tf.best_params_

    {'C': 59.94842503189409, 'penalty': 'l2'}
```

Here we are taking the penalty L1 & L2.

L2 penalty function uses the sum of the squares of the parameters and Ridge Regression encourages this sum to be small. L1 penalty function uses the sum of the absolute values of the parameters and Lasso encourages this sum to be small.

And we are taking logarithmically spaced C values.

Comparison of the sparsity (percentage of zero coefficients) of solutions when L1, L2 penalty are used for different values of C. Large values of C give more freedom to the model. Conversely, smaller values of C constrain the model more. In the L1 penalty case, this leads to sparser solutions.

After getting the best value of hyperparameters we are going for the model building and getting the better accuracy than previous one.

```
Train Accuracy :  0.9823974396275822
Test Accuracy  :  0.9512656386383473
f1-Score Test  :  0.9512871008844906
Classification Report :
               precision    recall  f1-score   support

    Negative       0.96      0.95      0.95      8067
     Neutral       0.93      0.93      0.93     13737
    Positive       0.97      0.97      0.97     19440

    accuracy                           0.95     41244
   macro avg       0.95      0.95      0.95     41244
weighted avg       0.95      0.95      0.95     41244

================================================================================================================================================
Confusion Matrix :
 [[ 7636   420    11]
 [  310 12790   637]
 [   22   610 18808]]
```

▾ Model Comparison (Before Hyperparameter Tuning & After Hyperparameter Tuning

```
tbl1=pd.DataFrame()
tbl1['Model']=pd.Series(['Logistic Regression With TFIDF( Before HP Tuning)','Logistic Regression With TFIDF( After HP Tuning)'])
tbl1['Accuracy']=pd.Series([acc_lr_tf,acc_tst])
tbl1['F1 Score']=pd.Series([f1_lr_tf,f1])
tbl1.set_index('Model')
```

Model	Accuracy	F1 Score
Logistic Regression With TFIDF(Before HP Tuning)	0.921322	0.921278
Logistic Regression With TFIDF(After HP Tuning)	0.951266	0.951287

From the above figure we can say that our model isn't overfitted ,as we are getting approximately similar training accuracy & testing accuracy.

Also after hyperparameter tuning we getting better accuracy & f1 score .

Also our accuracy & f1 score is coming similar so we can conclude that our data is'nt imbalanced, it was normally distributed.

5.2.8 Predicting the sentiments of some random data:

As we get the best model after hyper parameter tuning. Now we can go for the prediction of sentiments on some random data to test the performance & accuracy of our model.

▾ Predicting the random data on best model.

```
## Prediction on random data
#####################################
data=['Taste not to be believed. Buy a box for my office every week',
      "These are delicious and healthy snacks!  I with they were more affordable because they're really tasty and convenient.  I purchased these because they're lower in sugar than many other brands and really enjoy them.",
      "I like most of the flavors but this one is my favorite so far!!",
      "Excellent. Only complaint is they stick to the wrapper and are hard to remove. But taste and nutrition are great."]
data=vector.transform(data)
lr_tf_model.predict(data)
```

```
array(['Negative', 'Positive', 'Neutral', 'Neutral'], dtype='<U8')
```

Here, we go for the prediction on some random data . And according to the output we can say that our model is correctly predicting.

5.3 Conclusion:

According to our model evaluation we can conclude that Logistic Regression by TF-IDF vectorization method is the best one.

KNN Classifier with TFIDF Vectorization is the poor one. Because KNN algorithm works perfectly only on small data.

After hyperparameter tuning our Logistic Regression model will predict correctly 95 times out of 100 times. So we can use this model to predict further on this Amazon Prime Pantry Data.

Chapter 6

Clustering

Clustering or cluster analysis is a machine learning technique, which groups the unlabeled dataset. It can be defined as *"A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group."*

It does it by finding some similar patterns in the unlabelled dataset such as shape, size, color, behavior, etc., and divides them as per the presence and absence of those similar patterns.

It is an unsupervised learning method, hence no supervision is provided to the algorithm, and it deals with the unlabeled dataset.

After applying this clustering technique, each cluster or group is provided with a cluster-ID. ML system can use this id to simplify the processing of large and complex datasets

6.1 Clustering Algorithms

1. **K-Means algorithm:** The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of **O(n).**
2. **DBSCAN Algorithm:** It stands **for Density-Based Spatial Clustering of Applications with Noise**. It is an example of a density-based model similar to the mean-shift, but with some remarkable advantages. In this algorithm, the areas of high density are separated by the areas of low density. Because of this, the clusters can be found in any arbitrary shape.
3. **Agglomerative Hierarchical algorithm:** The Agglomerative hierarchical algorithm performs the bottom-up hierarchical clustering. In this, each data point is treated as a single cluster at the outset and then successively merged. The cluster hierarchy can be represented as a tree-structure.

6.1.1 K-Means Clustering Algorithm

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters

How to choose the value of "K number of clusters" in K-means Clustering?

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K. The method is given below:

6.1.2 Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.

- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method.

6.2 Python Implementation of K-means Clustering Algorithm

In the above section, we have discussed the K-means algorithm, now let's see how it can be implemented using Python.

Before implementation, let's understand what type of problem we will solve here. So, we have a dataset of **Prime Pantry**, which is the data of customers who gives reviews to the different brands present in the pantry

In the given dataset, we have **Customer_reviews, Price, Rating, Product_id, Brand, etc**. From this dataset, we need to calculate some patterns, as it is an unsupervised method, so we don't know what to calculate exactly.

The steps to be followed for the implementation are given below:

- **Data Pre-processing**
- **Finding the optimal number of clusters using the elbow method**
- **Training the K-means algorithm on the training dataset**
- **Visualizing the clusters**

Step-1: Data pre-processing Step

The first step will be the data pre-processing, as we did in our earlier topics Classification. But for the clustering problem, it will be different from other models.

- **Importing-Libraries**
  As we did in previous topics, firstly, we will import the libraries for our model, which is part of data pre-processing. The code is given below:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import seaborn as sns
sns.set()

from sklearn import preprocessing,metrics
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
import warnings
warnings.simplefilter(action='ignore')
```

- **Importing-the-Dataset:**

  Next, we will import the dataset that we need to use. It can be imported using the below code:

```
pantry_bkp=pd.read_csv('/clean_data_finall.zip',compression='zip')

pantry_bkp.head()
```

- **Extracting Independent Variables**

Here we don't need any dependent variable for data pre-processing step as it is a clustering problem, and we have no idea about what to determine.

```
X1=pantry_bkp.loc[:,['rating','price','Polarity']].values
```

Step-2: Finding the optimal number of clusters using the elbow method

In the second step, we will try to find the optimal number of clusters for our clustering problem. So, as discussed above, here we are going to use the elbow method for this purpose.

As we know, the elbow method uses the WCSS concept to draw the plot by plotting WCSS values on the Y-axis and the number of clusters on the X-axis. So we are going to calculate the value for WCSS for different k values ranging from 1 to 10. Below is the code for it:

```
[ ]  distortion=[]

     for k in range(2,10):
         kmeans=KMeans(n_clusters=k,random_state=22)
         kmeans.fit(X1)

         distortion.append(kmeans.inertia_)

     fig=plt.figure(figsize=(15,5))
     plt.plot(range(2,10),distortion,marker='o',markerfacecolor='red',markersize=10)
     plt.grid(True)
     plt.xlabel("K-Values")
     plt.ylabel("WCSS")
     plt.show()
```

As we can see in the above code, we have used **the KMeans** class of sklearn. cluster library to form the clusters.

Next, we have created the **wcss_list** variable to initialize an empty list, which is used to contain the value of wcss computed for different values of k ranging from 2 to 10.

After that, we have initialized the for loop for the iteration on a different value of k ranging from 2 to 10; since for loop in Python, exclude the outbound limit, so it is taken as 10 to include 9th value.

The rest part of the code is similar as we did in earlier topics, as we have fitted the model on a matrix of features and then plotted the graph between the number of clusters and WCSS.

Output: After executing the above code, we will get the below output:

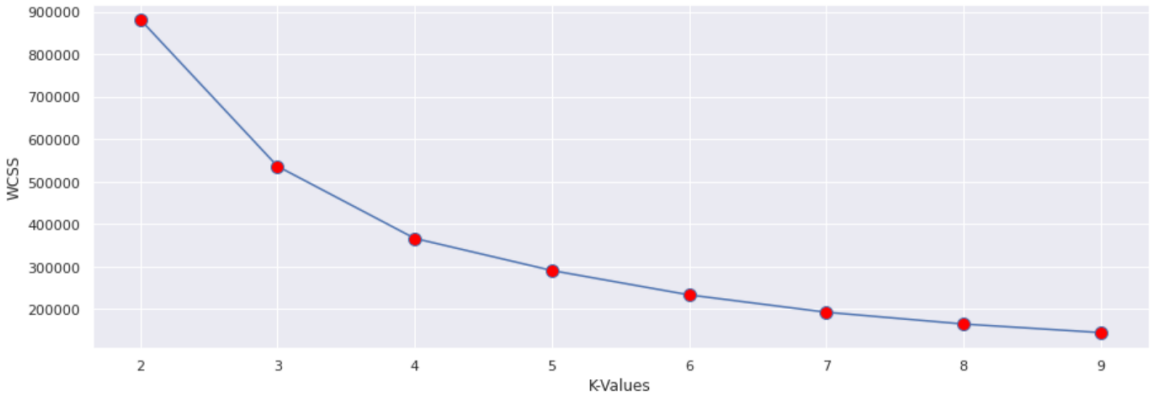


Fig 6.1: Elbow Curve

From the above plot, we can see the elbow point is at **3. So the number of clusters here will be 3.**

Step- 3: Training the K-means algorithm on the training dataset

As we have got the number of clusters, so we can now train the model on the dataset.

To train the model, we will use the same two lines of code as we have used in the above section, but here instead of using i, we will use 3, as we know there are 3 clusters that need to be formed. The code is given below:

```
[ ]  kmeans_3=KMeans(n_clusters=3,random_state=22)
     kmeans_3.fit(X1)
```

```
KMeans(n_clusters=3, random_state=22)
```

```
[ ]  cluster_labels=kmeans_3.fit_predict(X1)
     pantry_bkp['K_cluster']=cluster_labels
     pantry_bkp.head()
```

Step-4: Visualizing the Clusters

The last step is to visualize the clusters. As we have 3 clusters for our model, so we will visualize each cluster one by one.

To visualize the clusters will use scatter plot using plt.scatter() function of matplotlib.pyplot.

```
[ ]  plt.scatter(X1[cluster_labels==0,2] , X1[cluster_labels==0,1], c='Blue',label='Negative' )
     plt.scatter(X1[cluster_labels==1,2] , X1[cluster_labels==1,1], c='red' ,label='Neutral')
     plt.scatter(X1[cluster_labels==2,2] , X1[cluster_labels==2,1], c='green',label='Positive')


     plt.xlabel('Polarity' )
     plt.ylabel('Price')
     plt.legend(loc=(1,.76))
     plt.show()
```

```
# 3d scatterplot using matplotlib for rating, price and polarity

fig = plt.figure(figsize = (10,8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X1[cluster_labels == 0,0],X1[cluster_labels == 0,1],X1[cluster_labels == 0,2], s = 40 , color = 'blue', label = "Negative")
ax.scatter(X1[cluster_labels == 1,0],X1[cluster_labels == 1,1],X1[cluster_labels == 1,2], s = 40 , color = 'red', label = "Neutral")
ax.scatter(X1[cluster_labels == 2,0],X1[cluster_labels == 2,1],X1[cluster_labels == 2,2], s = 40 , color = 'green', label = "Positive")

ax.set_xlabel('Rating of a customer-->')
ax.set_ylabel('Price-->')
ax.set_zlabel('Polarity-->')
ax.legend()
plt.show()
```

In above lines of code, we have written code for each clusters, ranging from 1 to 3.

Output:

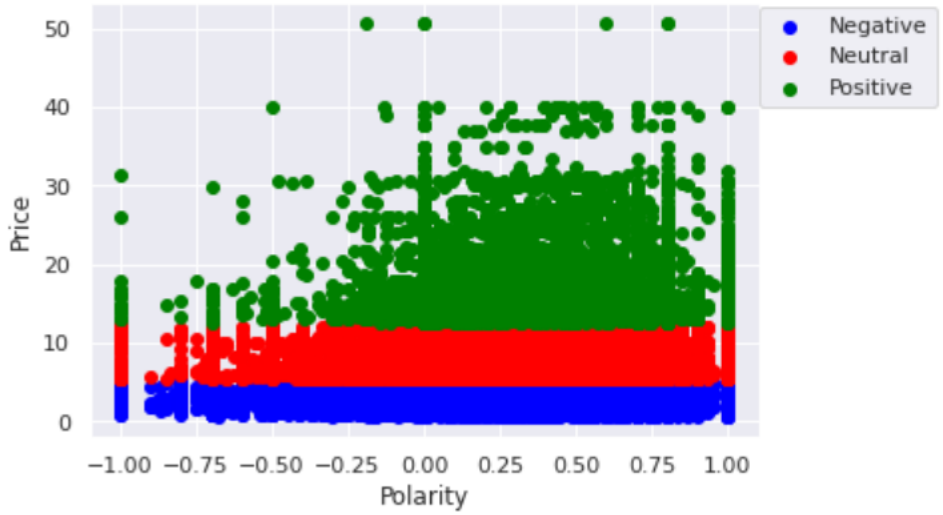


Fig 6.2: Visualization of Clusters in 2D

The output image is clearly showing the three different clusters with different colors.

The Positive reviews are more than that of Neutral and Negative.

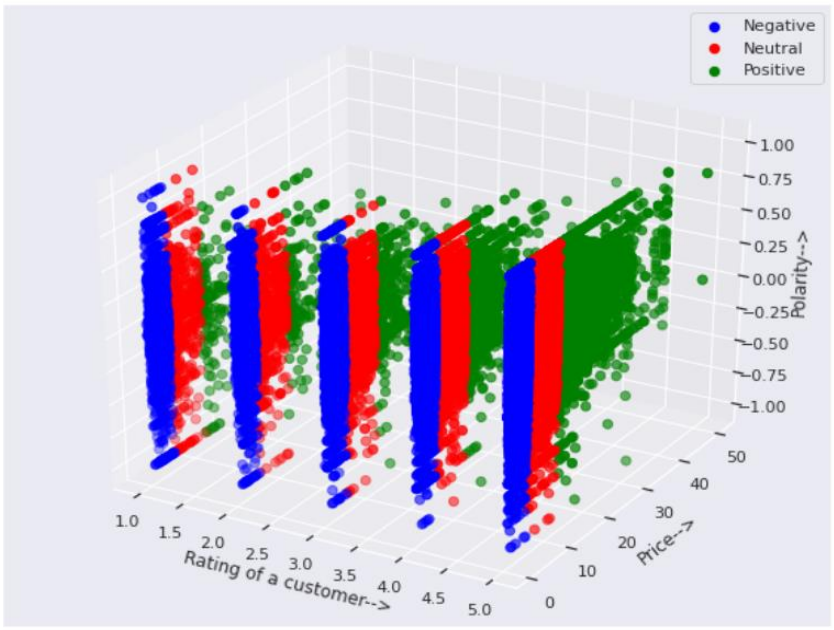


Fig 6.3: Visualization of Clusters in 3D

- From the above graph we can say that as the rating and the price increases the positive reviews also increases. For Low rating the reviews are very less.

6.3 Principal Component Analysis

Principal Component Analysis is basically a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. Each of the principal components is chosen in such a way so that it would describe most of them still available variance and all these principal components are orthogonal to each other. In all principal components first principal component has a maximum variance.

Objectives of PCA:

- It is basically a non-dependent procedure in which it reduces attribute space from a large number of variables to a smaller number of factors.
- PCA is basically a dimension reduction process but there is no guarantee that the dimension is interpretable.
- The main task in this PCA is to select a subset of variables from a larger set, based on which original variables have the highest correlation with the principal amount.

6.4 Python Implementation for PCA on Customer review

First we have to vectorize the review i.e clean_text column.

Vectorization of clean_text

```
[ ]

    tfidf =TfidfVectorizer()

    text=tfidf.fit_transform(pantry_bkp['clean_text'])

    # pd.DataFrame(text.toarray(),columns=tfidf.get_feature_names()).head()
```

Importing the required library for PCA:

```
[ ]  from sklearn.decomposition import PCA
```

Defined a function to fit the model predict the outcomes and to plot the clusters:

```
[ ]  def plot_pca(data, labels):
        max_label = max(labels)
        max_items = np.random.choice(range(data.shape[0]), size=10000, replace=False)

        pca = PCA(n_components=2).fit_transform(data[max_items,:].todense())



        idx = np.random.choice(range(pca.shape[0]), size=3000, replace=False)
        label_subset = labels[max_items]
        label_subset = [cm.hsv(i/max_label) for i in label_subset[idx]]

        fig= plt.figure( figsize=(14, 6))

        plt.scatter(pca[idx, 0], pca[idx, 1], c=label_subset)
        plt.title('PCA Cluster Plot')
        plt.legend(labels=['Positive','Negative','Neutral'])
        plt.tight_layout()


    plot_pca(text, cluster_labels)
```

Output:

Fig 6.4: PCA Clusters

Conclusion:

- According to our analysis as the rating is increasing the positive cluster density is increasing. And as the rating is decreasing the negative cluster density is increasing.

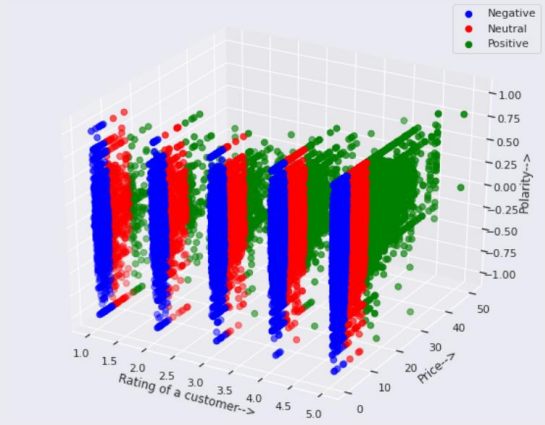


Fig 6.5: 3D visualization of clusters

- From the text blob the sentiments we are getting is approximately equal to the sentiments which we are getting from clustering.

```
pantry_bkp['Analysis'].value_counts()
```

```
Positive    88791
Neutral     41663
Negative     7026
Name: Analysis, dtype: int64
```

```
# label '0'=Positive
# label '1'=Neutral
# label '2'=Negative
pantry_bkp['K_cluster'].value_counts()
```

```
0    96181
1    35024
2     6275
Name: K_cluster, dtype: int64
```

Chapter 7
Time Series Analysis

Time series analysis is a statistical technique that deals with time series data, or trend analysis. Time series data means that data is in a series of particular time periods or intervals. In Time series Analysis the order of data matters and is fixed. Another very important thing about it is features are not necessarily independent of each other.

It is a set of observations, each one recorded at a time interval. A discrete time series is a set of observations recorded in a fixed interval. This might be hourly, daily, weekly, monthly, quarterly or yearly. These time series are often one dimensional, just a time/date and a value however, there are multiple factors to take into account when analyzing the data.

7.1 Types of Time Series Data

1. **Univariate Time Series**
   A time series that has single time-dependent variable. For example, Time-Stock Closing Price**.**

2. **Multivariate Time Series**
   A time series that has more than one time-dependent variable. For example, Time- Temperature, Humidity, Cloud Cover, Wind Speed**.**

7.2 Time Series Components

Time Series data have four components:

1. **Secular (Trend):** The trend shows the general tendency of the data to increase or decrease during a long period of time. A trend is a smooth, general, long-term, average tendency. It is not always necessary that the increase or decrease is in the same direction throughout the given period of time. A trend could be :
   - **Uptrend:** Time Series Analysis shows a general pattern that is upward then it is Uptrend.
   - **Downtrend:** Time Series Analysis shows a pattern that is downward then it is Downtrend.
   - **Horizontal or Stationary trend:** If no pattern observed then it is called a Horizontal or stationary trend.

2. **Cyclic (Trend):** The variations in a time series which operate themselves over a span of more than one year are the cyclic variations. This oscillatory movement has a period of oscillation of more than a year. One complete period is a cycle. This cyclic movement is sometimes called the 'Business Cycle'.

It is a four-phase cycle comprising of the phases of prosperity, recession, depression, and recovery. The cyclic variation may be regular are not periodic. The upswings and the downswings in business depend upon the joint nature of the economic forces and the interaction between them.

3. **Seasonal Variations:** These are the rhythmic forces which operate in a regular and periodic manner over a span of less than a year. They have the same or almost the same pattern during a period of 12 months. This variation will be present in a time series if the data are recorded hourly, daily, weekly, quarterly, or monthly.

   These variations come into play either because of the natural forces or man-made conventions. The various seasons or climatic conditions play an important role in seasonal variations. Such as production of crops depends on seasons, the sale of umbrella and raincoats in the rainy season, and the sale of electric fans and A.C. shoots up in summer seasons.

   The effect of man-made conventions such as some festivals, customs, habits, fashions, and some occasions like marriage is easily noticeable. They recur themselves year after year. An upswing in a season should not be taken as an indicator of better business conditions.

4. **Random or Irregular Movement:** There is another factor which causes the variation in the variable under study. They are not regular variations and are purely random or irregular. These fluctuations are unforeseen, uncontrollable, unpredictable, and are erratic. These forces are earthquakes, wars, flood, famines, covid-19 and any other disasters.

7.3 Objective of TSA:

Through Time Series Analysis we are going to predict/forecast the future sentiment of the customers shopping on Amazon Prime Pantry. By the sentiment we got through the Natural Language Processing we going to show how the sentiment of unstructured text data changes over time.

Data Set Understanding

For TSA we need date column and sentiment analysis column from dataset. As all the sentiment are in one column. So we need to separate every sentiment in unique column we will create crosstab and prepared data quarter-wise for analysis. Now the data is ready for Time Series Analysis.

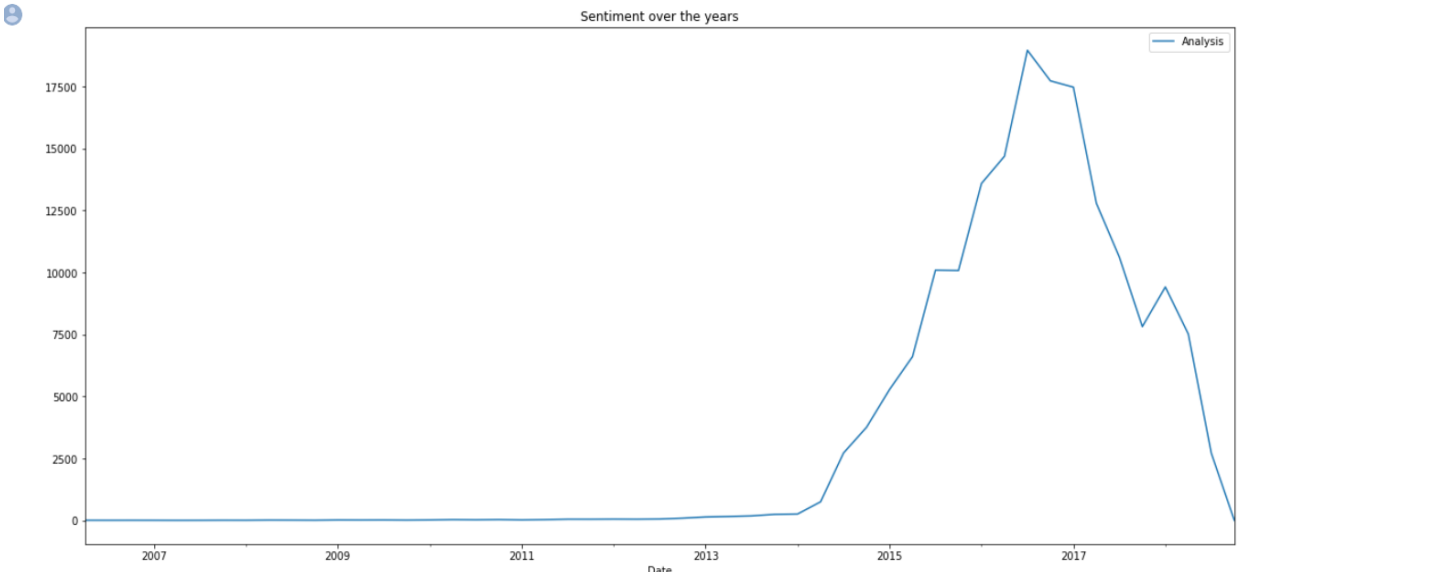
```
#Unstacking Analysis column into separate columns
#resampling data quaterly as the data is huge
pantry_dff=pd.crosstab(pantry_bkp['Date'],pantry_bkp['Analysis'])
pantry_dff=pantry_dff.resample('Q').sum()
pantry_dff['total']=pantry_dff['Negative']+pantry_dff['Neutral']+pantry_dff['Positive']

pantry_dff.head()
```

Analysis Date	Negative	Neutral	Positive	total
2006-06-30	0	3	0	3
2006-09-30	0	2	0	2
2006-12-31	0	3	0	3
2007-03-31	0	0	1	1
2007-06-30	0	0	0	0

As the data is present from 2006 onwards for analysis but the data-points values are constant or have no fluctuations upto year 2012. So in our analysis we are taking data from 2012 onwards

```
pantry_df.plot(figsize=(20,9),title='Sentiment over the years')
plt.show()
```




```
#Slicing the required data for analysis as before it all datapoints have constant or zero values.
Positive=pantry_dff['Positive'][27:]
Neutral=pantry_dff['Neutral'][27:]
Negative=pantry_dff['Negative'][27:]
```

```
#seeing plot of Analysis Column
for i in [Positive,Neutral,Negative]:
  i.plot(figsize=(20,9))
  plt.legend();
```

Fig 7.1: Sentiments over the years

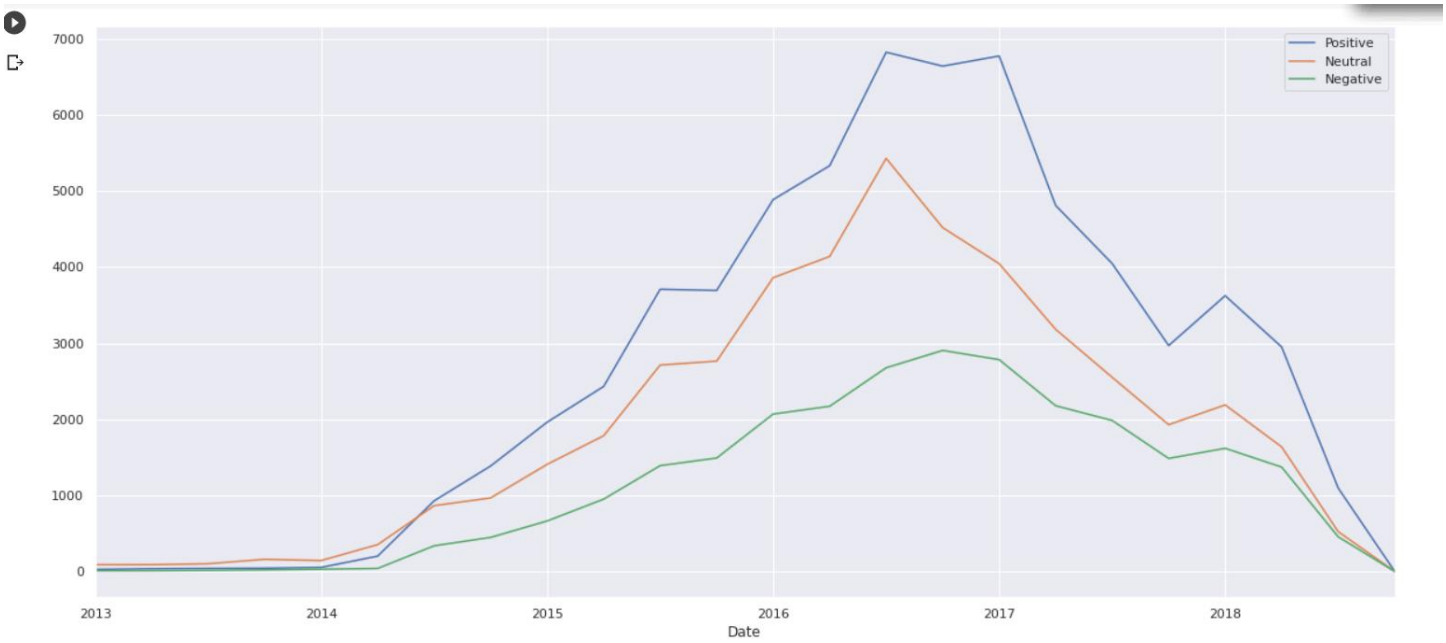


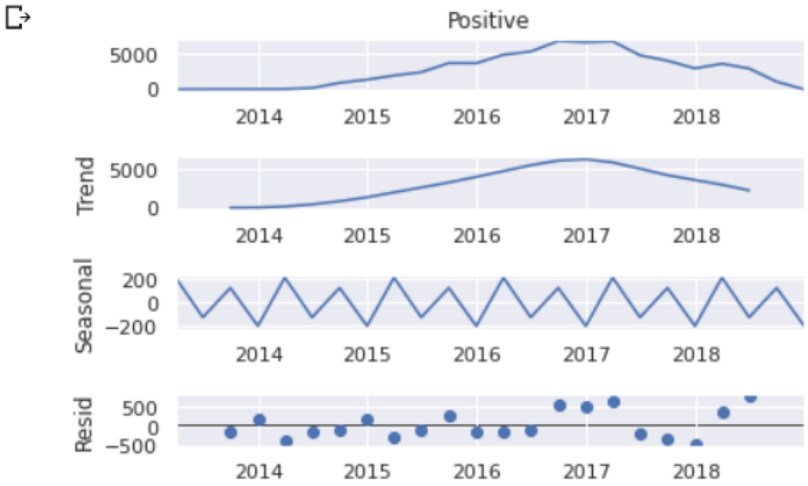
Fig 7.2: Sentiment Analysis in detail

7.4 Models taken for analysis:

In our analysis we have taken SARIMA model for analysis as there was seasonality present in our data. But to understand SARIMA we need to first know ARIMA.

Seasonal graph of positive sentiment showing seasonality repeating at start of every year

```
#seasonal decomposition for positive sentiment,period=4 as there are 4 months in each qaurter
decomposition=seasonal_decompose(Positive,period=4)
decomposition.plot();
```



ARIMA is a combination of multiple forecasting principles. The Autoregressive model (AR) is a representation of a random process. The autoregressive model specifies that the output depends on the previous inputs and a stochastic term (an imperfect predictable term) therefore making it a stochastic differential equation. Basically, autoregressive models take the previous steps into account when predicting and calculating the next step. The issue with the AR model is that temporary or single shocks affect the whole output indefinitely. To avoid this issue the AR process often has a lag value. The lag value is how many of the previous steps that should contribute more to the output than others. The AR model can be nonstationary as it can be represented by a unit root variable.

The MA (Moving Average) model in contrary to the AR model is always stationary. The moving average is a linear regression of the current value against the white noise or random shocks in the series, contrary to the AR model which is the linear regression to non-shock values.

The I in ARIMA is for integrated, which is the differencing of the previous observations in the series (subtracting an observation from a previous observation in the series to make the series stationary). Non-seasonal ARIMA is often described as ARIMA (p, d, q) where p is the number of lags in the AR model, d is the grade of differentiation (the number of time the data has previous values subtracted) and the q is the order of the MA model.

Seasonal Autoregressive Integrated Moving Average, SARIMA or Seasonal ARIMA, is an extension of ARIMA that explicitly supports univariate time series data with a seasonal component. All things remain same as ARIMA model but new seasonality elements are added. **SARIMA(p,d,q)(P,D,Q,m)**

There are four seasonal elements that are not part of ARIMA that must be configured; they are:

- **P**: Seasonal autoregressive order.
- **D**: Seasonal difference order.
- **Q**: Seasonal moving average order.
- **m**: The number of time steps for a single seasonal period.

Model selection is done in above part. Now before applying dataset to model we have to check whether the data is stationary or not. For this we need to check through Augmented Dickey-Fuller test, which is discussed below.

7.4.1 Augmented Dickey-Fuller test for Stationarity:

The Augmented Dickey-Fuller includes a lag variable in order to remove the autocorrelation from the results. There are three different versions of the Dickey-Fuller test. There is the normal unit root test, unit root with drift and lastly, a version for a unit root with a drift and a deterministic trend. We will use the normal one, as it is safer to use, even if we should have a drift or deterministic trend. Wrongly inclusion of the drift and deterministic trend reduces the power of the root test. The formula for the Dickey-Fuller test for a normal unit root is shown in the formula below.

$$\Delta y_t = a + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \ldots + \delta_{p-1} \Delta y_{t-p+1} + \epsilon_t$$

The yt is the variable of interest, t is the time index and ut is an error term. The Δ is an operator for the first difference level. α is a constant, β the coefficient on a time trend and p the lag order of the autoregressive process. Imposing the constraints $\alpha = 0$ and $\beta = 0$ corresponds to modelling a random walk and using the constraint $\beta = 0$ corresponds to modeling a random walk with a drift. The unit root test is then carried out under the null hypothesis $\gamma = 0$ against the alternative hypothesis of $\gamma < 0$. We did not set these parameters ourselves, instead, we used an already implemented method. We will discuss this further in the implementation. Once a value for the test statistic is computed it can be compared to the relevant critical value for the Dickey-Fuller Test. If the value is below the critical value, we can then reject the null hypothesis and prove stationarity.

Example of stationarity check on positive sentiment data from Python workbook:

```
[ ]  # dickey-fuller test for stationarity

     def checkStationarity(data):
         pvalue = adfuller(data)[1]
         if pvalue < 0.05:
             ret = "Data is Stationary. Proceed to model building"
         else:
             ret = "Data is not Stationary. Make it stationary"

         return(ret)
```

```
[ ]  # checking for stationarity of the positive sentiment data
     checkStationarity(Positive)
```

```
'Data is Stationary. Proceed to model building'
```

If suppose the data is not stationary, than first we have to make the data stationary and then proceed for model building. For making the data stationary we have three methods:

1. **Differencing-** one way to make a non-stationary time series stationary — compute the differences between consecutive observations. This is known as **differencing**.
2. **Log Transformation-** Log transformation can be used to stabilize the variance of a series with non-constant variance. This is done using the log() function. One limitation of log transformation is that it can be applied only to positively valued time series. Taking a log shrinks the values towards 0.
3. **EDA technique-** In this method we adjust the outlier by performing simple EDA.

After these we go for acf and pacf correlogram plot for p, q, P and Q values.

7.4.2 ACF and PACF Plot:

Autocorrelation function (ACF) is a measurement of how related the actual value is to the previous values including trend and seasonality. The values are plotted along a confidence interval band (typically 0.05, 0.025 etc). It is the q/Q component of ARIMA/SARIMA model.

Partial Autocorrelation function (PACF), unlike the ACF, finds the correlation between the residual values in the series, therefore it is only the partial function. The values are plotted along a confidence interval band (typically 0.05, 0.025 etc). It is the p/P component of ARIMA/SARIMA model.

Showing ACF and PACF plot:

```
#plotting correlogram for positive sentiment data
fig,ax=plt.subplots(1,2,figsize=(15,7))
plot_acf(Positive,lags=10,ax=ax[0])
plot_pacf(Positive,lags=10,ax=ax[1])
plt.show()
```

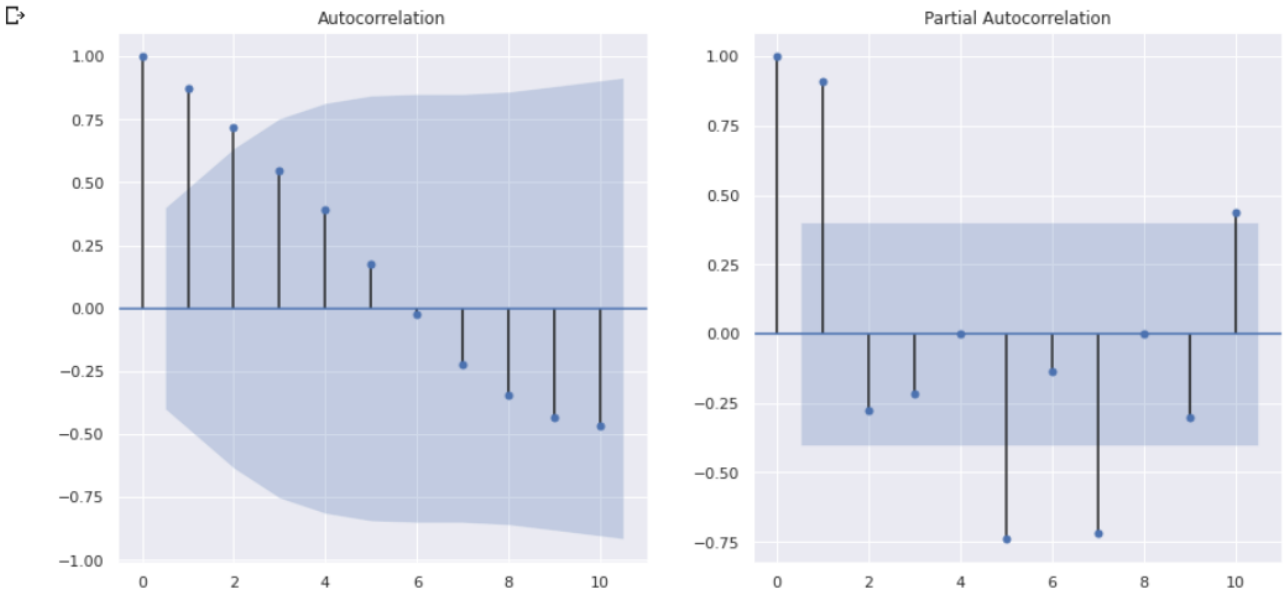


Fig 7.3: ACF and PACF plot

The value of p and q is determine by how many lines crossing the error bandwidth. It will give value of p and q for the data. For example here the value of p=[0,1,2,3,4] and q=[0,1,2]. In the same way we can go for P and Q value for data if there is seasonality in the dataset.

7.4.3 Training the Model:

The common practice is to have your training data be 70% of the original data set, and the testing data for verifying your accuracy will be the remaining 30%. Here we are splitting the data through the row indexing method.

Example of how we are splitting the data:

```
[ ]  # Splitting the data into train and test by row indexing
     ss=int(len(Positive)*0.70)
     train_pos=Positive[:ss]
     test_pos=Positive[ss:]
```

After splitting and getting p, q, P and Q values we are going to find the SARIMA model for which AIC score is minimum. Akaike information criterion (AIC) works by evaluating the model's fit on the training data and adding a penalty term for the complexity of the model (similar fundamentals to regularization.) The desired result is to find the lowest possible AIC, which indicates the best balance of model fit with generalizability.

```
p=[]
q=[]
P=[]
Q=[]
aic=[]
for i in range(2):
  for j in range(3):
    for k in range(2):
      for l in range(3):
        sarima_pos=SARIMAX(Positive,order=(i,0,j),seasonal_order=(k,0,l,4),enforce_stationarity=False).fit()
        p.append(i)
        q.append(j)
        P.append(k)
        Q.append(l)
        aic.append(sarima_pos.aic)
```

```
[ ]  df=pd.DataFrame({"p":p,"q":q,"P":P,"Q":Q,"AIC":aic})
     df.sort_values(by='AIC',ascending=True).head()
```

	p	q	P	Q	AIC
32	1	2	0	2	221.144916
35	1	2	1	2	222.556143
14	0	2	0	2	227.318174
17	0	2	1	2	228.425169

After this step we have define a function for model building and done prediction, forecasting for every sentiment whether it is positive, neutral or negative sentiment.

```
# defining the model for Time series analysis plot
def TSAplots(data=Positive,train=train_pos,p=1,d=0,q=2,P=0,Q=2,D=0,title="Trend of Positive Sentiment Count Prime Pantry"):
    global m2_pos
    m2_pos=SARIMAX(train,order=(p,d,q),seasonal_order=(P,D,Q,4)).fit()
    f2_pos=m2_pos.predict(len(train),len(data)-1)
    new_model_pos=SARIMAX(data,order=(p,d,q),seasonal_order=(P,D,Q,4)).fit()
    fc_pos=new_model_pos.forecast(10)
    plt.subplots(figsize=(12,8))
    plt.plot(data,color='orange',label='Actual')
    plt.plot(fc_pos.combine_first(data[:-2:-1]),color='green',label='Forecast')          #Combine two DataFrame objects by filling null values in one DataFrame with non-null values from other DataFrame
    plt.plot(f2_pos,color='blue',label='predicted test')
    plt.title(title,fontsize=14)
    plt.xlabel('Time Period (Quaterly)',fontsize=12)
    plt.ylabel('Count of Sentiment',fontsize=12)
    plt.legend(loc='left')
```

TSA plot showing prediction and forecasting for positive sentiment of customers:

```
# Time Series Analysis plot for Actual,Predicted and Forecast
TSAplots(data=Positive,train=train_pos,p=1,d=0,q=2,P=0,Q=2,D=0,title="Trend of Positive Sentiment Count Prime Pantry")
```

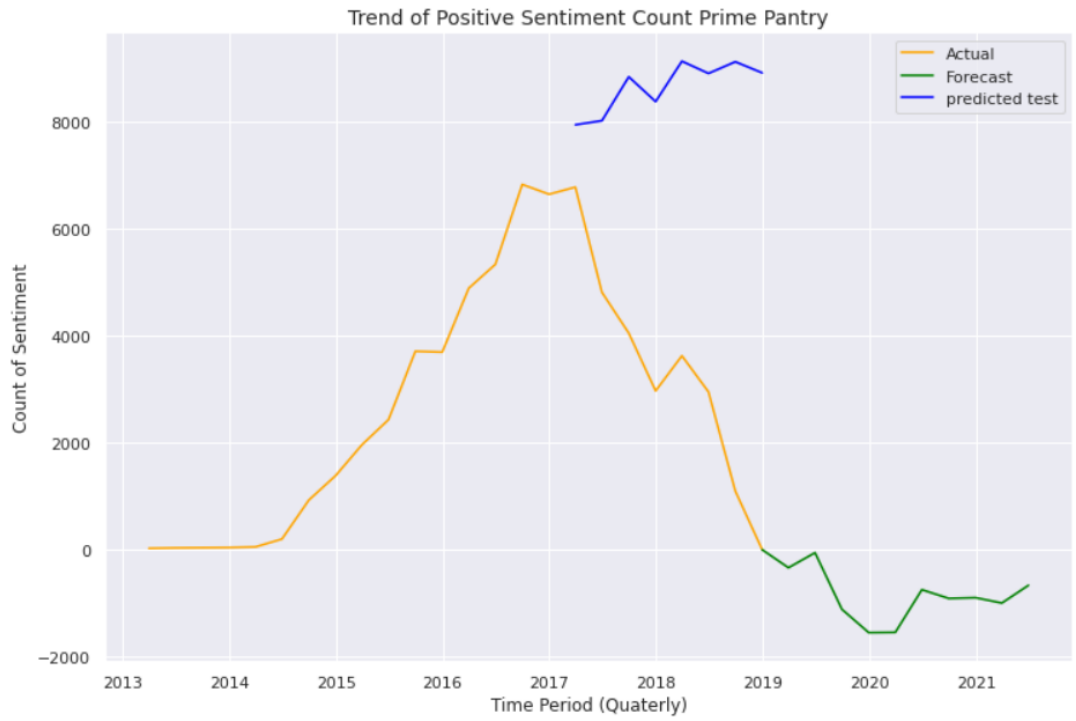


Fig 7.4: Trend of Positive Sentiments

From the above plot we can say that the positive sentiments of customers will decrease till 2020 and after that it will increase, as per shopping habits of people.

TSA plot showing prediction and forecasting for Neutral sentiment of customers:

```
# Time Series Analysis plot for Actual,Predicted and Forecast
TSAplots(data=Neutral,train=train_neu,p=1,d=4,q=2,P=0,Q=2,D=0,title="Trend of Neutral Sentiment Count Prime Pantry")
```

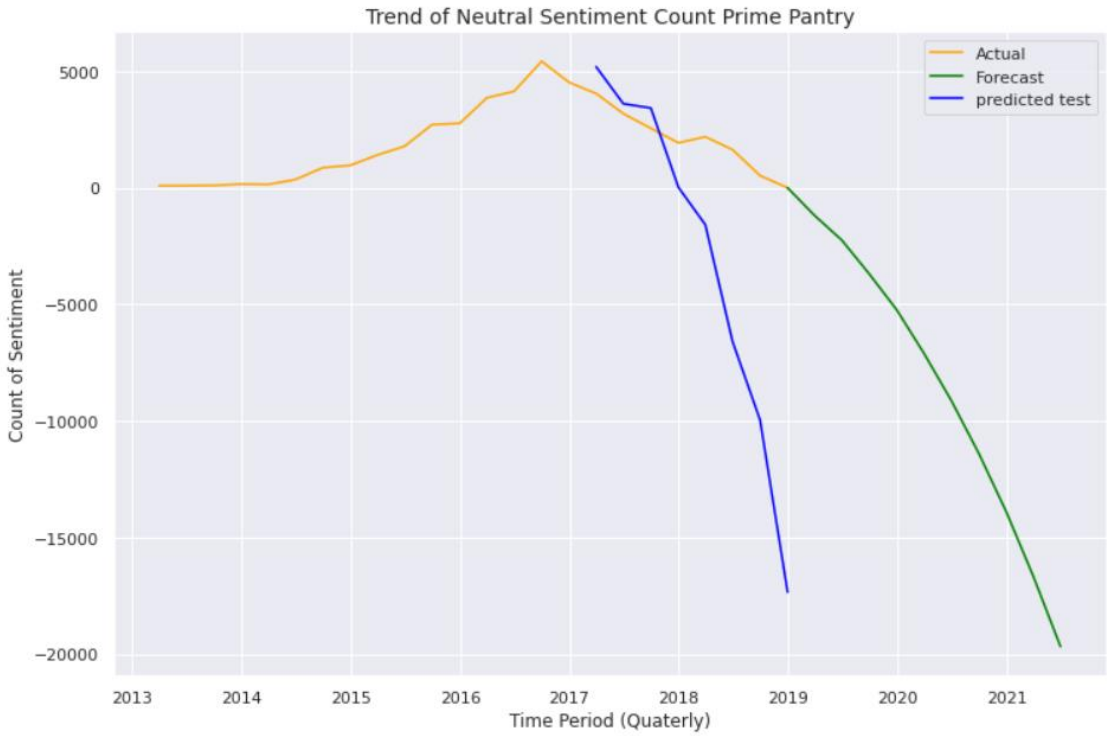


Fig 7.5: Trend of Neutral Sentiments

TSA plot showing prediction and forecasting for Negative sentiment of customers:

```
# Time Series Analysis plot for Actual,Predicted and Forecast
TSAplots(data=Negative,train=train_neg,p=2,d=0,q=2,P=0,Q=3,D=0,title="Trend of Negative Sentiment Count Prime Pantry")
```

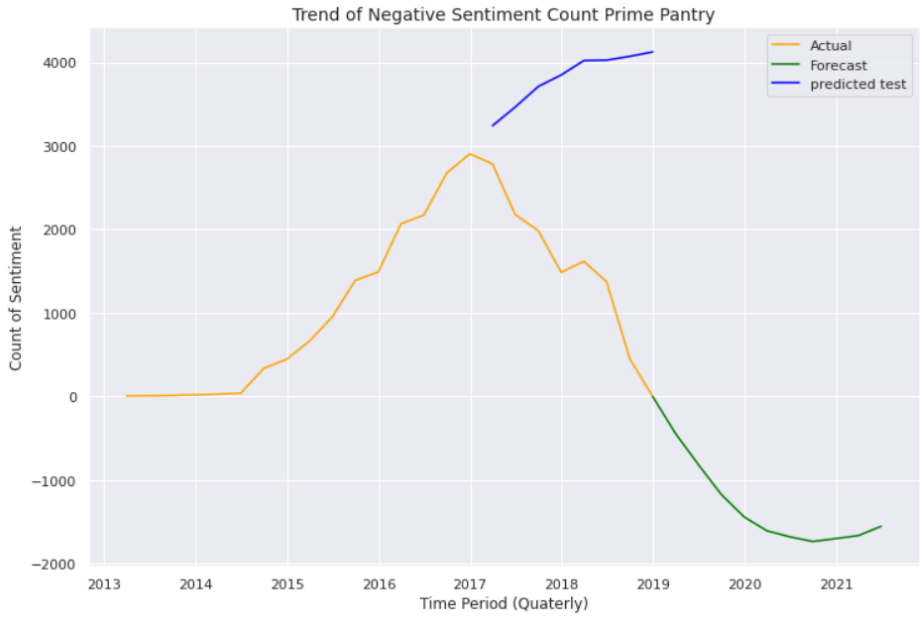


Fig 7.6: Trend of Negative Sentiments

At last we go for **Ljung Box test**. It is a way to test for the *absence* of serial autocorrelation, up to a specified lag *k*.

Ljung Box test Hypothesis-The null hypothesis of the Box Ljung Test, H_0, is that our model *does not* show lack of fit (or in simple terms—the model is just fine). The alternate hypothesis, H_a, is just that the model *does* show a lack of fit. A significant p-value in this test rejects the null hypothesis that the time series isn't autocorrelated. Output of one of the model test is displayed below:

```
[ ]  #perform Ljung-box test
     pvalue = sm.stats.acorr_ljungbox(m2_pos.resid,lags=[1],return_df=True)['lb_pvalue'].values
     if pvalue < 0.05:
         print("Reject H0. Bad model")
     else:
         print("Fail-to-Reject H0. Good model")

     Fail-to-Reject H0. Good model
```

7.5 Implementation:

For the experimenting and testing suite, we programmed these in python due to python having good documentation and there are multiple good libraries in python for math and statistical data handling. The libraries we used for the implementation are:

• Numpy: This is one of the most common math libraries. It is also really good for array handling and collections.

• Statsmodels: This is a great library with many forecasting models and other tools for statistical data handling.

• Pandas: This library has a lot of functionality for both series, reading external csv files and data handling in terms of series and other files. It is one of the more commonly used data analysis libraries.

• Matplotlib: It handles the plotting of data and has been used for all the plotted figures in the implementation.

Conclusion:

We analyzed our data set and experimented with the SARIMA model. From our analysis following conclusion is reached about each sentiments:

- From the TSA plot we can say that the positive sentiments of customers will decrease till 2020 and after that it will increase, as per shopping habits of people.

- From the plot we can say that the neutral sentiments of customers will gradually decrease ,as per shopping habits of people.

- From the plot we can say that the negative sentiments of customers will decrease till 2020 and after that it will slightly increase, as per shopping habits of people.

- Customers on whom brands should focus

```
[ ]  #Top 20 customer who shop more frequently
     shop_tworm.iloc[:,[0,3,4,-3,-1]].sort_values(by='Count_shopping',ascending=False).head(20)
```

	reviewerID	reviewerName	Analysis	Count_shopping	freq_time
18842	A35Q0RBM3YNQNF	M. Hill	Neutral	168	6 days 14:34:17.142857142
29907	ANDVNCX6JU4XW	SHERRY MCCAUGHTRY	Positive	124	0 days 08:19:21.290322580
765	A13J2PGKNMJG1K	LegoGirl	Neutral	118	11 days 11:35:35.593220338
29752	AMMNGUJK4HQJ5	Misty	Neutral	116	12 days 06:12:24.827586206
10362	A26K3T6L5NYO7L	PennyPincher	Neutral	111	10 days 00:25:56.756756756
10066	A25DP3DWUXSS48	KT	Neutral	93	11 days 22:27:05.806451612
17185	A2YKWYC3WQJX5J	ShannonOnTheLakes	Neutral	79	44 days 09:43:17.468354430
29271	AKPG8VQBS0MWR	Old Coast Customer	Neutral	78	10 days 15:23:04.615384615
20943	A3EF7PUYTF057Z	Gary R. Jordan	Positive	73	20 days 18:04:55.890410958
4764	A1JN63QBBNGB78	Elle S	Neutral	72	17 days 02:20:00
2756	A1BT9J2I6DC246	Debbie	Positive	72	14 days 19:40:00
26368	A92ZKEZI137M1	Lisa M. Rainer	Negative	67	4 days 05:22:23.283582089
21115	A3F9UAX22LLZWK	K K Schwartz	Positive	67	16 days 02:51:56.417910447
14668	A2O421DTA8J0RW	Dogs & Horses	Neutral	65	19 days 05:32:18.461538461
10501	A276RHM6BBPDTY	Ddee	Positive	64	20 days 06:45:00
2430	A1AB6D301MOTM0	Lynn G.	Neutral	63	7 days 10:17:08.571428571
32317	AXK37UZY8UPYP	Que Sera Sera	Positive	62	13 days 00:23:13.548387096
7833	A1W511P7B2QSQE	MariamG	Neutral	62	10 days 08:07:44.516129032

- Customers likely to churn out

```
#bottom 10 cutomers who shops very less.
shop_tworm.iloc[:,[0,3,4,-3,-1]].sort_values(by='Count_shopping',ascending=True).head(10)
```

	reviewerID	reviewerName	Analysis	Count_shopping	freq_time
0	A0526222H977CBZM4DK7	JAIME SCARPITTA	Negative	1	192 days
19987	A3AE8HSBCSLYX4	sly	Negative	1	35 days
6968	A1SU2TR45U1VB4	Chelsie Luchini	Neutral	1	74 days
19988	A3AE8HSBCSLYX4	sly	Neutral	1	35 days
6962	A1STPMGQSC12NS	valeri	Positive	1	226 days
19991	A3AEAMF75QS4WB	Amazon Customer	Neutral	1	199 days
19993	A3AECY1VS8T6V5	Y. Pope	Negative	1	814 days
20002	A3AG3ZHG78N4M5	Pat	Negative	1	683 days
20006	A3AGK4J9PHB6XE	Sbuxgirl36	Neutral	1	316 days
6952	A1SSOLFAUR915J	Crystal	Neutral	1	474 days

Chapter 8
Results and Conclusion

1. According to our model evaluation we can conclude that Logistic Regression by TF-IDF vectorization method is the best one. After hyperparameter tuning our Logistic Regression model will predict correctly 95 times out of 100 times. So we can use this model to predict further on this Amazon Prime Pantry Data.

2. According to our analysis as the rating is increasing the positive cluster density is increasing. And as the rating is decreasing the negative cluster density is increasing.

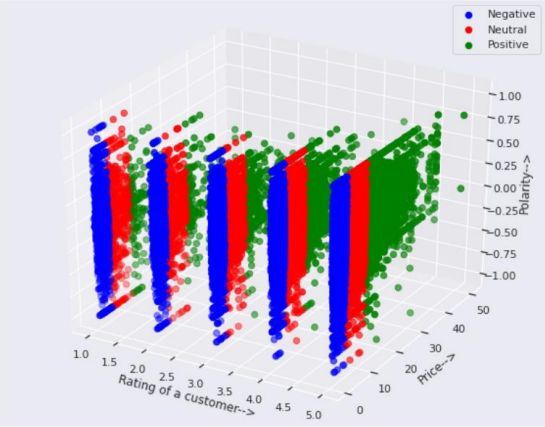


Fig 8.1: Clustering of Sentiment Analysis in 3D

3. a) From the TSA plot we can say that the positive sentiments of customers will decrease till 2020 and after that it will increase, as per shopping habits of people.

   b) From the TSA plot we can say that the neutral sentiments of customers will gradually decrease, as per shopping habits of people.

   c) From the plot we can say that the negative sentiments of customers will decrease till 2020 and after that it will slightly increase, as per shopping habits of people

4. Customer on whom brand should focus on:

	reviewerID	reviewerName	Analysis	Count_shopping	freq_time
18842	A35Q0RBM3YNQNF	M. Hill	Neutral	168	6 days 14:34:17.142857142
29907	ANDVNCX6JU4XW	SHERRY MCCAUGHTRY	Positive	124	0 days 08:19:21.290322580
765	A13J2PGKNMJG1K	LegoGirl	Neutral	118	11 days 11:35:35.593220338
29752	AMMNGUJK4HQJ5	Misty	Neutral	116	12 days 06:12:24.827586206
10362	A26K3T6L5NYO7L	PennyPincher	Neutral	111	10 days 00:25:56.756756756
10066	A25DP3DWUXSS48	KT	Neutral	93	11 days 22:27:05.806451612
17185	A2YKWYC3WQJX5J	ShannonOnTheLakes	Neutral	79	44 days 09:43:17.468354430
29271	AKPG8VQBS0MWR	Old Coast Customer	Neutral	78	10 days 15:23:04.615384615
20943	A3EF7PUYTF057Z	Gary R. Jordan	Positive	73	20 days 18:04:55.890410958
4764	A1JN63QBBNGB78	Elle S	Neutral	72	17 days 02:20:00
2756	A1BT9J2I6DC246	Debbie	Positive	72	14 days 19:40:00
26368	A92ZKEZI137M1	Lisa M. Rainer	Negative	67	4 days 05:22:23.283582089
21115	A3F9UAX22LLZWK	K K Schwartz	Positive	67	16 days 02:51:56.417910447
14668	A2O421DTA8J0RW	Dogs & Horses	Neutral	65	19 days 05:32:18.461538461
10501	A276RHM6BBPDTY	Ddee	Positive	64	20 days 06:45:00
2430	A1AB6D301MOTM0	Lynn G.	Neutral	63	7 days 10:17:08.571428571
32317	AXK37UZY8UPYP	Que Sera Sera	Positive	62	13 days 00:23:13.548387096
7833	A1W511P7B2QSQE	MariamG	Neutral	62	10 days 08:07:44.516129032

5. Customers likely to churn out:

```
#bottom 10 cutomers who shops very less.
shop_tworm.iloc[:,[0,3,4,-3,-1]].sort_values(by='Count_shopping',ascending=True).head(10)
```

	reviewerID	reviewerName	Analysis	Count_shopping	freq_time
0	A0526222H977CBZM4DK7	JAIME SCARPITTA	Negative	1	192 days
19987	A3AE8HSBCSLYX4	sly	Negative	1	35 days
6968	A1SU2TR45U1VB4	Chelsie Luchini	Neutral	1	74 days
19988	A3AE8HSBCSLYX4	sly	Neutral	1	35 days
6962	A1STPMGQSC12NS	valeri	Positive	1	226 days
19991	A3AEAMF75QS4WB	Amazon Customer	Neutral	1	199 days
19993	A3AECY1VS8T6V5	Y. Pope	Negative	1	814 days
20002	A3AG3ZHG78N4M5	Pat	Negative	1	683 days
20006	A3AGK4J9PHB6XE	Sbuxgirl36	Neutral	1	316 days
6952	A1SSOLFAUR915J	Crystal	Neutral	1	474 days

Amazon Prime Pantry

According to our analysis on the Amazon Prime Pantry dataset, we can conclude that:

- There are 46.49% customers with positive sentiments.
- 19.87% negative sentiment customers are involved.
- 33.64% customers have neutral sentiments.
- Sherry Mccaughtry is the most satisfied customer with 124 positive reviews.
- Lisa M. Rainer is the most dissatisfied customer 67 negative reviews.

Chapter 9
References

- https://www.analyticsvidhya.com
- https://stackoverflow.com
- https://www.javatpoint.com
- https://www.techtarget.com
- https://scikit-learn.org/stable
- https://www.statsmodels.org/stable/index.html