FULL STACK SCRUTINY

A
Project Work
Submitted as Minor Project in Partial fulfillment for the award of Graduate Degree in Bachelor of Engineering in Computer Science & Engineering.

Submitted to

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA BHOPAL (M.P)

Submitted By--

ANSHUL VERMA (0105CS191023) [TEAM LEADER]
AYUSH WAGHMARE (0105CS191028)
HARSHIT SHRIVASTAVA (0105CS191048)
MANISH NATHRANI (0105CS191062)

Under the Guidance of -

PROF. GOLDI JARBAIS

(Department of Computer Science & Engineering)

Oriental Institute of Science & Technology, Bhopal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JAN - JUNE 2022

Oriental Institute of Science & Technology, Bhopal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled

"FULL STACK SCRUTINY" being submitted by

ANSHUL VERMA (0105CS191023) [TEAM LEADER],

AYUSH WAGHMARE (0105CS191028),

HARSHIT SHRIVASTAVA (01015CS191048) and

MANISH NATHRANI (0105CS191062) students of **6th** Semester, B. Tech in Computer Science & Engineering have done their work as MINOR PROJECT-II for Partial fulfillment of the B.Tech degree from RGPV, Bhopal (M.P.) is a record of Bonafede work carried out by them/him/her under our supervision.

Guide
Department of Computer Science& Engineering

Head
Department of Computer Science & Engineering

ACKNOWLEDGEMENT

We take the opportunity to express our cordial gratitude and deep sense of indebtedness to our **Guide** for the valuable guidance and inspiration throughout the project duration. We feel thankful to her for innovative ideas, which led to successful submission of this minor project work. We feel proud and fortune to work under such an outstanding mentor in the field of Data Science. She has always welcomed our problems and helped us to clear our doubts. We will always be grateful to her for providing us moral support and sufficient time.

We owe sincere thanks to **Director, OIST** for providing us with moral support and necessary help during our project work in the Department.

At the same time, we would also like to thank **HOD, CSE** and all other faculty members and all non-teaching staff of department of Computer Science &Engineering for their valuable co-operation.

We would also thank to our Institution, faculty members and staff without whom this project would have been a distant reality. We also extend our heartfelt thanks to our family and well-wishers.

ANSHUL VERMA
0105CS191023

AYUSH WAGHMARE
0105CS191028

HARSHIT SHRIVASTAVA
0105CS191048

MANISH NATHRANI
0105CS191062

APPROVAL CERTIFICATE

This is to certify that the project entitled FULL STACK SCRUTINY, being submitted by –

ANSHUL VERMA (0105CS191023) [TEAM LEADER],

AYUSH WAGHMARE (0105CS191028)

HARSHIT SHRIVASTAVA (0105CS191048) and

MANISH NATHRANI (0105CS191062)

students of 6th Semester, Bachelor of Technology in Computer Science and Engineering have done their work as Minor Project for Partial fulfillment of the degree from RGPV, Bhopal (M.P.).

Guide Name:

Date:

Signature:

ABSTRACT

The emergence of many competitors and entrepreneurs has created a great deal of tension between competing businesses to find new buyers and keep old ones. As a result of the preceding, the need for exceptional customer service becomes appropriate, regardless of the size of the business. In addition, the ability of any business to understand each of its customers' needs will receive greater support in providing targeted customer services and developing customized customer service plans. This understanding is possible through structured customer service. Each segment contains customers who share similar market features. Big data ideas and machine learning have fostered more acceptance of the automated customer segmentation approach in favor of traditional market analytics that often do not work especially when the customer base is too large. In this paper, the k-Means clustering algorithm is used for this purpose. The sklearn library was developed for the k-Means algorithm (found in the appendix) and the program is trained using a two-factor dataset of 100 patterns obtained from the retail business. Features of the average number of customer purchases and the average number of monthly customer visits.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The main idea of this project is to find similarities between people who are currently customers and people who are not. Then, use this information to find groups of potential new customers (people who are not currently customers but have high similarities with people who are).

Sentiment analysis and opinion mining have been acquiring a crucial role in both commercial and research applications because of their possible applicability to several different fields. Therefore, a large number of companies have included the analysis of opinions and sentiments of customers as part of their mission. One of the most interesting applications of these approaches involves the automatic analysis of social network messages, on the basis of the feelings and emotions conveyed. This chapter aims to relate the most recent state-of-the-art sentiment-based techniques and tools to the affective characterization that may be inferred from social networks. The main result consists of a review of the most interesting methods employed to compare and classify messages on social media platforms and a description of advanced tools in this area.

Various researches on this process have led to formation of different data mining algorithms. We can use these algorithms directly on a dataset to create some models and get the inferences from the dataset. Popular data mining algorithms are Nearest Neighbor (CNN), Naive Bayes, Random Forest, Logistic regression, Decision tree, and Support vector machines.

1. Nearest neighbor (CNN) is a managed machine calculation which can be utilized to unravel a wide range of relapse and arrangement issues.

2. Naive Bayes calculation is a group of probabilistic calculations that exploits the likelihood hypothesis and Bayes hypothesis to anticipate the tag of a book.

3. Logistic relapse is a characterization calculation used to allot perception to set of discrete classes. A portion of the characterization issues are email spam, online misrepresentation identification, tumor harmful or kind. Calculated relapse essentially changes its yield utilizing the strategic sigmoid capacity to restore likelihood esteem.

4. Decision tree calculation goes under the class of directed learning. This calculation can be utilized to explain both the relapse and grouping issues. The choice tree utilizes tree portrayal to take care of the issues.

5. A support vector machine is an administered AI calculation which can be utilized for both relapse and grouping issues. In help vector machine, we plot every datum thing as a point in N-dimensional space (where n is the quantity of highlights) with the estimation of each component being a similar estimation of specific organizations.

6. Random timberland calculation is a technique that works by building numerous choice trees during preparing stage. The choice of greater part of the trees is picked by the arbitrary backwoods as an official conclusion. It utilizes include haphazardness and stowing when assembling an individual tree to attempt to make a wood of trees whose forecast is more exact and precise than that of any individual tree.

1.2 PROJECT OBJECTIVE AND SCOPE

When determining how to segment your customers, start by working through the following strategy.

I. Determine your Customer Segmentation Goals

- Think about why you're creating a customer segmentation strategy — ask yourself why you are spending time on segmentation and what you hope to derive from the process.

- To do this, refer to the list of common reasons businesses choose to segment customers we reviewed above. Determine which outcomes that you're

looking to achieve so you can develop the rest of your strategy in a way that will help you accomplish them.

- Something to note when developing your customer segmentation goals is that they're going to be unique to *your* business — segmenting customers isn't a one-size-fits-all process.

- For example, your number of goals will be unique to you based on your business's size, type, and industry as well as who your customers are. Additionally, your goals may be relevant cross-team (e.g. marketing, sales, service) or for one department specifically. So, when determining your customer segmentation goals, use your company's specific traits and business needs as a launching pad.

II. Segment your customers into groups of your Choice

- Once you have an idea of what you're looking to get out of the customer segmentation process, decide how you'll segment your customers. Refer back to the customer segmentation models and types to determine how you'll do this. There's no right or wrong answer here — it's dependent on your specific business, customers, and the goals you set in the previous step.

- For example, if you want to share targeted ads with your audience members and customers on the West Coast in hopes of boosting conversions in that region, you can geographically segment your customers.

III. Target and Reach your Customer Segments

- Once you've segmented your customers, it's time to determine how you'll target them across your organization. By ensuring all departments (e.g. marketing, sales, and service) understand how your customers are segmented, members of those departments will be able to effectively target your customers through their work.

- For example, think back to our example above about targeting ads to your West Coast customers:

- Marketing can tailor and customize content to attract, teach, and meet the needs of your West Coast audience members to boost leads and brand awareness.

- Sales can identify common traits shared by your most-qualified West Coast customers — as well as best ways to reach out to and communicate with them — to increase conversions.

- Service can use your customer segments to prepare materials and resources for them based on challenges West Coast customers are most likely to experience.

IV. Run Customer Segmentation Analysis.

- Analyzing your segmentation efforts will provide insight into the way you've organized your customers — this way, you can make updates and changes if needed.

- Check in with your marketing, sales, and service teams (as often as you want) to get their opinions on any necessary segmentation adjustments. You can also experiment with new ways of grouping your customers together to decide what makes the most sense.

- You can also gather feedback from your customers to more effectively segment them into appropriate groups. For example, you could conduct surveys to improve your behavioral segmentation by asking customers about their feature use and product-use habits/ tendencies. This will allow you to more accurately organize customers based on their specific behaviors.

V. Run Sentiment Analysis

- Sentiment analysis of in the domain of micro-blogging is a relatively new research topic so there is still a lot of room for further research in this area. Decent amount of related prior work has been done on sentiment analysis of user reviews, documents, web blogs/articles and general phrase level sentiment analysis. These differ from twitter mainly because of the limit of 140 characters per tweet which forces the user to express opinion compressed in very short text.

- The best results reached in sentiment classification use supervised learning techniques such as Naive Bayes and Support Vector Machines, but the manual labelling required for the supervised approach is very expensive. Some work has been done on unsupervised and semi-supervised approaches, and there is a lot of room of improvement. Various researchers testing new features and classification techniques often just compare their results to base-line performance. There is a need of proper and formal comparisons between these results arrived through different features and classification techniques in order to select the best features and most efficient classification techniques for particular applications. You can also achieve the objective by

    - To minimize your efforts and time investment.
    - To make a user-friendly atmosphere for the app users.

CHAPTER 2

BACKGROUND AND LITERATURE SURVEY

2.1 LITERATURE SURVEY

In Customer Segmentation, Customers can have different types of characteristics and can be of different importance to a company. For companies to know which customers are of significance, a segmentation of customer's needs to be done (McDonald & Dunbar, 2012). The theory of segmentation is the process were identifying characteristics of different customers and dividing them into groups. What companies often do when segmenting their customers is to divide them based on how much revenue they contribute to the company based on their purchase volumes (Batt, 2000)

Identifying and classifying customers leads to a better understanding of who the customers are and what type of demand the customers require. Some customer groups can have a high degree of innovation where changes within the customer group over time often occur. For 9 these types of customers, you need to be aware of the requirement changes to meet the customer demand in the best way which also fulfils the customer needs (Bottcher, Spott, Nauck & Kruse, 2009)

According to Sandström (2003) segmentation has its basis in the concept that consumers who take part of the company's products and services are not proportionally valuable. For companies, customers are of different significance and to be able to stay in the market, companies need to distribute their attention unevenly, meaning that they need to move attention from the non-profit consumers to the ones with higher profit. For a company to continue gaining profit, they need to target a lot of attention to the customers that consume their products or services frequently or in greater volumes to create groups that are fruitful (Sandström, 2003). When a company has the right knowledge about the customer requirements it will give them the ability to easier divide the customers into segmentation groups. Furthermore, the company can easier find out what satisfies their customers and even surprise them. This kind of information can be used for further improvements into their services or products. These days, customer service is as important for the customers as the actual product or service,

and it is important that the companies have this part set. Finally, segmenting customers can simplify the choices of how much and what the company should put emphasis on when it comes to the degree of services that the different groups should get (Buttle, 2009). Lambert (1990) state the importance of segmenting markets in emerging production industries as well as in service industries. All kinds of organizations need to find a method that fits to categorize the market into different segments to meet the customer demand in best way and increase the revenue for the company. According to Lambert (1990) when an organization starts the process of segmenting it is recommended to look at the customers from a need based point of view where the customer with high demand should be prioritized.

A company provides the market with either a service or a product. Because of this it is vital according to (Fang, Palmatier & Steenkamp 2008) for a company to reach the customer service elements to please their customers. Well established service companies have the right skillset and right knowledge to fulfil the demands, expectations and needs of their customers (Mattson, 2004). The concept of customer service can be defined as what a company does to include the purchasers, sellers and other groups that can boost their product or service. A successful customer segmentation within services benefits the company to enhance their relationship with their purchasers and sellers which also contributes to an enhanced competitiveness (Pauline, 2009).

Out in the market there are a huge range of service providing companies with many different types of customers. To be competitive and meet the customer demand in best way, service companies have different strategies of how to target and segment their customers (Anderson, Narus & Narayandas 2009).

Sentiment analysis is a growing area of Natural Language Processing with research ranging from document level classification (Pang and Lee 2008) to learning the polarity of words and phrases (e.g., (Hatzivassiloglou and McKeown 1997; Esuli and Sebastiani 2006)). Given the character limitations on tweets, classifying the sentiment of Twitter messages is most similar to sentencelevel sentiment analysis (e.g., (Yu and Hatzivassiloglou 2003; Kim and Hovy 2004)); however, the informal and specialized language used in tweets, as well as the very nature of the microblogging domain make Twitter sentiment analysis a very different task. It’s an open question how well the features and techniques used on more well-formed data will transfer to the microblogging domain. Just in the past

year there have been a number of papers looking at Twitter sentiment and buzz (Jansen et al. 2009; Pak and Paroubek 2010; O'Connor et al. 2010; Tumasjan et al. 2010; Bifet and Frank 2010; Barbosa and Feng 2010; Davidov, Tsur, and Rappoport 2010). Other researchers have begun to explore the use of part-of-speech features but results remain mixed. Features common to microblogging (e.g., emoticons) are also common, but there has been little investigation into the usefulness of existing sentiment resources developed on non-microblogging data.

2.2 SOFTWARE REQUIREMENT SPECIFICATION

I. Internal Interface Requirement

Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem. Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority. Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type

II. External Interface Requirement

We classify External Interface in 4 types, those are: User Interface: Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface

specification. Hardware interface: Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

III. NON - FUNCTIONAL

Effective customer segments must have some requirements to make the segmentation more efficient. These are:

- Measurable -

  Segments should be easily measurable after targeting. The size, profiles and purchasing power of the segments can be measured. Sometimes, segmentation variables are difficult to measure. For example, if any company wants to launch any product which is useful for left-handed people. In America, there are more than 3 million people are left-handed, which is somehow equal to the entire population of Canada. So, if they target their market towards left-handed segment, then the major problem may be that it will be difficult to identify and measure.

- Accessible -

  Markets segments should be effectively reached and served to the targeted market. Marketers have to make their products accessible for customers. Suppose, a fragrance company gets that heavy user of one of its brands are single men and women who stay out late and socialize a lot. Unless this group shops from company's outlets, its members will be difficult to identify.

- Substantial -

  The market segments should be large or profitable to serve. Segments should be huge possible homogeneous groups worth pursuing with an altered marketing program. For example, it would not pay to automobiles manufactures, if they manufacture cars especially for those customers whose heights are more than 6.5 feet.

- Differential -

  The segments are conceptually visible and respond differently to different marketing mix elements and plan. For example, if married and non-married women react similarly to a sale of particular brand of perfume, they do not count as separate segment.

- Actionable -

  Effective plans can be designed for serving and attracting the segments. For example, even though, one small airline identifies seven market segments, airlines' staff is too small to design separate marketing plan for each segment.

IV. Performance Requirements

If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

V. Safety Requirements

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.

VI. Other Requirements

- Linux Operating System/Windows
- Python Platform(Anaconda2,Spyder,Jupyter)
- NLTK package,
- Modern Web Browser

2.3 FEASIBILITY REPORT

Feasibility Study in Software Engineering is a study to evaluate feasibility of proposed project or system. Feasibility study is one of stage among important four stages of Software Project Management Process. As name suggests feasibility study is the feasibility analysis or it is a measure of the software product in terms of how much beneficial product development will be for the organization in a practical point of view.

Types of Feasibility Study -

The feasibility study mainly concentrates on bellow five mentioned areas. Among these Economic Feasibility Study is most important part of the feasibility analysis and Legal Feasibility Study is less considered feasibility analysis. Some study topics are as follows:

1) Technical Feasibility
2) Economic Feasibility
3) Schedule Feasibility

I. TECHNICAL FEASIBILITY

In Technical Feasibility current resources both hardware software along with required technology are analyzed/assessed to develop project. This technical feasibility study gives report whether there exists correct required resources and technologies which will be used for project development. Along with this, feasibility study also analyses technical skills and capabilities of technical team, existing technology can be used or not, maintenance and up- gradation is easy or not for chosen technology etc.

II. ECONOMIC FEASIBILITY

In Economic Feasibility study cost and benefit of the project is analyzed. Means under this feasibility study a detail analysis is carried out what will be cost of the project for development which includes all required cost for final development like hardware and software resource required, design and development cost and operational cost and so on.

After that it is analyzed whether project will be beneficial in terms of finance for organization or not.

- The project is economically feasible to begin with as no expensive hardware and software components is required.
- Similarly, all the tools and techniques to be used are open source and are easily available free of cost
- Data collection is done among us individuals which is economically feasible.

III. SCHEDULE FEASIBILITY

In Schedule Feasibility Study mainly timelines/deadlines is analyzed for proposed project which includes how many times teams will take to complete final project which has a great impact on the organization as purpose of project may fail if it can't be completed on time.

- To develop the project a proper time line has been projected to complete relevant portion of the project in scheduled time period.
- Most of the Necessary resources are searched on the web and are available to begin research in time.
- Also, all the related software packages are easily available which makes if more feasible.

2.4 INNOVATIVENESS AND USEFULNESS

I. It allows you to fine-tune your message

- When you segment your marketing efforts to specific groups of people, it allows you to hone in on specific messages that you want to advertise.
- It enables you to fine-tune your marketing message to align with exactly what the recipient is looking for, and therefore, increases the chances that they'll convert.

II. Increase your revenue

- By fine-tuning your marketing message, you'll see increases in your revenue because users will be more likely to make a purchase when they're delivered exactly what they need.
- In fact, segmented and targeted emails generate 58% of all revenue for a company, which isn't hard to believe.
- When you segment your emails, you'll also have a subject line that that is personalized to the recipients' needs, which can increase open rate by 26%. And it's obvious that the more emails are opened, the more sales you'll make.

III. You'll increase awareness for your brand

- Users love personalized emails. When they receive something that's made just for them, they'll feel more comfortable purchasing from your company because they know you care.
- This also builds brand awareness because customers will remember you as the company that sends them emails based on their interests, previous spending habits, and more.

2.5 MARKET POTENTIAL AND COMPETITIVE

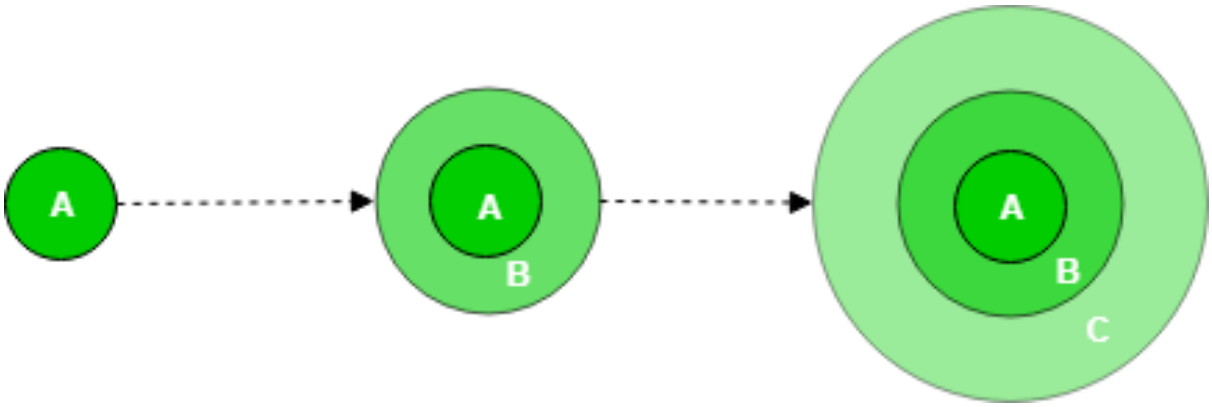
- The Idea is very unique as we using the three different types of concepts at the same place like Sentiment Analysis, Customer Segmentation and Behavior Analysis which gives us the Graphical Representation of Customers and their behaviors to make Business Tactics.
- A Person who has no experience in ML can use the Web App easily as it is very User Friendly and help them to visualize the Data.
- It consumes less time as it is deployed on a very powerful server (Heroku).

CHAPTER 3

PROCESS MODEL

3.1 PROPOSED METHODOLOGY

Incremental process model is also known as the Successive version model. First, a simple working system implementing only a few basic features is built and then that is delivered to the customer. Then thereafter many successive iterations/ versions are implemented and delivered to the customer until the desired system is released.



A, B, C are modules of Software Products that are incrementally developed and delivered.

Life Cycle Activities -

Requirements of Software are first broken down into several modules that can be incrementally constructed and delivered. At any time, the plan is made just for the next increment and not for any kind of long-term plan. Therefore, it is easier to modify the version as per the need of the customer. The development Team first undertakes to develop core features (these do not need services from other features) of the system.

Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions.

Each incremental version is usually developed using an iterative waterfall model of development.

As each successive version of the software is constructed and delivered, now the feedback of the Customer is to be taken and these were then incorporated into the next version. Each version of the software has more additional features than the previous ones.

3.2 SOFTWARE PROCESS MODEL

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieve

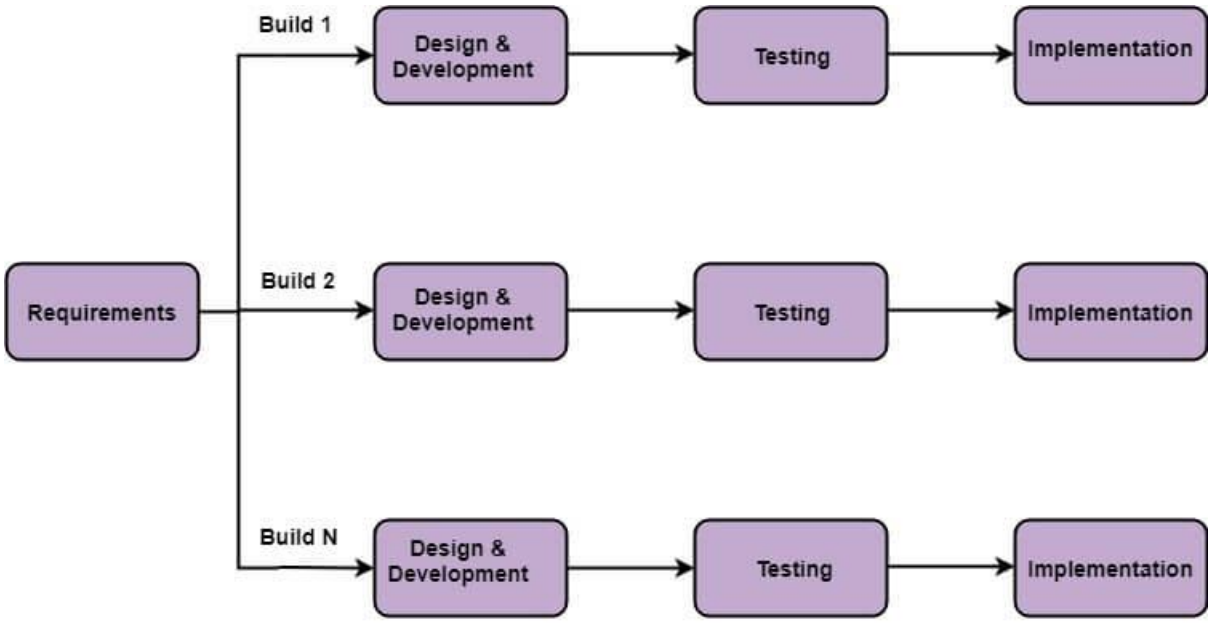


Fig: Incremental Model

The various phases of incremental model are as follows:

1. **Requirement analysis:** In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team.

To develop the software under the incremental model, this phase performs a crucial role.

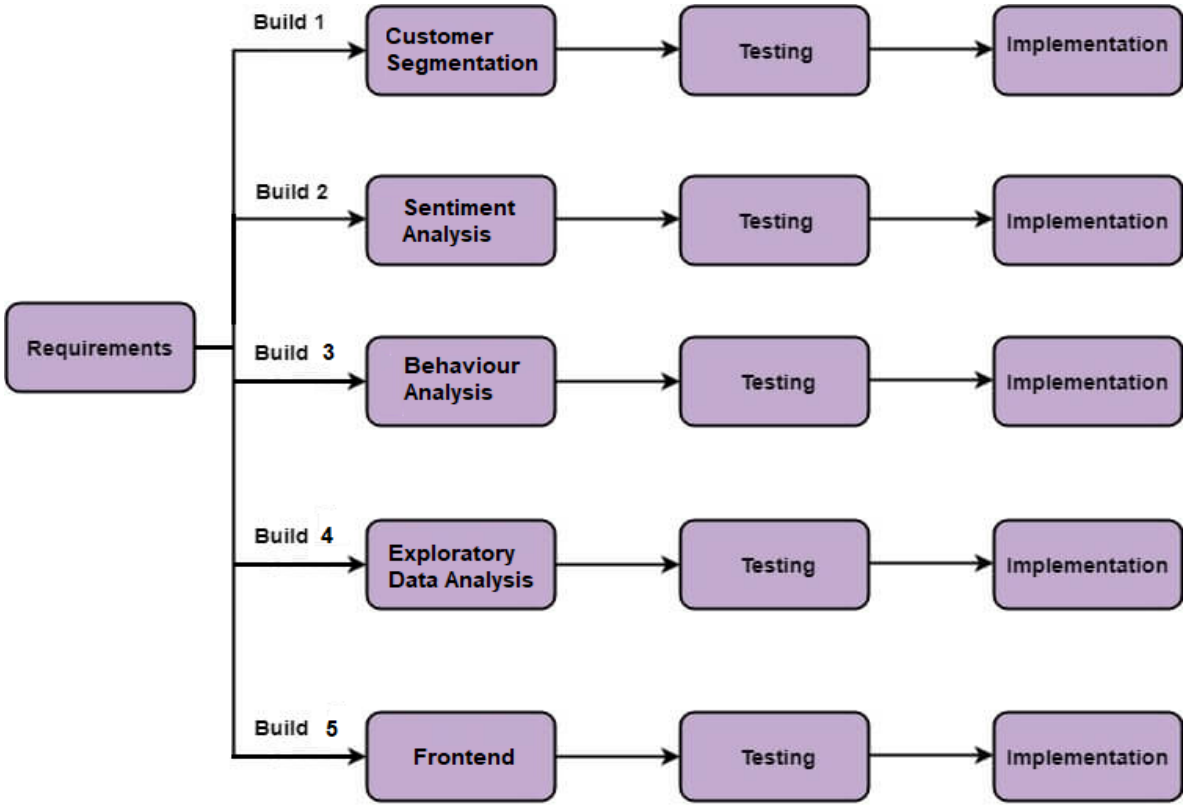
2. **Design & Development:** In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. **Testing:** In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behaviour of each task.

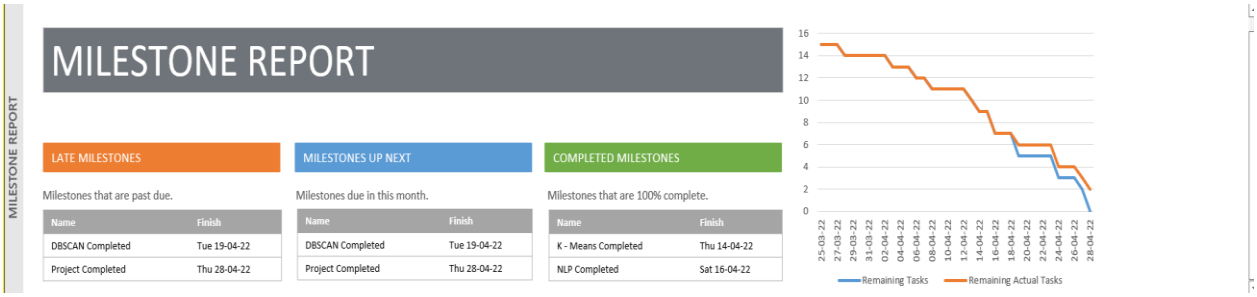
4. **Implementation:** Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

3.3 PROJECT PLAN

I. IMPLEMENTATION

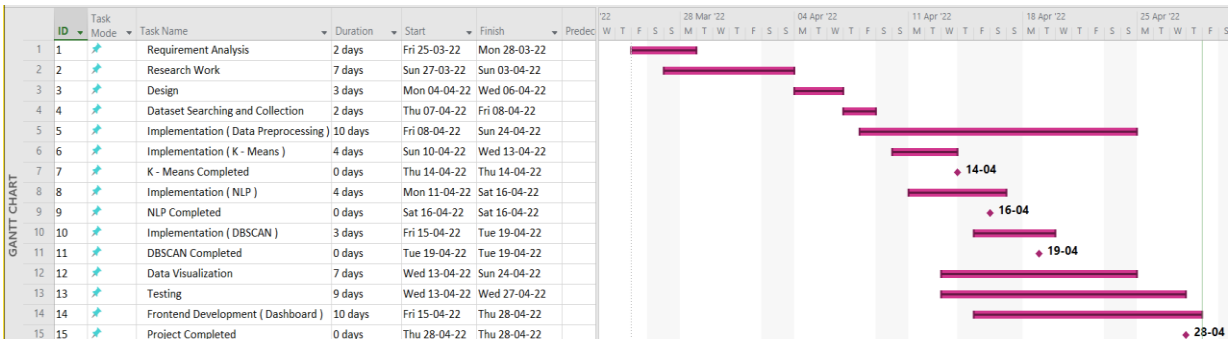


II. PROGRESS

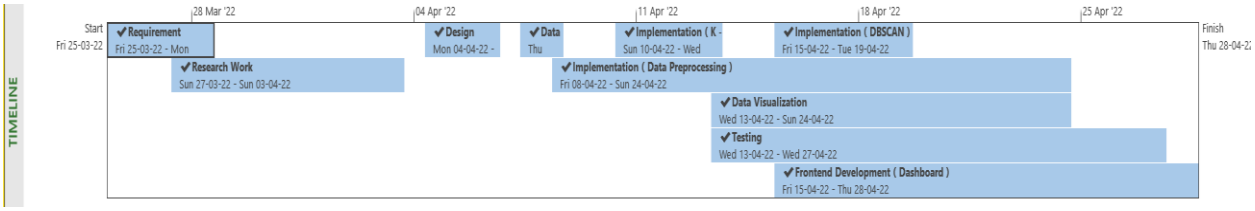


3.4 PROJECT ESTIMATION AND SCHEDULING

I. PROJECT ESTIMATION [GANTT CHART]



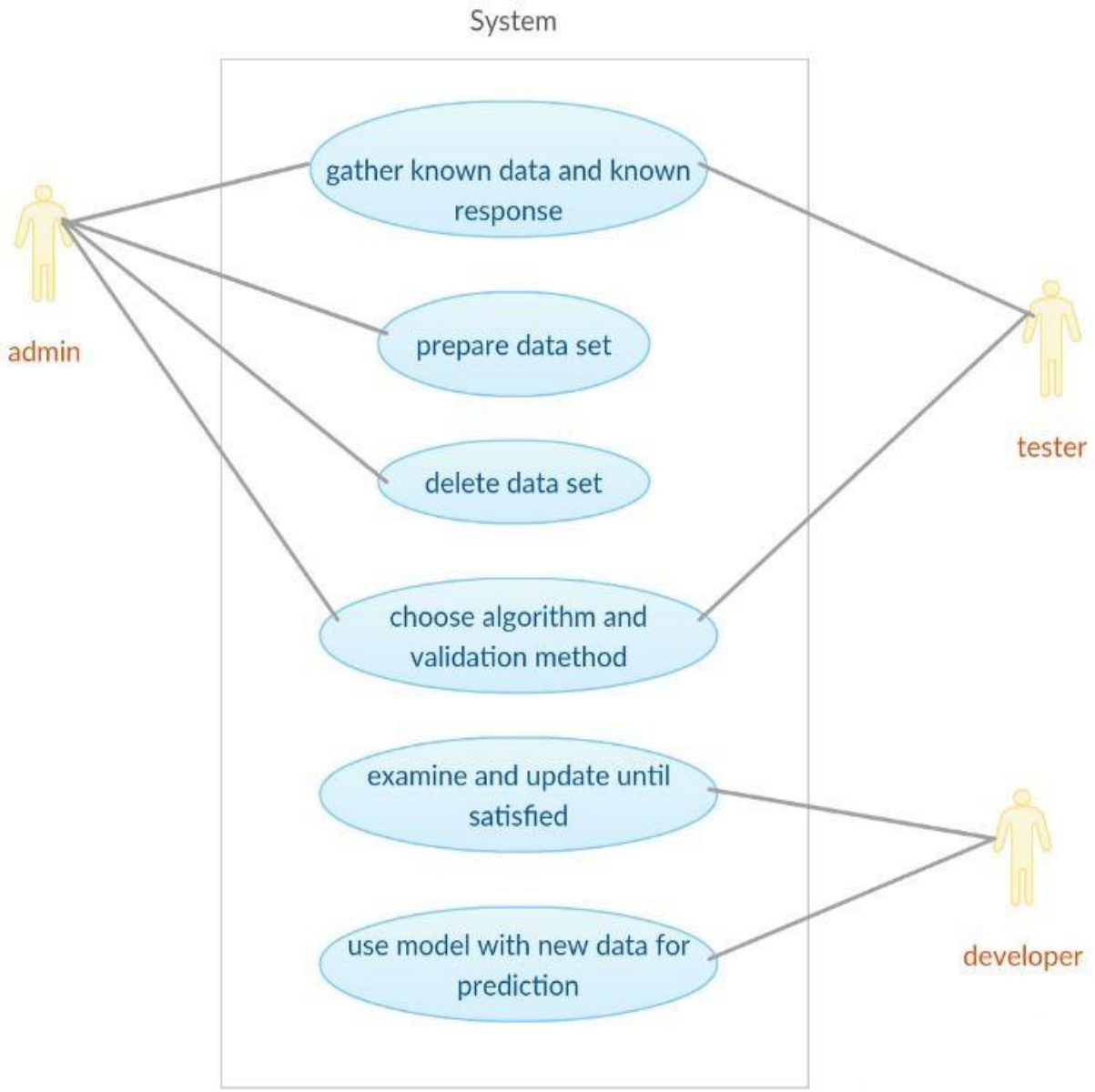
II. SCHEDULING



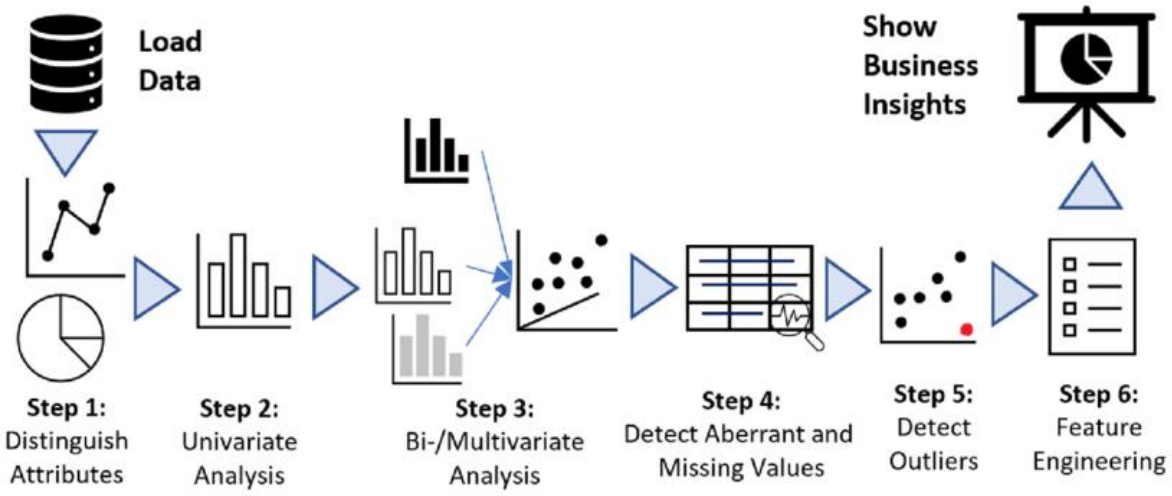
CHAPTER 4

DESIGN

4.1 USE CASE DIAGRAM

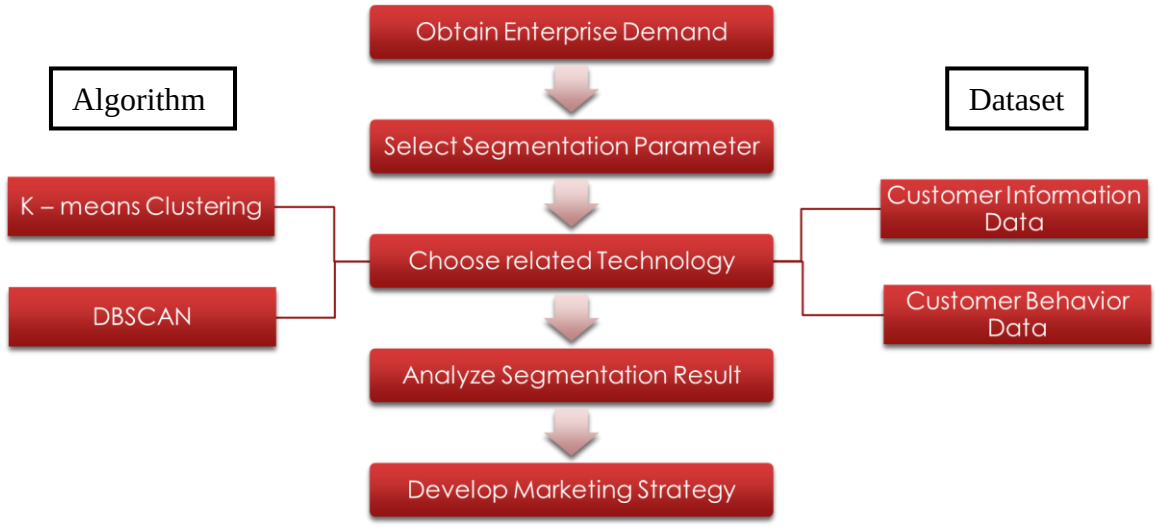


4.2 DATA FLOW DIAGRAM

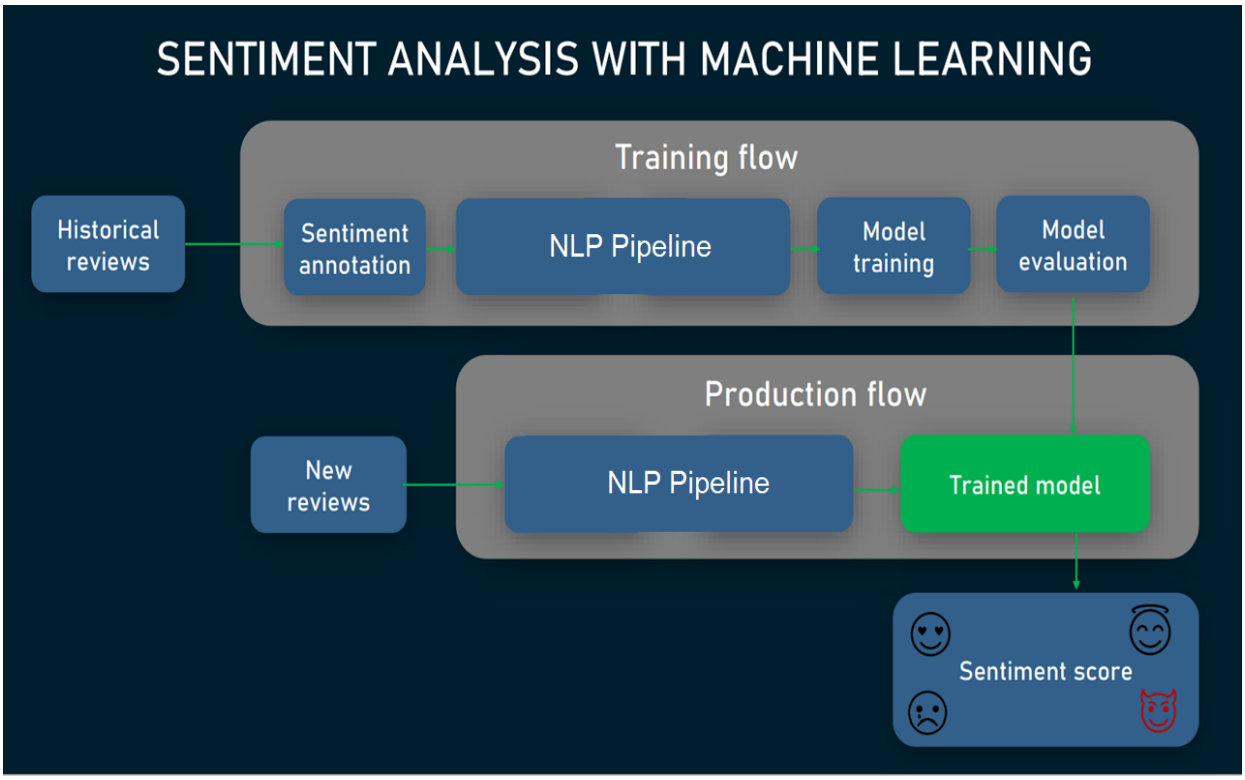


4.3 FLOW CHART

I. Customer Segmentation



II. Sentiment Analysis



4.4 ALGORITHM

I. Customer Segmentation

Customer segmentation is the process of dividing customers into groups based on common characteristics so companies can market to each group effectively and appropriately.

- K-mean Algorithm

  The K-means clustering algorithm computes centroids and repeats until the optimal centroid is found. It is presumptively known how many clusters there are. It is also known as the flat clustering algorithm. The number of clusters found from data by the method is denoted by the letter 'K' in K-means.

- DBSCAN

  DBSCAN - Density-Based Spatial Clustering of Applications with Noise. Finds core samples of high density and expands clusters from them. Good for data which contains clusters of similar density.

II. Sentiment Analysis

Sentiment analysis is the process of detecting positive or negative sentiment in text. It's often used by businesses to detect sentiment in social data, gauge brand reputation, and understand customers.

- NLP

  Natural language processing is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyse large amounts of natural language data.

CHAPTER 5

TECHNICAL DETAILS

5.1 SOFTWARE SPECIFICATIONS AND DETAILS

I. Python Libraries –

- matplotlib.
- seaborn.
- plotly
- sklearn.
- Pandas.
- Numpy
- Streamlit

II. Machine Learning Algorithms –

A. Customer Segmentation

- DBSCAN (Density Based Spatial Clustering of Application with Noise)
- K-means Clustering

B. Sentiment Analysis

- Natural Language Processing (NLP)
- Using different classifiers
  - SVM (Support Vector Machine)
  - Random Forest Classifier
  - Multinomial Naive Bayes

- Naive Bayes
- Combination of SVM and Multinomial Naive Bayes

III. Software Used –

- Anaconda
- Spyder
- Jupyter Notebooks
- Web Browser

5.2 HARDWARE REQUIREMENTS

- Processor- Quad core processor, 2.2 GHz with Turboboost upto 3.1 GHz.
- Motherboard – ASRock EPC612D8A Or better.
- RAM – 8 GB DDR4 2133 MHz's
- Hard Drive - 512 GB SSD.
- GPU – NVidia 1650 (4 GB VRAM) or Above (Optional)

CHAPTER 6

IMPLEMENTATION

I. CUSTOMER SEGMENTATION

In [ ]:
```
import math
import json
import os
import datetime

#PyData Packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb

#Sklearn Packages
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import MultiLabelBinarizer, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering

#Magic Commands
%matplotlib inline
```

In [ ]:
```
portfolio = pd.read_json("./portfolio.json", orient = 'records', lines = True )
```

In [ ]:
```
portfolio.shape
```

In [ ]:
```
transcript = pd.read_json("./transcript.json", orient = 'records', lines = True )
```

In [ ]:
```
transcript.head()
```

In [ ]:
```
transcript.shape
```

In [ ]:
```
portfolio.info()
```

In [ ]:
```
profile.info()
```

In [ ]:
```
transcript.info()
```

Portfolio Dataset Cleaning

In [ ]:
```
portfolio['offer_type'].value_counts()
```

In [ ]:
```
portfolio['channels'].astype('str').value_counts()
```

In [ ]:
```
portfolio.rename(columns={'id' : 'offer_id'}, inplace=True)
```

In [ ]:
```
offer_dummies = pd.get_dummies(portfolio['offer_type'], prefix='offer')
```

In [ ]:
```
portfolio = pd.concat([portfolio, channels_df], axis = 1)
```

In [ ]:
```
portfolio.rename(columns={'email' : 'channel_email', 'mobile' : 'channel_mobile', 'social' : 'channel_social',
                          'web' : 'channel_web'}, inplace = True)
```

In [ ]:
```
portfolio.drop(columns=['channels', 'offer_type'], inplace=True)
```

Profile Dataset Cleaning

In [ ]:
```
profile['gender'].value_counts(dropna=False)
```

In [ ]:
```
profile['age'].value_counts()
```

In [ ]:
```
profile.dropna(inplace=True)
```

In [ ]:
```
profile.rename(columns={'id' : 'customer_id'}, inplace = True)
```

In [ ]:
```
profile['became_member_on'] = profile['became_member_on'].apply(lambda x: datetime.datetime.strptime(str(x), "%Y%m%d")
                                                                .date())
```

In [ ]:
```
end_date = pd.Timestamp('2018-08-01')
end_date
```

In [ ]:
```
end_date = pd.to_datetime(end_date)
end_date
```

In [ ]:
```
profile['days_as_member'] = np.abs((pd.to_datetime(profile['became_member_on']) - end_date).dt.days)
```

In [ ]:
```
age_range_labels = ['11-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80+']
age_range_bins = [11, 20, 30, 40, 50, 60, 70, 80, 110]
pd.cut(profile['age'], bins=age_range_bins, labels = age_range_labels, right=False)
```

In [ ]:
```
profile['age_range'] = pd.cut(profile['age'], bins=age_range_bins, labels = age_range_labels, right=False)
```

In [ ]:
```
pd.Categorical(profile['age_range'], ordered = True, categories = age_range_labels)
```

In [ ]:
```
profile['age_range'] = pd.Categorical(profile['age_range'], ordered = True, categories = age_range_labels)
```

Transcript Dataset Cleaning

In [ ]:
```
transcript['event'].value_counts()
```

In [ ]:
```
transcript.rename(columns={'person' : 'customer_id'}, inplace=True)
```

In [ ]:
```
pd.merge(left = transcript, right = profile, how = 'inner', left_on = 'customer_id', right_on = 'customer_id')
```

In [ ]:
```
transcript = pd.merge(left = transcript, right = profile, how = 'inner', left_on = 'customer_id', right_on = 'customer_id')
transcript.head()
```

In [ ]:
```
transcript = transcript.drop(columns = {'age', 'became_member_on', 'gender', 'income', 'days_as_member', 'age_range'})
transcript
```

In [ ]:
```
event_dummies = pd.get_dummies(transcript['event'])
event_dummies
```

In [ ]:
```
event_dummies.rename(columns = {'offer completed' : 'offer_completed', 'offer received' : 'offer_received',
                                'offer viewed' : 'offer_viewed'}, inplace=True)
event_dummies
```

In [ ]:
```
transcript = pd.concat([transcript, event_dummies], axis = 1)
transcript
```

In [ ]:
```
transcript['days'] = transcript['time']/24
```

In [ ]:

In [ ]:
```
transcript['offer-or-amount'] = transcript['value'].apply(lambda x: list(x.keys())[0])
transcript
```

In [ ]:
```
transcript['value'].apply(lambda x: list(x.values())[0])
```

In [ ]:
```
transcript['id-or-amount'] = transcript['value'].apply(lambda x: list(x.values())[0])
transcript
```

In [ ]:
```
trans_offer_dict = {'offer id': 'offer_id'}
transcript['offer-or-amount'].replace(trans_offer_dict, inplace = True)
```

In [ ]:
```
transcript[transcript['offer-or-amount'] == 'offer_id']
```

In [ ]:
```
transcript_offer = transcript[transcript['offer-or-amount'] == 'offer_id']
```

In [ ]:
```
transcript_offer.rename(columns={'id-or-amount' : 'offer_id'}, inplace=True)
```

In [ ]:
```
transcript_offer.drop(columns = ['event', 'value', 'offer-or-amount', 'transaction'], axis = 1, inplace = True)
```

In [ ]:
```
transcript[transcript['offer-or-amount'] == 'amount']
```

In [ ]:
```
transcript_amount = transcript[transcript['offer-or-amount'] == 'amount']
```

In [ ]:
```
transcript_amount.rename(columns = {'id-or-amount': 'amount'}, inplace = True)
```

In [ ]:
```
transcript_amount.drop(columns = ['event', 'value', 'offer-or-amount', 'offer_completed',
                                  'offer_received', 'offer_viewed'], inplace = True)
```

Exploratory Data Analysis

General Age Ranges of our Customer ?

In [ ]:
```
profile['age_range'].value_counts().sort_index()
```

In [ ]:
```
df_temp = profile['age_range'].value_counts().sort_index()
df_temp = df_temp.reset_index()
df_temp.rename(columns={'index': 'age_range', 'age_range' : 'no_of_cust'}, inplace = True)
df_temp
```

In [ ]:
```
df_temp.to_csv('./Dashboard/profile_1.csv', index = False)
```

In [ ]:
```
import plotly.express as px

fig = px.bar(df_temp, x='age_range', y='no_of_cust',
             color = 'age_range',
             labels={'no_of_cust':'Number of Customers', 'age_range' : 'Age Range'},
             height=600,
             title='Age Distribution of Starbucks Customers')
fig.show()
```

What are the salary ranges of people across different age groups?

In [ ]:
```
profile.to_csv('./Dashboard/profile_2.csv', index = False)
```

In [ ]:
```
import plotly.express as px

fig = px.violin(profile, y="income", x="age_range", color="age_range",
                box=True, title='Starbucks Income Distribution across age Ranges',
                labels={'age_range' : 'Age Ranges', 'income' : 'Income'})
#fig = px.violin(profile, y="income", x="age_range", color="age_range", box=True, points='all')
fig.show()
```

In [ ]:
```
import plotly.express as px

fig = px.scatter(profile, x="age", y="income", color='age',
                 title='Starbucks Customers - Income Distribution Across Individual Ages',
                 labels={'age' : 'Age', 'income' : 'Income'})
fig.show()
```

What does the correlation between number of days an offer has been open vs. final transaction amount have to tell us?

In [ ]:
```
transcript_amount.to_csv('./Dashboard/transcript_amount.csv', index = False)
```

In [ ]:
```
import plotly.express as px

fig = px.scatter(transcript_amount, x="days", y="amount",
                 color='days', title='Days Reward Open vs. Final Amount Spent',
                 labels={'days' : 'Days', 'amount' : 'Amount'})
fig.show()
```

In [ ]:
```
import plotly.express as px

fig = px.scatter(transcript_amount, x="days", y="amount",
                 color='days', title='Days Reward Open vs. Final Amount Spent' ,
                 labels={'days' : 'Days', 'amount' : 'Amount'}, range_y=(0,50))
fig.show()
```

Do gender distributions have any major effect on our data here?

In [ ]:
```
profile['age_range'].sort_index()
```

In [ ]:
```
import plotly.express as px
fig = px.histogram(profile.sort_values(by='age_range'), x='age_range',
             color='gender', barmode='group',
             height=600, title='Distribution of Genders Across Age Ranges',
             labels={'age_range' : 'Age Ranges', 'count' : 'Number of Peoples'})
fig.show()
```

In [ ]:
```
plt.figure(figsize=(15,10))
sb.barplot(data = profile, x = 'age_range', y = 'income', hue = 'gender', palette='cool_r')
plt.title('Distribution of Income Across Genders and Ages')
plt.xlabel('Age Ranges')
plt.ylabel('Income')
plt.style.use('seaborn')
```

Analysis of gender across our customer data

In [ ]:
```
customer_id_1 = '0610b486422d4921ae7d2bf64640c50b'
customer_attributes = profile[profile['customer_id'] == customer_id_1]
customer_attributes
```

In [ ]:
```
age = customer_attributes['age'].values[0]
age
```

In [ ]:
```
became_member_on = customer_attributes['became_member_on'].values[0]
became_member_on
```

In [ ]:
```
gender = customer_attributes['gender'].values[0]
gender
```

In [ ]:
```
income = customer_attributes['income'].values[0]
income
```

In [ ]:
```
days_as_member = customer_attributes['days_as_member'].values[0]
days_as_member
```

In [ ]:
```
age_range = customer_attributes['age_range'].values[0]
age_range
```

In [ ]:
```
customer_transactions = pd.DataFrame()
num_discounts = 0
num_bogos = 0
```

In [ ]:
```
transcript_offer[transcript_offer['customer_id'] == customer_id_1]
```

In [ ]:
```
total_viewed = transcript_offer[transcript_offer['customer_id'] == customer_id_1]['offer_viewed'].sum()
total_viewed
```

In [ ]:
```
total_received = transcript_offer[transcript_offer['customer_id'] == customer_id_1]['offer_received'].sum()
total_received
```

In [ ]:
```
total_completed = transcript_offer[transcript_offer['customer_id'] == customer_id_1]['offer_completed'].sum()
total_completed
```

In [ ]:
```
total_spent = transcript_amount[transcript_amount['customer_id'] == customer_id_1]['amount'].sum()
total_spent
```

In [ ]:
```
avg_spent = transcript_amount[transcript_amount['customer_id'] == customer_id_1]['amount'].mean()
avg_spent
```

In [ ]:
```
num_transactions = transcript_amount[transcript_amount['customer_id'] == customer_id_1]['transaction'].sum()
num_transactions
```

In [ ]:
```
if total_viewed > 0:
    percent_completed = total_completed / total_received
else:
    percent_completed = 0
```

In [ ]:
```
offer_completed_df = transcript_offer[transcript_offer['customer_id'] == customer_id_1]
offer_completed_df
```

In [ ]:
```
completed_offer_list = offer_completed_df['offer_id'].tolist()
completed_offer_list
```

In [ ]:
```
completed_bogo = portfolio[portfolio['offer_id'].isin(completed_offer_list)]['offer_bogo'].sum()
completed_bogo
```

In [ ]:
```
portfolio[portfolio['offer_id'].isin(completed_offer_list)]
```

In [ ]:
```
portfolio[portfolio['offer_id'].isin(completed_offer_list)]
```

In [ ]:
```
completed_discount = portfolio[portfolio['offer_id'].isin(completed_offer_list)]['offer_discount'].sum()
completed_discount
```

In [ ]:
```
offers_received = transcript_offer[transcript_offer['customer_id'] == customer_id_1]['offer_id'].values.tolist()
offers_received
```

In [ ]:
```
for x in offers_received:
    #Iterating through all the bogo offers
    if x in ['ae264e3637204a6fb9bb56bc8210ddfd', '4d5c57ea9a6940dd891ad53e9dbe8da0',
             '9b98b8c7a33c4b65b9aebfe6a799e6d9', 'f19421c1d4aa40978ebb69ca19b0e20d']:
        num_bogos += 1
    #Iterating through all the discount offers
    elif x in ['0b1e1539f2cc45b7b9fa7c272da2e1d7', '2298d6c36e964ae4a3e7e9706d1fb8c2',
               'fafdcd668e3743c1bb461111dcafc2a4', '2906b810c7d4411798c6938adc9daaa5']:
        num_discounts += 1
```

In [ ]:
```
if num_bogos > 0:
    bogo_percent_completed = completed_bogo / num_bogos
else:
    bogo_perecent_completed = 0.0
```

In [ ]:
```
if num_discounts > 0:
    discount_percent_completed = completed_discount / num_discounts
else:
    discount_percent_completed = 0.0
```

In [ ]:
```
person_entry_temp = {'customer_id': customer_id_1,
                     'age': age,
                     'age_range': age_range,
                     'gender': gender,
                     'income': income,
                     'became_member_on': became_member_on,
                     'days_as_member': days_as_member,
                     'total_completed': total_completed,
                     'total_viewed': total_viewed,
                     'total_received': total_received,
                     'percent_completed': percent_completed,
                     'total_spent': total_spent,
                     'avg_spent': avg_spent,
                     'num_transactions': num_transactions,
                     'completed_bogo': completed_bogo,
                     'num_bogos': num_bogos,
                     'bogo_percent_completed': bogo_percent_completed,
                     'completed_discount': completed_discount,
                     'num_discounts': num_discounts,
                     'discount_percent_completed': discount_percent_completed}
```

In [ ]:
```
#Creating a function that will build our master DataFrame_customer transactions
#Special thanks to this post for assisting how to append DataFrames: https://pandas.pydata.org/pandas-docs/stable/reference

def generate_transactions(customer_ids, portfolio = portfolio, profile = profile,
                          transcript_offer = transcript_offer, transcript_amount = transcript_amount):

    #Initializing a list to append our individual customer dictionaries to
    person_entries = []

    for customer_id in customer_ids:
        #Pulling customer attributes from the 'profile' DataFrame
        customer_attributes = profile[profile['customer_id'] == customer_id]
        age = customer_attributes['age'].values[0]
        became_member_on = customer_attributes['became_member_on'].values[0]
        gender = customer_attributes['gender'].values[0]
        income = customer_attributes['income'].values[0]
        days_as_member = customer_attributes['days_as_member'].values[0]
        age_range = customer_attributes['age_range'].values[0]

        #Initializing variables we will use later on down
        customer_transactions = pd.DataFrame()
        num_discounts = 0
        num_bogos = 0
```

```
    #Establishing new features from respective transcript_offer and transcript_amount DataFrames
    total_viewed = transcript_offer[transcript_offer['customer_id'] == customer_id]['offer_viewed'].sum()
    total_received = transcript_offer[transcript_offer['customer_id'] == customer_id]['offer_received'].sum()
    total_completed = transcript_offer[transcript_offer['customer_id'] == customer_id]['offer_completed'].sum()
    total_spent = transcript_amount[transcript_amount['customer_id'] == customer_id]['amount'].sum()
    avg_spent = transcript_amount[transcript_amount['customer_id'] == customer_id]['amount'].mean()
    num_transactions = transcript_amount[transcript_amount['customer_id'] == customer_id]['transaction'].sum()

    #Determining percentage of completed bogo and discount offers
    if total_viewed > 0:
        percent_completed = total_completed / total_received
    else:
        percent_completed = 0

    #Determining the completed offers in order to determine the completed bogos and completed discounts
    offer_completed_df = transcript_offer[transcript_offer['customer_id'] == customer_id]
    completed_offer_list = offer_completed_df['offer_id'].tolist()
    completed_bogo = portfolio[portfolio['offer_id'].isin(completed_offer_list)]['offer_bogo'].sum()
    completed_discount = portfolio[portfolio['offer_id'].isin(completed_offer_list)]['offer_discount'].sum()

    #Determining all the offers received by an individual customer
    offers_received = transcript_offer[transcript_offer['customer_id'] == customer_id]['offer_id'].values.tolist()

    #Determining whether or not offer is a bogo or discount offer and incrementing appropriate variables
    for x in offers_received:
        #Iterating through all the bogo offers
        if x in ['ae264e3637204a6fb9bb56bc8210ddfd', '4d5c57ea9a6940dd891ad53e9dbe8da0',
                 '9b98b8c7a33c4b65b9aebfe6a799e6d9', 'f19421c1d4aa40978ebb69ca19b0e20d']:
            num_bogos += 1
        #Iterating through all the discount offers
        elif x in ['0b1e1539f2cc45b7b9fa7c272da2e1d7', '2298d6c36e964ae4a3e7e9706d1fb8c2',
                   'fafdcd668e3743c1bb461111dcafc2a4', '2906b810c7d4411798c6938adc9daaa5']:
            num_discounts += 1

    #Determining the percentage of bogo & discount offers actually completed versus how many of each were offered
    if num_bogos > 0:
        bogo_percent_completed = completed_bogo / num_bogos
    else:
        bogo_perecent_completed = 0.0

    if num_discounts > 0:
        discount_percent_completed = completed_discount / num_discounts
    else:
        discount_percent_completed = 0.0

    #Adding all features to a unified dictionary, 'person_entry'
    person_entry = {'customer_id': customer_id,
                    'age': age,
                    'age_range': age_range,
                    'gender': gender,
                    'income': income,
                    'became_member_on': became_member_on,
                    'days_as_member': days_as_member,
                    'total_completed': total_completed,
                    'total_viewed': total_viewed,
                    'total_received': total_received,
                    'percent_completed': percent_completed,
                    'total_spent': total_spent,
                    'avg_spent': avg_spent,
                    'num_transactions': num_transactions,
                    'completed_bogo': completed_bogo,
                    'num_bogos': num_bogos,
                    'bogo_percent_completed': bogo_percent_completed,
                    'completed_discount': completed_discount,
                    'num_discounts': num_discounts,
                    'discount_percent_completed': discount_percent_completed}

    #Appending person_entry to master customer_transactions DataFrame
    person_entries.append(person_entry)

#Building our final DataFrame from the person_entries list
customer_transactions = pd.DataFrame(person_entries)

return customer_transactions
```

In [ ]:
```
#Generating a list of customer ids to iterate through
customer_ids = profile['customer_id'].to_list()

#Iterate through customer IDs to form master customer_transactions DataFrame
customer_transactions = generate_transactions(customer_ids)
customer_transactions.head()
```

In [ ]:
```
customer_transactions.to_csv('customer_transactions.csv', index = False)
```

In [ ]:
```
customer_transactions = pd.read_csv('customer_transactions.csv')
```

In [ ]:
```
customer_transactions['Label'] = df_cluster['labels']
```

In [ ]:
```
customer_transactions.to_csv('./Dashboard/customer_transactions.csv')
```

In [ ]:
```
customer_transactions.rename(columns={'index':'labels'})
```

In [ ]:
```
labels = list(df_cluster['labels'])
```

In [ ]:
```
labels = pd.DataFrame(labels)
labels.rename(columns={0:'labels'}, inplace=True)
labels
```

In [ ]:
```
cols_to_drop = ['customer_id', 'became_member_on', 'age_range']
customer_transactions.drop(columns = cols_to_drop, inplace = True)
customer_transactions
```

In [ ]:
```
gender_dict = {'M': 0, 'F': 1, 'O': 2}
customer_transactions['gender'].replace(gender_dict, inplace = True)
customer_transactions
```

In [ ]:
```
#Filling in avg_amount null values with Imputer
imputer = SimpleImputer(strategy = 'median')
cust_trans_imputed = pd.DataFrame(imputer.fit_transform(customer_transactions))
cust_trans_imputed
```

In [ ]:
```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(cust_trans_imputed)
scaled_data
```

In [ ]:
```
column_names = customer_transactions.columns.values.tolist()
customer_transactions_master = pd.DataFrame(scaled_data, columns = column_names)
customer_transactions_master.head()
```

In [ ]:
```
customer_transactions_master.to_csv('customer_transactions_scaled.csv', index = False)
```

In [ ]:
```
#Using the single_link_model, we'll append the results as a new column to our master
customer_transactions_master_1 = customer_transactions_master
single_link_model = AgglomerativeClustering(n_clusters = 6, linkage = 'single')
customer_transactions_master_1['single_link_cluster'] = single_link_model.fit_predict(customer_transactions_master)
customer_transactions_master_1
```

In [ ]:
```
customer_transactions_master_1['single_link_cluster'].value_counts()
```

Number of Clusters using K-Means

In [ ]:
```
customer_transactions_master
```

In [ ]:
```
#Building a function to quickly iterate through various KMeans models that takes in data and number of cluster as inputs
#Returns the appropriate k_means score
def kmeans_score(data, n_clusters):
    kmeans = KMeans(n_clusters = n_clusters)
    model = kmeans.fit(data)
    score = np.abs(model.score(data))
    return score
```

In [ ]:
```
#Running the kmeans_score function for a number of different cluster patterns
scores = []
for i in range(1, 20):
    score = kmeans_score(customer_transactions_master, i)
    print("For n_clusters = ", i,"The K-Means Score is :", score)
    scores.append(score)
```

In [ ]:
```
k_means_result = pd.DataFrame(scores, centers).reset_index()
```

In [ ]:
```
k_means_result = k_means_result.rename(columns={'index':'no_of_clusters', 0:'sum_of_squared_error'})
```

In [ ]:
```
k_means_result.to_csv('./Dashboard/K_means_result.csv')
```

In [ ]:
```
#Plotting results from the previous cell; noting that we begin seeing after about 6 clusters
centers = range(1, 20)
plt.plot(centers, scores, linestyle = '--', marker = 'o');
plt.style.use('seaborn');
plt.title('KMeans SSE Scores as K Increases');
plt.ylabel('SSE');
plt.xlabel('Number of Clusters (K)');
```

In [ ]:
```
import plotly.graph_objects as go
import numpy as np

fig = go.Figure(data=go.Scatter(x=list(centers), y=scores))
fig.update_layout(title='KMeans SSE Scores as K Increases',
                  xaxis_title='SSE',
                  yaxis_title='Number of Clusters (K)')
fig.show()
```

In [ ]:
```
km = KMeans(n_clusters=5)
y_predicted = km.fit_predict(customer_transactions_master)
y_predicted
```

In [ ]:
```
customer_transactions_master_1['cluster_1']=y_predicted
customer_transactions_master_1.head()
```

In [ ]:
```
km.cluster_centers_[0,0]
```

In [ ]:
```
customer_transactions_master_1.to_csv('./Dashboard/cust_tran_master_1.csv')
```

In [ ]:
```
df1 = customer_transactions_master_1[customer_transactions_master_1.cluster_1==0]
df2 = customer_transactions_master_1[customer_transactions_master_1.cluster_1==1]
df3 = customer_transactions_master_1[customer_transactions_master_1.cluster_1==2]
df4 = customer_transactions_master_1[customer_transactions_master_1.cluster_1==3]
df5 = customer_transactions_master_1[customer_transactions_master_1.cluster_1==4]
plt.scatter(df1.age,df1['income'],color='green')
plt.scatter(km.cluster_centers_[0,0],km.cluster_centers_[0,1],color='purple',marker='*',label='centroid')
plt.legend()
```

In [ ]:
```
plt.scatter(df2.age,df2['income'],color='red')
plt.scatter(km.cluster_centers_[1,0],km.cluster_centers_[1,1],color='purple',marker='*',label='centroid')
```

In [ ]:
```
plt.scatter(df3.age,df3['income'],color='black')
plt.scatter(km.cluster_centers_[2,0],km.cluster_centers_[2,1],color='purple',marker='*',label='centroid')
```

In [ ]:
```
plt.scatter(df4.age,df4['income'],color='blue')
plt.scatter(km.cluster_centers_[3,0],km.cluster_centers_[3,1],color='purple',marker='*',label='centroid')
```

In [ ]:
```
plt.scatter(df5.age,df5['income'],color='orange')
plt.scatter(km.cluster_centers_[4,0],km.cluster_centers_[4,1],color='purple',marker='*',label='centroid')
```

Between Age and Gender

In [ ]:
```
scores = []
for i in range(1, 20):
    score = kmeans_score(customer_transactions_master[['age', 'gender']], i)
    print("For n_clusters = ", i,"The K-Means Score is :", score)
    scores.append(score)
```

In [ ]:
```
value = range(1,20)
k_means_result_2 = pd.DataFrame(scores, value).reset_index()
```

In [ ]:
```
k_means_result_2.rename(columns={'index':'Number_of_Clusters', 0:'K_Means_Score'}, inplace=True)
```

In [ ]:
```
k_means_result_2.to_csv('./Dashboard/K_means_result_2.csv')
```

In [ ]:
```
import plotly.graph_objects as go
import numpy as np

fig = go.Figure(data=go.Scatter(x=list(centers), y=scores))
fig.update_layout(title='KMeans SSE Scores as K Increases',
                  xaxis_title='SSE',
                  yaxis_title='Number of Clusters (K)')
fig.show()
```

Between Age and Income

In [ ]:
```
scores = []
for i in range(1, 20):
    score = kmeans_score(customer_transactions_master[['age', 'income']], i)
    print("For n_clusters = ", i,"The K-Means Score is :", score)
    scores.append(score)
```

In [ ]:
```
value = range(1,20)
k_means_result_3 = pd.DataFrame(scores, value).reset_index()
```

In [ ]:
```
k_means_result_3.rename(columns={'index':'Number_of_Clusters', 0:'K_Means_Score'}, inplace=True)
```

In [ ]:
```
k_means_result_3.to_csv('./Dashboard/K_means_result_3.csv')
```

In [ ]:
```
import plotly.graph_objects as go
import numpy as np

fig = go.Figure(data=go.Scatter(x=list(centers), y=scores))
fig.update_layout(title='KMeans SSE Scores as K Increases',
                  xaxis_title='SSE',
                  yaxis_title='Number of Clusters (K)')
fig.show()
```

Between Gender and Income

In [ ]:
```
scores = []
for i in range(1, 20):
    score = kmeans_score(customer_transactions_master[['gender', 'income']], i)
    print("For n_clusters = ", i,"The K-Means Score is :", score)
    scores.append(score)
```

In [ ]:
```
value = range(1,20)
k_means_result_4 = pd.DataFrame(scores, value).reset_index()
```

In [ ]:
```
k_means_result_4.rename(columns={'index':'Number_of_Clusters', 0:'K_Means_Score'}, inplace=True)
```

In [ ]:
```
k_means_result_4.to_csv('./Dashboard/K_means_result_4.csv')
```

In [ ]:
```
import plotly.graph_objects as go
import numpy as np

fig = go.Figure(data=go.Scatter(x=list(centers), y=scores))
fig.update_layout(title='KMeans SSE Scores as K Increases',
                  xaxis_title='SSE',
                  yaxis_title='Number of Clusters (K)')
fig.show()
```

In [ ]:
```
km = KMeans(n_clusters=5)
y_predicted = km.fit_predict(customer_transactions_master[['gender', 'income']])
y_predicted
```

In [ ]:
```
customer_transactions_master_1['cluster_2']=y_predicted
customer_transactions_master_1.head()
```

In [ ]:
```
df1 = customer_transactions_master_1[customer_transactions_master_1.cluster_2==0]
df2 = customer_transactions_master_1[customer_transactions_master_1.cluster_2==1]
df3 = customer_transactions_master_1[customer_transactions_master_1.cluster_2==2]
df4 = customer_transactions_master_1[customer_transactions_master_1.cluster_2==3]
df5 = customer_transactions_master_1[customer_transactions_master_1.cluster_2==4]
plt.scatter(df1.gender,df1['income'],color='green')
plt.scatter(df2.gender,df2['income'],color='black')
plt.scatter(df3.gender,df3['income'],color='blue')
plt.scatter(df4.gender,df4['income'],color='orange')
plt.scatter(df5.gender,df5['income'],color='red')
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color='purple',marker='*',label='centroid')
plt.legend()
```

DBSCAN

In [ ]:
```
from sklearn.cluster import DBSCAN
```

In [ ]:
```
dbscan =  DBSCAN(eps=35, min_samples=10)
labels = dbscan.fit_predict(customer_transactions_master[['age', 'gender']])
```

In [ ]:
```
np.unique(labels)
```

In [ ]:
```
from sklearn.manifold import TSNE

plot_kwds = {'alpha' : 0.25, 's' : 80, 'linewidths':0}

projection = TSNE().fit_transform(customer_transactions_master)
plt.scatter(*projection.T, **plot_kwds)
```

In [ ]:
```
import hdbscan
import seaborn as sns
```

In [ ]:
```
clusterer = hdbscan.HDBSCAN(min_cluster_size=890, min_samples=10).fit(customer_transactions_master)
```

In [ ]:
```
clusterer = hdbscan.HDBSCAN(min_cluster_size=890, min_samples=10).fit(customer_transactions_master)

plt.figure(figsize=(8, 8), dpi=80)
color_palette = sns.color_palette('Paired', 12)
cluster_colors = [color_palette[x] if x >= 0
                  else (0.5, 0.5, 0.5)
                  for x in clusterer.labels_]
cluster_member_colors = [sns.desaturate(x, p) for x, p in
                         zip(cluster_colors, clusterer.probabilities_)]
plt.scatter(*projection.T, s=50, linewidth=0, c=cluster_member_colors, alpha=0.25)
```

In [ ]:
```
len(cluster_member_colors)
```

In [ ]:
```
np.unique(clusterer.labels_)
```

In [ ]:
```
clusterer.labels_.size
```

In [ ]:
```
import plotly.express as px
fig = px.scatter(*projection.T, color=clusterer.labels_)
fig.show()
```

In [ ]:
```
label_normal = clusterer.labels_
label_normal = pd.DataFrame({"label": clusterer.labels_})

plt.figure(figsize=(8, 6))
ax = sns.countplot(x='label', data=label_normal)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

In [ ]:
```
label_normal = label_normal.groupby(by = ['label']).size().reset_index(name = 'count')
```

In [ ]:
```
import plotly.express as px

fig = px.bar(label_normal, x='label', y='count',
             color='label',
             height=600)
fig.show()
```

In [ ]:
```
# Save dataset with each cluster
df_cluster = customer_transactions_master.copy()
df_cluster['labels'] = clusterer.labels_
df_cluster
```

In [ ]:
```
df_cluster.to_csv('final_segmentation.csv')
```

In [ ]:
```
df_cluster = pd.read_csv('final_segmentation.csv')
```

In [ ]:
```
import seaborn as sns
personal_details = ['gender', 'income', 'age', 'labels']
sns.pairplot(df_cluster[personal_details], hue='labels', kind='reg');
```

In [ ]:
```
fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (30, 18))

#Visualizing clusters across age ranges
sns.countplot(data = df_cluster, x = 'age', hue = 'labels', ax = axes[0,0]);
axes[0,0].set_title('Distribution of Clusters Across Age Ranges');
axes[0,0].set_xlabel('Age');
axes[0,0].set_ylabel('Number of People');

#Visualizing clusters across incomes and age ranges
sns.scatterplot(data = df_cluster, x = 'age', y = 'income', hue = 'labels',
```

In [ ]:
```
fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (30, 18))

#Visualizing clusters across age ranges
sns.countplot(data = df_cluster, x = 'age', hue = 'labels', ax = axes[0,0]);
axes[0,0].set_title('Distribution of Clusters Across Age Ranges');
axes[0,0].set_xlabel('Age');
axes[0,0].set_ylabel('Number of People');

#Visualizing clusters across incomes and age ranges
sns.scatterplot(data = df_cluster, x = 'age', y = 'income', hue = 'labels',
                palette = sns.color_palette('muted', n_colors = 4), ax = axes[0,1]);
axes[0,1].set_title('Distribution of Clusters Across Income and Age');
axes[0,1].set_xlabel('Age');
axes[0,1].set_ylabel('Income');

#Visualizing clusters across age ranges
sns.countplot(data = df_cluster, x = 'gender', hue = 'labels', ax = axes[1,0]);
axes[1,0].set_title('Distribution of Clusters Across Genders');
axes[1,0].set_xlabel('Genders');
axes[1,0].set_ylabel('Number of People');
plt.sca(axes[1,0]);
plt.xticks(range(3), ['Male', 'Female', 'Other']);
```

In [ ]:
```
personal_details = ['gender', 'income', 'age', 'days_as_member', 'labels']
sb.pairplot(df_cluster[personal_details], hue = 'labels', kind = 'reg');
```

In [ ]:
```
cust_transaction_2 = pd.read_csv('customer_transactions.csv')
cust_transaction_2
```

In [ ]:
```
cust_transaction_2['labels'] = df_cluster['labels']
```

In [ ]:
```
fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (20, 13))

#Visualizing clusters across age ranges
sb.countplot(data = cust_transaction_2, x = 'age_range', hue = 'labels', ax = axes[0,0]);
axes[0,0].set_title('Distribution of Clusters Across Age Ranges');
axes[0,0].set_xlabel('Age Ranges');
axes[0,0].set_ylabel('Number of People');

#Visualizing clusters across incomes and ages
sb.scatterplot(data = cust_transaction_2, x = 'age', y = 'income', hue = 'labels',
               palette = sb.color_palette('muted', n_colors = 5), ax = axes[0,1]);
axes[0,1].set_title('Distribution of Clusters Across Income and Age');
axes[0,1].set_xlabel('Age');
axes[0,1].set_ylabel('Income');

#Visualizing how long members have been members in a stacked histogram
days_as_member_0 = cust_transaction_2[cust_transaction_2['labels'] == -1]['days_as_member']
days_as_member_1 = cust_transaction_2[cust_transaction_2['labels'] == 0]['days_as_member']
days_as_member_2 = cust_transaction_2[cust_transaction_2['labels'] == 1]['days_as_member']
days_as_member_3 = cust_transaction_2[cust_transaction_2['labels'] == 2]['days_as_member']
days_as_member_4 = cust_transaction_2[cust_transaction_2['labels'] == 3]['days_as_member']

axes[1,0].hist(days_as_member_0, stacked = True);
axes[1,0].hist(days_as_member_1, stacked = True);
axes[1,0].hist(days_as_member_2, stacked = True);
axes[1,0].hist(days_as_member_3, stacked = True);
axes[1,0].hist(days_as_member_4, stacked = True);
axes[1,0].legend(title = 'clusters', labels = ['cluster 0', 'cluster 1', 'cluster 2', 'cluster 3', 'cluster 4']);
axes[1,0].set_title('Number of Days as Member Across the Clusters');
axes[1,0].set_xlabel('Number of Days as Member');
axes[1,0].set_ylabel('Number of People');

#Visualizing clusters across age ranges
sb.countplot(data = cust_transaction_2, x = 'gender', hue = 'labels', ax = axes[1,1]);
axes[1,1].set_title('Distribution of Clusters Across Genders');
axes[1,1].set_xlabel('Genders');
axes[1,1].set_ylabel('Number of People');
plt.sca(axes[1,1]);
plt.xticks(range(3), ['Male', 'Female', 'Other']);
```

In [ ]:
```
behavioral_attributes = ['labels', 'avg_spent', 'bogo_percent_completed', 'discount_percent_completed', 'total_received']
sb.pairplot(cust_transaction_2[behavioral_attributes], hue = 'labels', kind = 'reg');
```

In [ ]:

```
#Visualizing the behavioral attributes in more detail
fig, axes = plt.subplots(nrows = 3, ncols = 2, figsize = (20, 18))

#Visualizing the average amount spent by customers in each respective cluster
avg_spent_0 = cust_transaction_2[cust_transaction_2['labels'] == -1]['avg_spent']
avg_spent_1 = cust_transaction_2[cust_transaction_2['labels'] == 0]['avg_spent']
avg_spent_2 = cust_transaction_2[cust_transaction_2['labels'] == 1]['avg_spent']
avg_spent_3 = cust_transaction_2[cust_transaction_2['labels'] == 2]['avg_spent']
avg_spent_4 = cust_transaction_2[cust_transaction_2['labels'] == 3]['avg_spent']
avg_spent_bins = np.arange(min(cust_transaction_2['avg_spent']), max(cust_transaction_2['avg_spent'] + 2.5), 2.5)
axes[0,0].hist(avg_spent_0, bins = avg_spent_bins, stacked = True);
axes[0,0].hist(avg_spent_1, bins = avg_spent_bins, stacked = True);
axes[0,0].hist(avg_spent_2, bins = avg_spent_bins, stacked = True);
axes[0,0].hist(avg_spent_3, bins = avg_spent_bins, stacked = True);
axes[0,0].hist(avg_spent_4, bins = avg_spent_bins, stacked = True);
axes[0,0].set_xlim([0, 40]);
axes[0,0].legend(title = 'labels', labels = ['cluster 0', 'cluster 1', 'cluster 2', 'cluster 3', 'cluster 4'],
                 frameon = True);
axes[0,0].set_title('Distribution of Average Amount Spent by Each Customer in Each Cluster');
axes[0,0].set_xlabel('Amount Spent');
axes[0,0].set_ylabel('Number of People');

#Visualizing the total number of offers sent to the customers
sb.countplot(data = cust_transaction_2, x = 'total_received', hue = 'labels', ax = axes[0,1]);
axes[0,1].set_title('Total Number of Offers Sent to Customers in Each Cluster');
axes[0,1].set_xlabel('Number of Total Offers Sent');
axes[0,1].set_ylabel('Number of People');

#Visualizing the number of BOGO offers sent to the customers
sb.countplot(data = cust_transaction_2, x = 'num_bogos', hue = 'labels', ax = axes[1,0]);
axes[1,0].set_title('Number of BOGO Offers Sent to Customers in Each Cluster');
axes[1,0].set_xlabel('Number of BOGO Offers Sent');
axes[1,0].set_ylabel('Number of People');
axes[1,0].set_xlim([0, 6.5]);

#Visualizing the number of discount offers sent to the customers
sb.countplot(data = cust_transaction_2, x = 'num_discounts', hue = 'labels', ax = axes[1,1]);
axes[1,1].set_title('Number of Discount Offers Sent to Customers in Each Cluster');
axes[1,1].set_xlabel('Number of Discount Offers Sent');
axes[1,1].set_ylabel('Number of People');
axes[1,1].set_xlim([0, 6.5]);

#Visualizing the mean of the how often customers completed bogo offers in each cluster
bogo_pct_completed_0 = cust_transaction_2[cust_transaction_2['labels'] == -1]['bogo_percent_completed'].mean()
bogo_pct_completed_1 = cust_transaction_2[cust_transaction_2['labels'] == 0]['bogo_percent_completed'].mean()
#Visualizing the number of discount offers sent to the customers
sb.countplot(data = cust_transaction_2, x = 'num_discounts', hue = 'labels', ax = axes[1,1]);
axes[1,1].set_title('Number of Discount Offers Sent to Customers in Each Cluster');
axes[1,1].set_xlabel('Number of Discount Offers Sent');
axes[1,1].set_ylabel('Number of People');
axes[1,1].set_xlim([0, 6.5]);

#Visualizing the mean of the how often customers completed bogo offers in each cluster
bogo_pct_completed_0 = cust_transaction_2[cust_transaction_2['labels'] == -1]['bogo_percent_completed'].mean()
bogo_pct_completed_1 = cust_transaction_2[cust_transaction_2['labels'] == 0]['bogo_percent_completed'].mean()
bogo_pct_completed_2 = cust_transaction_2[cust_transaction_2['labels'] == 1]['bogo_percent_completed'].mean()
bogo_pct_completed_3 = cust_transaction_2[cust_transaction_2['labels'] == 2]['bogo_percent_completed'].mean()
bogo_pct_completed_4 = cust_transaction_2[cust_transaction_2['labels'] == 3]['bogo_percent_completed'].mean()
bogo_pct_completed_data = [bogo_pct_completed_0, bogo_pct_completed_1,
                           bogo_pct_completed_2, bogo_pct_completed_3, bogo_pct_completed_4]
y_pos = np.arange(len(bogo_pct_completed_data))
bogo_tick_labels = ['cluster 0', 'cluster 1', 'cluster 2', 'cluster 3', 'cluster 4']
axes[2,0].barh(y_pos, bogo_pct_completed_data, tick_label = bogo_tick_labels);
axes[2,0].set_xlim([0, 1]);
axes[2,0].set_title('Average Times Each Cluster Redeemed a BOGO Offer');
axes[2,0].set_xlabel('Percentage of Success');
axes[2,0].set_ylabel('Clusters');

#Visualizing the mean of the how often customers completed discount offers in each cluster
discount_pct_completed_0 = cust_transaction_2[cust_transaction_2['labels'] == -1]['discount_percent_completed'].mean()
discount_pct_completed_1 = cust_transaction_2[cust_transaction_2['labels'] == 0]['discount_percent_completed'].mean()
discount_pct_completed_2 = cust_transaction_2[cust_transaction_2['labels'] == 1]['discount_percent_completed'].mean()
discount_pct_completed_3 = cust_transaction_2[cust_transaction_2['labels'] == 2]['discount_percent_completed'].mean()
discount_pct_completed_4 = cust_transaction_2[cust_transaction_2['labels'] == 3]['discount_percent_completed'].mean()
discount_pct_completed_data = [discount_pct_completed_0, discount_pct_completed_1,
                               discount_pct_completed_2, discount_pct_completed_3, discount_pct_completed_4]
y_pos = np.arange(len(discount_pct_completed_data))
discount_tick_labels = ['cluster 0', 'cluster 1', 'cluster 2', 'cluster 3', 'cluster 4']
axes[2,1].barh(y_pos, discount_pct_completed_data, tick_label = discount_tick_labels);
axes[2,1].set_xlim([0, 1]);
axes[2,1].set_title('Average Times Each Cluster Redeemed a Discount Offer');
axes[2,1].set_xlabel('Percentage of Success');
axes[2,1].set_ylabel('Clusters');


#discount_percent_cmompleted
#bogo_percent_completed
```

II. SENTIMENT ANALYSIS

Importing libraries

In [1]:
```
import numpy as np
import pandas as pd
```

Importing dataset

In [ ]:
```
df = pd.read_csv('./Coffee_ratings_and_sentiments.csv', encoding = 'ISO-8859-1')
df
```

In [3]:
```
df_rev = df[['review_text', 'rating', 'num_rating', 'cat_rating']]
```

In [4]:
```
df_rev.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7621 entries, 0 to 7620
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   review_text  7616 non-null   object
 1   rating       7616 non-null   object
 2   num_rating   7616 non-null   float64
 3   cat_rating   7616 non-null   object
dtypes: float64(1), object(3)
memory usage: 238.3+ KB
```

In [5]:
```
df_new = df_rev.dropna()
```

In [ ]:
```
df_new.drop(columns=['rating'], inplace=True)
```

In [8]:
```
df_new.head()
```

Out[8]:

	review_text	num_rating	cat_rating
0	11/25/2016 1 check-in Love love loved the vib...	5.0	HIGH
1	12/2/2016 Listed in Date Night: Austin, vibe ...	4.0	HIGH
2	11/30/2016 1 check-in Listed in food seating ...	4.0	HIGH
3	11/25/2016 Very cool vibe! Good drinks Nice s...	2.0	LOW
4	12/3/2016 1 check-in They are location within...	4.0	HIGH

Data cleaning

In [ ]:
```
import re
import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
```

In [10]:
```
corpus=[]

for i in range(0, 7616):
    review = re.sub('[^a-zA-Z]', ' ', df_new['review_text'][i])
    review = review.lower()
    review = review.split()
    review = [ps.stem(word) for word in review if not word in set(all_stopwords)]
    review = ' '.join(review)
    corpus.append(review)
```

In [11]:
```
len(corpus)
```

Out[11]:
```
7616
```

In [43]:
```
corpus[:2]
```

Out[43]:
```
['check love love love vibe everi corner coffe shop style swing order matcha coffe muy fantastico order get drink pretti streamlin order ipad includ bev
erag select rang coffe alcohol desir level sweet checkout system got coffe within minut hope typic heart feather coffe found list possibl vibe may ide
a',
 'list date night austin vibe austin beauti love vibe instagram worthi definit prepar gonna cost pretti penni food food decent noth rave probabl back so
mewher uniqu nice']
```

Data transformation

In [14]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=176)
```

In [15]:
```
X = cv.fit_transform(corpus).toarray()
y = df_new.iloc[:, -2].values
```

In [16]:
```
X.shape
```

Out[16]:
```
(7616, 176)
```

In [17]:
```
y.shape
```

Out[17]:
```
(7616,)
```

In [18]:
```
# Saving BoW dictionary to later use in prediction
import pickle
bow_path = './c1_BoW_Sentiment_Model.pkl'
pickle.dump(cv, open(bow_path, "wb"))
```

Dividing dataset into training and test set

In [19]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

Model fitting (Naive Bayes)

In [20]:
```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Out[20]:
```
GaussianNB()
```

In [21]:
```
# Exporting NB Classifier to later use in prediction
import joblib
joblib.dump(classifier, './c2_Classifier_Sentiment_Model')
```

Out[21]:
```
['./c2_Classifier_Sentiment_Model']
```

Model performance

In [22]:
```
y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)

accuracy_score(y_test, y_pred)
```

Out[22]:
```
[[ 28   8   3   0  11]
 [ 36  21  13   6  27]
 [ 28  13  39  35  49]
 [ 28  30  33 139 221]
 [ 30  19  23 105 579]]
0.5288713910761155
```

In [23]:
```
accuracy_score(y_test, y_pred)
```

Out[23]:
```
0.5288713910761155
```

In [24]:
```
accuracy = {}

for feat in range(1,200):
    from sklearn.feature_extraction.text import CountVectorizer
    cv = CountVectorizer(max_features=feat)

    X = cv.fit_transform(corpus).toarray()
    y = df_new.iloc[:, -2].values

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

```
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

```
    acc = accuracy_score(y_test, y_pred)

    accuracy[feat] = acc

    if(feat%10 == 0):
        print("[INFO] " + str(feat) + ' Features Completed' )
```

```
[INFO] 10 Features Completed
[INFO] 20 Features Completed
[INFO] 30 Features Completed
[INFO] 40 Features Completed
[INFO] 50 Features Completed
[INFO] 60 Features Completed
[INFO] 70 Features Completed
[INFO] 80 Features Completed
[INFO] 90 Features Completed
[INFO] 100 Features Completed
[INFO] 110 Features Completed
[INFO] 120 Features Completed
[INFO] 130 Features Completed
[INFO] 140 Features Completed
[INFO] 150 Features Completed
[INFO] 160 Features Completed
[INFO] 170 Features Completed
[INFO] 180 Features Completed
[INFO] 190 Features Completed
```

```
res = dict((v,k) for k,v in accuracy.items())
res[max(accuracy.values())]
```

```
176
```

```
max(accuracy.values())
```

```
0.5288713910761155
```

```
y_pred = classifier.predict(X_test)
print(y_pred)
```

In [27]:
```
y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[5. 5. 2. ... 3. 5. 2.]
```

In [29]:
```
dataset = df_new['review_text']
y_actual = df_new['num_rating']
```

In [30]:
```
y_actual
```

Out[30]:
```
0       5.0
1       4.0
2       4.0
3       2.0
4       4.0
       ...
7611    4.0
7612    5.0
7613    4.0
7614    3.0
7615    4.0
Name: num_rating, Length: 7616, dtype: float64
```

In [31]:
```
y_pred = classifier.predict(X)
print(y_pred)
```

```
[5. 5. 3. ... 5. 1. 5.]
```

In [ ]:
```
dataset['predicted_label'] = y_pred.tolist()
dataset.head()
```

In [ ]:
```
dataset['predicted_label'] = y_pred.tolist()
dataset.head()
```

In [33]:
```
from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix , classification_report
import pandas as pd
import seaborn as sns
```

In [34]:
```
# Source code credit for this function: https://gist.github.com/shaypal5/94c53d765083101efc0240d776a23823
def print_confusion_matrix(confusion_matrix, class_names, figsize = (10,10), fontsize=14):
    """Prints a confusion matrix, as returned by sklearn.metrics.confusion_matrix, as a heatmap.

    Arguments
    ---------
    confusion_matrix: numpy.ndarray
        The numpy.ndarray object returned from a call to sklearn.metrics.confusion_matrix.
        Similarly constructed ndarrays can also be used.
    class_names: list
        An ordered list of class names, in the order they index the given confusion matrix.
    figsize: tuple
        A 2-long tuple, the first value determining the horizontal size of the ouputted figure,
        the second determining the vertical size. Defaults to (10,7).
    fontsize: int
        Font size for axes labels. Defaults to 14.

    Returns
    -------
    matplotlib.figure.Figure
        The resulting confusion matrix figure
    """
    df_cm = pd.DataFrame(
        confusion_matrix, index=class_names, columns=class_names,
    )
    fig = plt.figure(figsize=figsize)
    try:
        heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
    except ValueError:
        raise ValueError("Confusion matrix values must be integers.")
    heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=fontsize)
    heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=45, ha='right', fontsize=fontsize)
    plt.ylabel('Truth')
    plt.xlabel('Prediction')
```

In [35]:
```
rate = df_new['num_rating'].unique()
rate
```

Out[35]:
```
array([5., 4., 2., 3., 1.])
```

In [ ]:
```
cm = confusion_matrix(y_actual,y_pred)
print_confusion_matrix(cm,rate)
```

In [37]:
```
print(classification_report(y_actual, y_pred))
```

```
              precision    recall  f1-score   support

         1.0       0.16      0.79      0.27       278
         2.0       0.27      0.27      0.27       460
         3.0       0.28      0.25      0.26       738
         4.0       0.53      0.32      0.40      2360
         5.0       0.68      0.66      0.67      3780

    accuracy                           0.50      7616
   macro avg       0.38      0.46      0.37      7616
weighted avg       0.55      0.50      0.51      7616
```

In [38]:
```
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
import plotly.io as pio
```

In [39]:
```
df_new = pd.DataFrame(df_new)
```

In [ ]:
```
fig = px.histogram(df_new, x="num_rating")
fig.update_layout(barmode='stack')
fig.show()
```

In [ ]:
```
fig = px.histogram(y_pred)
fig.update_layout(barmode='stack')
fig.show()
```

III. BEHAVIOR ANALYSIS

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
import plotly.io as pio
```

In [ ]:
```
df = pd.read_csv("./Starbucks satisfactory survey.csv")
df.head()
```

In [3]:
```
df.shape
```

Out[3]: (122, 21)

In [ ]:
```
#As the column names are very big sentences, we have compresed them down to short variable names.
df.rename({'1. Your Gender' : 'Gender',
           '2. Your Age' : 'Age',
           '3. Are you currently....?' : 'Working_Status',
           '4. What is your annual income?' : 'Annual_Income',
           '5. How often do you visit Starbucks?' : 'Visit_Frequency',
           '6. How do you usually enjoy Starbucks?' : 'Prefered Form of Consumption',
           '7. How much time do you normally  spend during your visit?' : 'Visit_Time',
           "8. The nearest Starbucks's outlet to you is...?" : 'Nearest_Outlet',
           '9. Do you have Starbucks membership card?' : 'Membership_Status',
           '10. What do you most frequently purchase at Starbucks?' : 'Frequently_Purchased',
           '11. On average, how much would you spend at Starbucks per visit?' : 'Avg_Spend',
           '12. How would you rate the quality of Starbucks compared to other brands (Coffee Bean, Old Town White Coffee..) to be:' : 'Overall_Rating',
           '13. How would you rate the price range at Starbucks?' : 'Price_Rating',
           '14. How important are sales and promotions in your purchase decision?' : 'Promotion_Rating',
           '15. How would you rate the ambiance at Starbucks? (lighting, music, etc...)' : 'Ambiance_Rating',
           '16. You rate the WiFi quality at Starbucks as..' : 'Wifi_Rating',
           '17. How would you rate the service at Starbucks? (Promptness, friendliness, etc..)' : 'Service_Rating',
           '18. How likely you will choose Starbucks for doing business meetings or hangout with friends?' : 'Meeting_Rating',
           '19. How do you come to hear of promotions at Starbucks? Check all that apply.' : 'Promo_Ways',
           '20. Will you continue buying at Starbucks?' : 'Customer_Loyalty'} ,
          inplace = True , axis = 1)
```

In [5]:
```
#Cleaned the data by deleting the tuples having na/NA values
df=df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 121 entries, 0 to 121
Data columns (total 21 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Timestamp                    121 non-null    object
 1   Gender                       121 non-null    object
 2   Age                          121 non-null    object
 3   Working_Status               121 non-null    object
 4   Annual_Income                121 non-null    object
 5   Visit_Frequency              121 non-null    object
 6   Prefered Form of Consumption 121 non-null    object
 7   Visit_Time                   121 non-null    object
 8   Nearest_Outlet               121 non-null    object
 9   Membership_Status            121 non-null    object
 10  Frequently_Purchased         121 non-null    object
 11  Avg_Spend                    121 non-null    object
 12  Overall_Rating               121 non-null    int64
 13  Price_Rating                 121 non-null    int64
 14  Promotion_Rating             121 non-null    int64
 15  Ambiance_Rating              121 non-null    int64
 16  Wifi_Rating                  121 non-null    int64
 17  Service_Rating               121 non-null    int64
 18  Meeting_Rating               121 non-null    int64
 19  Promo_Ways                   121 non-null    object
 20  Customer_Loyalty             121 non-null    object
dtypes: int64(7), object(14)
memory usage: 20.8+ KB
```

In [ ]:
```
#for showing number of unique values in each column
for column in df.columns:
    print('{} column: {} unique values'.format(column,df[column].nunique()))
```

In [7]:
```
#Replacing repeated values into a single value thus reducing number of unique values.
df['Prefered Form of Consumption'] = df['Prefered Form of Consumption'].replace({'Never':'None',
                                                                                 'never':'None',
                                                                                 'Never buy':'None',
                                                                                 'Never ':'None',
                                                                                 'I dont like coffee':'None'})
df['Frequently_Purchased'] = df['Frequently_Purchased'].replace({'Never buy any':'Nothing',
                                                                 'never':'Nothing',
                                                                 'Never':'Nothing',
                                                                 'Jaws chip ':'Jaws Chip',
                                                                 'cake ':'Cake'})
```

In [8]:
```
#for showing number of unique values in each column
for column in df.columns:
    print('{} column: {} unique values'.format(column,df[column].nunique()))
```

```
Timestamp column: 121 unique values
Gender column: 2 unique values
Age column: 4 unique values
Working_Status column: 4 unique values
Annual_Income column: 5 unique values
Visit_Frequency column: 5 unique values
Prefered Form of Consumption column: 4 unique values
Visit_Time column: 5 unique values
Nearest_Outlet column: 3 unique values
Membership_Status column: 2 unique values
Frequently_Purchased column: 18 unique values
Avg_Spend column: 4 unique values
Overall_Rating column: 5 unique values
Price_Rating column: 5 unique values
Promotion_Rating column: 5 unique values
Ambiance_Rating column: 5 unique values
Wifi_Rating column: 5 unique values
Service_Rating column: 5 unique values
Meeting_Rating column: 5 unique values
Promo_Ways column: 31 unique values
Customer_Loyalty column: 2 unique values
```

In [9]:
```
df.shape
```

Out[9]:
```
(121, 21)
```

In [10]:
```
df.to_csv('./Dashboard/Behaviour_Analysis.csv')
```

Components of the customers

In [ ]:
```
fig = px.scatter_3d(df,x='Age', y='Working_Status', z='Annual_Income',color='Working_Status',symbol='Gender')
fig.show()
```

Gender Count

In [ ]:
```
fig = px.histogram(df, x="Gender")
fig.update_layout(barmode='stack')
fig.show()
```

Working Status of Customers

In [ ]:
```
fig = px.histogram(df, x="Working_Status")
fig.update_layout(barmode='stack')
fig.show()
```

Age Ranges of the Customer

In [ ]:
```
fig = px.histogram(df, x="Age")
fig.update_layout(barmode='stack')
fig.show()
```

Age Ranges of the Customer

In [ ]:
```
fig = px.histogram(df, x="Age")
fig.update_layout(barmode='stack')
fig.show()
```

Visiting Frequency

Histogram

In [ ]:
```
fig = px.histogram(df, x="Visit_Frequency")
fig.update_layout(barmode='stack')
fig.show()
```

Pie Chart

In [ ]:
```
# Construct a DataFrame by 'visitingFrequency' and its value_counts()
dfF = df['Visit_Frequency'].value_counts()
dfF = pd.DataFrame({'Frequency':dfF.index, 'Counts':dfF.values})

labels = dfF['Frequency']
values = dfF['Counts']
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.show()
```

In this survey 62.3% of the customers rarely come to Starbucks.

How does people buy starbucks?

Histogram

In [ ]:
```
fig = px.histogram(df, x="Prefered Form of Consumption")
fig.update_layout(barmode='stack')
fig.show()
```

Pie Chart

In [ ]:
```
# same process as Pie chart of visting frequency
dfM = df['Prefered Form of Consumption'].value_counts()
dfM = pd.DataFrame({'Method':dfM.index, 'Counts':dfM.values})

labels = dfM['Method']
values = dfM['Counts']
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.show()
```

Time spending in Starbucks of each Customers

In [ ]:
```
dfT = df['Visit_Time'].value_counts()
dfT = pd.DataFrame({'Time':dfT.index, 'Counts':dfT.values})

labels = dfT['Time']
values = dfT['Counts']
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.show()
```

Location

Distance of the nearest Starbuck of the customers

In [ ]:
```
dfL = df['Nearest_Outlet'].value_counts()
dfL = pd.DataFrame({'Distance':dfL.index, 'Counts':dfL.values})

labels = dfL['Distance']
values = dfL['Counts']
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.show()
```

Membership Card

How many customers (don't)have menbership card?

In [ ]:
```
dfMem = df['Membership_Status'].value_counts()
dfMem = pd.DataFrame({'Membership(Y/N)':dfMem.index, 'Counts':dfMem.values})

labels = dfMem['Membership(Y/N)']
values = dfMem['Counts']
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.show()
```

Product sales analysis

How much money did people pay to Starbucks during this visit?

In [ ]:
```
dfP = df['Frequently_Purchased'].value_counts()
dfP = pd.DataFrame({'Product':dfP.index, 'Counts':dfP.values})
```

Product sales analysis

How much money did people pay to Starbucks during this visit?

In [ ]:
```
dfP = df['Frequently_Purchased'].value_counts()
dfP = pd.DataFrame({'Product':dfP.index, 'Counts':dfP.values})

labels = dfP['Product']
values = dfP['Counts']
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.update_layout(margin = dict(t=0, l=0, r=0, b=0))
fig.show()
```

Money spend

How much money did people pay to Starbucks during this visit?

In [ ]:
```
dfS = df['Avg_Spend'].value_counts()
dfS = pd.DataFrame({'Costs':dfS.index, 'Counts':dfS.values})

labels = dfS['Costs']
values = dfS['Counts']
fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)])
fig.update_layout(margin = dict(t=0, l=0, r=0, b=0))
fig.show()
```

Components of customers & moneySpend

In [ ]:
```
fig = px.scatter_3d(df,x='Age', y='Working_Status', z='Annual_Income',color='Avg_Spend',symbol='Gender')
fig.show()
```

Which part of starbucks satisfies customers the most/least?

In [11]:
```
# Caculate the sum of each column and store it into a DataFrame

# dict Score
Score = {}
Score['Comparing with other coffee shop']=df['Overall_Rating'].sum()
Score['Price']=df['Price_Rating'].sum()
Score['Ambient'] = df['Ambiance_Rating'].sum()
Score['Wifi'] = df['Wifi_Rating'].sum()
Score['Service'] = df['Service_Rating'].sum()
Score['MhRating'] = df['Meeting_Rating'].sum()

# Convert Score into DataFrame
score_df = pd.DataFrame(Score.items(), columns=['part','score'])
score_df
```

Out[11]:

	part	score
0	Comparing with other coffee shop	446
1	Price	352
2	Ambient	455
3	Wifi	394
4	Service	454
5	MhRating	426

In [ ]:
```
fig = px.bar(score_df, x='part', y='score')
fig.show()
```

Which part satisfies customers the most?

In [12]:
```
score_df[score_df['score']==score_df['score'].max()]
```

Out[12]:

	part	score
2	Ambient	455

Which part satisfies customers the least?

In [13]:
```
score_df[score_df['score']==score_df['score'].min()]
```

Out[13]:

	part	score
1	Price	352

In [14]:
```
#Tells us about the count, mean, median, min/max values and gives IQR i.e. Q1(25%),Q3(75%)
df.describe()
```

Out[14]:

	Overall_Rating	Price_Rating	Promotion_Rating	Ambiance_Rating	Wifi_Rating	Service_Rating	Meeting_Rating
count	121.000000	121.000000	121.000000	121.000000	121.000000	121.000000	121.000000
mean	3.685950	2.909091	3.818182	3.760331	3.256198	3.752066	3.520661
std	0.913173	1.072381	1.064581	0.931171	0.962020	0.829468	1.033595
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	3.000000	2.000000	3.000000	3.000000	3.000000	3.000000	3.000000
50%	4.000000	3.000000	4.000000	4.000000	3.000000	4.000000	4.000000
75%	4.000000	4.000000	5.000000	4.000000	4.000000	4.000000	4.000000
max	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

In [15]:
```
#Extractig columns with integer values
num_cols = df.select_dtypes(include='int64').columns
num_cols
```

Out[15]:
```
Index(['Overall_Rating', 'Price_Rating', 'Promotion_Rating', 'Ambiance_Rating',
       'Wifi_Rating', 'Service_Rating', 'Meeting_Rating'],
      dtype='object')
```

In [ ]:
```
def plot_numeric(df,x):
    plt.boxplot(df[i], vert=True)
    plt.show()
    print('Quantiles')
    print(df[i].quantile([0.01,0.03,0.1,0.2,0.5,0.75,0.9,1.0]))

for i in num_cols:
    print("Plots for Column: "+ i)
    plot_numeric(df,i)
    print("\n")
```

In [16]:
```
df['Visit_Frequency'] = df['Visit_Frequency'].replace({'Never':0, 'Rarely':1,'Monthly':2,'Weekly':3,"Daily":4})
```

In [17]:
```
#counts how frequently customers visit Starbucks
df['Visit_Frequency'].value_counts()
#1-rarely
#2-monthly
#3-weekly
#4-daily
```

Out[17]:
```
1    76
2    26
3     9
0     8
4     2
Name: Visit_Frequency, dtype: int64
```

In [18]:
```
#Giving age-groups- numerical rankings
df['Age'] = df['Age'].replace({'Below 20':1, 'From 20 to 29':2,'From 30 to 39':3,'40 and above':4})
```

In [20]:
```
#Creating a new database with only columns having integer values in order to do modelling
df1=df[['Age','Visit_Frequency','Overall_Rating','Price_Rating',
        'Promotion_Rating','Ambiance_Rating',
        'Wifi_Rating','Service_Rating','Meeting_Rating']]
```

In [21]:
```
#Giving X all the values accept the target variable(Visit frequency)
#Y=target variable
X = df1.drop('Visit_Frequency', axis = 1)
y = df1['Visit_Frequency']
```

In [22]:
```
from sklearn.model_selection import train_test_split
```

In [23]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, stratify = y, test_size = 0.25) #training data=30% of dataset
```

In [24]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
```

In [25]:
```
tree = DecisionTreeClassifier()
rf = RandomForestClassifier()
```

DEFINING DIFFERENT CLASSIFICATION MODELS AND CHECKING ACCURACIES OF THE SAME, THEREBY FINDING THE MOST ACCURATE/SUITABLE MODEL.

In [26]:
```
Dt_model = tree.fit(X_train,y_train)
```

In [27]:
```
Dt_model
```

Out[27]:
```
DecisionTreeClassifier()
```

In [28]:
```
score_acc1 = accuracy_score(y_test, Dt_model.predict(X_test))
score_acc1
```

Out[28]:
```
0.4838709677419355
```

In [29]:
```
rf_model = rf.fit(X_train,y_train)
```

In [30]:
```
rf_model
```

Out[30]:
```
RandomForestClassifier()
```

In [31]:
```
score_acc2 = accuracy_score(y_test, rf_model.predict(X_test))
score_acc2
```

Out[31]:
```
0.5161290322580645
```

We can see that random forest claasifier has the highest accuracy score

In [32]:
```
rf_model.predict(X.sample())
#1-rarely
#2-monthly
#3-weekly
#4-daily
```

Out[32]:
```
array([1], dtype=int64)
```

IV. DASHBOARD

```
import pandas as pd
import streamlit as st
import plotly.express as px
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import seaborn as sns
from streamlit_option_menu import option_menu
from sklearn.metrics import confusion_matrix , classification_report

#st.set_page_config(page_title='STARBUCKS ANALYSIS')
#st.subheader('Was the tutorial helpful?')

st.set_option('deprecation.showPyplotGlobalUse', False)


st.set_page_config(page_title="Starbucks Analysis", page_icon=":bar_chart:",
layout="wide")
#st.header('STARBUCKS ANALYSIS')


selected = option_menu(
        menu_title="STARBUCKS ANALYSIS",
        options=["Customer Segmentation", "Sentiment Analysis", "Behaviour Analysis"],
        icons=["clipboard-data", "emoji-laughing", "bar-chart-steps"],
        menu_icon="cast",
        default_index=0,
        orientation="horizontal",
        )

if selected == "Customer Segmentation":
    st.title("Customer Segmentation")

    st.subheader('Age Distribution')
    profile_1 = pd.read_csv('./profile_1.csv')

    fig_1 = px.bar(profile_1, x='age_range', y='no_of_cust',
                color = 'age_range',
                labels={'no_of_cust':'Number of Customers', 'age_range' : 'Age Range'}, title='Age
Distribution of Starbucks Customers',
                width=1200, height=800)

    st.plotly_chart(fig_1)

    st.subheader('What are the salary ranges of people across different age groups?')

    profile_2 = pd.read_csv('./profile_2.csv')
```

```
    fig_2 = px.violin(profile_2, x="age_range", y="income", color="age_range", box=True,
title='Starbucks Income Distribution across age Ranges', labels={'age_range' : 'Age Ranges',
'income' : 'Income'}, width=1200, height=700)
    #fig = px.violin(profile, y="income", x="age_range", color="age_range", box=True,
points='all')
    st.plotly_chart(fig_2)

    fig_3 = px.scatter(profile_2, x="age", y="income", color='age', title='Starbucks Customers
- Income Distribution Across Individual Ages', labels={'age' : 'Age', 'income' : 'Income'},
height=700, width=1200)
    st.plotly_chart(fig_3)

    transcript_amount = pd.read_csv('./transcript_amount.csv')
    st.subheader('What does the correlation between number of days an offer has been open vs.
final transaction amount have to tell us?')
    fig_4 = px.scatter(transcript_amount, x="days", y="amount", color='days', title='Days
Reward Open vs. Final Amount Spent', labels={'days' : 'Days', 'amount' : 'Amount'},
height=800, width=1200)
    #st.plotly_chart(fig_4)

    fig_5 = px.scatter(transcript_amount, x="days", y="amount", color='days', title='Days
Reward Open vs. Final Amount Spent' , labels={'days' : 'Days', 'amount' : 'Amount'},
range_y=(0,50), height=800, width=1200)
    #st.plotly_chart(fig_5)

    fig_6 = px.histogram(profile_2.sort_values(by='age_range'), x='age_range',
                color='gender', barmode='group',
                height=800, width=1200, title='Distribution of Genders Across Age Ranges',
labels={'age_range' : 'Age Ranges', 'count' : 'Number of Peoples'})
    st.plotly_chart(fig_6)

    st.subheader("Distribution of Income Across Genders and Ages")

    fig_7 = plt.figure(figsize=(15,10))
    sns.barplot(data = profile_2, x = 'age_range', y = 'income', hue = 'gender', palette='cool_r')
    fig_7 = plt.xlabel('Age Ranges')
    fig_7 = plt.ylabel('Income')
    fig_7 = plt.style.use('seaborn')
    st.pyplot(fig_7)

    st.subheader('Number of Clusters using K-Means')
    k_means_result = pd.read_csv('./K_means_result.csv')
    fig = px.line(k_means_result, x='no_of_clusters', y='sum_of_squared_error',
markers=True, height=800, width=1200, title=('K Means SSE Scores as K Increases'))
    fig.update_layout(title='KMeans SSE Scores as K Increases',
                    yaxis_title='SSE',
                    xaxis_title='Number of Clusters (K)')
    st.plotly_chart(fig)

    st.subheader('Scattered Plot for each and Every Cluster')
    cust_tran_master_1 = pd.read_csv('./cust_tran_master_1.csv')
```

```
    fig = px.scatter(cust_tran_master_1, y="income", x="age", color="cluster_1",
symbol="cluster_1", render_mode='webgl', height=800, width=1200, labels=({'age' : 'Age',
'income' : 'Income', 'cluster_1' : 'Different Clusters (k = 5)'}))
    fig.update_layout(legend_orientation="h")
    st.plotly_chart(fig)

    st.subheader('Number of Customers in a Particular Cluster')
    fig = px.histogram(cust_tran_master_1, x="cluster_1", labels=({'cluster_1' : 'Cluster
Number', 'count' : 'Count'}), height=800, width=1200)
    fig.update_layout(bargap=0.2)
    st.plotly_chart(fig)

    fig = px.pie(cust_tran_master_1,
cust_tran_master_1['cluster_1'],color_discrete_sequence=px.colors.sequential.PuBuGn_r,heig
ht=700, width=1200)
    st.plotly_chart(fig)

    st.subheader('Clustering between Age and Gender')
    k_means_result_2 = pd.read_csv('./K_means_result_2.csv')
    fig = px.line(k_means_result_2, x='Number_of_Clusters', y='K_Means_Score',
markers=True, height=800, width=1200, title=('K Means SSE Scores as K Increases'))
    fig.update_layout(title='KMeans SSE Scores as K Increases',
                      yaxis_title='SSE',
                      xaxis_title='Number of Clusters (K)')
    st.plotly_chart(fig)

    st.subheader('Clustering between Age and Income')
    k_means_result_3 = pd.read_csv('./K_means_result_3.csv')
    fig = px.line(k_means_result_3, x='Number_of_Clusters', y='K_Means_Score',
markers=True, height=800, width=1200, title=('K Means SSE Scores as K Increases'))
    fig.update_layout(title='KMeans SSE Scores as K Increases',
                      yaxis_title='SSE',
                      xaxis_title='Number of Clusters (K)')
    st.plotly_chart(fig)

    st.subheader('Clustering between Gender and Income')
    k_means_result_4 = pd.read_csv('./K_means_result_4.csv')
    fig = px.line(k_means_result_4, x='Number_of_Clusters', y='K_Means_Score',
markers=True, height=800, width=1200, title=('K Means SSE Scores as K Increases'))
    fig.update_layout(title='KMeans SSE Scores as K Increases',
                      yaxis_title='SSE',
                      xaxis_title='Number of Clusters (K)')
    st.plotly_chart(fig)

    st.header('DBSCAN')

    DBSCAN = pd.read_csv('./Scattered_Plot_for_DBSCAN.csv')
    fig = px.scatter(DBSCAN, x="x", y="y", color='cluster', title='DBSCAN Cluster Result
Visualization', labels={'cluster' : 'Cluster', 'x' : 'X', 'y':'Y'}, height=700, width=1200)
    fig.update_layout(legend_orientation="h")
    st.plotly_chart(fig)
```

```
    st.subheader('Customers in each Clusters')
    fig = px.histogram(DBSCAN, x="cluster", labels={{'cluster' : 'Cluster Number', 'count' :
'Count'}), height=800, width=1200)
    fig.update_layout(bargap=0.2)
    st.plotly_chart(fig)

    st.subheader('Age Classification on the basis of the Clusters')
    customer_transaction = pd.read_csv('./customer_transactions.csv')
    fig = px.histogram(customer_transaction, x="age_range", color="Label", height=800,
width=1200).update_xaxes(categoryorder='total descending')
    st.plotly_chart(fig)

    st.subheader('Income Classification on the basis of the Clusters')
    fig = px.histogram(customer_transaction, x="income", color="Label", height=800,
width=1200).update_xaxes(categoryorder='total descending')
    st.plotly_chart(fig)

    st.subheader('Customer Days as a Member Classification on the basis of the Clusters')
    fig = px.histogram(customer_transaction, x="days_as_member", color="Label",
height=800, width=1200).update_xaxes(categoryorder='total descending')
    st.plotly_chart(fig)

    st.subheader('Dataset')
    st.dataframe(customer_transactions)

if selected == "Sentiment Analysis":
    st.title("Sentiment Analysis")

    df = pd.read_csv('./Sentiment_Analysis_Result.csv')
    rate = [5.0, 4.0, 3.0, 2.0, 1.0]

    df_2 = pd.read_csv('./Ratings.csv')
    fig = px.histogram(df_2, x="num_rating", color="num_rating", height=800, width=1200,
labels={'num_rating' : 'Number of Rating', 'count' : 'Count'})
    fig.update_layout(barmode='stack')
    st.plotly_chart(fig)


#     def print_confusion_matrix(confusion_matrix, class_names, figsize = (3,3), fontsize=7):
#         df_cm = pd.DataFrame(
#             confusion_matrix, index=class_names, columns=class_names,
#         )
#         fig = plt.figure(figsize=figsize)
#         try:
#             heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
#         except ValueError:
#             raise ValueError("Confusion matrix values must be integers.")
#         heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right',
fontsize=fontsize)
#         heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=45, ha='right',
fontsize=fontsize)
#         plt.ylabel('Truth')
```

```
#         plt.xlabel('Prediction')
#         st.pyplot(fig)
#
#     cm = confusion_matrix(df['y_actual'],df['y_pred'])
#     print_confusion_matrix(cm,rate)

    st.subheader('Dataset')
    st.dataframe(df)

if selected == "Behaviour Analysis":
    st.title("Behaviour Analysis")

    df = pd.read_csv('./Behaviour_Analysis.csv')
    st.subheader('Components of the Customers ( Age, Working Status, Annual Income )')
    fig = px.scatter_3d(df,x='Age', y='Working_Status', z='Annual_Income',
labels={'Annual_Income' : 'Annual Income', 'Working_Status' : 'Working Status'},
color='Working_Status',symbol='Gender', height=800, width=1200)
    st.plotly_chart(fig)

    st.subheader('Gender Count')
    fig = px.histogram(df, x="Gender", color="Gender", width=1200, height=800)
    fig.update_layout(barmode='stack')
    st.plotly_chart(fig)

    st.subheader('Working Status of Customers')
    fig = px.histogram(df, x="Working_Status", color='Working_Status', width=1200,
height=800, labels={'count' : 'Count', 'Working_Status' : 'Working Status'})
    fig.update_layout(barmode='stack')
    st.plotly_chart(fig)


    st.subheader('Age Ranges of the Customer')
    fig = px.histogram(df, x="Age", color='Age', width=1200, height=800, labels={'count' :
'Count'})
    fig.update_layout(barmode='stack')
    st.plotly_chart(fig)

    st.subheader("Product sales analysis")
    st.text("How much money did people pay to Starbucks during this visit?")
    st.text("")
    dfP = df['Frequently_Purchased'].value_counts()
    dfP = pd.DataFrame({'Product':dfP.index, 'Counts':dfP.values})

    layout = go.Layout(width=1200, height=800)
    labels = dfP['Product']
    values = dfP['Counts']
    fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)], layout=layout)
    fig.update_layout(margin = dict(t=0, l=0, r=0, b=0))
    st.plotly_chart(fig)


    st.subheader('Visiting Frequency')
```

```
    fig_1 = px.histogram(df, x="Visit_Frequency", color='Visit_Frequency', width=1200,
height=800, labels={'count' : 'Count', 'Visit_Frequency' : 'Visit Frequency'}, title="Histogram
Representation")
    fig_1.update_layout(barmode='stack')
    st.plotly_chart(fig_1)

    dfF = df['Visit_Frequency'].value_counts()
    dfF = pd.DataFrame({'Frequency':dfF.index, 'Counts':dfF.values})

    labels = dfF['Frequency']
    values = dfF['Counts']
    fig_2 = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3, title='Pie Chart
Representation')], layout=layout)
    st.plotly_chart(fig_2)

    st.subheader('How does people buy starbucks?')
    fig_1 = px.histogram(df, x="Prefered Form of Consumption", color='Prefered Form of
Consumption', width=1200, height=800, labels={'count' : 'Count', 'Visit_Frequency' : 'Visit
Frequency'}, title="Histogram Representation")
    fig_1.update_layout(barmode='stack')
    st.plotly_chart(fig_1)


    dfM = df['Prefered Form of Consumption'].value_counts()
    dfM = pd.DataFrame({'Method':dfM.index, 'Counts':dfM.values})

    labels = dfM['Method']
    values = dfM['Counts']
    st.text("Pie Chart Representation")
    fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)], layout=layout)
    st.plotly_chart(fig)

    st.subheader("Time spending in Starbucks by each Customers")
    dfT = df['Visit_Time'].value_counts()
    dfT = pd.DataFrame({'Time':dfT.index, 'Counts':dfT.values})

    labels = dfT['Time']
    values = dfT['Counts']
    fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)], layout=layout)
    st.plotly_chart(fig)

    st.subheader("Location : Distance of the nearest Starbuck of the customers")
    dfL = df['Nearest_Outlet'].value_counts()
    dfL = pd.DataFrame({'Distance':dfL.index, 'Counts':dfL.values})

    labels = dfL['Distance']
    values = dfL['Counts']
    fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)], layout=layout)
    st.plotly_chart(fig)
```

```
    st.subheader("Membership Card: How many customers (don't )have menbership card?")
    dfMem = df['Membership_Status'].value_counts()
    dfMem = pd.DataFrame({'Membership(Y/N)':dfMem.index, 'Counts':dfMem.values})

    labels = dfMem['Membership(Y/N)']
    values = dfMem['Counts']
    fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)], layout=layout)
    st.plotly_chart(fig)

    st.subheader("Money Spend")
    st.text("How much money did people pay to Starbucks during this visit?")
    dfS = df['Avg_Spend'].value_counts()
    dfS = pd.DataFrame({'Costs':dfS.index, 'Counts':dfS.values})

    labels = dfS['Costs']
    values = dfS['Counts']
    fig = go.Figure(data=[go.Pie(labels=labels, values=values, hole=.3)], layout=layout)
    fig.update_layout(margin = dict(t=0, l=0, r=0, b=0))
    st.plotly_chart(fig)

    st.subheader("Components of customers & Money Spend")
    fig = px.scatter_3d(df,x='Age', y='Working_Status',
z='Annual_Income',color='Avg_Spend',symbol='Gender', labels={'Working_Status' :
'Working Status', 'Annual_Income' : 'Annual Income', 'Avg_Spend' : 'Average Spend'},
width=1200, height=800)
    st.plotly_chart(fig)

    st.subheader("Which part of starbucks satisfies customers the most/least?")
    Score = {}
    Score['Comparing with other coffee shop']=df['Overall_Rating'].sum()
    Score['Price']=df['Price_Rating'].sum()
    Score['Ambient'] = df['Ambiance_Rating'].sum()
    Score['Wifi'] = df['Wifi_Rating'].sum()
    Score['Service'] = df['Service_Rating'].sum()
    Score['MhRating'] = df['Meeting_Rating'].sum()

    # Convert Score into DataFrame
    score_df = pd.DataFrame(Score.items(), columns=['part','score'])
    fig = px.bar(score_df, x='part', y='score', color='part', labels={'part' : 'Part', 'score' : 'Score'},
width=1200, height=800)
    st.plotly_chart(fig)

    st.subheader('Dataset')
    st.dataframe(df)
```

CHAPTER 7

TESTING

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves execution of a software component or system component to evaluate one or more properties of interest. Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools.

7.1 IMPORTANCE OF TESTING

1. Software testing is really required to point out the defects and errors that were made during the development phases. Example: Programmers may make a mistake during the implementation of the software. There could be many reasons for this like lack of experience of the programmer, lack of knowledge of the programming language, insufficient experience in the domain, incorrect implementation of the algorithm due to complex logic or simply human error.

2. It's essential since it makes sure that the customer finds the organization reliable and their satisfaction in the application is maintained. If the customer does not find the testing organization reliable or is not satisfied with the quality of the deliverable, then they may switch to a competitor organization. Sometimes contracts may also include monetary penalties with respect to the timeline and quality of the product. In such cases, if proper software testing may also prevent monetary losses.

3. It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence. As explained in the previous point, delivering good quality product on time builds the customers confidence in the team and the organization.

4. Testing is necessary in order to provide the facilities to the customers like the delivery of the high quality product or software application which requires

lower maintenance cost and hence results into more accurate, consistent and reliable results. High quality product typically has fewer defects and requires lesser maintenance effort, which in turn means reduced costs.

5. Testing is required for an effective performance of software application or product.

6. It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development. Proper testing ensures that bugs and issues are detected early in the life cycle of the product or application. If defects related to requirements or design are detected late in the life cyle, it can be very expensive to fix them since this might require redesign, re-implementation and retesting of the application.

7. It's required to stay in the business. Users are not inclined to use software that has bugs. They may not adopt a software if they are not happy with the stability of the application. In case of a product organization or startup which has only one product, poor quality of software may result in lack of adoption of the product and this may result in losses which the business may not recover from.

7.2 TYPES OF TESTING

7.2.1. FUNCTIONAL TESTING:

Functional Testing is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions. During functional testing, Black Box Testing technique is used in which the internal logic of the system being tested is not known to the tester. Functional testing is normally performed during the levels of System Testing and Acceptance Testing.
Typically, functional testing involves the following steps:

- Identify functions that the software is expected to perform.
- Create input data based on the function's specifications.
- Determine the output based on the function's specifications.
- Execute the test case.
- Compare the actual and expected outputs.

Functional testing is more effective when the test conditions are created directly from user/business requirements. When test conditions are created from the system documentation (system requirements/ design documents), the defects in that documentation will not be detected through testing and this may be the cause of end-users' wrath when they finally use the software.

7.2.2. NON-FUNCTIONAL TESTING

NON-FUNCTIONAL TESTING is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing. An excellent example of a non-functional test would be to check how many people can simultaneously login into a software. Non-functional testing is equally important as functional testing and affects client satisfaction. The Non-Functional requirements were also not given proper attention in the earlier test cycles. However, this has changed now. Non-functional tests are now most important as they consider all the application performance and security issues these days. This testing has a greater impact on applications when it comes to the performance of the application under high user traffic. This testing ensures that your application is stable and is able to handle load under extreme conditions.

7.2.3. REGRESSION TESTING

REGRESSION TESTING is a type of software testing that intends to ensure that changes (enhancements or defect fixes) to the software have not adversely affected it. The likelihood of any code change impacting functionalities that are not directly associated with the code is always there and it is essential that regression testing is conducted to make sure that fixing one thing has not broken another thing.

During regression testing, new test cases are not created but previously created test cases are re-executed. Regression [noun] literally means the act of going back to a previous place or state; return or reversion. In an ideal case, a full regression test is desirable but oftentimes there are time/resource constraints. In such cases, it is essential to do an impact analysis of the changes to identify areas of the software that have the highest probability of being affected by the change and that have the highest impact to users in case of malfunction and focus testing around those areas. Due to the scale and importance of regression testing, more and more companies and projects are adopting regression test automation tools.

7.2.4. PERFORMANCE TESTING

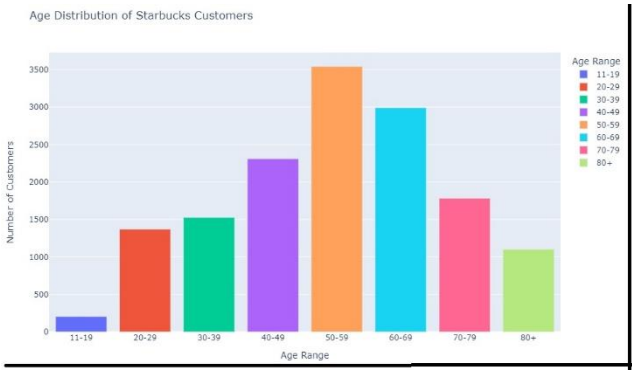
Performance testing is the process of determining the speed, responsiveness and stability of a computer, network, software program or device under a workload.

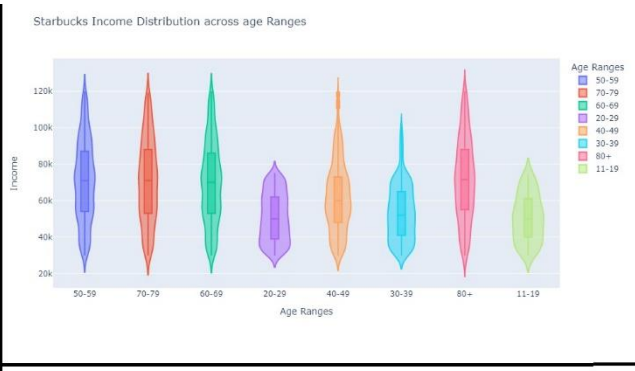
Performance testing can involve quantitative tests done in a lab, or occur in the production environment in limited scenarios. Typical parameters include processing speed, data transfer rate, network bandwidth and throughput, workload efficiency and reliability. For example, an organization can measure the response time of a program when a user requests an action or the number of millions of instructions per second (MIPS) at which a mainframe function.

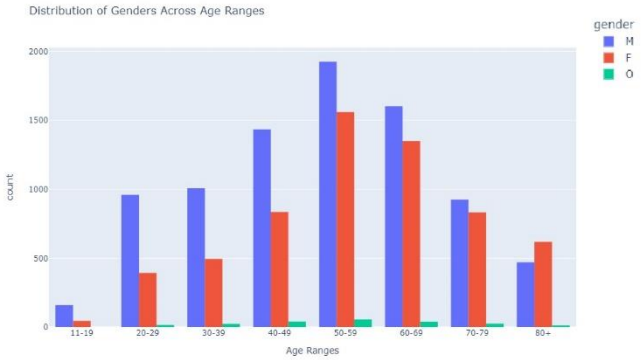
CHAPTER 8

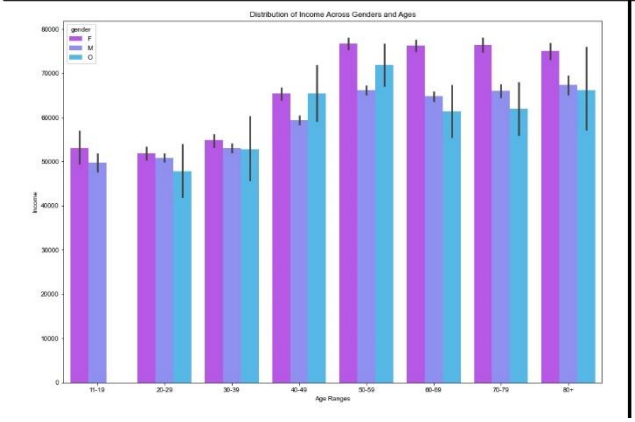
RESULT

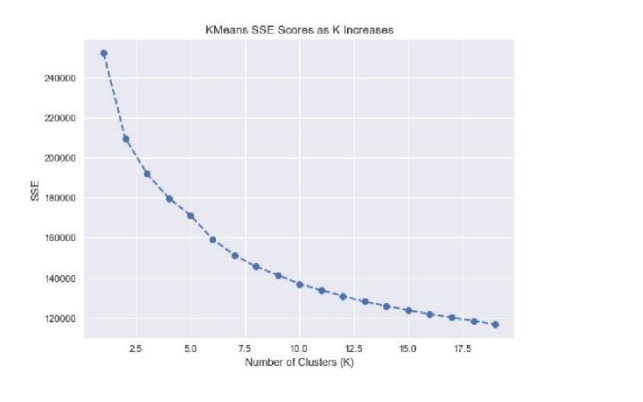
1. Customer Segmentation

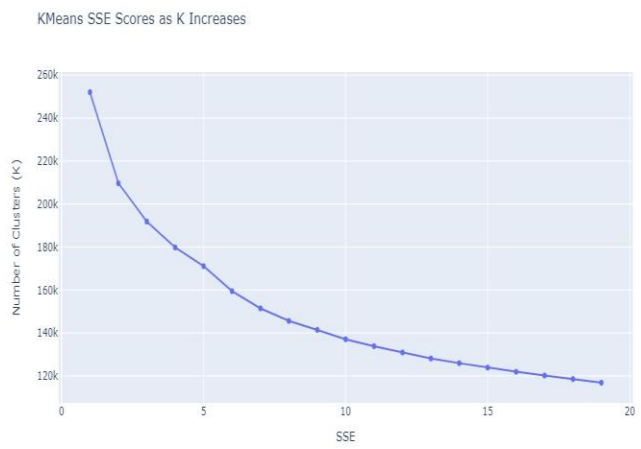


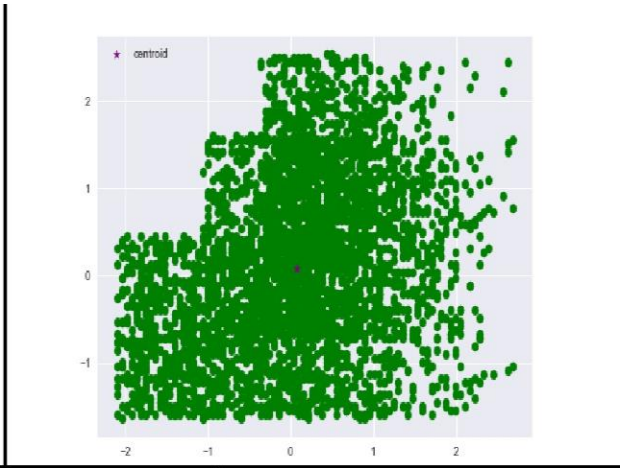


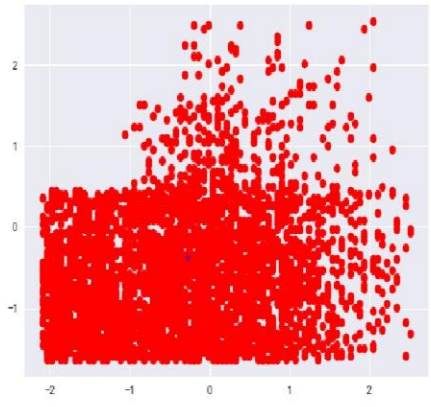


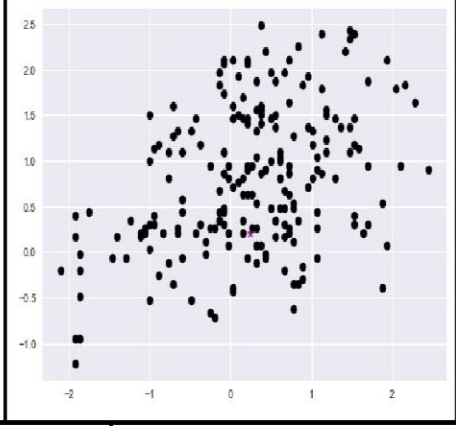


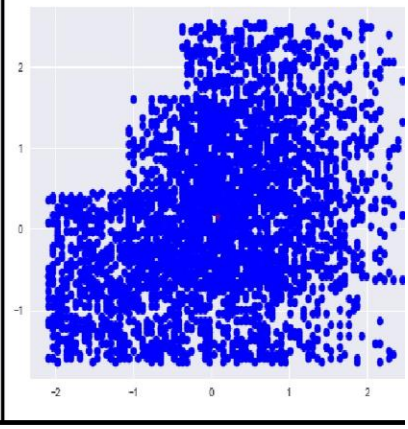


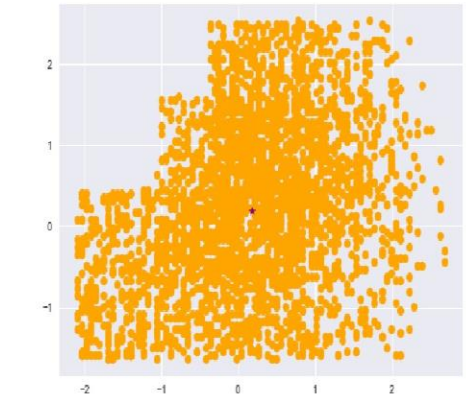

KMeans SSE Scores as K Increases
Number of Clusters (K)
SSE

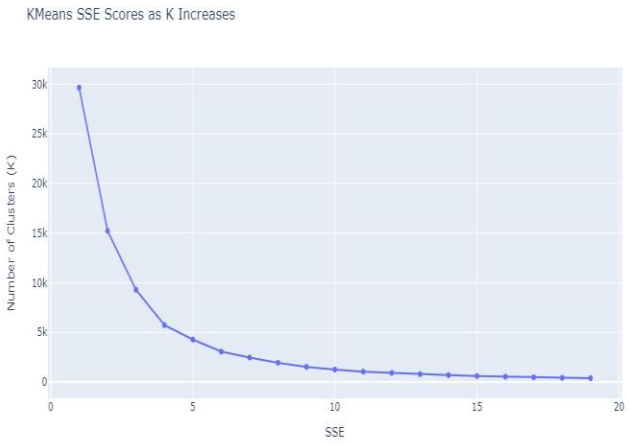

centroid

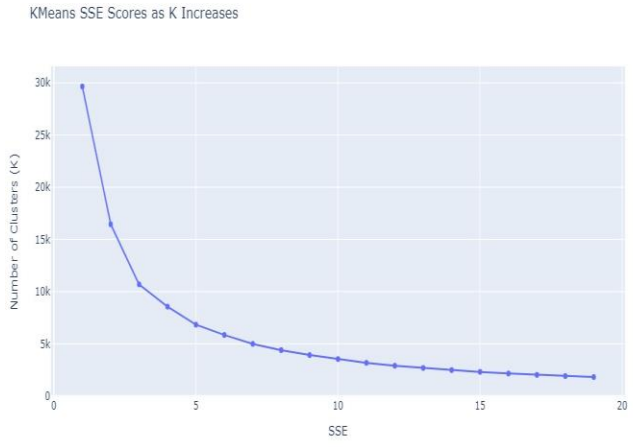


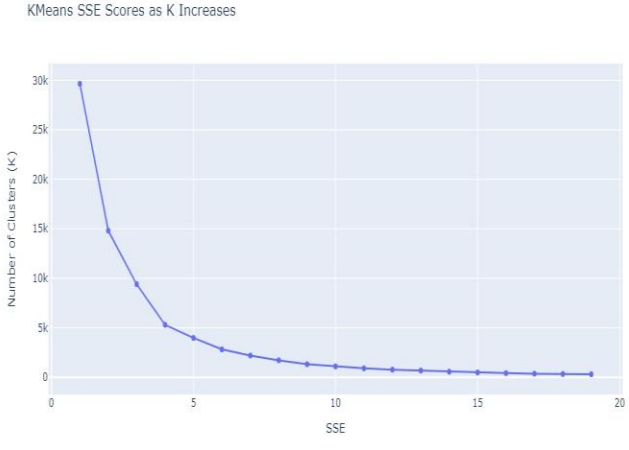


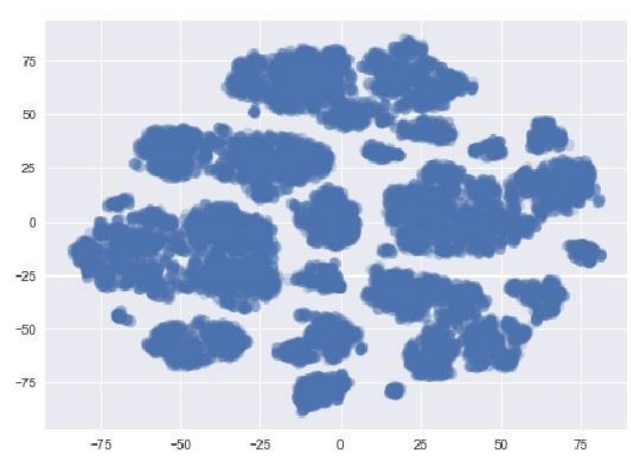


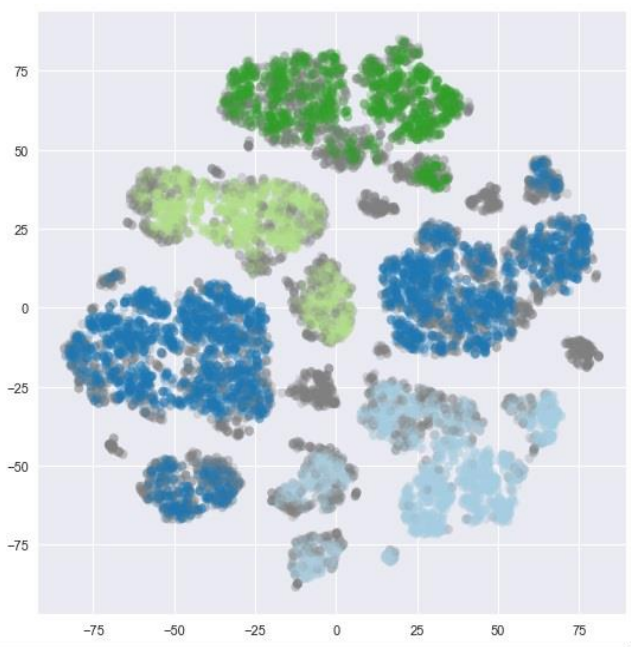


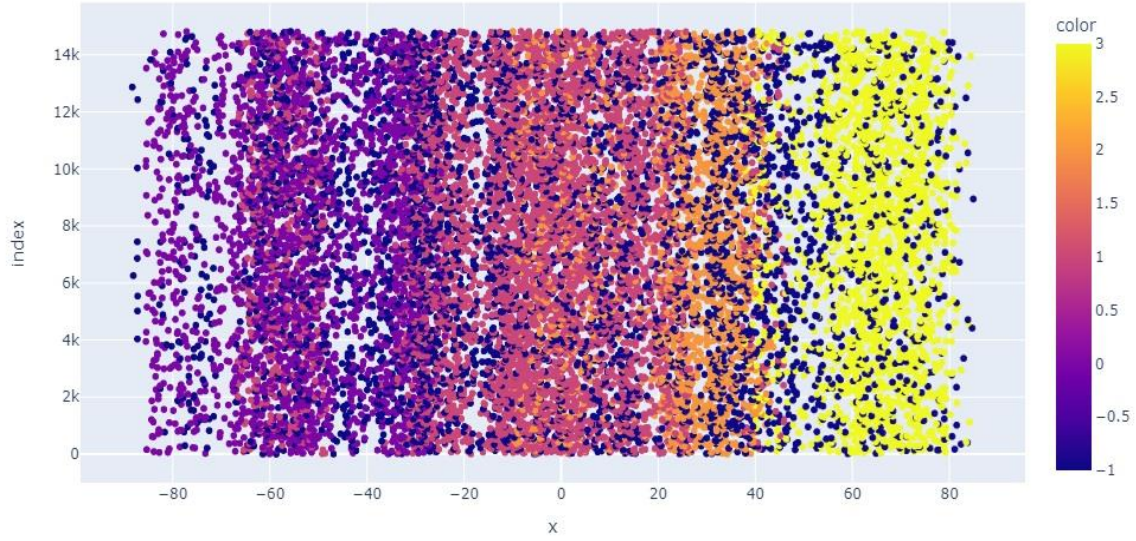

KMeans SSE Scores as K Increases
Number of Clusters (K)
SSE

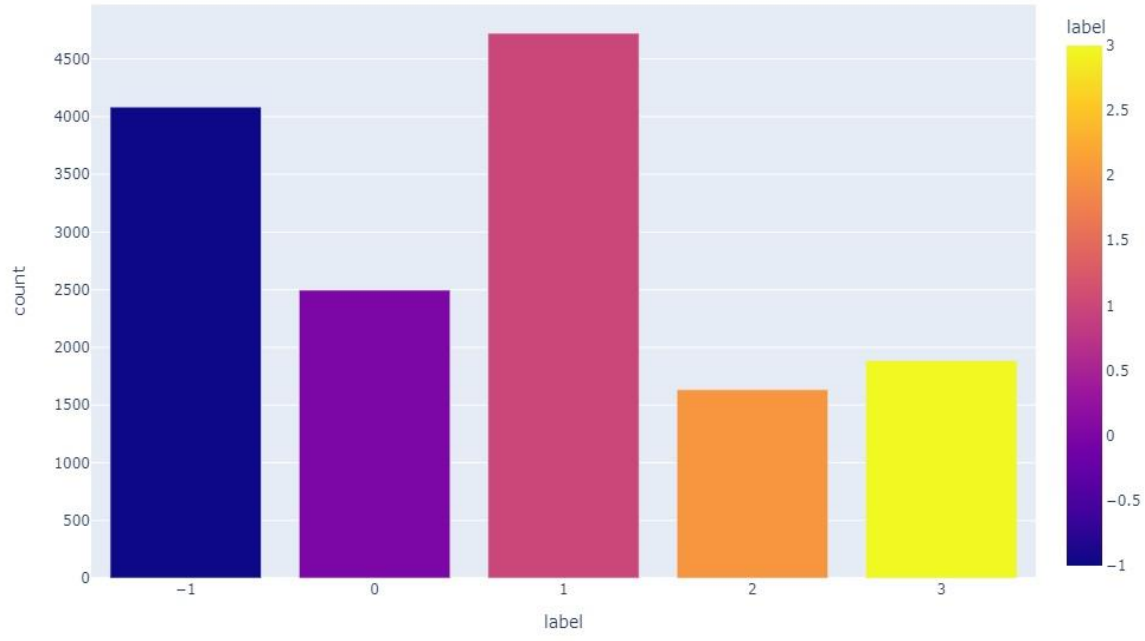

KMeans SSE Scores as K Increases
Number of Clusters (K)
SSE

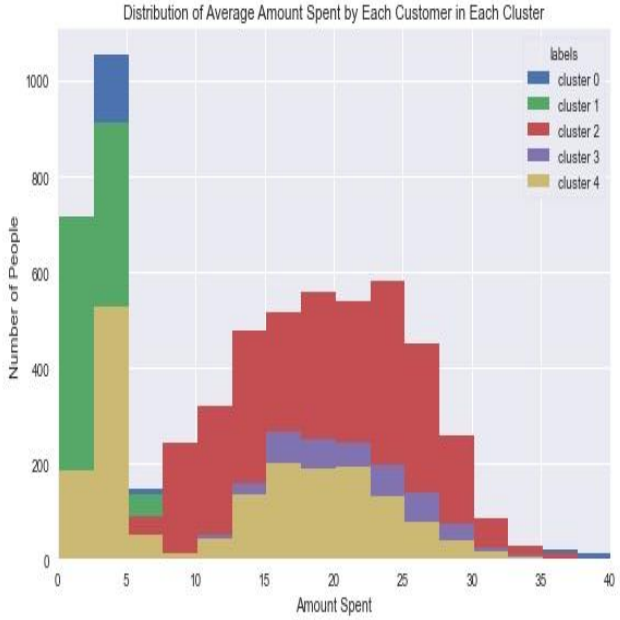

KMeans SSE Scores as K Increases
Number of Clusters (K)
SSE

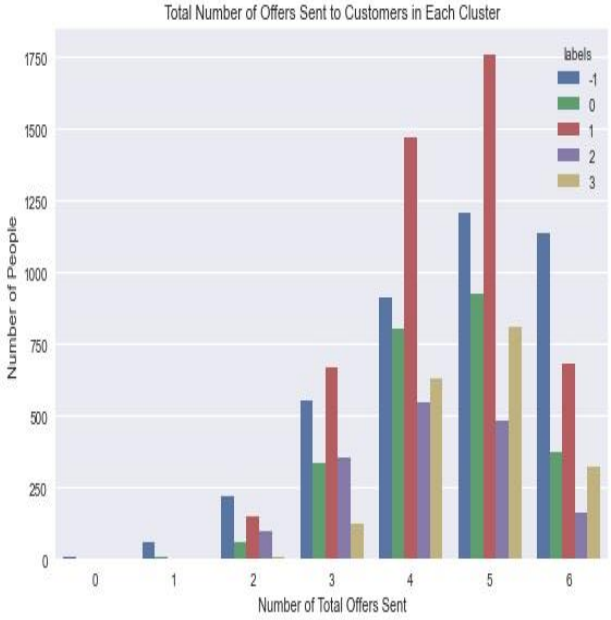

75
50
25
0
-25
-50
-75
-75
-50
-25
0
25
50
75

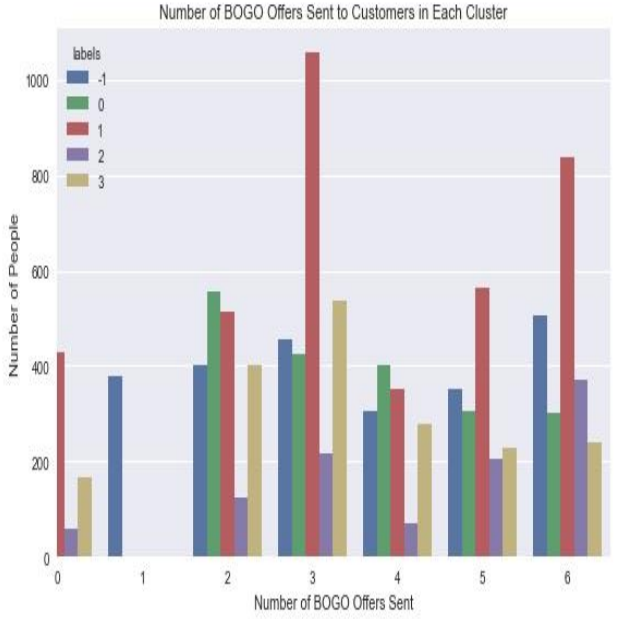

75
50
25
0
-25
-50
-75
-75
-50
-25
0
25
50
75

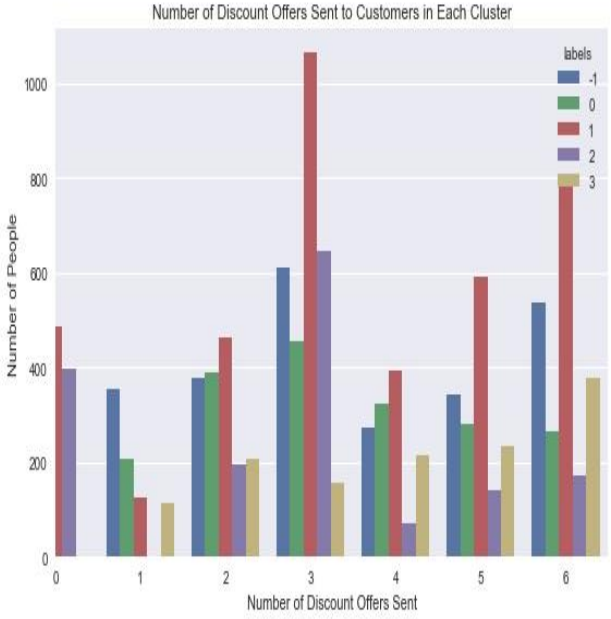

index
14k
12k
10k
8k
6k
4k
2k
0
-80
-60
-40
-20
0
20
40
60
80
x
color
3
2.5
2
1.5
1
0.5
0
-0.5
-1

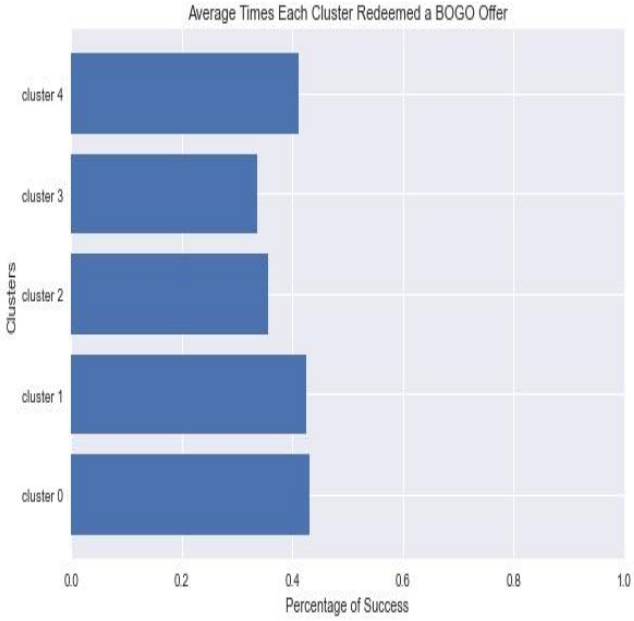

count
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-1
0
1
2
3
label
label
3
2.5
2
1.5
1
0.5
0
-0.5
-1

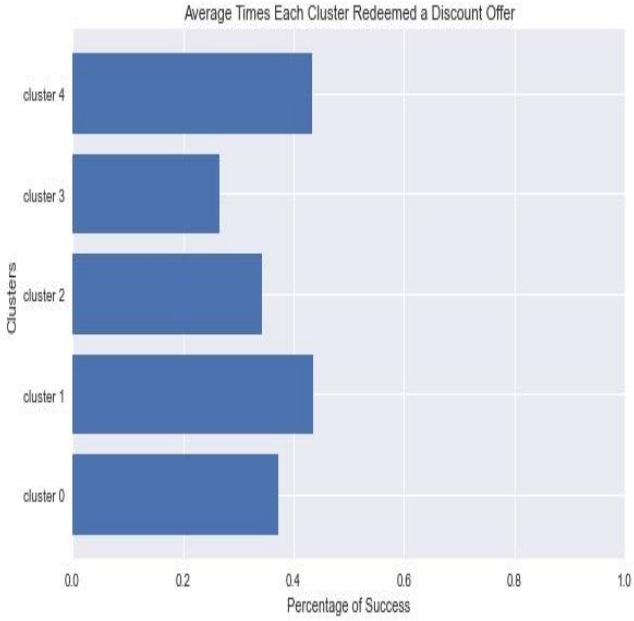

Distribution of Average Amount Spent by Each Customer in Each Cluster
labels
cluster 0
cluster 1
cluster 2
cluster 3
cluster 4
Number of People
Amount Spent


Total Number of Offers Sent to Customers in Each Cluster
labels
-1
0
1
2
3
Number of People
Number of Total Offers Sent

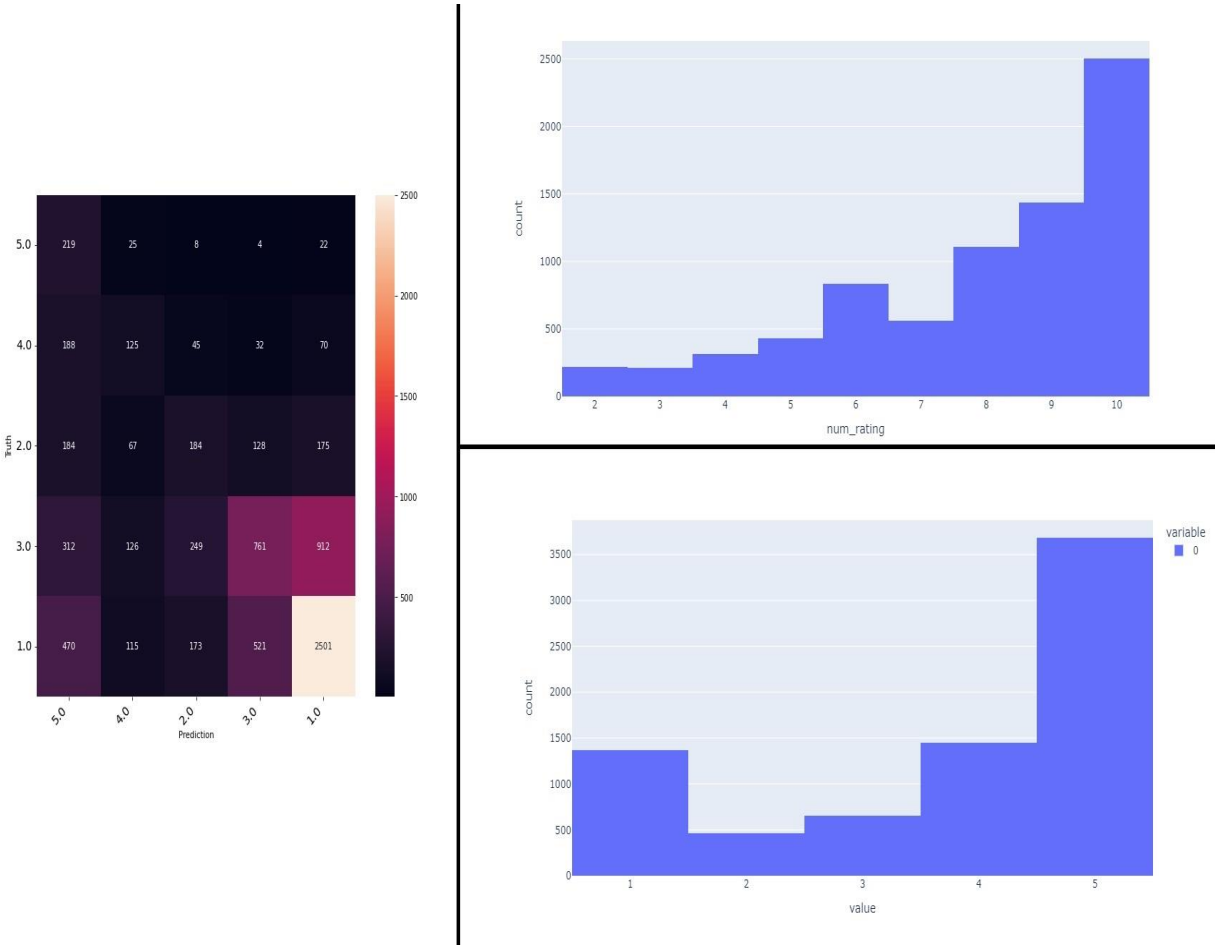

Number of BOGO Offers Sent to Customers in Each Cluster
labels
-1
0
1
2
3
Number of People
Number of BOGO Offers Sent


Number of Discount Offers Sent to Customers in Each Cluster
labels
-1
0
1
2
3
Number of People
Number of Discount Offers Sent

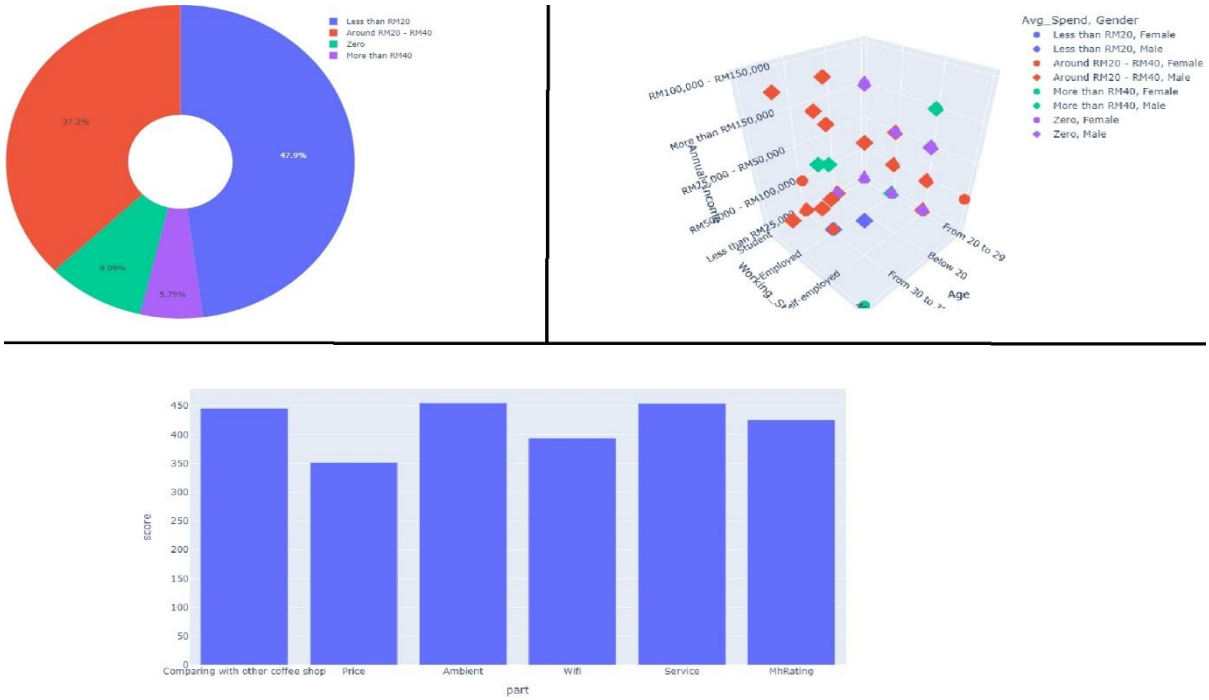

Average Times Each Cluster Redeemed a BOGO Offer
cluster 4
cluster 3
cluster 2
cluster 1
cluster 0
Clusters
Percentage of Success

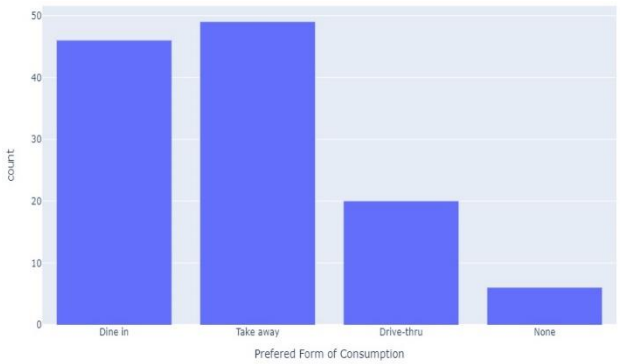

Average Times Each Cluster Redeemed a Discount Offer
cluster 4
cluster 3
cluster 2
cluster 1
cluster 0
Clusters
Percentage of Success

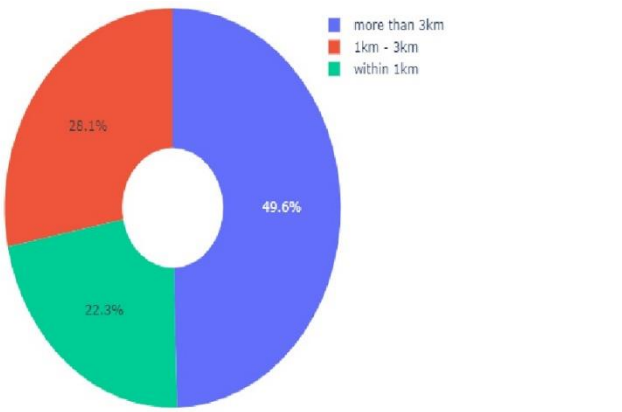
2. Sentiment Analysis

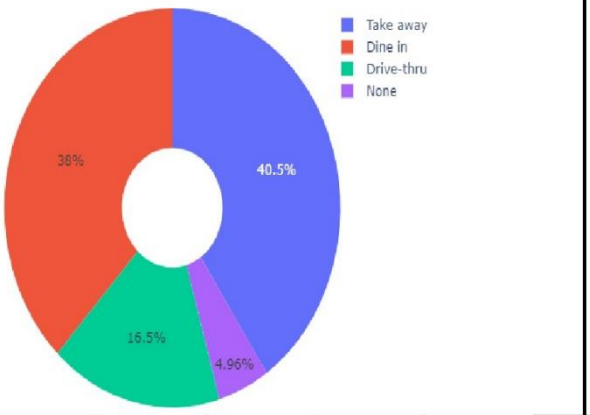


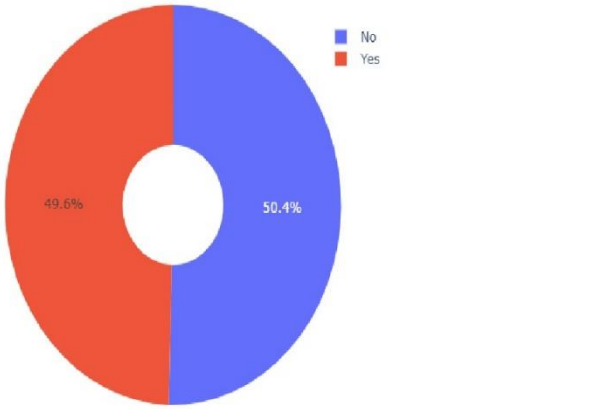
3. Behavior Analysis

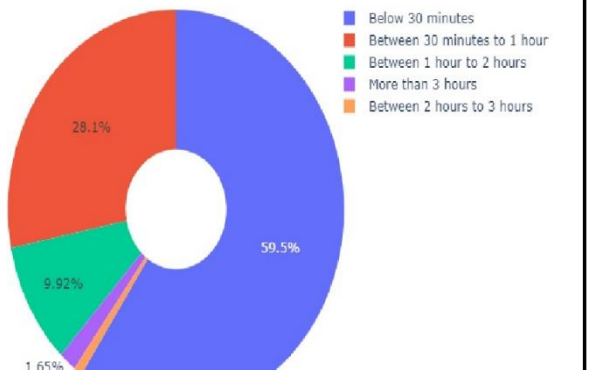


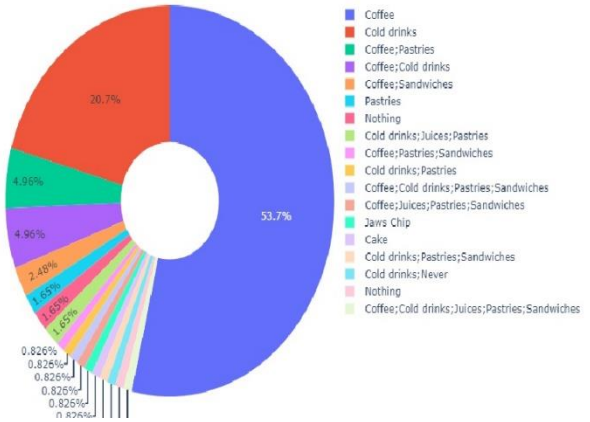

count
Dine in
Take away
Drive-thru
None
Prefered Form of Consumption

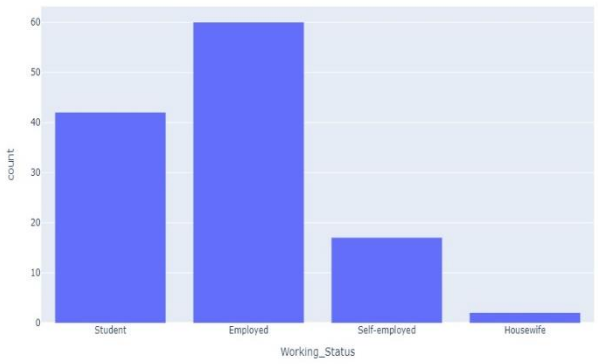

more than 3km
1km - 3km
within 1km
28.1%
49.6%
22.3%

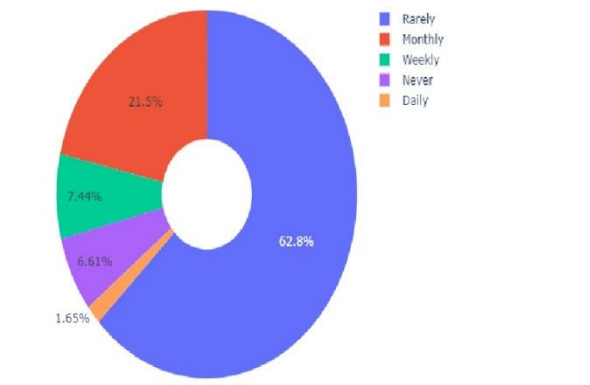

Take away
Dine in
Drive-thru
None
38%
40.5%
16.5%
4.96%

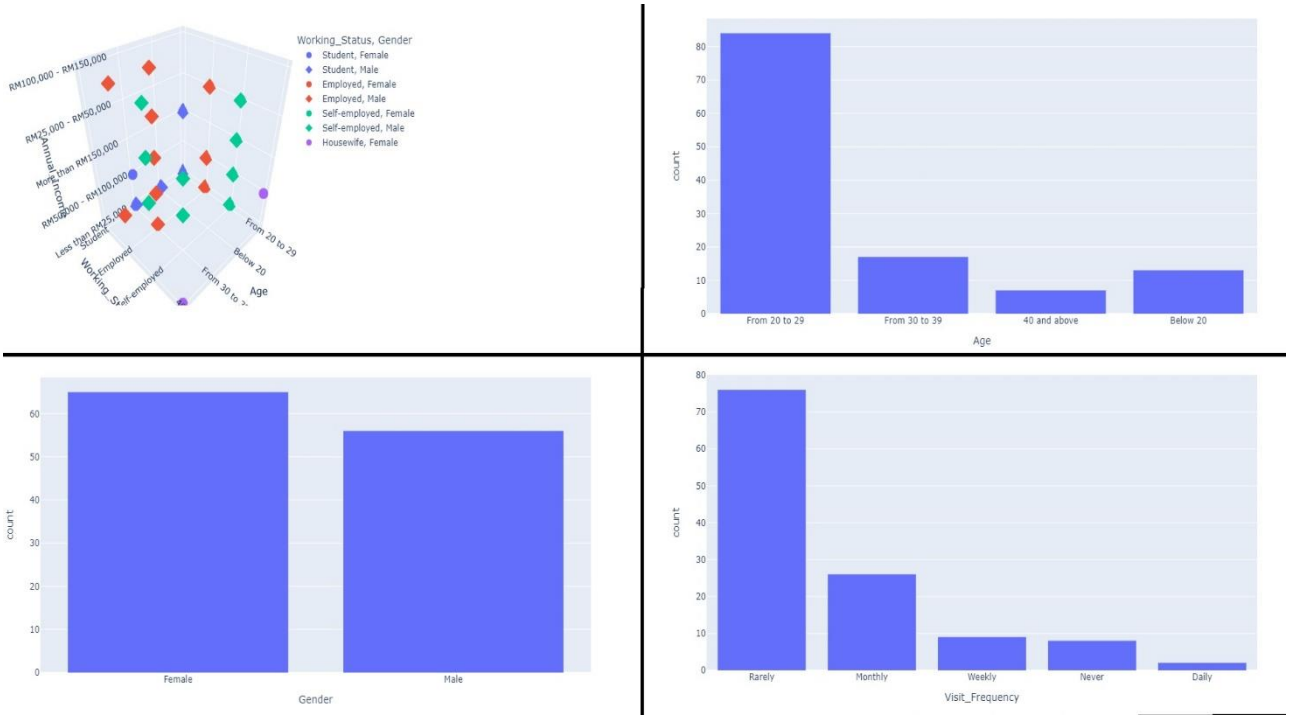

No
Yes
49.6%
50.4%


Below 30 minutes
Between 30 minutes to 1 hour
Between 1 hour to 2 hours
More than 3 hours
Between 2 hours to 3 hours
28.1%
59.5%
9.92%

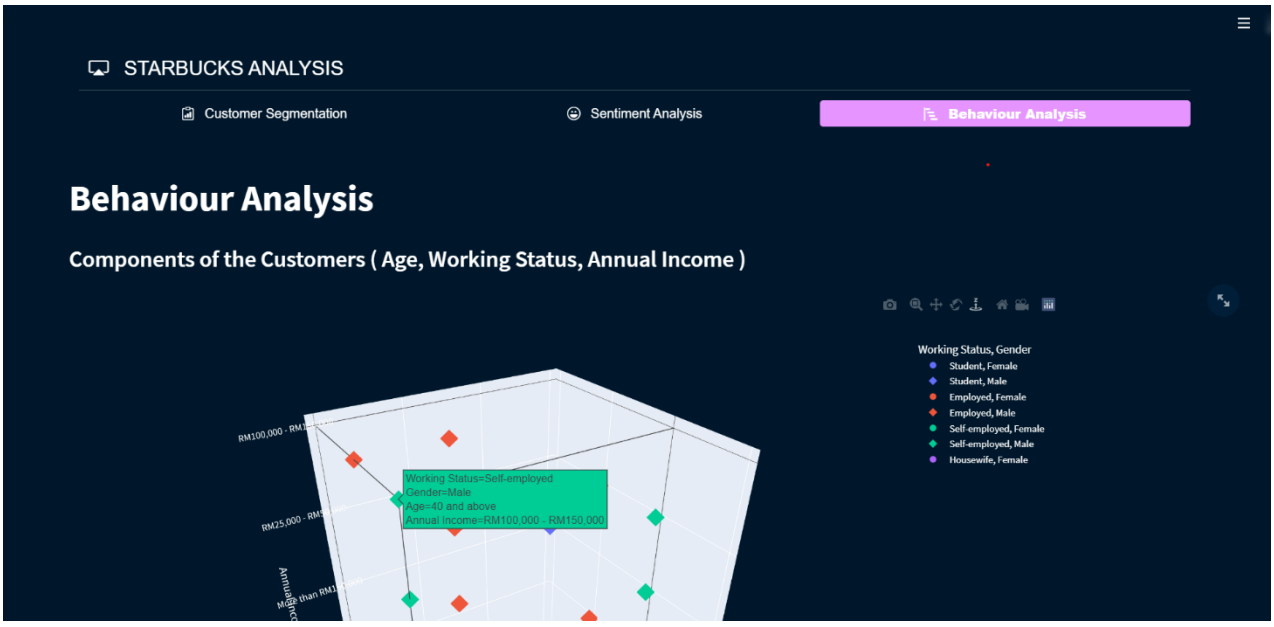

Coffee
Cold drinks
Coffee;Pastries
Coffee;Cold drinks
Coffee;Sandwiches
Pastries
Nothing
Cold drinks;Juices;Pastries
Coffee;Pastries;Sandwiches
Cold drinks;Pastries
Coffee;Cold drinks;Pastries;Sandwiches
Coffee;Juices;Pastries;Sandwiches
Jaws Chip
Cake
Cold drinks;Pastries;Sandwiches
Cold drinks;Never
Nothing
Coffee;Cold drinks;Juices;Pastries;Sandwiches
20.7%
53.7%
4.96%
4.96%
2.48%
1.65%
1.65%
1.65%
0.826%
0.826%
0.826%
0.826%
0.826%

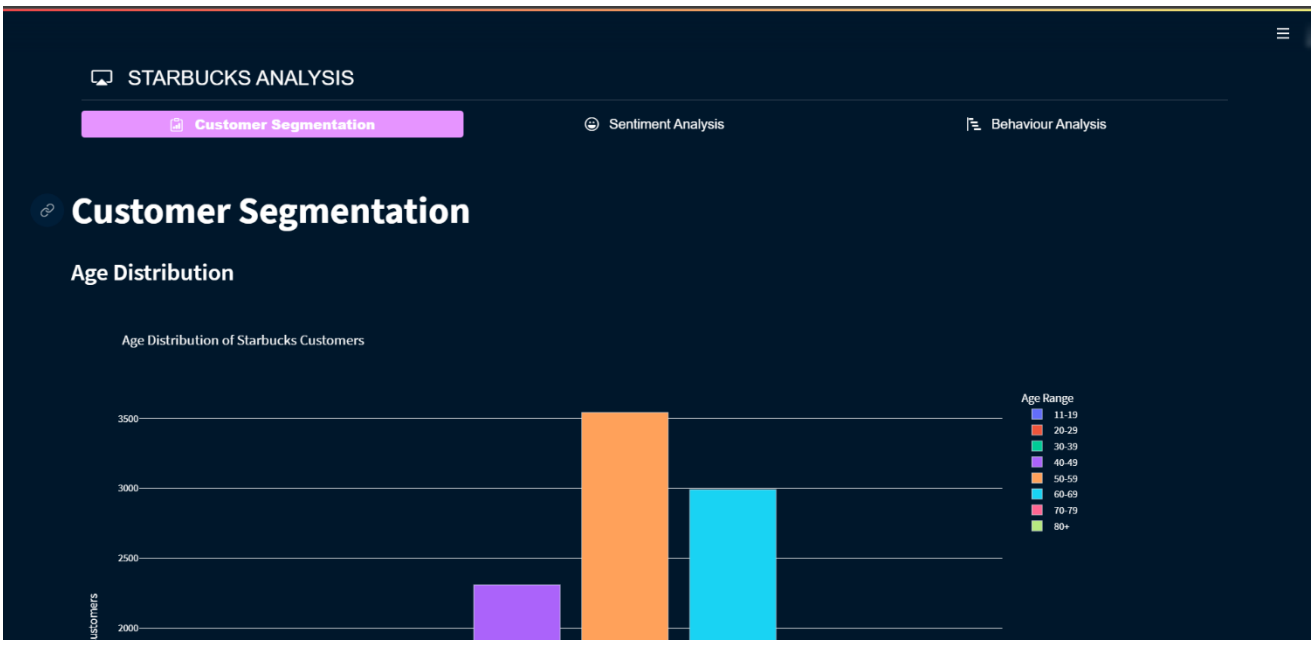

count
Student
Employed
Self-employed
Housewife
Working_Status

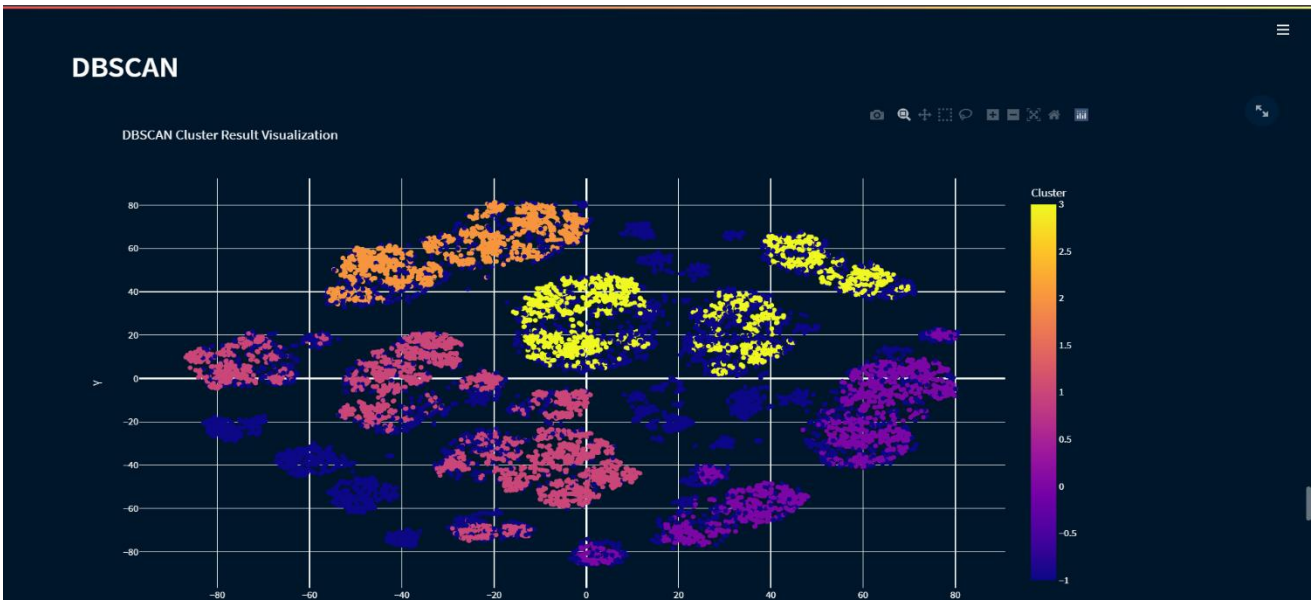

Rarely
Monthly
Weekly
Never
Daily
21.5%
7.44%
6.61%
1.65%
62.8%

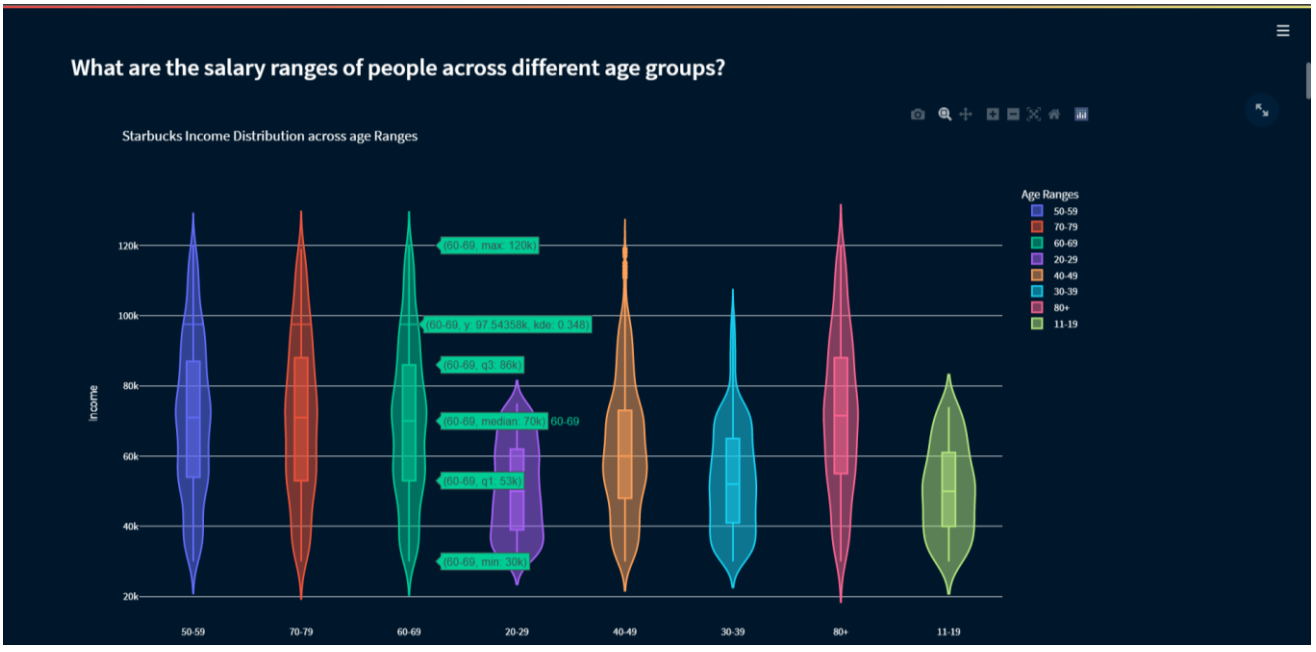


4. Dashboard (Frontend)




STARBUCKS ANALYSIS
Customer Segmentation
Sentiment Analysis
Behaviour Analysis
Customer Segmentation
Age Distribution
Age Distribution of Starbucks Customers
Age Range
11-19
20-29
30-39
40-49
50-59
60-69
70-79
80+
3500
3000
2500
2000


DBSCAN
DBSCAN Cluster Result Visualization
Cluster


What are the salary ranges of people across different age groups?
Starbucks Income Distribution across age Ranges
Age Ranges
50-59
70-79
60-69
20-29
40-49
30-39
80+
11-19
Income
120k
100k
80k
60k
40k
20k
(60-69, max: 120k)
(60-69, q3: 86k)
(60-69, median: 70k)
(60-69, q1: 53k)
(60-69, min: 30k)
50-59
70-79
60-69
20-29
40-49
30-39
80+
11-19

CHAPTER 9

CONCLUSION

- For the most part, the cannabis industry is in its nascent stages. The intense federal criminalization of cannabis for years totally hampered professional research into all facets of cannabis, from cultivation to retail to consumption. As a result, dispensaries are learning how to navigate not just a thicket of regulations and other constraints, but also an unclear road of consumer behaviour.

- Conducting direct research with consumers and products is not possible, so retailers must look inward to uncover the behaviours of their customers. Despite many retailers outside of cannabis have had tremendous success with traditional customer segmentation analysis, the supply of skilled analysts willing and capable to serve the cannabis industry is far smaller. To compound this, there is plenty of data in the cannabis industry— due to the enforcement of a traceability system— but few ways to access it. Although dashboards and elaborate interfaces have reasoned the responsibility of finding patterns or commonalities in retail data, none of them provides statistics or data that is rich enough to make advanced insights such as customer segmentations. As a result, it is necessary to bring in tools that are specifically built for situations such as this: machine learning.

- It has a capacity of analyzing the needs of the Customer.

- It has a target of reaching the Products & services for the particular group of Customers.

- For Customer Segmentation we are using DBSCAN over K-Means Algorithm as DBSCAN is more efficient & K-means is a Show stopper.

- The task of sentiment analysis, especially in the domain of micro-blogging, is still in the developing stage and far from complete. So we propose a couple of ideas which we feel are worth exploring in the future and may result in further improved performance

- Right now, we have worked with only the very simplest unigram models; we can improve those models by adding extra information like closeness of the word with a negation word. We could specify a window prior to the word (a window could for example be of 2 or 3 words) under consideration and the effect of negation may be incorporated into the model if it lies within that window. The closer the negation word is to the unigram word whose prior polarity is to be calculated, the more it should affect the polarity. For example, if the negation is right next to the word, it may simply reverse the polarity of that word and farther the negation is from the word the more minimized ifs effect should be.

CHAPTER 10

FUTURE ENHANCEMENT

We have done this project with as minimum flaws as possible and can further be enhanced by including major identification of statistics of people and improving the accuracy of the output. All the census data also can be collected to train the dataset even more to get more accurate outputs. Based on the social data accumulated, we can conclude that mall customer segmentation system can be used in a wide range of applications across a variety of domains including:

- Making the Project Live such that all the Excel or csv files will be updated regularly as per the data changes in the Company and as a result the Plots on the Dashboard will be Updated.

- Identifying interests of people while they are buying items from a mall

- Grouping people efficiently

- Identifying many more clusters to segment the products to improve the sales of the product

In this project we have implemented k-means algorithm, it can be further enhanced by using few complex algorithms such as conventional neural networks algorithms. All the census data also can be collected to train the dataset even more to get more accurate outputs There is no need for privacy invasion of users like other applications like amazon etc.

CHAPTER 11

REFERENCES

- https://technologyadvice.com/blog/marketing/customer-segmentation-methods/

- https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17#:~:text=Sentiment%20analysis%20is%20contextual%20mining,service%20while%20monitoring%20online%20conversations.

- https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15

- https://thesai.org/Downloads/Volume10No2/Paper_48-A_Study_on_Sentiment_Analysis_Techniques.pdf

- https://www.researchgate.net/publication/313737530_Review_on_Customer_Segmentation_Technique_on_Ecommerce#:~:text=This%20paper%20will%20review%20customer%20segmentation%20using%20data%2C,and%20survey%20data%20were%20as%20the%20external%20data.

Thank You