CE301- Individual Capstone Project

Project Title	: Applications in Data Clustering
Name	: Yew KaiShen
Registration number	: 2204263
Supervisor	: Panagiotis KANELLOPOULOS
Second accessor	: Nikolaos THOMOS
Degree course	: Computer Science

Acknowledgements

The very first person that I would like to begin with and acknowledge is my supervisor, Dr. Panagiotis KANELLOPOULOS, who had weekly meetings with me and made sure that I was always on schedule with my capstone project. Other than that, he also gave me some ideas on my technical stuff such as coding part and assisted me in how to make my project better.

Another person I would like to acknowledge is my Malaysian friend, YuXin who helped me to solve some issues on the Gitlab. It was very lcuky that he also used the same software as me, Visual Studio Code, and because of that, he taught me how to link between Visual Studio Code and Gitlab and how to push and commit file correctly to the Gitlab.

Last but not least, I would like thank my family for their support throughout this difficult time, espicially my father, who always worried my studies.

Abstract/Summary

Nowadays, businesses will rely on new innovations which will produce new products as many as possible to try to attract more customers but it will leave them frustrated as there are too many options and also companies have no idea on which type of customers they should target for product sales. This is where the machine learning plays an important part and few algorithms will be applied to discover some hidden patterns in their customers' purchasing behaviours. By doing this, it will help the companies make a suitable marketing strategy and improve decision-making in the future. This can be done by applying customer and it is used to segmenting the customers with similar behaviours into one group and dissimilar behaviours will be in different groups.

In this project, clustering algorithms are being implemented to group the customers and the results obtained from different clustering algorithms will be compared. Elbow method is being used to find the number of clusters most suitable to the dataset. However, there is no accuracy for cluster analysis because you use it when you don't know the clusters.

Table of Contents

List of figures

List of Tables

List of Symbols

ML – Machine Learning

GMM – Gaussian Mixture Model

DBSCAN – Density-based Spatial Clustering of Applications with Noise

AHC – Agglomerative Hierarchical Clustering

1.0 Introduction

1.1 Background of Study

Data clustering or cluster analysis is an unsupervised machine learning and it is designed to use a data analysis technique of grouping a set of objects where these objects in one cluster have the same characteristics and objects in a different cluster are relatively different. However, the main goal of clustering is to explore different patterns or hidden information from a raw dataset.

Clustering applications have been developing rapidly over the years such as image classification, recommendation engines, document analysis, etc., and especially in business industries, they keep producing products to attract more and more customers. Hence, the customers' information they need to store will become massive. This information such as age, marital status, birth of date, purchasing behaviors, and so on is extremely important as it will help the business to make the right decision on which type of customers they should target.

Hence, customer segmentation has come to play an increasingly important part in the business industry as it can group customers with similar characteristics into one group and similar characteristics will be in dissimilar different groups. By doing this, it will improve the business competitiveness by making the right decision on which type of customers they should target.

Hence, the aim is to develop an approach to discovering hidden patterns, information from a raw dataset and applying different clustering algorithms, and choosing the best outcome by comparing different results obtained from different algorithms.

1.2 Problem Statements

One of the reasons why a business will fail is they don't know how to When the number of customers, as well as the information of customers, is increasing, this will lead the companies to a big problem which is they don't know what and how to deal with the information to keep their customers and know what type of customers they should focus on.

1.3 Objectives

The foundation for sending the correct message to the right customer at the right time is laid by customer segmentation. It allows the companies to deeply comprehend the drives and wants of their customers, not only to provide better customer service but also the ability to run more efficient and cost-effective campaigns. Even though the number of customers is increasing rapidly, the company still doesn't know what and how to deal with the information of customers to know their customers' preferences and likings since every customer is unique. Hence, the objectives of this project are as follows:

- develop an approach to explore hidden information from a raw dataset
- apply different clustering algorithms
- compare the results obtained from different clustering algorithms
- choose the most suitable clustering algorithm

2.0 Literature Review

There is various type of machine learning in the Information Technology industry and this project will mainly focus on unsupervised machine learning. Unsupervised machine learning is the opposite of supervised learning as the data set does not contain any labeled data which means a training set is not needed so the machine learning algorithm will be more complex and time-consuming compared to supervised. The well-known unsupervised machine-learning technique that will be discussed throughout the project is clustering or cluster analysis. Clustering is an unsupervised machine-learning problem used for exploratory data analysis. The main purpose of cluster analysis is to group the data set into different clusters where the data that has similar characteristics to each other will be in one group, and dissimilar characteristics will be in a different group. Cluster analysis has always been broadly used in many applications such as customer segmentation, document analysis, image recognition, and so on[1]. Due to a programming-intensive project, I will split it into two parts which are the applications and algorithms. By doing this, you will have a deep understanding the concept of applications of data clustering.

2.1 Applications

There are a few sectors or fields where cluster analysis techniques have been successfully applied such as image segmentation, pattern recognition, academic industry, customer segmentation, and so on.

2.1.1 Image Segmentation

Image segmentation is playing an important role in image processing and it can help us to analyze and identify what are the objects inside the image[2]. For instance, if we have an image and we would like to know if there is something that we want inside the image, we may need image segmentation to help us to identify and separate each object individually and group the similar object in one group and different objects in another group. In other words, image segmentation is the classification of an image into different groups.

2.1.2 Pattern recognition

Sometimes machine learning algorithms can also be used to automatically identify patterns and regularities in data by using one of the data analysis methods which is the pattern recognition[3]. This data or information can be taken from different types of data such as text, images or sounds or any data as long as it can be recognizable. Besides that, it can also recognize and identify any unfamiliar objects and shapes from different angles, and even the objects are not clear. One of the benefits of using pattern recognition is it can be applied to different fields such as speech recognition, license plate recognition, and voice-based and face-based authentication. There are some advantages of using pattern recognition as it can help us to solve the problem of fake biometric detection and analyze objects from different angles. On the other hand, it is very a complex machine-learning algorithm and it takes a long time to process it.

2.1.3 Academic industry

The main purpose of applying cluster analysis to the academic industry is to classify if the student is active or non-active and help to maximize their academic performance[4]. Due to the evolution of technology, it is very normal for students who are studying in college and university to own a laptop for their studies, thus, the student's information relating to educational institutions is increasing rapidly. It is very useful to apply cluster analysis in the academic field as it has the ability to mine the student's data and analyze their e-learning behavior. With the use of data mining from these vast volumes, the educational management will be able to make academic decisions. Hence, early prediction and decisions at an early stage will help to identify active students and assist the non-active students by putting more effort in developing educational programs to improve the student's overall performance.

2.1.4 Customer segmentation

Customer segmentation is one the most important tools in the market industry as it has unlimited potential that we can't see and it can lead businesses towards more efficient ways to advertise their products and create a new one if needed[5]. The basic concept of customer segmentation is to assist the business company to classify their customers based on some criteria such as customers' value, purchasing behavior, preferences, and some other information. By doing this, the business company will know which group of customers they should put more attention to as the customers belong to the same group have similar characteristics while different group of customers have distinct characteristic. There are some advantages of applying customer segmentation to the market industry such as the companies will have a deeper understanding about their customer by knowing what they like to purchase or not, how often they purchase and how much they spent every time they make a purchase, increase the company value and profit by putting more effort to their target customer.

2.2 Types of Clustering Method and Algorithms

There are several types of clustering method and each of type of clustering method has their own different clustering algorithms[6]. Since there are too many clustering method and algorithms, I will pick four well-known clustering method and one clustering algorithm for each method. Also, I will apply these four clustering algorithms into my capstone project. Hence, I will split it into 2 different categories, one is clustering method, and the other one is clustering algorithm.

2.2.1 Centroid-based Clustering Method

When we talk about centroid-based clustering, the K-means clustering algorithm is one of the most widely used algorithms for this method where K is the number of clusters that we have to set the number manually[7]. Centroid-based clustering methods are designed to partition the data into 'K' number of clusters and the 'K' number is the thing we have to figure out what number is the most suitable number to be applied to the data set. In other words, you can think of it as a post office

location problem, where your task is to figure out which location is the best to place a post office by giving a number of residents where 'post office' is the center of the cluster and the 'residents' is the data that is clustered. By figuring out what number is the best for 'K', we have to keep measuring the distance between each data again and again and point out the nearest cluster's centroid by applying some distance calculation which I will explain later. There are some pros and cons to applying centroid-based clustering, such as it is the fastest centroid-based algorithm and also it can apply to large data sets, but the performance will be affected if there are too many outliers in the data and the 'K' number has to choose wisely.

2.2.1.1 K-Means Clustering

K-Means is one of the most popular and simplest unsupervised machine learning algorithms that is being used to group data points into a specified number of clusters (K)[8]. Once you figured out what is the optimal number of clusters, basically you have achieved the objective of the K-means algorithm. The number of clusters (K), also refers to the number of centroids and will allocate every data point to the nearest cluster. The process of K-means is very simple, you can simply set a random number of clusters and performs iterative calculations until you achieve a reasonable result.

2.2.2 Density-based Clustering Method

This algorithm is designed to discover the clusters in different sizes and shapes. It is defined as each cluster having a typical density value where this value is higher than outside the cluster. Data points with lower density are considered as outliers or noise points. One of the most common algorithms in this method is Density-based Spatial Clustering of Applications with Noise (DBSCAN)[9]. This algorithm has two parameters that we need to set manually which are the distance and the minimum of data points that are enough to form a cluster. The concept of this algorithm is the density value of a data point is the number of data points within the radius of the data point, where there is no limit to the number of data points. Thus, the high density of a data point is defined in a separate cluster from other clusters where these clusters have low density. However, it does not perform well if the data set

is high dimensional. In other words, if the data set has too many variables, this algorithm is not well fitted as the data points will be too close to each other and will generate only 1 cluster.

2.2.2.1 Density-based Spatial Clustering of Applications with Noise (DBSCAN)

Density-based Spatial Clustering of Applications with Noise (DBSCAN) is one of the density-based clustering methods. It is quite different from the well-known clustering algorithm which is the K-Means, in K-Means, the cluster is depending on the mean value so each data point in the data set is playing an important role and a small change in a data point will affect the whole result and you have to set the number of cluster 'K' manually, but in DBSCAN, you don't have to specify the number, instead, you need to calculate the distance between each data point and what is the minimum of data points that enough to form a cluster[10]. Sometimes, DBSCAN will produce a more reasonable result compared to K-Means. In other words, it can produce the result in a different shape and size depending on the data. However, K-Means will perform poorly if the data set contains outliers but DBSCAN can handle the outliers. When comes to the implementation of DBSCAN, it has more steps compared to other algorithms that I have applied.

2.2.3 Distribution-based Clustering Method

Distribution-based clustering method is designed to assign each data point to the cluster that corresponds to the distribution that has the highest likelihood. It is a type of clustering method that will assume the data point is generated from the model, the data point corresponds to a cluster, and assign each data point to the cluster with the highest likelihood[11]. Likelihood is a measurement of how likely a data point is belong the a particular cluster. For instance, if we want to perform customer segmentation and group the customers based on their purchasing habits. The algorithm will assume that each cluster has its own purchasing habit and this cluster will try to assign each data point to the appropriate cluster. Overall, it is designed to discover the hidden patterns in a data set and group the similar data point together based on the likelihood.

2.2.3.1 Gaussian Mixture Model

A Gaussian mixture model is a probability model that combines different Gaussian distributions to represent a probability distribution. In machine learning, it is always used for clustering and density estimation. In other words, assume there is a number of distributions, and each of these distributions represents a cluster, and GMM is used to group the data points that belong to the distribution[12]. It has a parameter that we need to set manually which is the 'n_components' and it also represents the number of clusters in the data. The number of clusters will have a significant impact on the result as if the number is too small, the data points will tend to be together and clusters as well, but if the number is too big, there will be too many clusters and each cluster will have fewer data points.

2.2.4 Hierarchical Clustering Method

Hierarchical clustering method is a type of clustering algorithm that will assign each data point as a cluster and merge these clusters together based on the measurement until there is only one single cluster[13]. There are two types of clustering algorithms, Agglomerative and Divisive. Agglomerative Hierarchical clustering is a bottom-up clustering which means it will assign each data point as a cluster and merge them with the most similar criteria until there is only one cluster. Whereas, the divisive algorithm is a top-down clustering method which means it will start with one single data point, also known as a cluster that contained all the data points, and divide the big cluster into smaller cluster until there is only one small cluster. This means the result generated from AHC is represented as a dendrogram, which looks like a tree diagram. The concept of hierarchical clustering is from 'n' groups to 1 group or vice versa. So, we have to find and pick the smallest distance between two data points and connect them to become one group. Once the group is formed, this group will find another group with the smallest distance and merge. Eventually, it will end up with a single cluster. To calculate the distance, it has to follow three criteria such as single linkage; complete linkage, and average linkage. Single linkage means it will take two data points with the closest distance and merge them together. Complete linkage refers to two furthest data points and links together. Average linkage refers to the average distance between all paired points. Moreover, it is also one of the methods to improve the performance of AHC by measuring the distance between each cluster, which also

determines the quality of the clustering result. To determine which linkage is the best fit, we can run all the linkages and then visualize it by using the Within-Cluster Sum of Square method where a lower score means the linkage is not well fitted to the data set; a higher score means the linkage is well fitted.

2.2.4.1 Agglomerative Hierarchical

Agglomerative Hierarchical clustering is using a technique called dendrogram to group similar objects. In other words, is a bottom-up clustering approach where it will start with big clusters and will have sub-clusters and these sub-clusters will have other sub-clusters, and so on until all the data points belong to a single cluster. Thus, this will produce a result that looks like a tree[14].

3.0 Implementation

Data clustering is one of the data mining tools but it can also be considered in the field of data science. Data mining is defined as a subfield of data science that is designed to focus on discovering and extracting useful data, and hidden information from an unlabelled data set. Specifically, data clustering is a common data mining technique used to identify groups of data where these data have common criteria. Whereas, data science is the study of how to extract useful information from structured and unstructured data by applying different tools and techniques. Hence, the most challenging part of my project is the implementation part as it contains many different steps and does not have a fixed process flow. Thus, whenever I need to modify or manipulate the original data set, I will create a copy of the data set and name it with the name that relates to the step that I am going to perform. Besides that, my project is all about coding and it is an intensive programming project that contains a lot of different sections and has to make sure everything is working correctly in order to achieve an understandable result. I will split it into nine different sections where I will explain briefly in each section.

3.1 Data Preparation

3.1.1 Data Collection

```
import matplotlib.pyplot as plt
from yellowbrick.cluster import KElbowVisualizer
from yellowbrick.cluster import SilhouetteVisualizer
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import silhouette_score
from kneed import KneeLocator
pd.options.display.max_columns = None
pd.options.display.max_rows = None
```

✓ 26.6s

```
# import the data file as data
data = pd.read_csv('C:\\Users\\yew_k\\OneDrive\\Desktop\\CE301-Individual Capstone Project\\marketing_campaign.csv')
```

✓ 0.1s

```
# print the first five row of the data in order to understand the contents
data.head()
```

✓ 0.1s

Outputs are collapsed ···

```
# to check null values and the data types
data.info()
```

✓ 0.1s

Outputs are collapsed ···

```
# check what are the categorical values
print("Total categories in the feature Marital_Status:\n", data["Marital_Status"].value_counts(), "\n")
print("Total categories in the feature Education:\n", data["Education"].value_counts())
```

Figure 1: Data Collection

In the picture shown above, there are few packages and libraries provided by Python that have been imported for the use of data analysis, visualization and clustering. The first four packages are the most widely used for data mining, like 'pandas' which is used to manipulate and analyze the data set, 'numpy' is imported to interact with numerical values, and the last two 'seaborn' and 'matplotlib' are designed for statistical data to create static and interactive visualization. And the rest of the scripts are describing various modules from different libraries that are being imported and these modules and libraries are the most important tools for me to perform data clustering. There are two modules such as StandardScaler and LabelEncoder, the most important modules as it is used to make sure the data

set is cleaned and ready to perform different clustering algorithms. Some of the tools are used to select the optimal number of clusters by visualizing the silhouette scores and reducing the dimensions of the data. After importing the modules and libraries, it is ready to import the data file, and I named it 'data' by using 'pd.read_csv' along with the file location, and the location can be changed based on the users. Next, I will print the first five rows of the data to get a rough understanding and check what the data types that existed in my data set by using the 'head()' and 'info()'. I saw there are two object data types and I have to check what are the values because I need to ensure all the data are numerical by converting them from object to integer which I will perform in the later section.

3.1.2 Data Cleaning

After collecting and understanding the dataset, I will be performing data cleaning which is the process of correcting or removing inaccurate or missing data and reformating incorrect data types in the data set. This process is one of the most important steps in data mining as if the data is not in the correct data types, the algorithms will be unreliable, and the results will be inaccurate even if the results may look reasonable.

```
# drop the null values
data = data.dropna()
print("The total number of data-points after removing the rows with missing values are:", len(data))
```
✓ 0.2s

Figure 2: Dropping data

The first thing I will do in this step is to remove all the outliers by using the 'data.dropna()', and you can see I specified the 'data' which I assigned before. Instead of filling back the value by the mean or median of the specific column because I think removing is safer compared to filling back the missing value.

```
# convert the Dt_Customer data type to date time
data["Dt_Customer"] = pd.to_datetime(data["Dt_Customer"])
dates = []
for i in data["Dt_Customer"]:
    i = i.date()
    dates.append(i)
# check the latest and oldest date of the customer enrolment
print("The newest customer's enrolment date in therecords:",max(dates))
print("The oldest customer's enrolment date in the records:",min(dates))
```

✓ 0.2s

```
The newest customer's enrolment date in therecords: 2014-12-06
The oldest customer's enrolment date in the records: 2012-01-08
```

```
# create a list to store the days of each customer enrolment
days = []
day1 = max(dates)
for i in dates:
    totalDays = day1 - i
    days.append(totalDays)
data["Customer_Days"] = days
data["Customer_Days"] = pd.to_numeric(data["Customer_Days"], errors="coerce")
```

✓ 0.0s

Figure 3: Add column Customer_Days

Next, I will convert the data type of the 'Dt_Customer' column to the date time format which is easier for me to understand, and extract it for doing some calculations. I will append the values to a 'dates' list and find the max value by using the 'max()'. Then, the max value will be used for calculating how many days the customer has been registered for each customer in the company, and convert the values to numeric by using 'pd.to_numeric' and storing in a new column 'Customer_Days'.

```
#Age of customer
data["Age"] = 2022-data["Year_Birth"]

#Number of children
data["Children"]=data["Kidhome"]+data["Teenhome"]

#Education in 3 different level
data["Education"]=data["Education"].replace({"Basic":"Undergraduate","2n Cycle":"Undergraduate",
"Graduation":"Graduate", "Master":"Postgraduate", "PhD":"Postgraduate"})

#Total Spent
data["Total_Spent"] = data["MntWines"]+ data["MntFruits"]+ data["MntMeatProducts"]+ data["MntFishProducts"]
+data["MntSweetProducts"]+ data["MntGoldProds"]

data['Is_Parent']=np.where(data.Children>0,1,0)

#Marital Status
data["Marital_Status"]=data["Marital_Status"].replace({"Married":"Partner", "Together":"Partner",
"Absurd":"Alone", "Widow":"Alone", "YOLO":"Alone", "Divorced":"Alone", "Single":"Alone",})

#Product Names
data=data.rename(columns={"MntWines": "Wines","MntFruits":"Fruits","MntMeatProducts":"Meat",
"MntFishProducts":"Fish","MntSweetProducts":"Sweets","MntGoldProds":"Gold"})

# drop some redundat attributes
to_drop = ["Dt_Customer", "Z_CostContact", "Z_Revenue", "Year_Birth",
           "ID","Complain","Response",'AcceptedCmp3', 'AcceptedCmp4',
           'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2']
data = data.drop(to_drop, axis=1)
```

Figure 4: Manipulating Data

After that, I will be performing feature engineering where I will manipulate some of the attributes by adding, removing, or renaming. I will add a new column 'Age' and store the value by subtracting the current year 2022 and 'Year_Birth'. A new column also be created and named 'Children' and take two values from 'Teenhome' and 'Kidhome' and sum up. Then, I will rename the 'Education', 'Marital_Status', and all the product names values for a better understanding, add a new column 'Total_Spent', and insert the value from summing up all the products. A new column 'Is_Parent' check if the customer is a parent by checking the 'Children' value if more than 0 then yes or no. After manipulating all the values, I will drop some of the redundant or useless features where these features don't affect our result.

```
data.describe()
✓ 0.1s
Outputs are collapsed ···

#Dropping the outliers by setting a cap on Age and income.
data = data[(data["Age"]<90)]
data = data[(data["Income"]<600000)]
print("The total number of data-points after removing the outliers are:", len(data))
✓ 0.0s
```

Figure 5: Checking and dropping outliers

The last step of data cleaning is also one of the most important steps, I will check if there is any outliers by using '.describe()' and checking the mean and max value and removing the outliers or setting a range and removing the values if outside the range.

3.2 Exploratory Data Analysis

```
#correlation matrix
corrmat= data.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corrmat,annot=True, center=0)
✓ 4.0s
```

Figure 6: Create correlation matrix

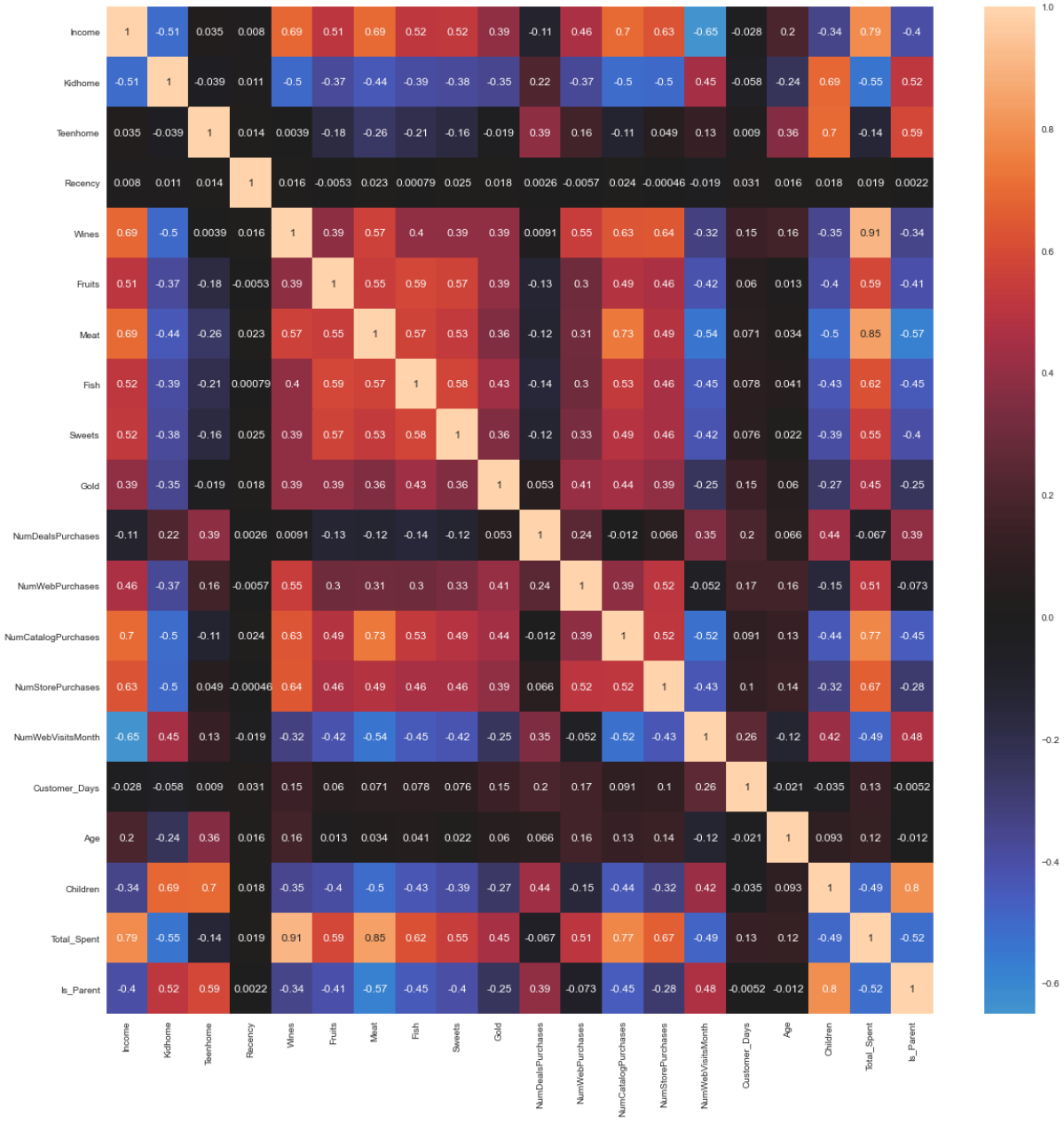

Figure 7: Correlation Matrix

In this section, I will use a correlation matrix and visualize it using a heatmap by using two modules which are '.corr()' and 'sns.heatmap(). The first module is used to calculate the correlation coefficient between two variables for all the columns of the data set and will produce a result in a square matrix format where each value represents the correlation between two variables. If the value is closer to one means the two variables have a high positive correlation, whereas if the two variables have a negative correlation, the value will be closer to -1. The second is the library from Seaborn which is used to plot the correlation matrix as a heatmap

3.3 Data Preprocessing

```
# get the object attributes
data_prep=data.copy()
s = (data_prep.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables in the dataset:", object_cols)
```

✓ 0.0s

```
Categorical variables in the dataset: ['Education', 'Marital_Status']
```

```
#Label Encoding the object dtypes.
LE=LabelEncoder()
for i in object_cols:
    data_prep[i]=data_prep[[i]].apply(LE.fit_transform)
print("All features are now numerical")
```

✓ 0.1s

```
All features are now numerical
```

```
data_prep.info()
```

✓ 0.1s

Figure 8: Change data type

After visualizing the heatmap in the previous step, I will perform a step called data preprocessing which includes the steps of transforming or encoding the data types so that the machine is ready to decode it. The main purpose of this step is to ensure the algorithms easily interpret the data features and the model will produce an accurate and precise result. First, I need to create a copy of the original data by using 'data.copy()' and name the new data set as 'data_prep'. Then, I will check the data type of each column if the data type is object. Then, I will use a library from scikit-learn which is used for encoding the categorical values. It is used to transform all the categorical values into numerical labels so that they can be used as input to the machine learning models.

3.4 Data Standardization

3.4.1 Standard Scaler

```
#Scaling
scaler = StandardScaler()
scaler.fit(data_prep)
data_scaled = pd.DataFrame(scaler.transform(data_prep),columns= data_prep.columns )
print("All features are now scaled")
```

✓ 0.0s

```
All features are now scaled
```

```
#Scaled data to be used for reducing the dimensionality
print("Dataframe to be used for further modelling:")
data_scaled.head()
```

✓ 0.0s

```
Dataframe to be used for further modelling:
```

	Education	Marital_Status	Income	Kidhome	Teenhome	Recency	Wines	Fruits	
0	-0.893586	-1.349603	0.287105	-0.822754	-0.929699	0.310353	0.977660	1.552041	1
1	-0.893586	-1.349603	-0.260882	1.040021	0.908097	-0.380813	-0.872618	-0.637461	-0
2	-0.893586	0.740959	0.913196	-0.822754	-0.929699	-0.795514	0.357935	0.570540	-0
3	-0.893586	0.740959	-1.176114	1.040021	-0.929699	-0.795514	-0.872618	-0.561961	-0
4	0.571657	0.740959	0.294307	1.040021	-0.929699	1.554453	-0.392257	0.419540	-0

Figure 9: Apply Standard Scaler

After transforming all the categorial data types into numerical labels, I will convert the data into a standard format that computer can easily understand and interpret by using Standard Scaler which is used for scaling numerical data. Next, I need to specific why data set I want to scale it by using 'scaler.fit()' where the scaler is an instance of the StandardScaler class. Then, I will apply the fitted scaler on the original data set 'data_prep' to transform all the data features into a scaled data set and name the scaled data set as 'data_scaled'. Last, I will print the first five rows of the data set to see how the data set looks like after scaling. Standardized data will allow the system to communicate and exchange data consistently and easier to process and analyze the data.

3.5 Dimensionality Reduction

3.5.1 Principal Component Analysis (PCA)

```
pca = PCA()
pca.fit(data_scaled)
```
✓ 0.6s

```
▾ PCA
PCA()
```

```
pca.explained_variance_ratio_
```
✓ 0.0s

Outputs are collapsed ···

```
plt.figure(figsize=(10,8))
plt.plot(range(1,23), pca.explained_variance_ratio_.cumsum(), marker='o',linestyle='--')
plt.title('Explained Variance by Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
```
✓ 0.2s

Outputs are collapsed ···

```
pca = PCA(n_components=10)
pca.fit(data_scaled)
pca.transform(data_scaled)
data_pca = pca.transform(data_scaled)
```
✓ 0.1s

Figure 10: Apply Principal Component Analysis

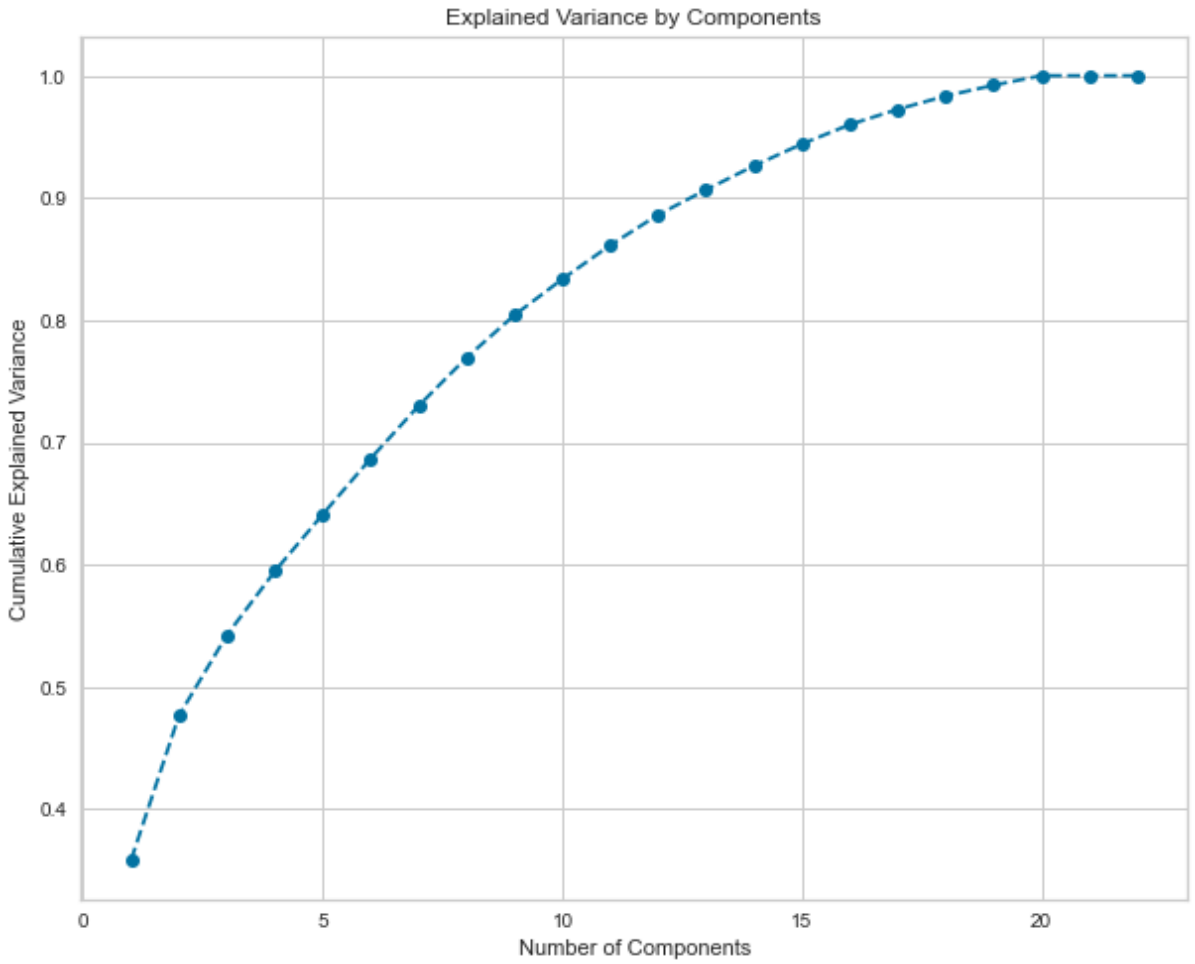

Figure 11: Principal Component Analysis

Dimensionality reductions is a technique used to reduce the number of input variables in the training data[15]. This technique is used when only dealing with high dimensional data set, which means your data set will have many different data variables. There are many techniques that can be used for dimensionality reduction, and I will perform one of the most widely-used technique which is the Principal Component Analysis, to transform a high dimension of data set into a low dimension while keeping as much information as possible[16]. To perform this technique, I will take the 'data_scaled' from the previous step to fit the PCA instance by using '.fit()'. The '.fit()' method of the PCA class computes the pricinpal components of the data that explain the maximum of variance in the data. After fitting the scaled data, I will plot the cumulative explained varaince by components by putting x-axis as number of components and y-axis as the cumulative sum of the percentages of explained variance ratio. From the plot, I will choose 10 components as the explained variance ratio is between 80% to 90% and it can avoid overfitting. Next, I will create a new PCA model and set the component

as 10 and fit the model to the 'data_scaled' and project the 'data_scaled' onto the new principal components and store it in a new data set 'data_pca'. And this 'data_pca' is ready to be used to find the optimal number of cluster.

3.6 Find optimal number of clusters

There are many methods to be used to determine the optimal number of clusters, but it is also one of the methods to evaluate the model. In this step, I used three methods to determine the optimal number of clusters which are Sum of Square Error (SSE), Within-Cluster of Sum of Square (WCSS), and Silhouette.

3.6.1 Sum of Square Errors (SSE)

```
kmeans_kwargs={
    "init":"random",
    "n_init": 10,
    "random_state": 1,
}

sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(data_pca)
    sse.append(kmeans.inertia_)

#visualize results
plt.plot(range(1, 11), sse)
plt.xticks(range(1, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
```

✓ 7.6s

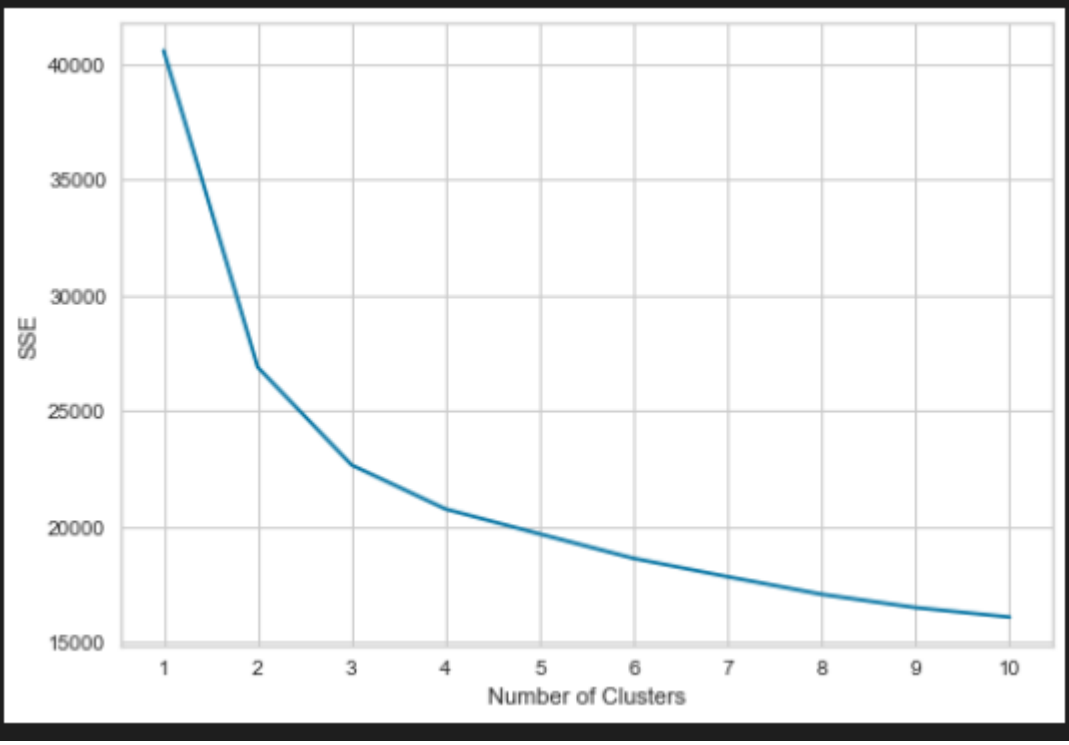

Figure 12: Apply Sum of Square Errors

The sum of square errors measures the variation of modelling errors. In other words, the lower the error, the model the better explain the data, while higher error means the model poorly explain the data[17].

3.6.2 Within-Cluster Sum of Square

```
from warnings import simplefilter
simplefilter(action='ignore',category=FutureWarning)

wcss = []
for i in range(1,10):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(data_pca)
    wcss.append(kmeans.inertia_)
```

✓ 6.1s

```
plt.figure(figsize = (10,10))
plt.plot(range(1,10), wcss, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.title('K-means Clustering')
plt.show()
```

Figure 13: Apply WCSS

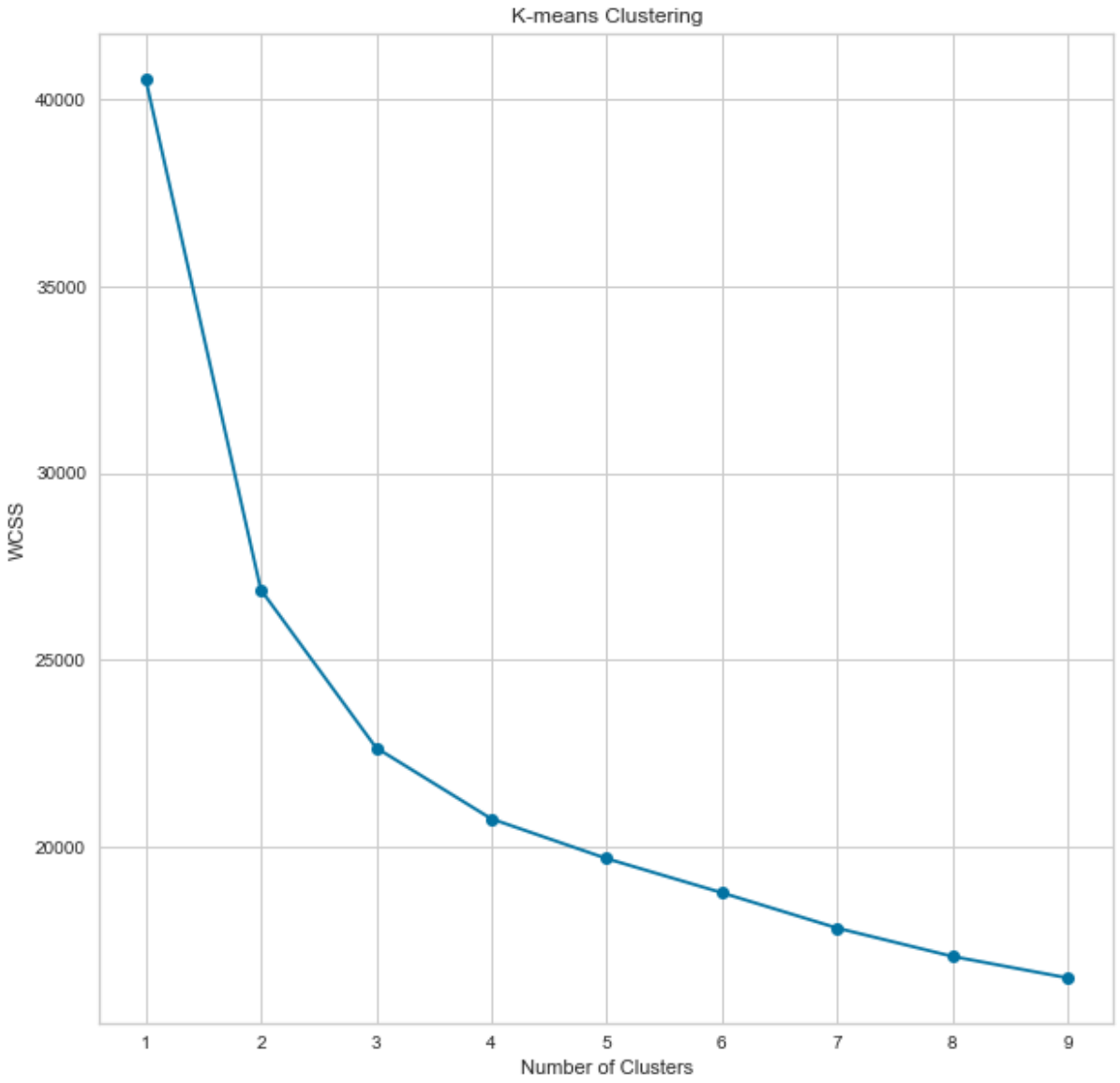

Figure 14: WCSS

Whereas, WCSS is the sum of squared distance between the data point and the centroid of the cluster[18]. There is one common thing in these two methods is as the number of cluster increase, the value will start to decrease. But, we always don't look for the lowest value, instead, we will look for the Elbow point. It is a point where the number of cluster increase, the value does not affect much when decrease. In order words, it is meaningless if we take the largest number of clusters.

3.6.3 Silhouette Analysis

```
fig, ax = plt.subplots(3, 2, figsize=(15,8))
for i in [2, 3, 4, 5, 6, 7]:
    '''
    Create KMeans instance for different number of clusters
    '''
    km = KMeans(n_clusters=i, init='k-means++', n_init=10, max_iter=100, random_state=42)
    q, mod = divmod(i, 2)
    '''
    Create SilhouetteVisualizer instance with KMeans instance
    Fit the visualizer
    '''
    visualizer = SilhouetteVisualizer(km, colors='yellowbrick', ax=ax[q-1][mod])
    visualizer.fit(data_pca)
```

6.6s

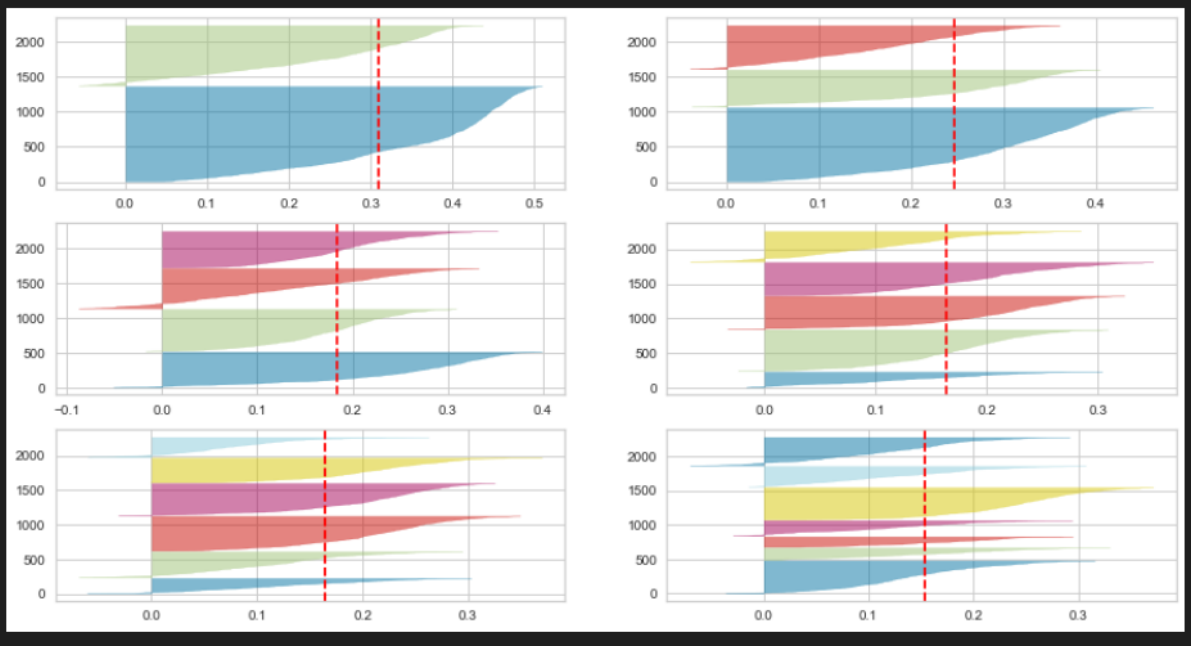

Figure 15: Silhouette Analysis

The silhouette analysis is a method of interpretation and validation of consistency within clusters of data and the value is just within -1 and 1. If the value is closer to 1 means the clusters are well separated from each other, while closer to 0 means the clusters are not well separated or the distance are not significant; closer to -1 means the data point is assigned to the wrong cluster. But, we always don't go for the score closer to 1 as it means the number of clusters is smaller, instead, I will look the score and the number of clusters, and analyze the size of the silhouette plot. Then, I will choose the best score based on some criteria such as the fluctuation of the size and the thickness of the cluster.

3.6.4 Elbow Method

```
from warnings import simplefilter
simplefilter(action='ignore',category=FutureWarning)

# Quick examination of elbow method to find numbers of clusters to make.
print('Elbow Method to determine the number of clusters to be formed:')
Elbow_M = KElbowVisualizer(KMeans(), k=10)
Elbow_M.fit(data_pca)
Elbow_M.show()
```

✓ 7.2s

```
Elbow Method to determine the number of clusters to be formed:
```

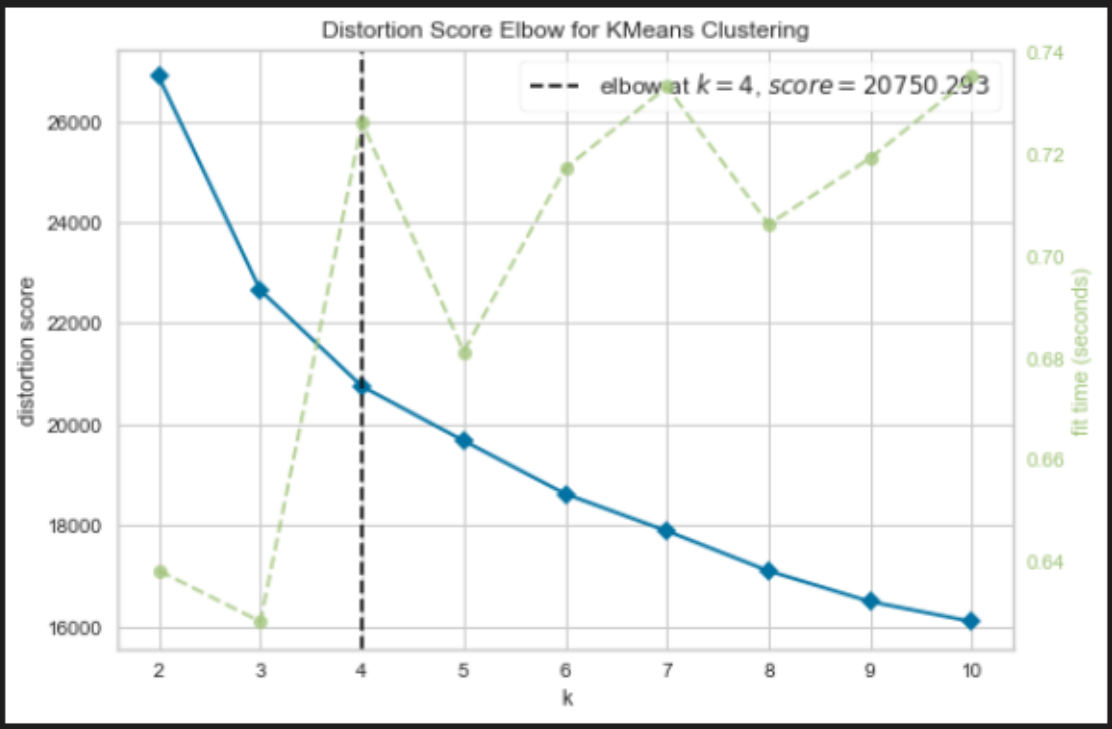

Figure 16: Elbow Method

Elbow method is also one of the techniques to find the optimal number of clusters. The idea behind this technique is to measure the WCSS. When we plot the graph of Elbow method, it will look like an Elbow and we will check the elbow point, where this point does not have big changes when the number of clusters (K) increase[19]. From the code, I will simply set a number (K) as 10, and it will check the WCSS for each cluster. I will fit the reduced data set 'data_pca' to the KElbowVisualizer library. Eventually, it shows me the optimal number of cluster is 4.

3.7 Model Building

Knowing the optimal number of clusters, you are ready to build a model. In this section, I have implemented four different clustering algorithm which are Gaussian Mixture Model, Density-based Spatial Clustering of Applications with Noise, Agglomerarive Hierarchical Clustering and K-Mean Clustering. The steps to implement each algorithm is quite similar but the parameter I am going to set have to be cautious as it will affect the result.

3.7.1 Gaussian Mixture Model (GMM)

```
gmm = GaussianMixture(n_components=4, random_state=42).fit(data_pca)
gmm_labels = gmm.predict(data_pca)
data_Gaussian_PCA_Cluster = data_prep.copy()
data_Gaussian_PCA_Cluster['Cluster'] = gmm_labels
✓ 0.3s
```

Figure 17: Implementation of GMM

The picture shown above is showing the Python code performing the Gaussian Mixture Model (GMM). First, I will create an instance 'gmm' for the GMM and specify the number of components that I have found in Section 6. This represents a GMM model with 4 clusters. Then, I will fit this model with reduced data 'data_pca'. Then, I will use the '.predict()' method from GMM to assign each data in the data set to one of the four clusters based on their probability density estimated from the GMM model. Next, I will create a variable 'labels' where each element corresponds to the cluster label assigned to the corresponding data set. Since it is an algorithm, I will create a copy of the original data set 'data_prep' and name it 'data_Gaussian_PCA_Cluster', and add a new column 'Cluster' at the end of the data set and I will assign the cluster label from 'labels' to the column. Hence, this allows me to visualize and analyze the result.

3.7.2 Density-based Spatial Clustering of Application with Noise (DBSCAN)

```
[31]  data_pca.shape
      ✓ 0.0s

      (2212, 10)

[32]  nn = NearestNeighbors(n_neighbors=20).fit(data_pca)
      distances, indices= nn.kneighbors(data_pca)
      distances = np.sort(distances, axis=0)
      distances = distances[:,1]
      plt.figure(figsize=(10,8))
      plt.plot(distances)
      ✓ 0.3s

      Outputs are collapsed ···

[33]  kneedle = KneeLocator(x = range(1, len(distances)+1), y = distances, S = 1.0,
                            curve = "concave", direction = "increasing", online=True)

      # get the estimate of knee point
      print(kneedle.knee_y)
      ✓ 0.1s

      4.515448979260546

[34]  dbscan = DBSCAN (eps=4.52, min_samples=20).fit(data_pca)
      dbscan_labels = dbscan.labels_
      data_DBSCAN_PCA_Cluster = data_prep.copy()
      data_DBSCAN_PCA_Cluster['Cluster'] = dbscan_labels
      ✓ 0.3s
```

Figure 18: Implementation of DBSCAN

When comes to implementation, there are two paraments which are minPts and eps(ε) that I need to calculate first before building the model. The first parameter, minPts, represents the minimum number of data points for a cluster to be formed, while the second parameter, eps(ε), refers to a measurement of the distance that will be used to find any nearby data points. Besides that, there are three different points, which are core, border, and noise. A core point is when the data point has at least a minimum number of data points within the distance, a border point is the point where it has at least one core point and a noise point is the data point where it has less than the minimum number within the distance. The minPts can be calculated from the number of dimensions of your data set, and the number of dimensions also refers to the number of data features of the dataset. As a rule of thumb,

minPts is double the dimensions (minPts = 2*dim). For instance, if your data set has 2 data features, then your minPts will be 4. And to calculate the eps(ε), I will perform nearest neighbor distance analysis by using NearestNeighbors from scikit-learn and display the result. First, I will create an instance for the NearestNeighbors and specify the 'n_neighbors' as 20 and fit the data to this model. Then, I will use the kneighbors method of the NearestNeighbors 'nn.kneighbors' to compute the distances between each data point and the nearest neighbors, and will return two arrays which are distances and indices. Distances refer to the distances of each data point to the nearest neighbors and indices refer to the corresponding indices of those neighbors. Then, I will sort the distances in ascending order and select the second column of the distances as the first column is always to the nearest neighbor which means the distance is 0. Then, I will plot the distances. It is difficult to determine the exact value from the graph based on the elbow point. Thus, I will run a knee point detection by applying KneeLocator which takes a few arguments, x, y, S, curve, direction, and online. X is the range of the distances, and y is the distance to the nearest neighbor, S is the sensitivity where a smaller value will detect better, curve is set to concave as the shape is like a knee, direction is set to 'increasing' as the graph is in an increasing curve, and online is set to True as the algorithm will assume all the data points are in sequential order. Eventually, I got the eps(ε) value which is 4.52. After I have two required parameters for the DBSCAN, it is ready to implement the algorithm. It is almost the same as the previous algorithm, but the parameters are changed. I will create a DBSCAN object and set the two parameters, eps and min_samples, and fit the reduced data set. Then assign the 'labels_' to the 'labels' and place the value to the new data set 'data_DBSCAN_PCA_Cluster' which I have created a copy from the original data set.

3.7.3 Agglomerative Hierarchical Clustering

```
#Initiating the Agglomerative Clustering Model
AC = AgglomerativeClustering(n_clusters=4)
#Fit and predict model
AC.fit_predict(data_pca)
#Label model
ac_labels = AC.labels_
#Add the Cluster feature to the original dataframe
data_Agglo_PCA_Cluster = data_prep.copy()
data_Agglo_PCA_Cluster['Cluster'] = ac_labels
```

✓ 0.2s

Figure 19: Implementation of AHC

The picture above is showing the Python code using the Agglomerative clustering algorithm to perform hierarchical clustering. As usual, I will create an object for the AgglomerativeClustering with the parameter which is the number of clusters set to 4. Then, I will fit the model onto the reduced data set and predict the cluster labels for each data point. Then, I will create a copy of the original data set and name it 'data_Agglo_PCA_Cluster' and add a new column 'Cluster' and assign the 'label' value which I have assigned the resulting cluster labels to a new variable 'label'.

3.7.4 K-Means Clustering

```
#Initiating the KMean Clustering Model
kmeans = KMeans(n_clusters=4, random_state=42)
#Fit model
kmeans.fit(data_pca)
#Predict model
k_labels = kmeans.predict(data_pca)
#Add the Cluster feature to the original dataframe
data_KMean_PCA_Cluster = data.copy()
data_KMean_PCA_Cluster['Cluster'] = k_labels
```

✓ 0.7s

Figure 20: Implementation of K-Means

Figure Implementation of K-Means

As usual, I will create an object 'kmeans' represents the KMeans and set the parameter manually, n_clusters to 4 and random_state to 42 where n_clusters is the number of clusters that I calculated using WCSS, SSE, and Silhouette, and random_state is the random number used by KMeans for centroid initialization and the value is by default which is 42. Then, I will fit the model onto the reduced data set and predict the cluster labels for each data point based on the distance between the data point and the centroid of the cluster. The following step is to create a copy of the original data set and add a new column called 'Cluster' based on the assigned cluster labels.

3.8 Model Evaluation

After building the model, it is time to evaluate the model. There are many ways to evaluate the model but the way that I used to evaluate the model is by checking the number of data in each separate cluster in different four datasets by using the Seaborn library to visualize it. Some people will use the SSE, WCSS, or Silhouette to evaluate the model by determining the optimal number of clusters but I have used it before model building so it also considers model evaluation just the process flow is changed but the concept is still the same. However, I will evaluate each cluster of different four models that I have built just now.

3.8.1 Gaussian Mixture Model

```
ax = sns.countplot(x=data_Gaussian_PCA_Cluster['Cluster'], data=data_Gaussian_PCA_Cluster)
for p in ax.patches:
    ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.show()
```

✓ 0.2s

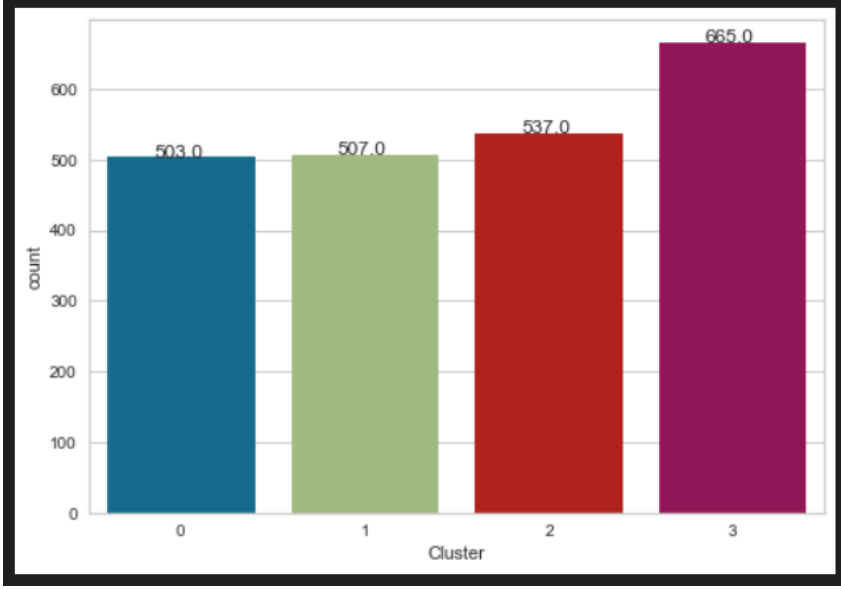

Figure 21: Cluster Distribution of GMM

First, I will evaluate the Gaussian Mixture Model by checking how many data points are in each cluster and deciding if the result is reasonable by comparing the size of different clusters. I will use countplot chart provided by Seaborn Library 'sns.countplot()' which will take two arguments 'x' and 'data' where 'x' refers to the 'Cluster' column of the data set and 'data' refers to the data set that I want

to visualize. Then, I will iterate over each bar in the countplot chart that I have created just now and add a total number of the data points that belong to the cluster by using 'ax.annotate()' function. From the chart, I can claim that most of the customer belongs to cluster 2 and the rest are well separated to cluster 0,1, and 3. By doing this, it can let me have a better understanding of the clustering result and decide if the result is reasonable or not by checking the size of the cluster after evaluation. Hence, I decided to explore this model with further analysis.

3.8.2 Density-based Spatial Clustering of Applications with Noise

```
ax = sns.countplot(x=data_DBSCAN_PCA_Cluster['Cluster'], data=data_DBSCAN_PCA_Cluster)
for p in ax.patches:
    ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.show()
```

✓ 0.1s

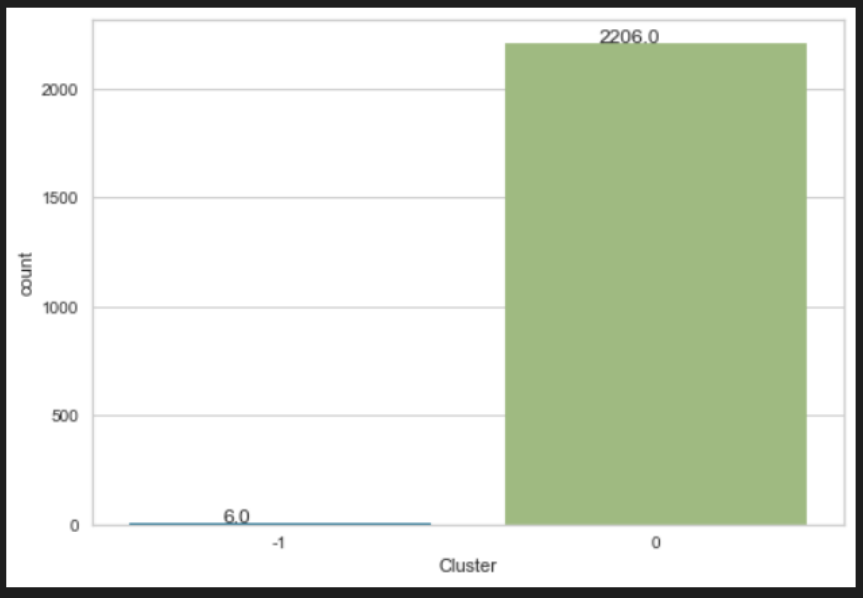

Figure 22: Cluster Distribution of DBSCAN

Unfortunately, when it comes to DBSCAN, the data points are poorly separated where there is only one cluster which is cluster 0, and cluster -1 means the data point is not within the distance from the centroid. The reason why having this result is that this model does not require the number of clusters (K), the model itself has the function to calculate the distance between the data point and the centroid of the cluster and what is the minimum of data points that enough to form a cluster. There are some reasons why DBSCAN only produces one cluster, such as the density of the data set being high

enough for all the data points connected, the minPts value being too high, or the eps value being too small. However, at least I will know that this model is not suitable for this data set and decided not to explore this model further.

3.8.3 Agglomerative Hierarchical Clustering

```
ax = sns.countplot(x=data_Agglo_PCA_Cluster['Cluster'], data=data_Agglo_PCA_Cluster)
for p in ax.patches:
    ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.show()
```

✓ 0.2s

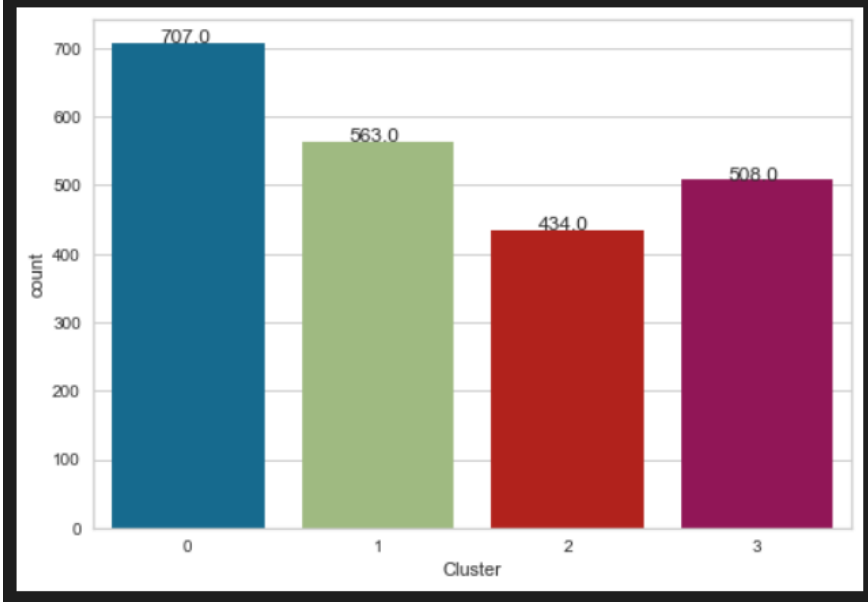

Figure 23: Cluster Distribution of AHC

The third module that I will evaluate is Agglomerative Hierarchical Model. The steps to evaluate this model are almost the same as the Gaussian Mixture Model, but I have to change the value of the arguments. First, I need to change the 'x' argument where I need to specify the data set which belongs to the model 'data_Agglo_PCA_Cluster'. From the chart, I can claim that 32% of the customer belongs to Cluster 0, 25% of the customer belongs to Cluster 1, 20% of the customer are Cluster 2, and 23% of the customer is Cluster 3. Even if the cluster is not well separated, but still consider a reasonable result. Hence, I decided to explore what are the types of customers in each cluster.

3.8.4 K-Means

```
ax = sns.countplot(x=data_KMean_PCA_Cluster['Cluster'], data=data_KMean_PCA_Cluster)
for p in ax.patches:
    ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.show()
```

✓ 0.2s

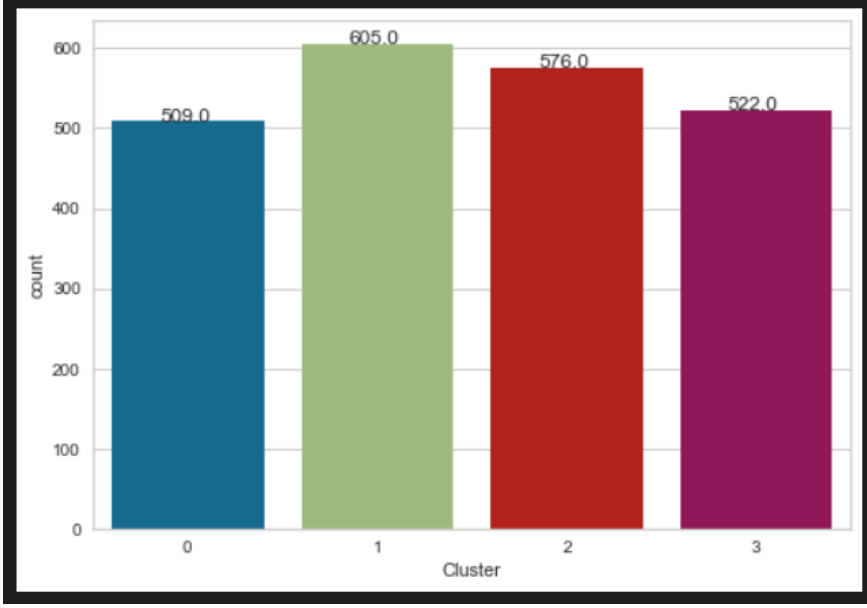

Figure 24: Cluster Distribution of K-Means

The last model that will be evaluated is the K-Means model. The steps to evaluate this K-Means model are almost the same as the previous model, and I have to change the value of the argument which is the 'x' argument where I need to specify the data set which belongs to the model 'data_Kmean_PCA_Cluster'. From the chart, I will consider the clustering result is well separated, the number of customers in each cluster is quite even. Hence, I decided to analysis this model by exploring what are the types of customers in each cluster.

3.9 Profiling

In this section, I will explore what are the types of customers in each cluster obtained from different clustering algorithms. But, I will only explore 3 clustering algorithms which are Gaussian Mixture Mode, Agglomerative Hierarchical Clustering, and K-Means Clustering as the DBSCAN has only one cluster. In each clustering algorithm, I will perform 4 different types of segmentation which are demographic segmentation, behavioral segmentation, psychographic segmentation, and value-based segmentation. Demographic segmentation is one of the most simplest but effective types of segmentation where it is based on personal objective information such as age, gender, and income. Psychographic segmentation refers to the customers' needs such as which products they spent the most. Next, behavioral segmentation refers to the interaction between the business and customers such as how often they visit your business. Last is value-based segmentation which refers to how many purchases have been made by the customers through different platforms like online or in stores. For that, I will be profiling each cluster formed on different clustering algorithms based on the types of customer segmentation. Hence, each clustering algorithm will have 4 different types of customer segmentation and I will be profiling what are the types of customers in the cluster by showing the result for each type of segmentation and giving an overview description.

3.9.1 Type of customer segmentation

3.9.1.1 Demographic Segmentation

For this segmentation, I will take some of the data variables that belong to the demographic such as age, income, kidhome, teenhome, children, is_parent education and marital_status.

3.9.1.2 Behavioral Segmentation

For this segmentation, I will take some of the data variables that belong to the behavioral such as recency, numwebvisitsmonth, and customer_days.

3.9.1.3 Psychographic Segmentation

For this segmentation, I will take some of the data variables that belong to the psychographic segmentation such as what are the products they spent the most like wines, fruits, meat, fish, and gold.

3.9.1.4 Value-based Segmentation

In this segmentation, I will take the data variables that explained how many purchases they have made in different ways such as through websites, in stores, or catalogs.

3.9.2 Algorithms

3.9.2.1 Gaussian Mixture Model

```
pal=['red','green','blue','orange']
plt.figure(figsize = (10,8) )
sns.kdeplot(data =data_Gaussian_PCA_Cluster, x = 'Age' , hue = 'Cluster' , palette=pal)
plt.title('Age Distribution' ,fontsize = 15)
plt.show()
```

✓ 0.4s

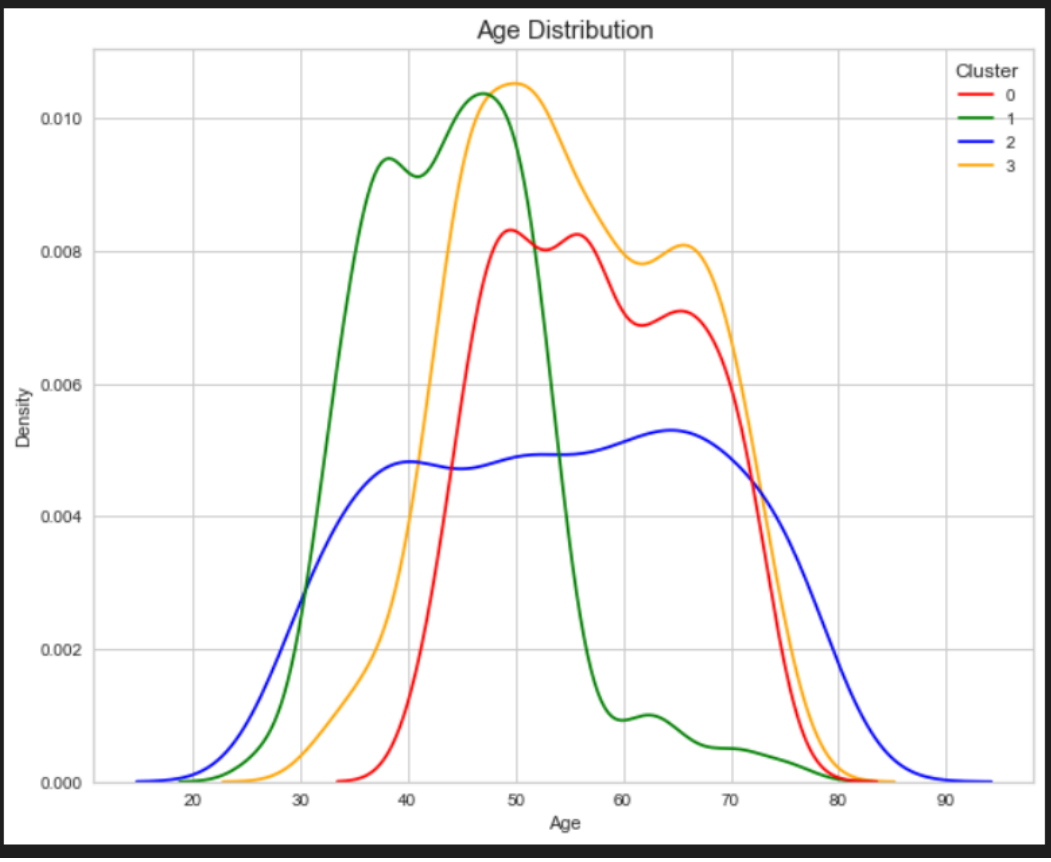

Figure 25: Age Distribution of GMM

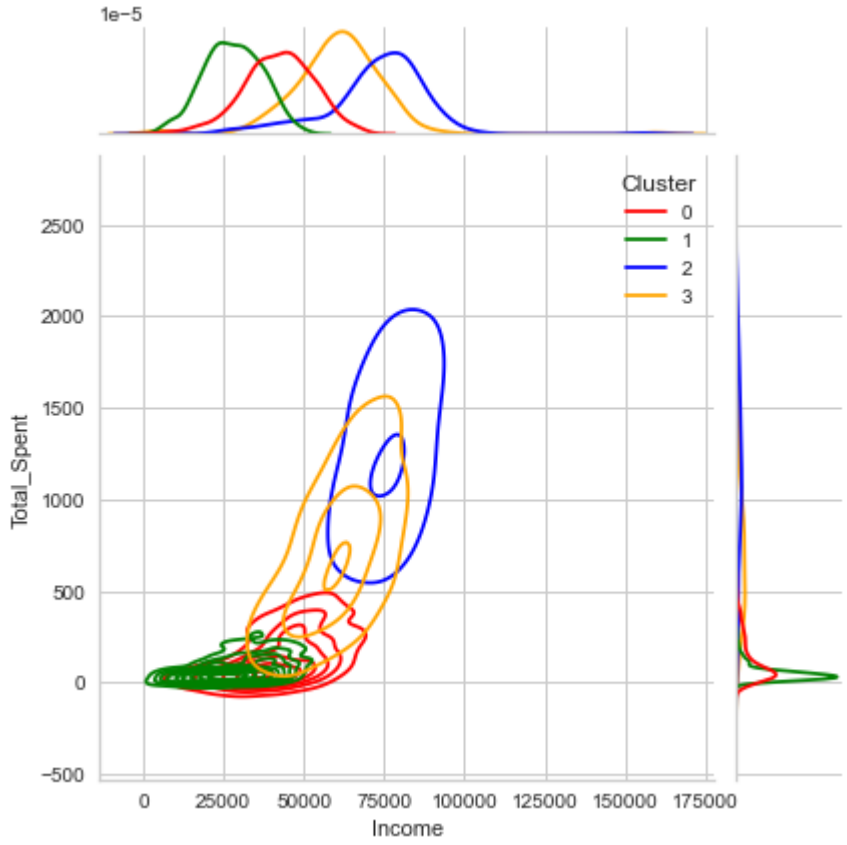

Figure 26: Income vs Total_Spent of GMM

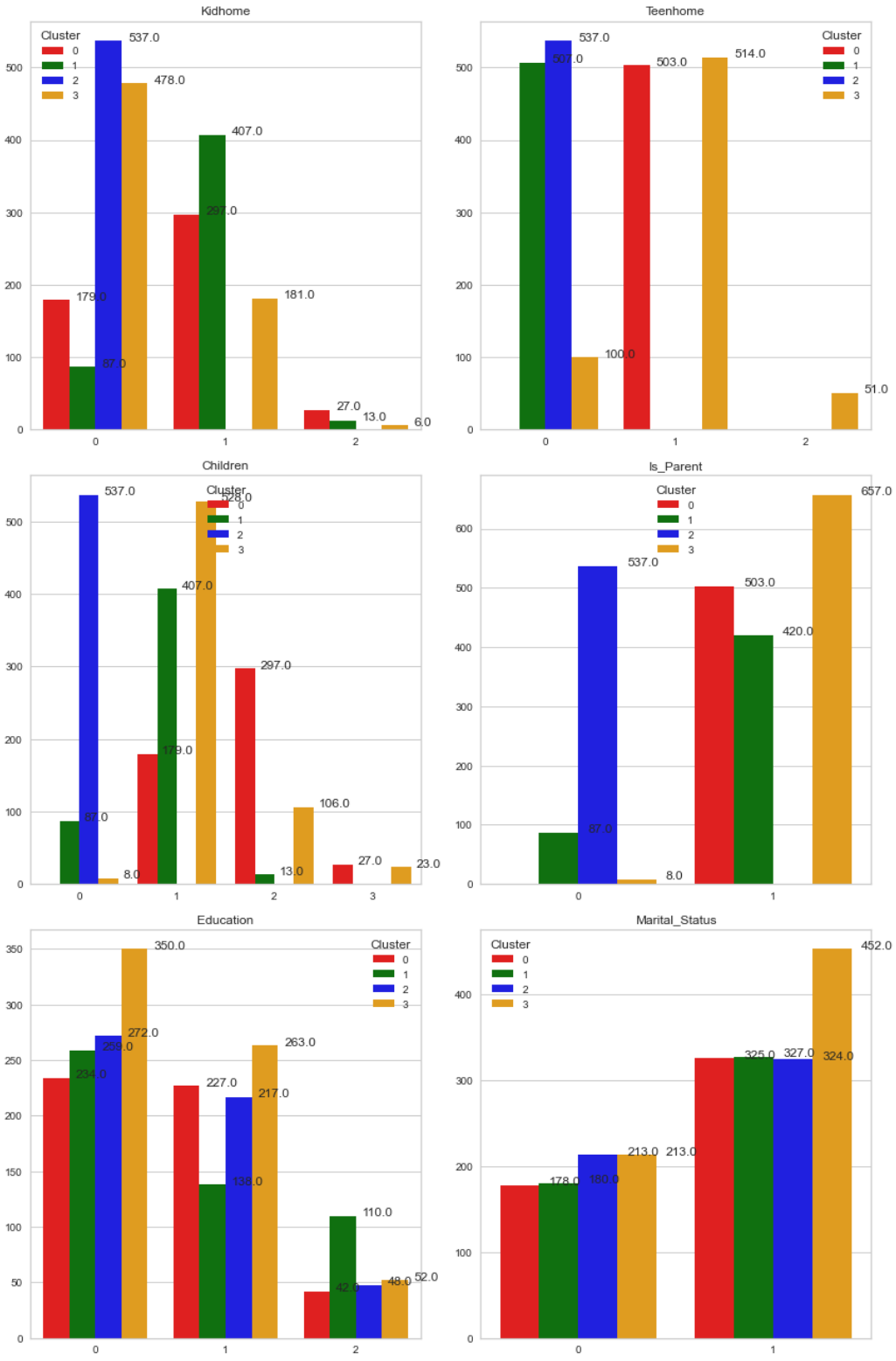

Figure 27: Count plot of GMM

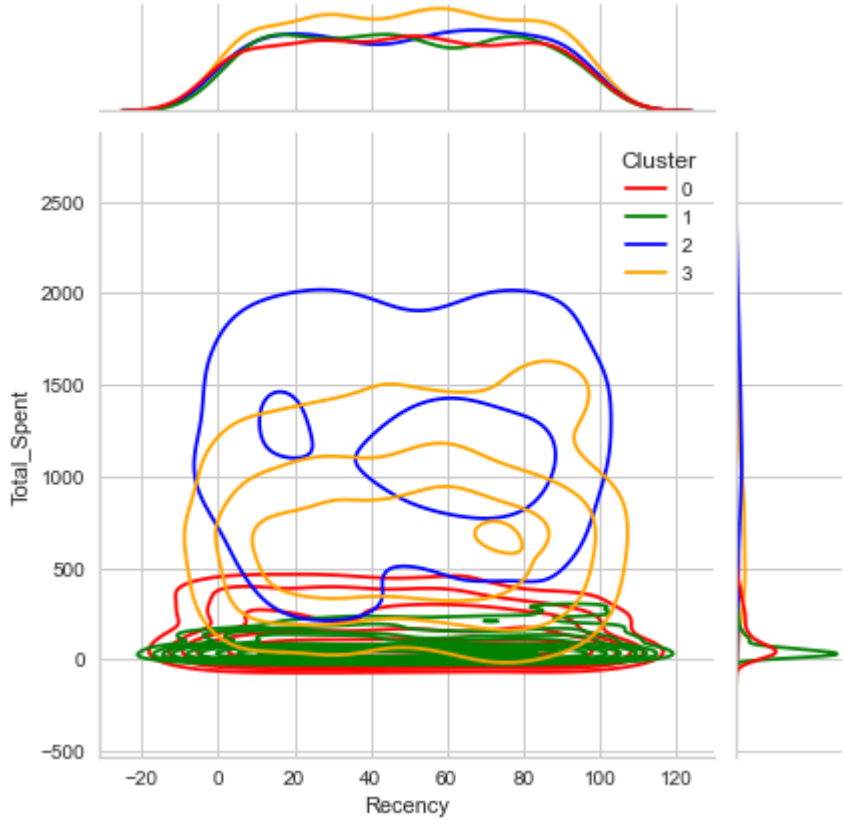

Figure 28: Recency vs Total_Spent of GMM

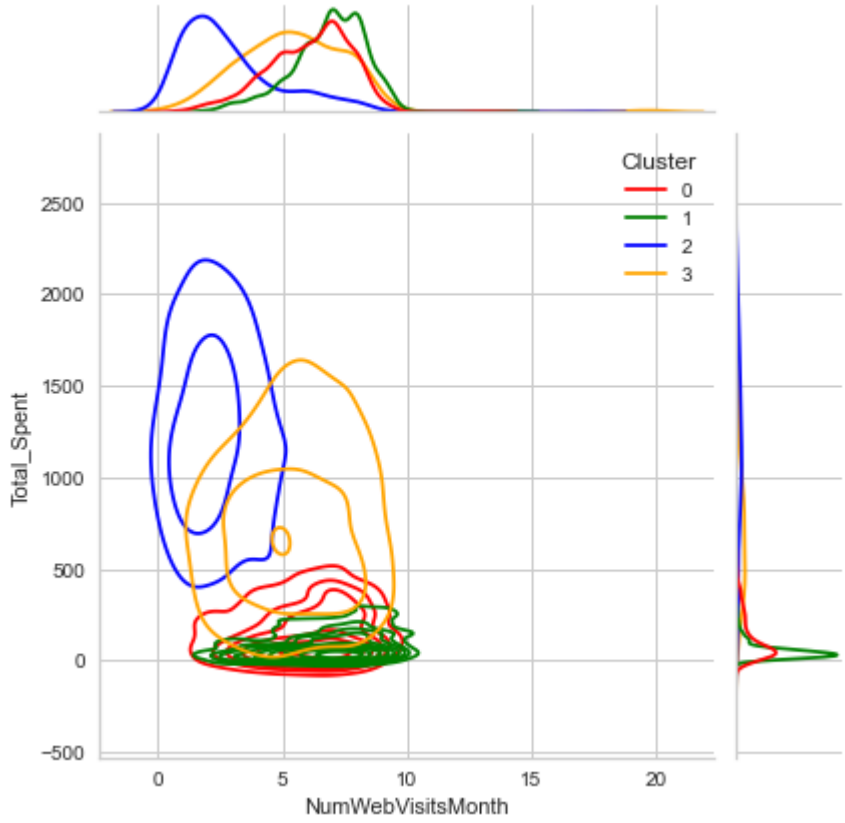

Figure 29: NumWebVisitsMonth vs Total_Spent of GMM

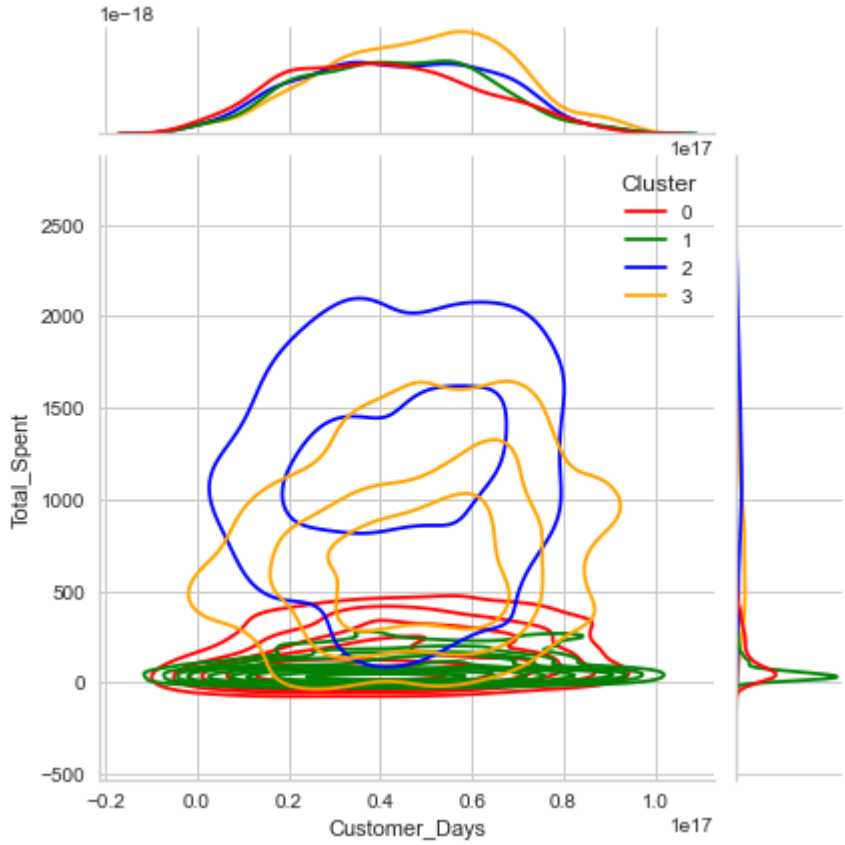

Figure 30: Customer_Days vs Total_Spent of GMM

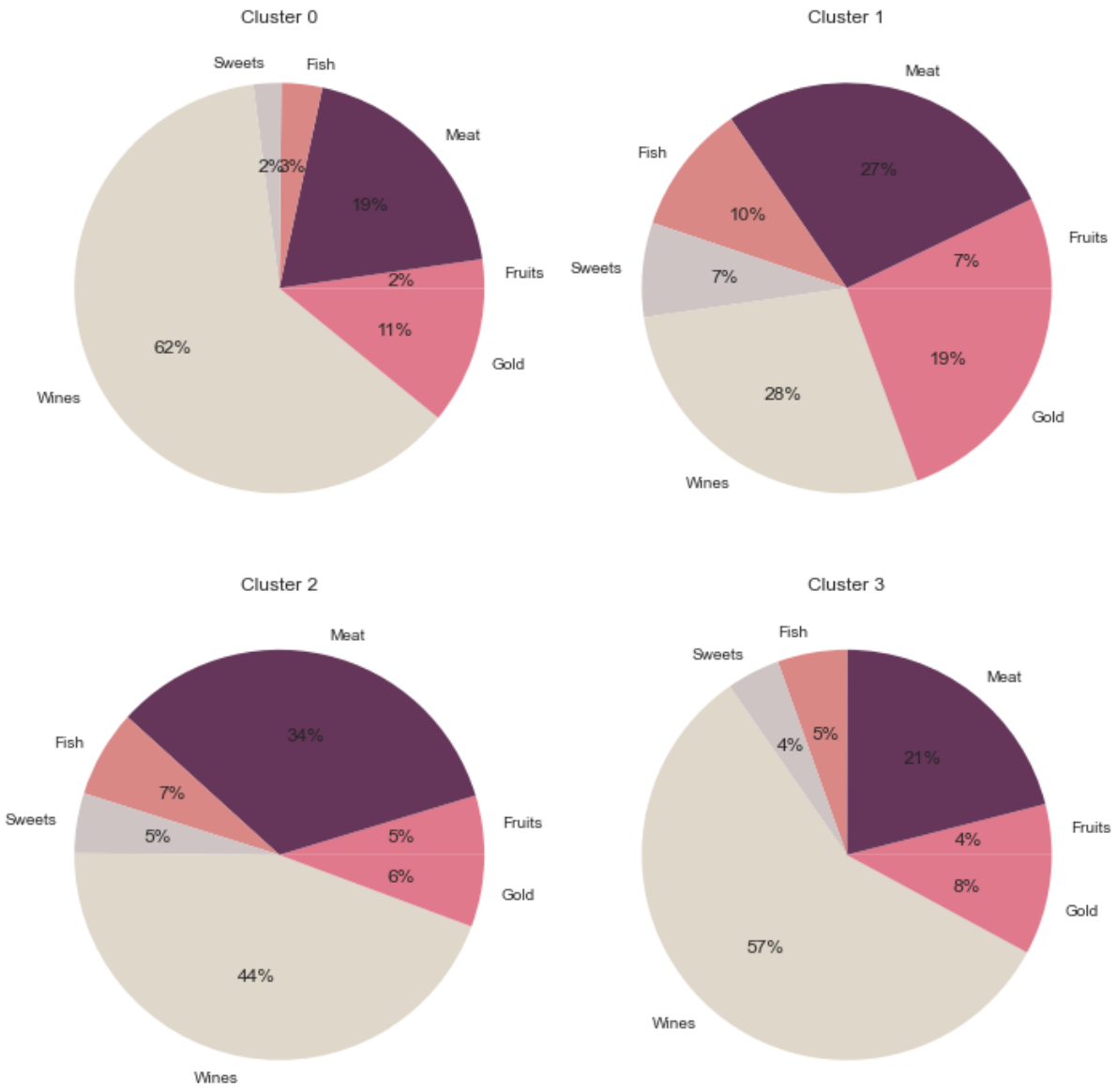

Figure 31: Pie Chart of Products of GMM

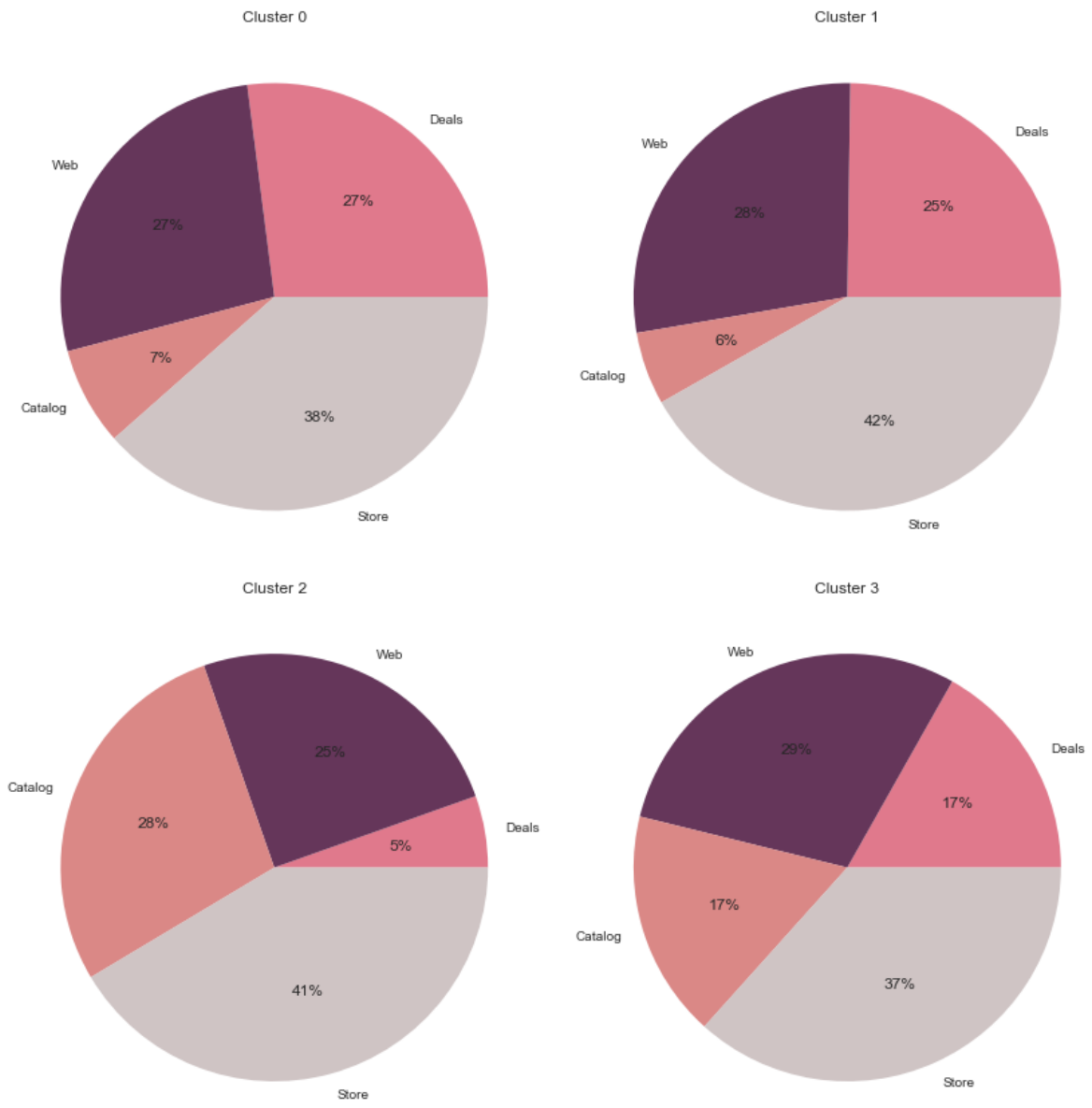

Figure 32: Pie Chart of Purchases of GMM

After going through all the figures that obtained from the GMM, I can conclude the types of customers in each cluster are as follows:

Cluster 0	Cluster 1	Cluster 2	Cluster 3
Low income low spent	Low income low spent	High income high spent	Average income average spent
57% of customer is older	Definitely younger	Definitely younger	Most of the customer is older
Some has 1 child; some has 2 children	Most has 1 child	Definitely no children	Most has 1 child; less has 2 children
Definitely is parent	Most is parent	Definitely not parent	Definitely is parent
Half graduate; half postgraduate	Half graduate; half postgraduate	Half graduate; half postgraduate	Most graduate
Most married	Most married	Most married	Most married
Spent most on wines	Spent most on meat and wines	Spent most on meat and wines	Spent most on wines
38% in store	42% in store	41% in store	37% in store

Table 1: Overview of GMM

3.9.2.2 Agglomerative Hierarchical Model

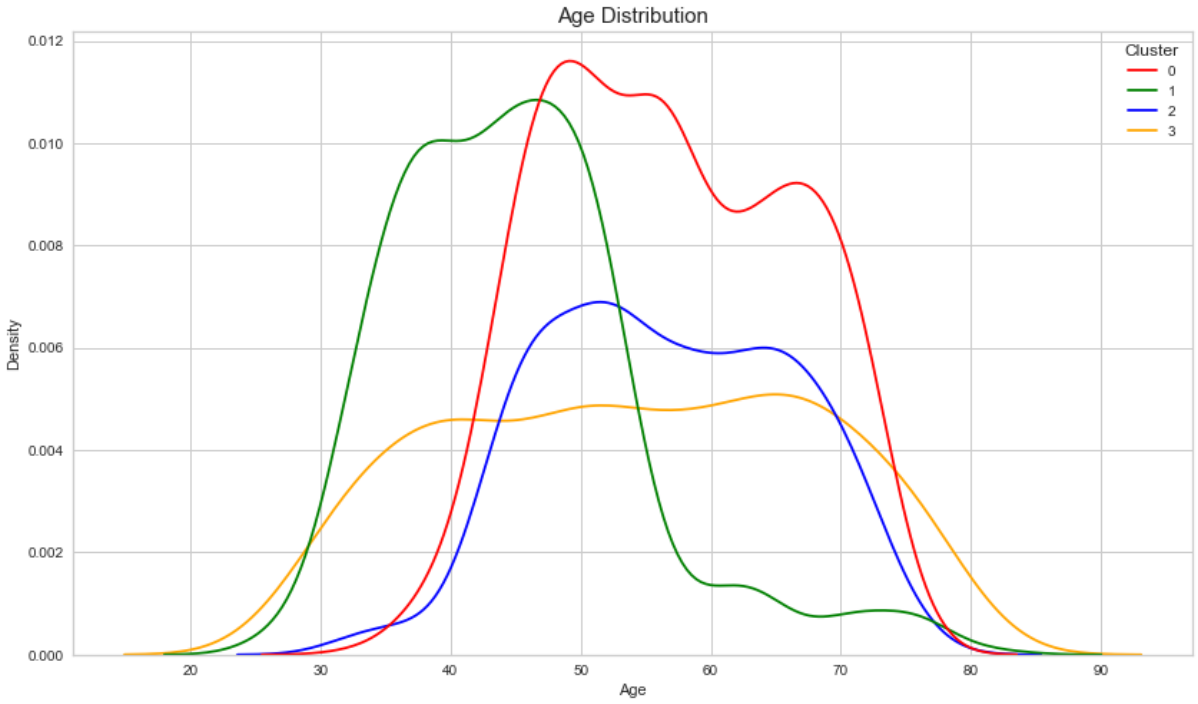

Figure 33: Age Distribution of AHC Model

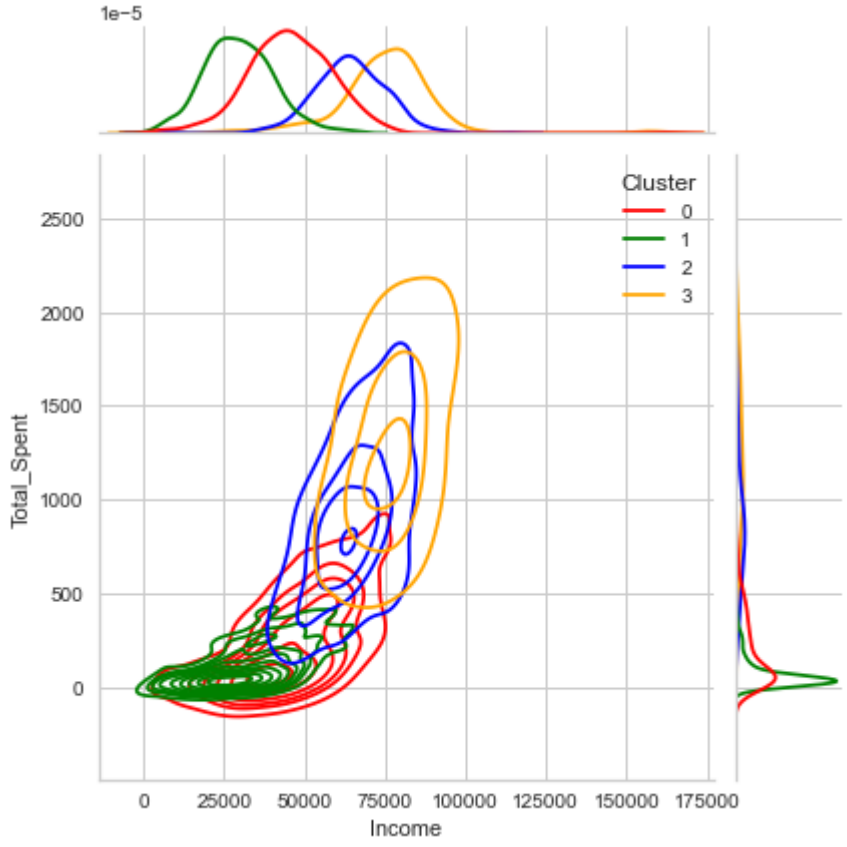

Figure 34: Income vs Total_Spent of AHC Model

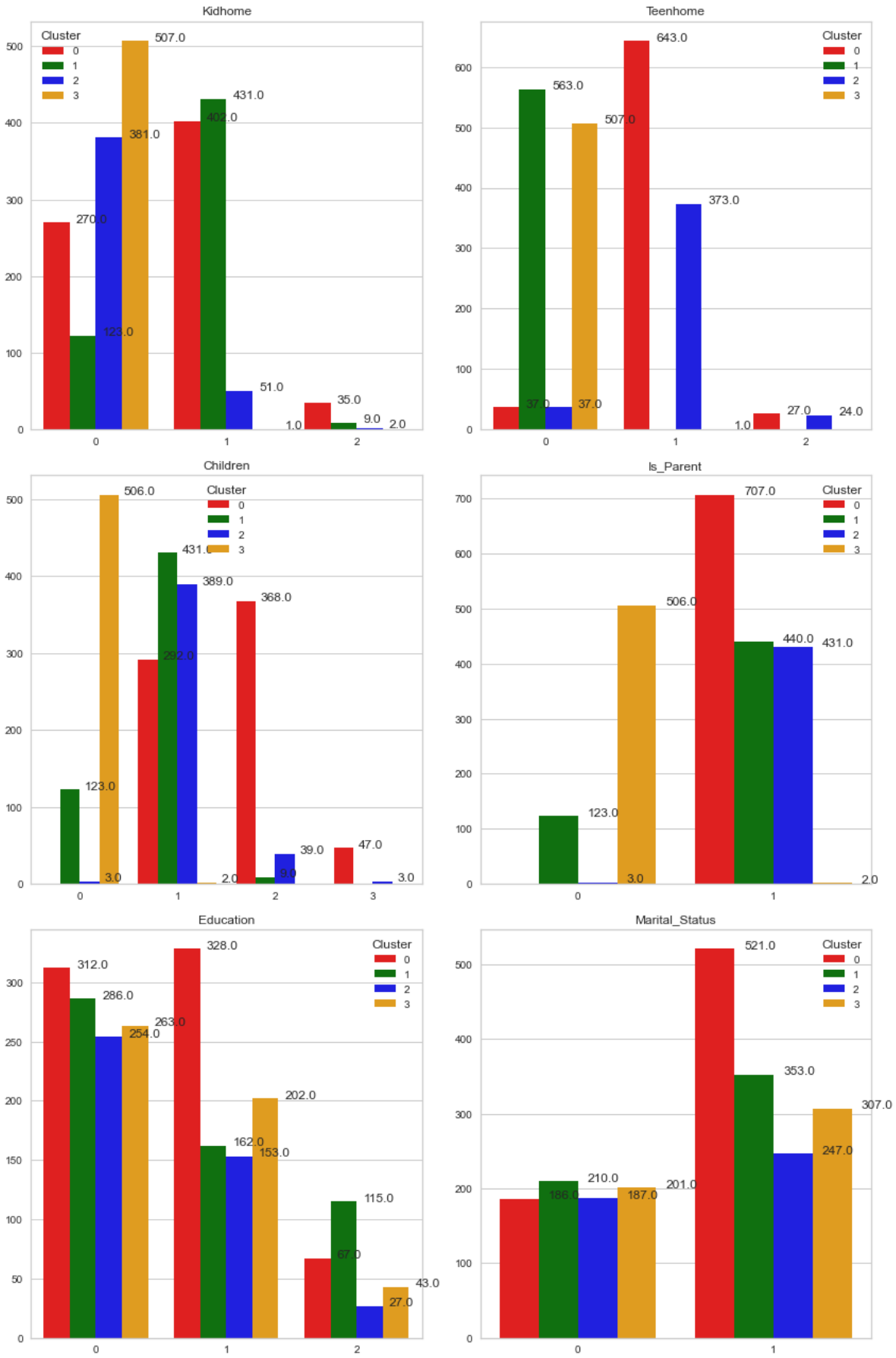

Figure 35: Count Plot of AHC Model

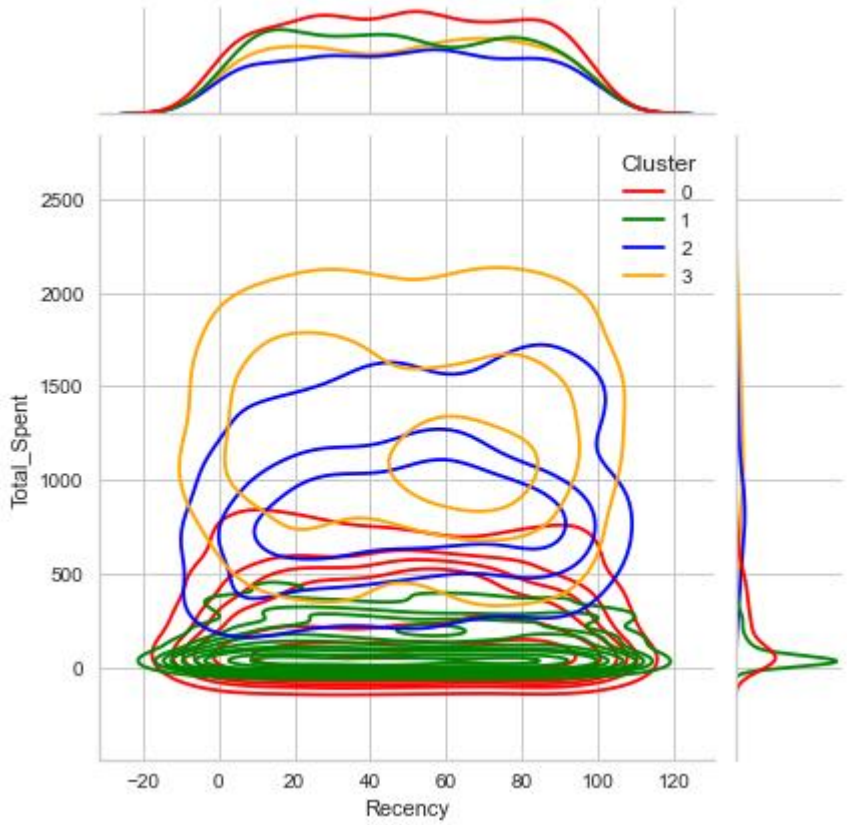

Figure 36: Recency vs Total_Spent of AHC Model

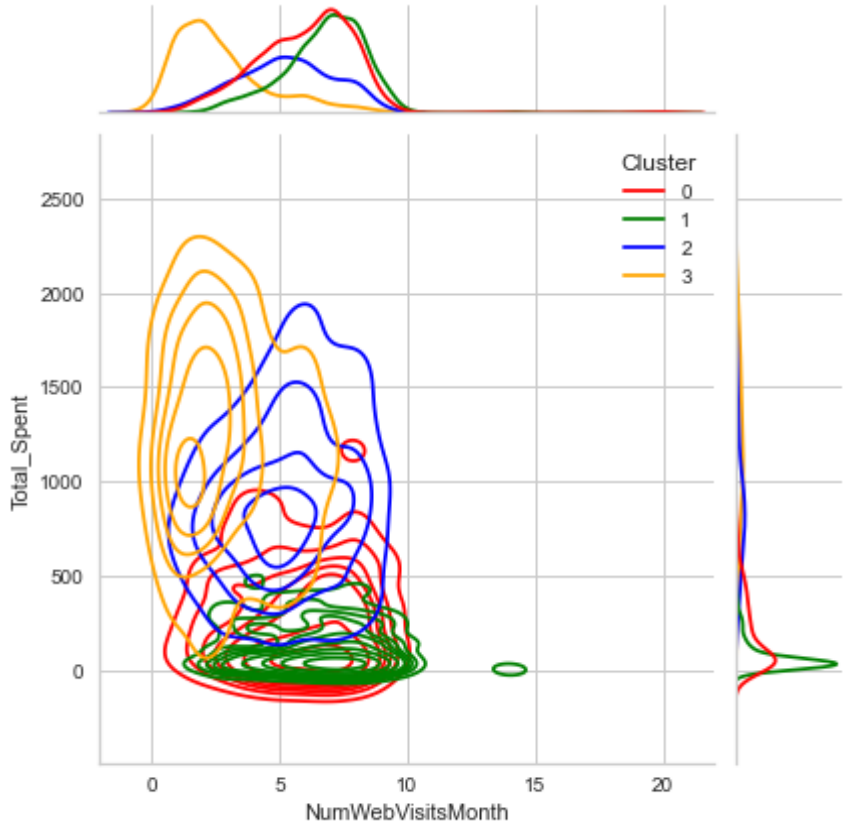

Figure 37: NumWebVisitsMonth vs Total_Spent of AHC Model

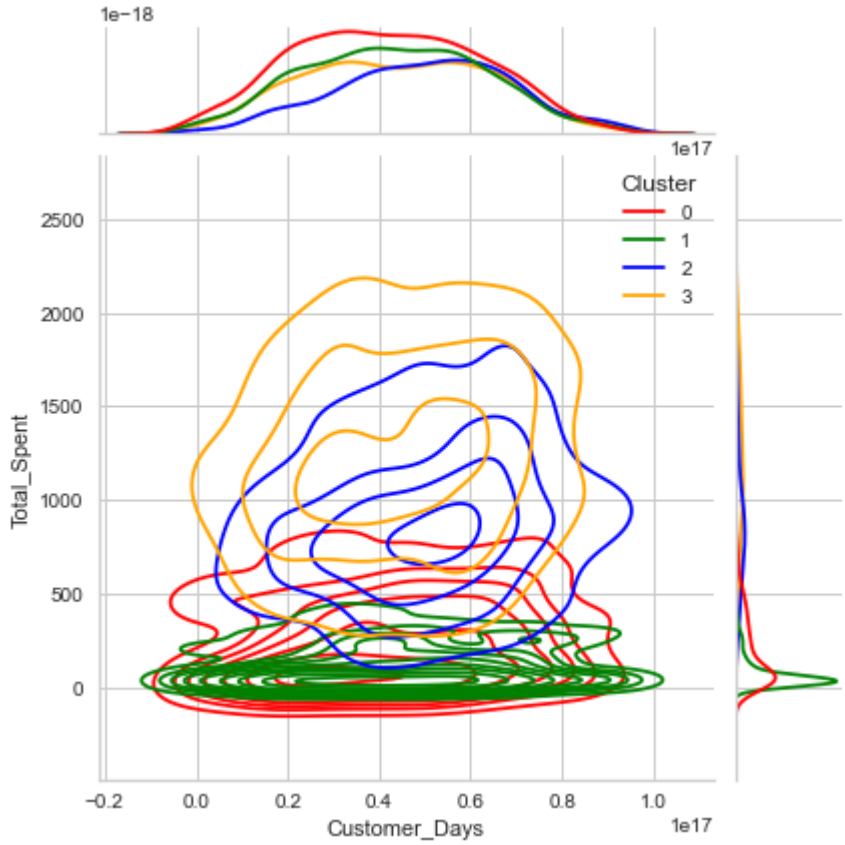

Figure 38: Customer_Days vs Total_Spent of AHC Model

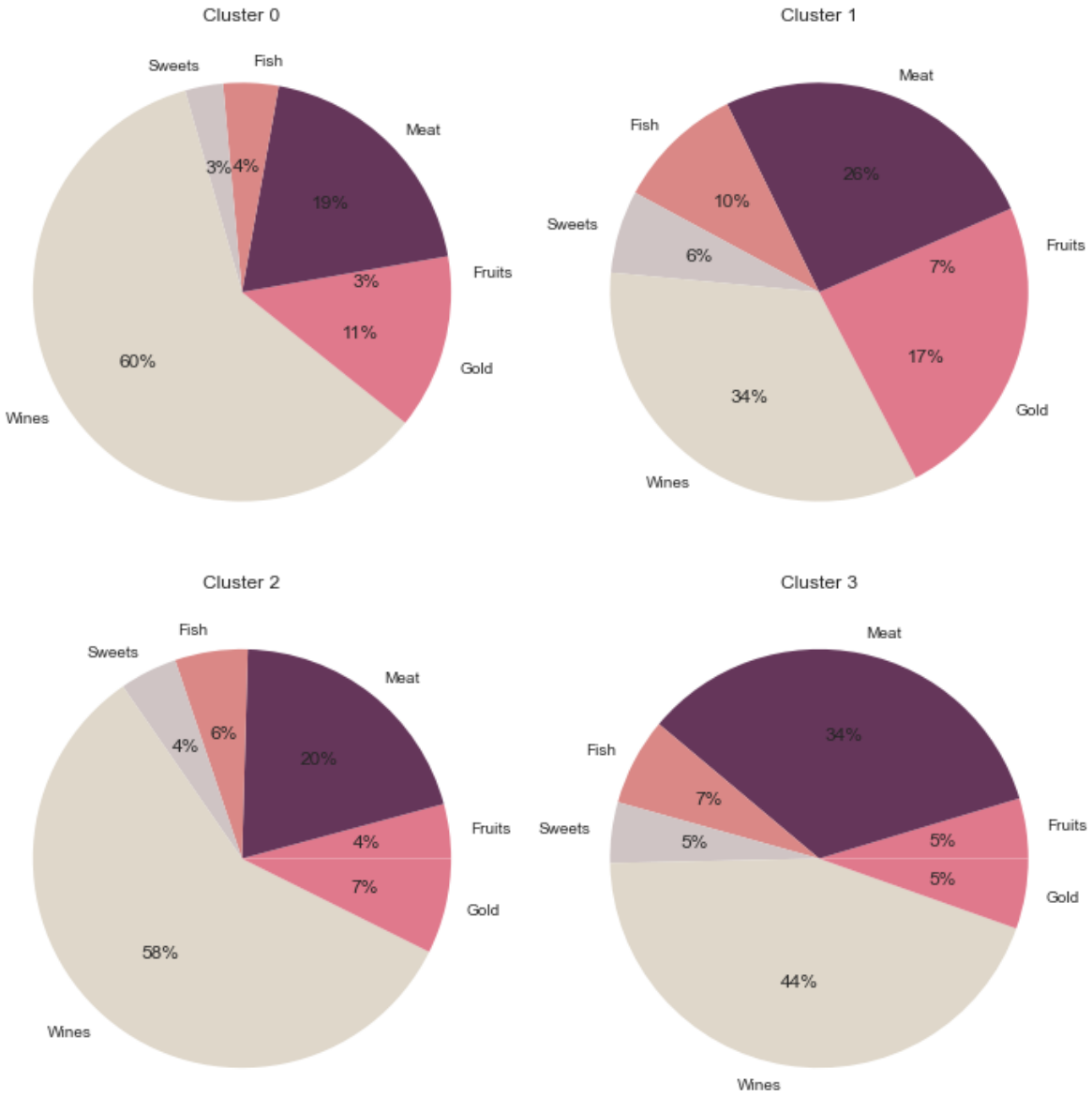

Figure 39: Pie Chart of Products of AHC Model

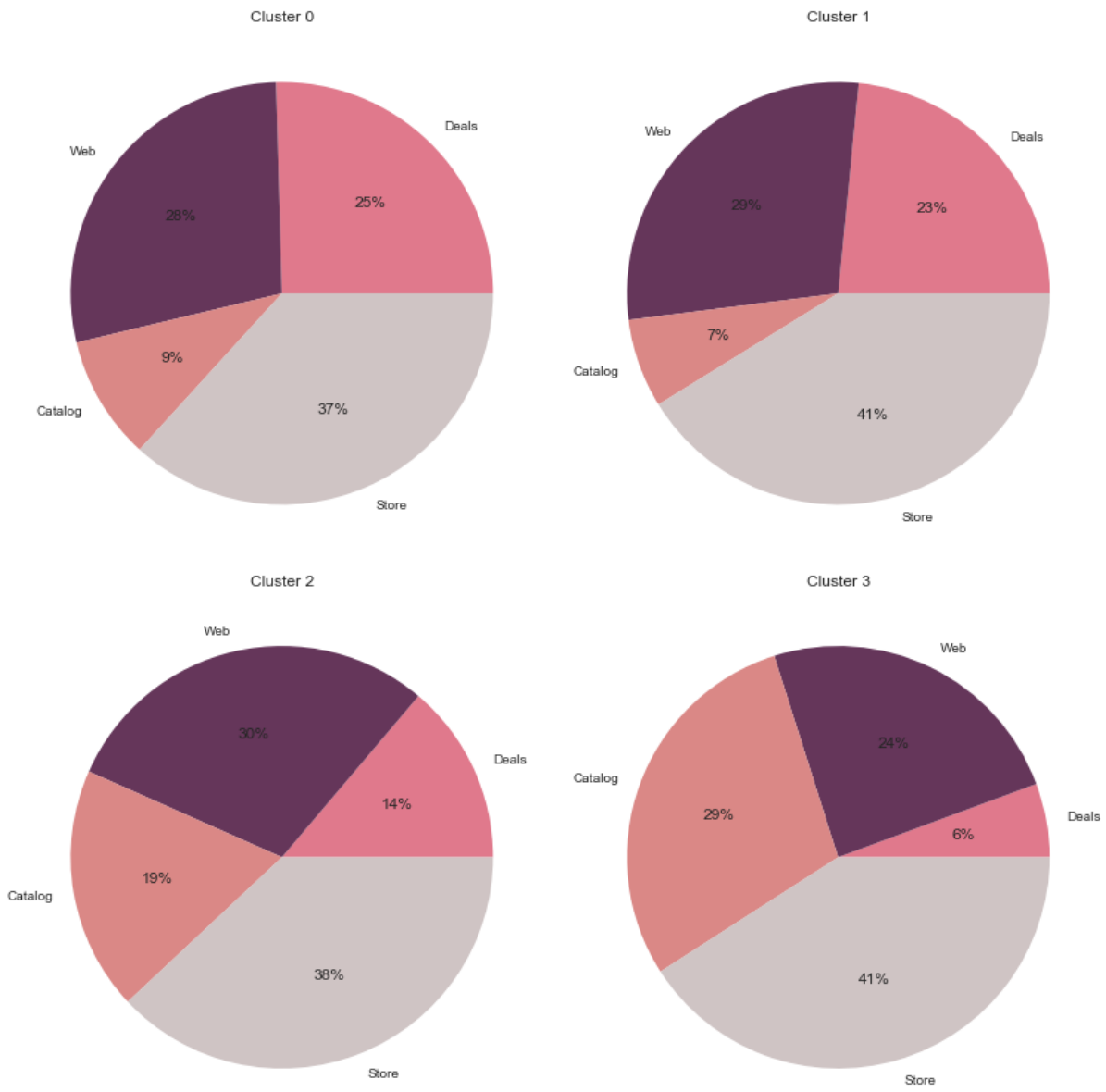

Figure 40: Pie Chart of Purchases of AHC Model

After analysing all the graphs and chart that generated from the AHC model, I can conclude the types of customers in each cluster are as follows:

Cluster 0	Cluster 1	Cluster 2	Cluster 3
Average income average spent	Low income low spent	High income above average spent	High income high spent
Half of customer is older	Definitely younger	Definitely younger	Definitely younger
Most has 1 or 2 children	Most has 1 child	Most has 1 child	Definitely no child
Definitely is parent	Most is parent	Definitely is parent	Definitely not parent
Most graduate and postgraduate	Most is graduate, average on undergraduate and postgraduate	Most graduate, some postgraduate	Average on graduate and postgraduate
Most married	Above average married	Half married	Above average married
Spent most on wines, second on meat	Spent most on wines	Spent most wines	Spent most on wines
37% in store	41% in store	38% in store	41% in store

Table 2: Overview of AHC Model

3.9.2.3 K-Means Model

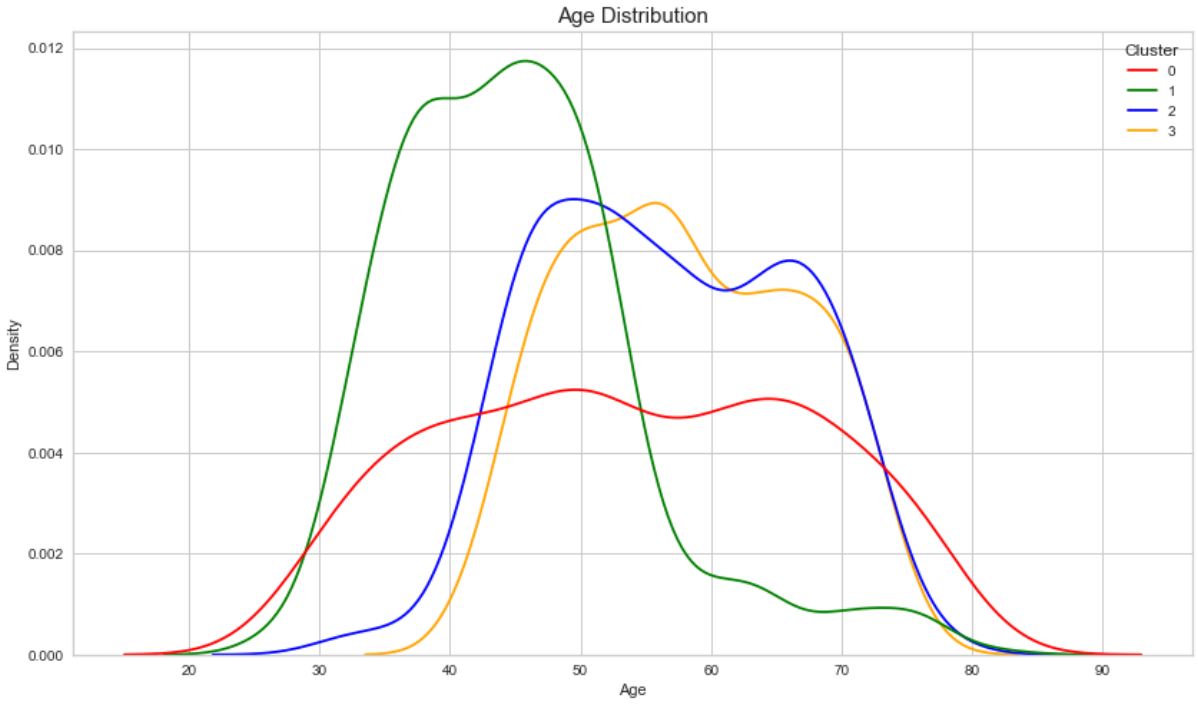

Figure 41: Age Distribution of K-Means model

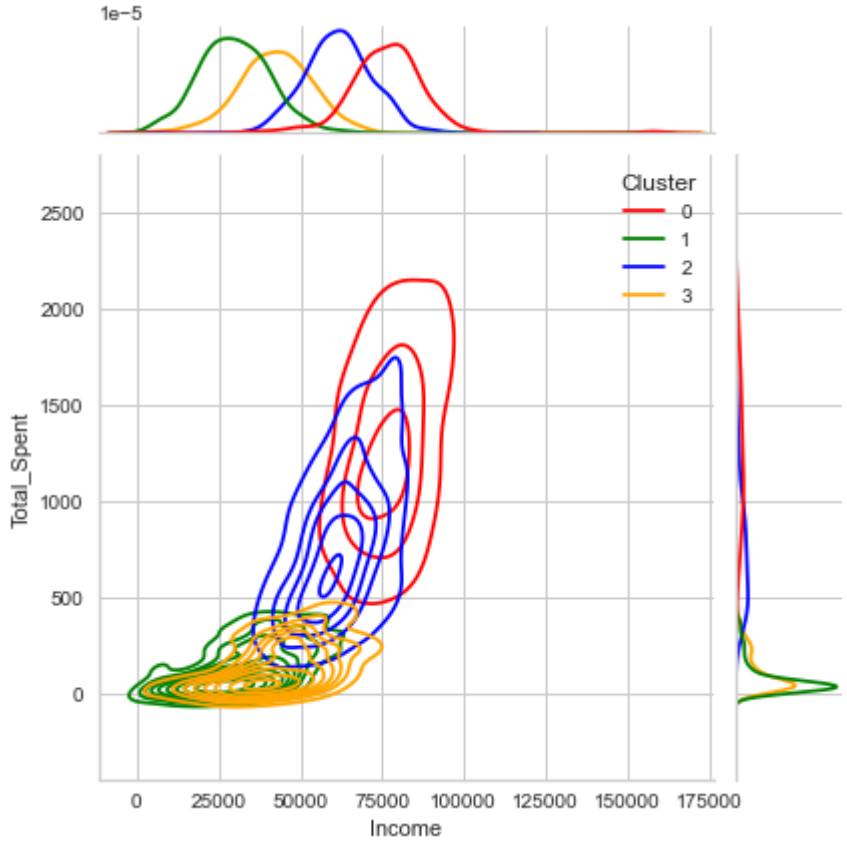

Figure 42: Income vs Total_Spent of K-Means Model

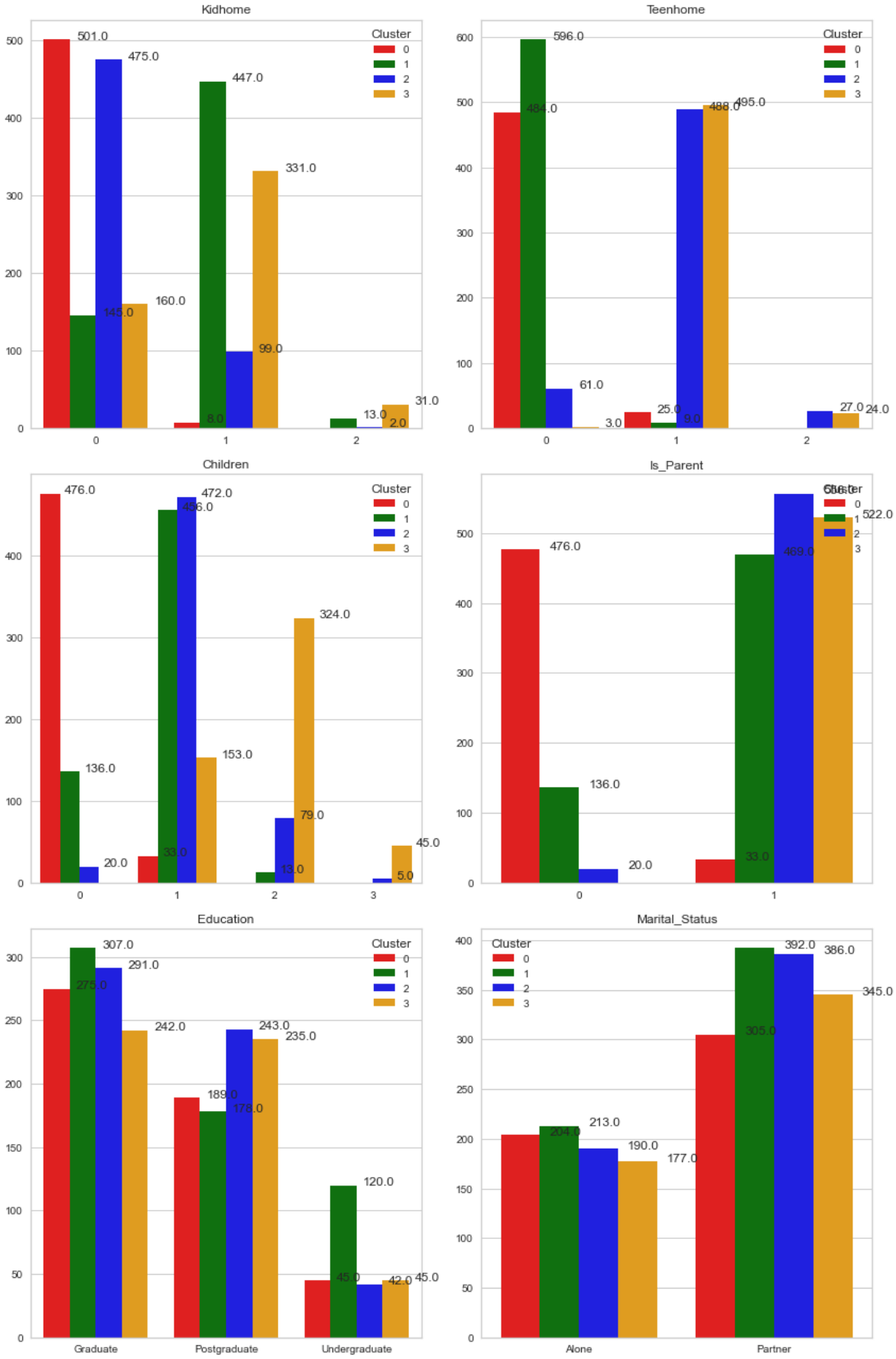

Figure 43: Count Plot of K-Means Model

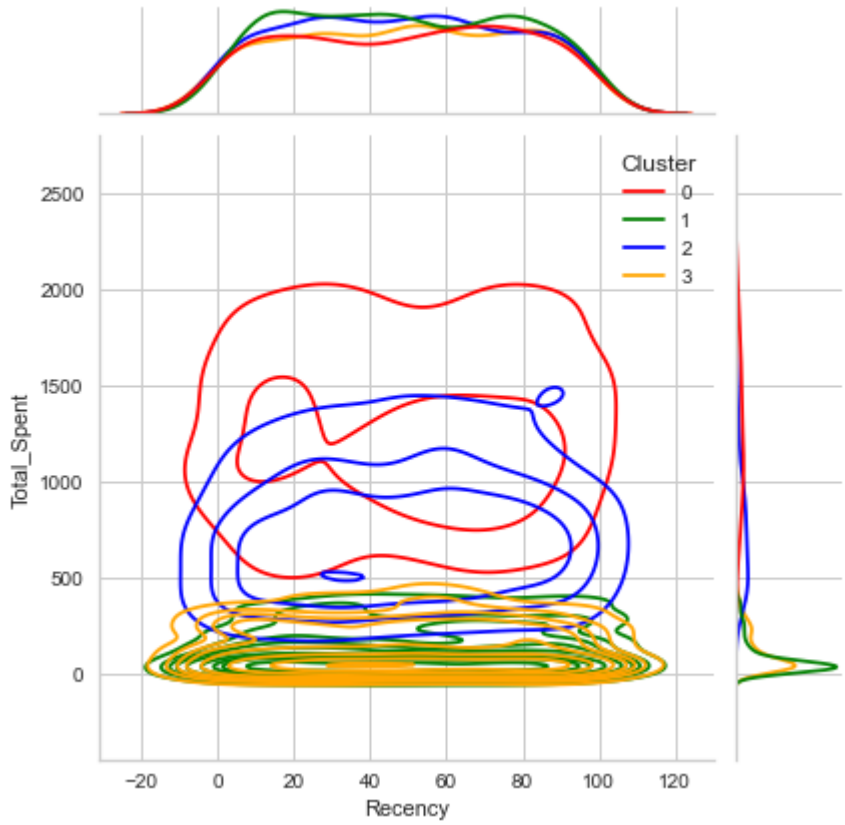

Figure 44: Recency vs Total_Spent of K-Means Model

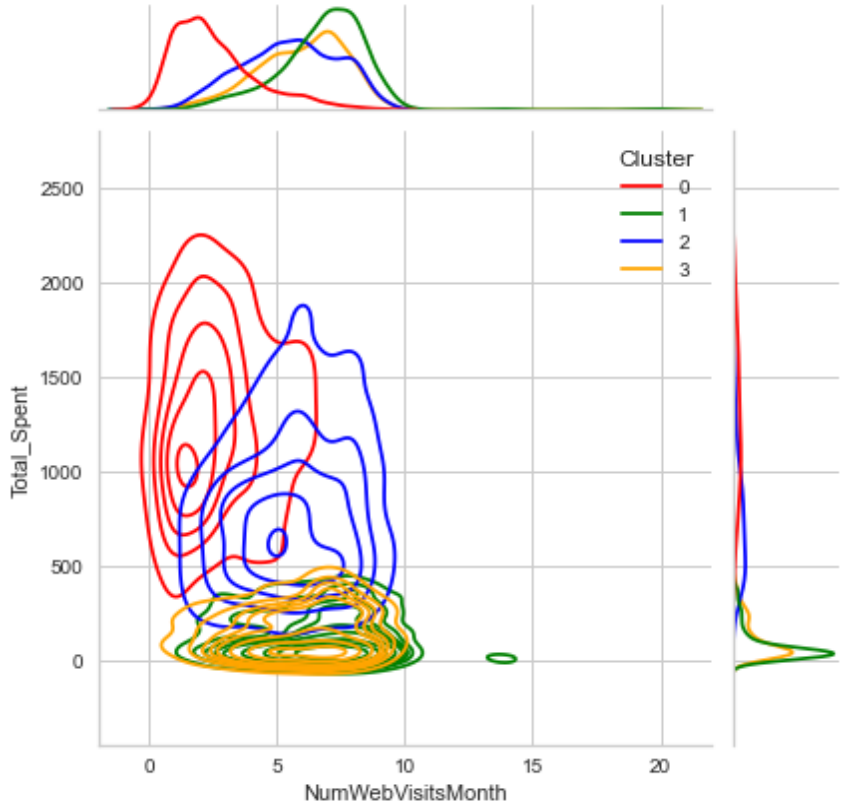

Figure 45: NumWebVisitsMonth vs Total_Spent of K-Means Model

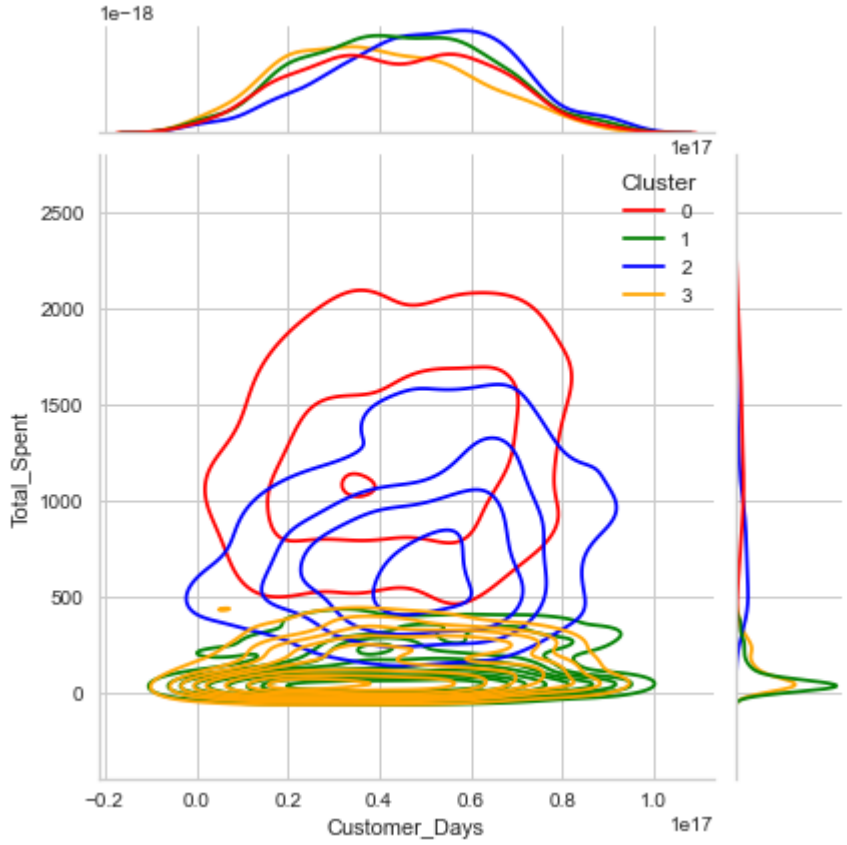

Figure 46: Customer_Days vs Total_Spent of K-Means Model

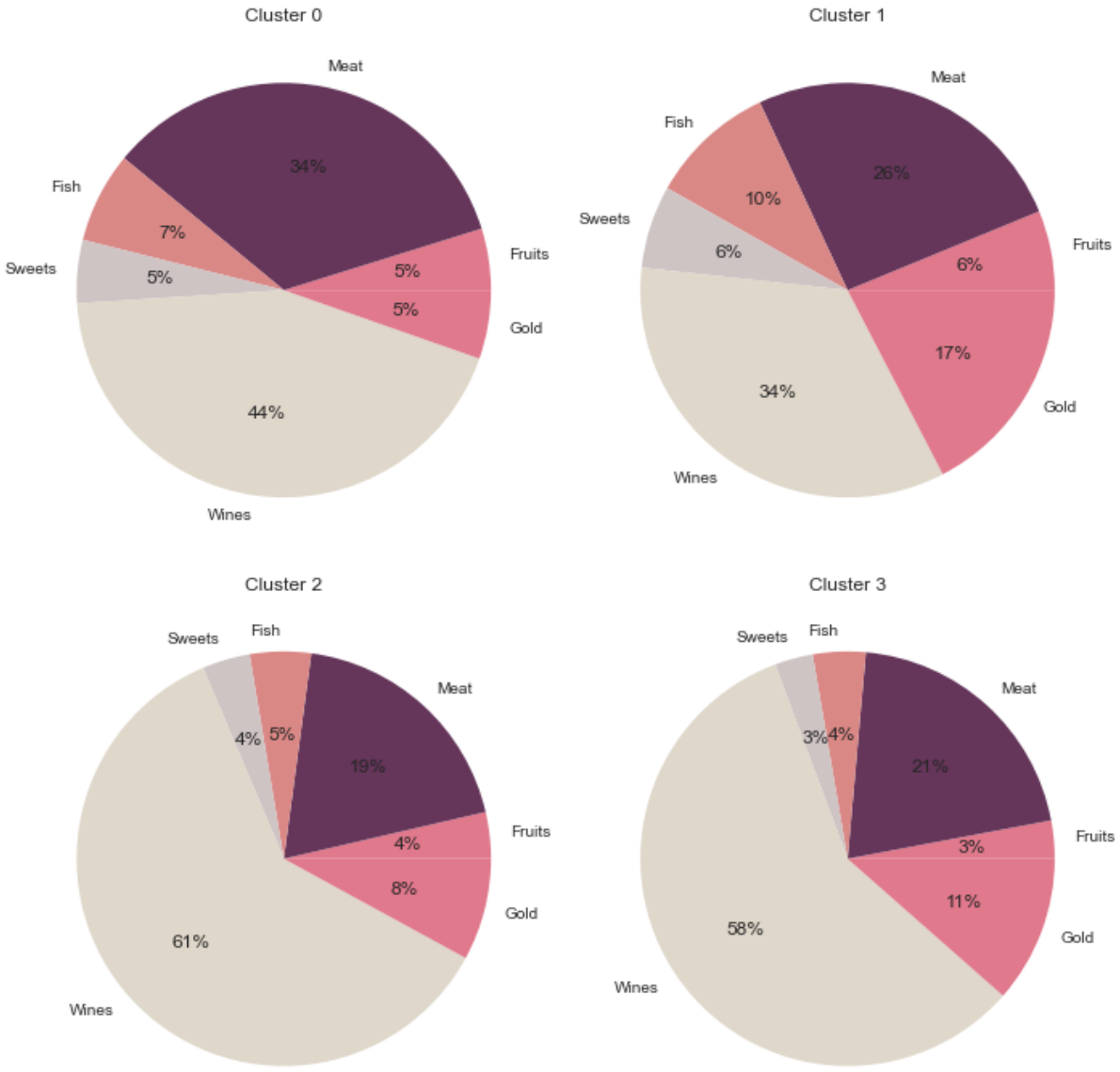

Figure 47: Pie Chart of Products of K-Means Model

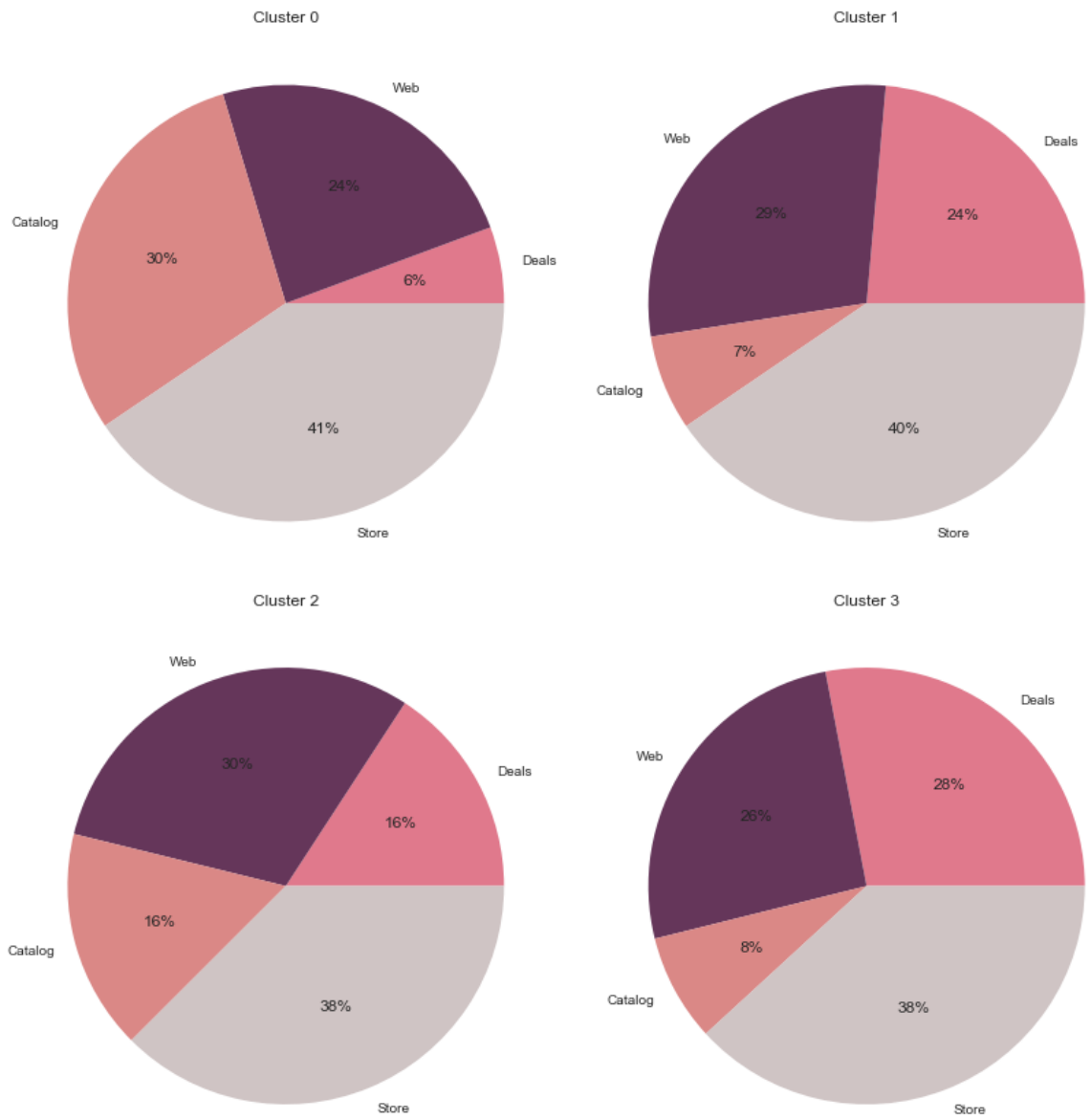

Figure 48: Pie Chart of Purchases of K-Means Model

After visualizing all the results that generated from the K-Means model, I can conclude the types of customers in each cluster are as follows:

Cluster 0	Cluster 1	Cluster 2	Cluster 3
High income high spent	Low income low spent	Average income average spent	Low income low spent
Half of customer is younger	Most younger	Most younger	Most younger
Definitely no child	Most has 1 child and some 0 child	Most has 1 child and some has 2 children	Above average has 2 children, some has 1 chid
Definitely not parent	Most is parent	Definitely is parent	Definitely is parent
Above average graduate, some postgraduate	Above average graduate, some postgraduate and undergraduate	Half graduate, half postgraduate	Half graduate, half postgraduate
Above average married	Above average married	Above average married	Above average married
Spent most on wines, second on meat	Spent most on wines	Spent most wines	Spent most on wines
37% in store	37% in store	30% in web	46% in store

Table 3: Overview of K-Means

3.9.3 Overview

As you can see from the analyzed result above, there are only 3 clustering algorithms where one clustering algorithm is not being analyzed which is the DBSCAN as this algorithm only generated one cluster. Hence, I decided to drop it. Comparing all these 3 clustering algorithms, I have to choose one that fits the data set better by calculating the Silhouette score for each algorithm and comparing them. It is a metric that is used to determine which algorithm will fill the data set better by comparing all the Silhouette scores. It is a measurement of how well the data point is assigned to the cluster compared to other clusters. If the Silhouette score of the algorithm is high, this means the cluster is well-fitted and well-separated from other clusters. Hence, the algorithm with the highest Silhouette score is considered the best-fitting model for the data set. However, the Silhouette score is not the only method to evaluate the clustering algorithm, it is worth using other methods to evaluate such as the elbow method.

Eventually, I calculated the Silhouette score for these 3 clustering algorithms that I applied and it showed the K-Means algorithm is the best-fitting model among those 3 algorithms. Hence, this is the goal throughout this project and I will show this cluster result generated by K-Means to the company and choose which cluster is more suitable to achieve the goal or solve the problem.

4.0 Project Planning

4.1 JIRA

A project plan is playing an important role in a big project because it can provide a shared vision with the supervisor for what the project aims to accomplish. This shared can let the supervisor keeps track of my project progress in order to make sure that I keep working on it and deliver the results. Hence, there is a planning tool provided by School which is the JIRA and it is used to plan, track and manage projects.

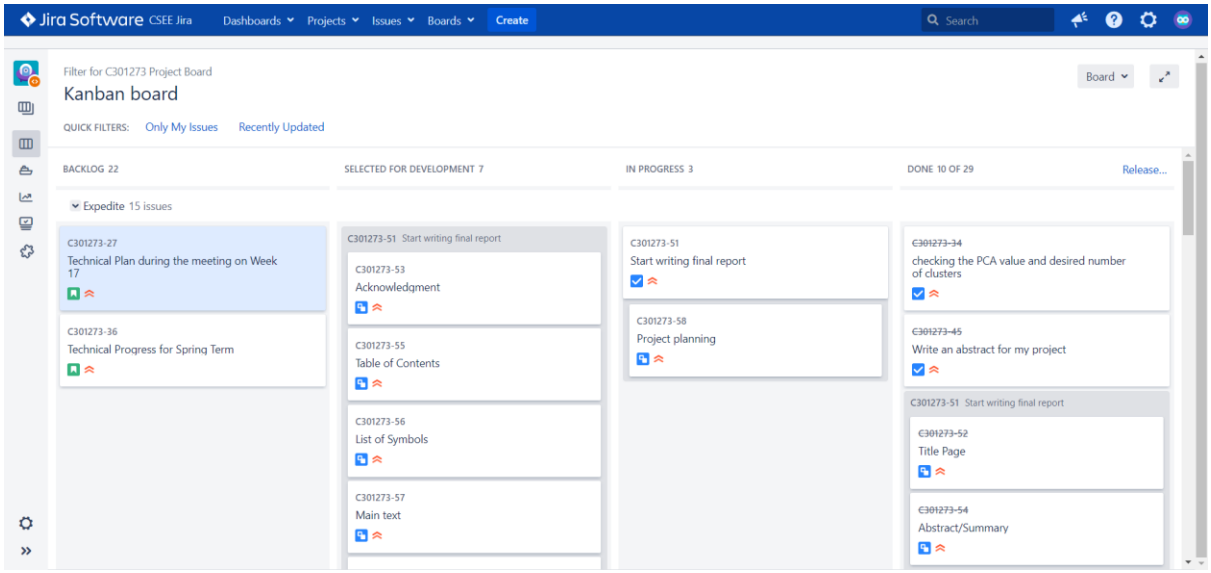

Figure 49: JIRA;s Kanban Board

The figure above shows a picture of what the Kanban board looks like, and there are four sections which are Backlog, Selected for Development, In Progress, and Done.

In the backlog, there is a list of feedback from my supervisor and a bunch of issues that I have created describing what I am going to accomplish on a project. Every week I will have a meeting with my supervisor to discuss my progress on my project and he will give feedback and it will show up on the backlog. Thus, I will review the feedback and reflect on my progress and accomplish what I have discussed with my supervisor. Other than that, whenever I create a new task for my project, it will appear on the backlog and the supervisor will know it. Then I can choose which task I want to develop in the next step by dragging the task from Backlog to Selected For Development. Once I start

working on the task, I will drag it to the In Progress column. Last, I will drag to the Done column once I completed the task.

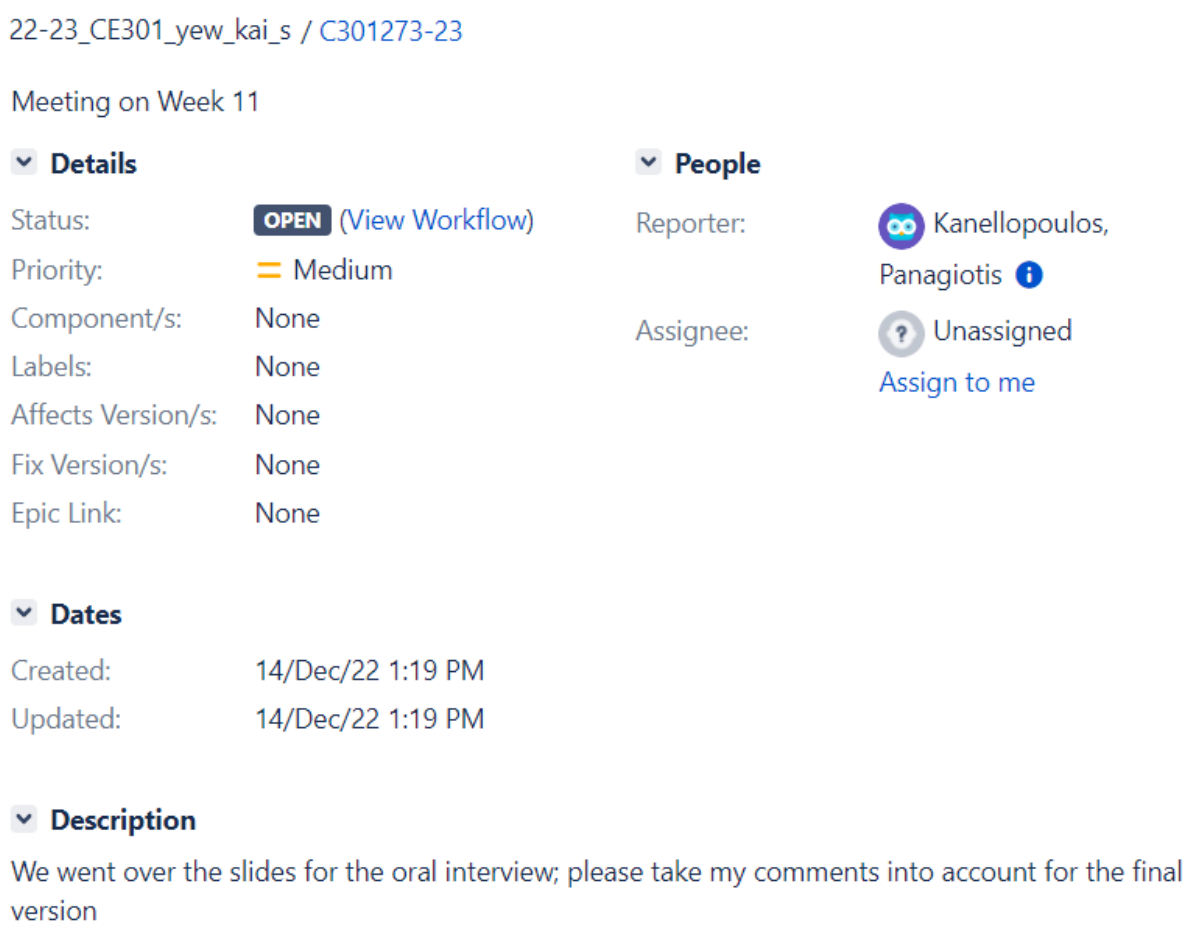

Figure 50: Description of a Backlog

I will take one example from the backlog. The figure above, it is showing the feedback from the supervisor under the Description after our weekly meeting which happened on Week 11, and the Dates is showing when this feedback is created. This feature allows me to revise what I have been discussed with the supervisor and reflect what are the mistakes that I have made or any changes I should make.

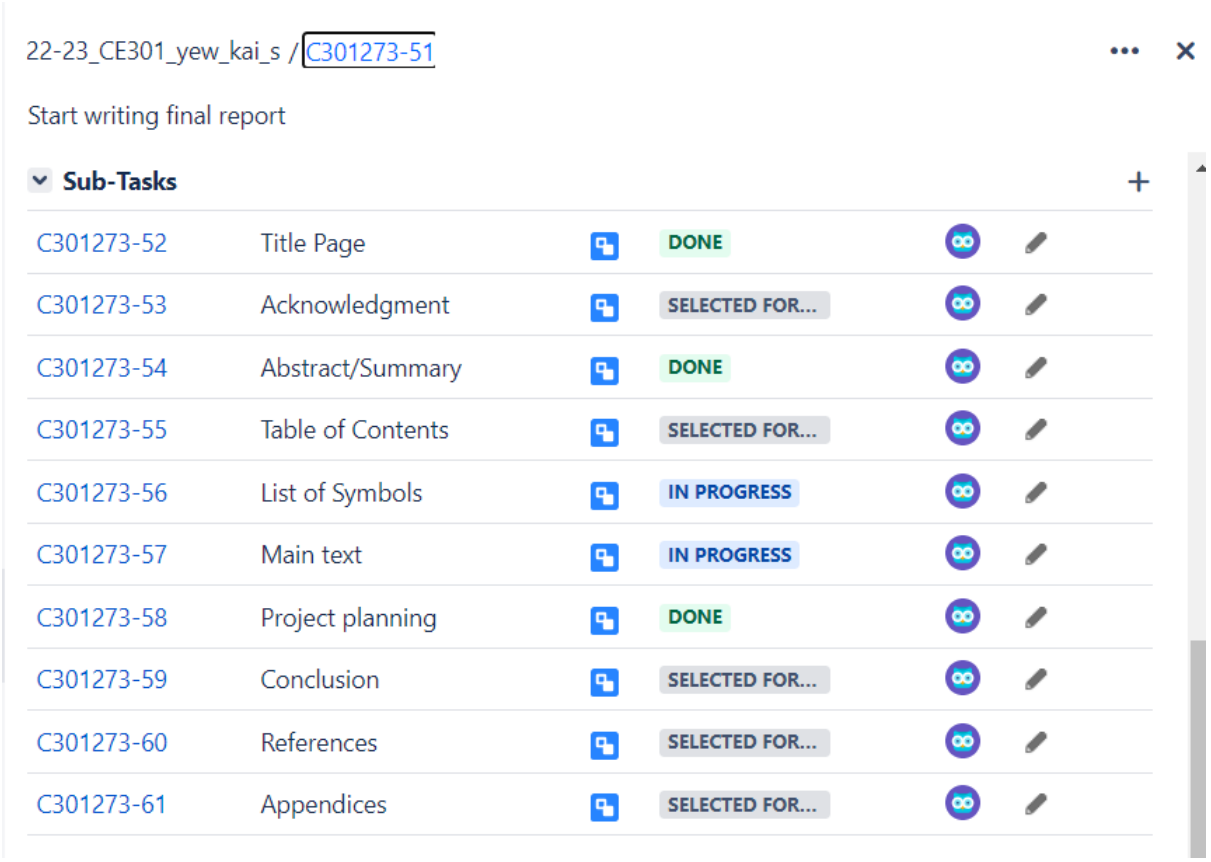

Figure 51: Main-task and Sub-task

I will take one example of how I have been using this feature all the time. The figure above is showing the task that I have created for this project. The status of this task I have set to In Progress which means I am working on this task and I will set the priority to Highest as this will remind me that I have to finish task before other tasks. And I have created a subtask named, Project Planning, and the Dates is almost the same because once I created this task, I have start working on this task. By doing this, I have to complete all the subtasks and then will only consider this main task as completed.

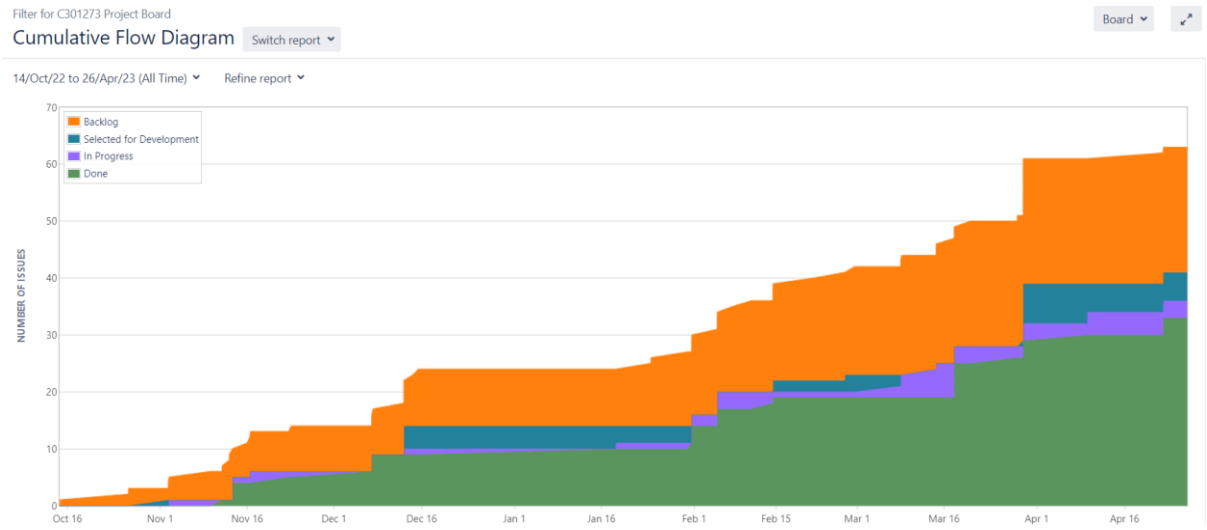

Figure 52: JIRA's Burn Down Chart

The figure above is showing the progress flow of my project from the first day I started working on it. There are four colors and they have their use. You can see the orange is keep increasing over time as every time after the weekly meeting, my supervisor will give feedback and leave it on the backlog. I started to make some progress on 2nd November as I have been doing some research on my project for two weeks. There were two bottlenecks that happened on December 2022 and February 2023 because there were some technical issues on my project and I have some other module assignments that need to submit before the due date. Hence, there is not much progress during these two bottlenecks. On the other hand, there is not much purple and blue as I always forget to create the tasks that I have completed.

Overall, I found it very useful in my project as my supervisor can see my progress and give some feedback during the weekly meeting. Especially the Kanban board, it gives me a visualization of my workflow and improves efficiency by setting the priority of each task and limiting the work-in-progress. But, there are some mistakes that I have made and I need some time to reflect and adapt so I could do better in the future. For instance, I always forget to update the Kanban every time I have a new task or I have completed the task. This will let my supervisor thinks that I didn't put any effort into my project and worries about my progress.

4.2 GitLab

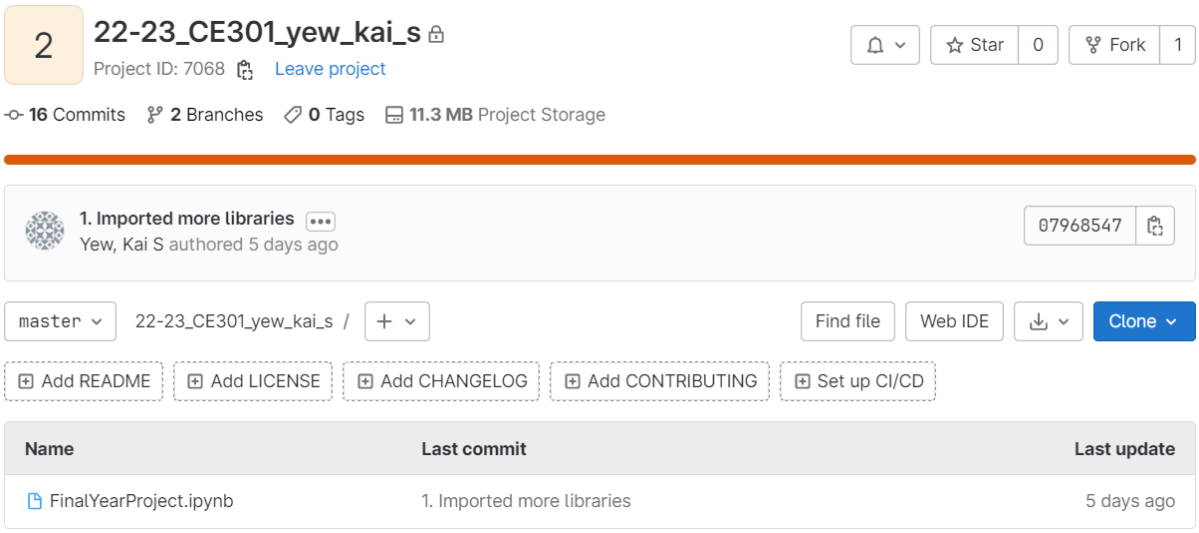

Figure 53: GitLab

Another project management tool provided by School is Gitlab which can give me a visualization of my project timelines by tracking all the issues that I have committed.

4.3 Reflection

Unfortunately, there were some mistakes that I made on the start day of my project. The first mistake was I should upload the code file from the first day I start my project every time I add a new function or make any changes but I only uploaded it less than 5 times which is bad as my supervisor can't see my progress. After I realized that, I straight-committed the latest file with the most completed parts and onwards I will commit every time I add a new function or make any changes. And the second mistake was the way I upload the code file is wrong as I delete the old version file and upload the new version file. But I solved the mistake by asking my friend the correct way to use Gitlab and how to commit and push a file every time I want to make a change. At first, this is my first time using project management tool and I spent so much time on the technical part such as coding, which lead me to forget to update the Kanban board. Using these two project management tools, JIRA and GitLab, truly made me realise the importance of planning and managing project in terms of completion time, project execution. In a nutshell, I would like to thank JIRA and GitLab as I have learned a lot from using these two project management tools such as improving my project management and time management. JIRA, allows me to have an overview of the whole process of my project, track and manage the tasks and issues that need to be done in the given time throughout the project.

5.0 Conclusions

I would like to say thank you to my supervisor, Panagiotis KANELLOPOULOS, as he helped me a lot throughout this project, for instance, he always had a meeting with me every week to discuss what tasks I have done and what are the problems I'm facing. Sometimes, he will give me some ideas on the project when I was stuck on some parts. I was so proud of myself as I never thought I would complete this capstone project. It did take me a lot of time to do research and understand the whole project as this project is about Data Science and my major is Computer Science. I learned Python and some modules related to Data Science during my second year and this helped me to have a basic understanding and the concept of unsupervised machine learning. When I start doing this project, I only planned to implement one algorithm but it ended up with four algorithms which are K-Means, DBSCAN, GMM, and AHC. As I already know what are K-Means, thus, I need to spend extra time to do some research on the other algorithms by reading some articles. At an early stage, it was difficult for me to understand other algorithms as my major is not Data Science and I have no one can ask for help. Somehow, I did manage to get through it. The goal of this project is to implement different clustering algorithms, compare and determine which algorithm is the best for the dataset, and I found K-Means is the best algorithm among the four algorithms, which means I have achieved the goal. There are some limitations to this project, for instance, I should integrate with other tools such as Tableau, which is a software tool that can allow me to visualize the data interactively. But, the technical part of this project took me so much time which left me with no time to use Tableau. In the future, I hope I can spend some time on how to integrate Tableau and Python and implement more algorithms such as BIRCH, OPTICS, and Affinity Propagation. Besides that, I hope I can improve the models that I have applied in this project by setting the parameters more accurately so the performance of the model will be higher and the result generated will be more accurate.

6.0 References

[1] Oyelade, J., Isewon, I., Oladipupo, O., Emebo, O., Omogbadegun, Z., Aromolaran, O., ... & Olawole, O. (2019, July). Data clustering: Algorithms and its applications. In *2019 19th International Conference on Computational Science and Its Applications (ICCSA)* (pp. 71-81). IEEE.

[2] Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N. and Terzopoulos, D., 2021. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, *44*(7), pp.3523-3542.

[3] Wan, Y., Gong, X. and Si, Y.W., 2016. Effect of segmentation on financial time series pattern matching. *Applied Soft Computing*, *38*, pp.346-359.

[4] Miguéis, V.L., Freitas, A., Garcia, P.J. and Silva, A., 2018. Early segmentation of students according to their academic performance: A predictive modelling approach. *Decision Support Systems*, *115*, pp.36-51.

[5] Marcus, C., 1998. A practical yet meaningful approach to customer segmentation. *Journal of consumer marketing*, *15*(5), pp.494-504.

[6] Milligan, G.W. and Cooper, M.C., 1987. Methodology review: Clustering methods. *Applied psychological measurement*, *11*(4), pp.329-354.

[7] Uppada, S.K., 2014. Centroid based clustering algorithms—A clarion study. *International Journal of Computer Science and Information Technologies*, *5*(6), pp.7309-7313.

[8] Uppada, S.K., 2014. Centroid based clustering algorithms—A clarion study. *International Journal of Computer Science and Information Technologies*, *5*(6), pp.7309-7313.

[9] Kriegel, H.P., Kröger, P., Sander, J. and Zimek, A., 2011. Density-based clustering. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, *1*(3), pp.231-240.

[10] Ram, A., Sharma, A., Jalal, A.S., Agrawal, A. and Singh, R., 2009, March. An enhanced density based spatial clustering of applications with noise. In *2009 IEEE International Advance Computing Conference* (pp. 1475-1478). IEEE.

[11] Xu, X., Ester, M., Kriegel, H.P. and Sander, J., 1998, February. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings 14th International Conference on Data Engineering* (pp. 324-331). IEEE.

[12] Reynolds, D.A., 2009. Gaussian mixture models. *Encyclopedia of biometrics*, *741*(659-663).

[13] Reynolds, D.A., 2009. Gaussian mixture models. *Encyclopedia of biometrics*, *741*(659-663).

[14] Bouguettaya, A., Yu, Q., Liu, X., Zhou, X. and Song, A., 2015. Efficient agglomerative hierarchical clustering. *Expert Systems with Applications*, *42*(5), pp.2785-2797.

[15] Van Der Maaten, L., Postma, E. and Van den Herik, J., 2009. Dimensionality reduction: a comparative. *J Mach Learn Res*, *10*(66-71), p.13.

[16] Abdi, H. and Williams, L.J., 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, *2*(4), pp.433-459.

[17] Thinsungnoena, T., Kaoungkub, N., Durongdumronchaib, P., Kerdprasopb, K. and Kerdprasopb, N., 2015. The clustering validity with silhouette and sum of squared errors. *learning*, *3*(7).

[18] Brusco, M.J. and Steinley, D., 2007. A comparison of heuristic procedures for minimum within-cluster sums of squares partitioning. *Psychometrika*, *72*, pp.583-600.

[19] Liu, F. and Deng, Y., 2020. Determine the number of unknown targets in open world based on elbow method. *IEEE Transactions on Fuzzy Systems*, *29*(5), pp.986-995.