TECH MAHINDRA

Tech Mahindra, Hyderabad, Telangana, India

A REPORT ON

CUSTOMER SEGMENTATION ON WHOLESALE DATA USING DATASCIENCE

BY

VNS Keerthana Sudina

(179303166)

Department of Computer and Communication Engineering

MANIPAL UNIVERSITY JAIPUR

ACKNOWLEDGEMENTS

The technical internship opportunity I had at EMEA Oneness Lab at Tech Mahindra Ltd was a great chance of learning and professional development. This Internship has helped me gain a real-time experience of working in an office. I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, to attain desired career objectives.

I would like to express my deepest gratitude to my industry mentor, Mr. Srikanth Rao Amarachinta, Project Manager, for sharing his invaluable experience and guidance and also for allowing me to do an internship at their esteemed organization.

I am highly indebted to Technical Supervisors Mr. Vijay Kumar Botla, Mr. Banudeva Reddy Bandi and Ms. Arpitha Ravi for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would also acknowledge my college for enabling us to go through an industrial internship to get hands-on experience of working in a professional environment

TABLE OF CONTENT

TABLE OF FIGURES

ABSTRACT

This report is related to the project being worked upon in the duration of the Technical Internship Program. The project is based on the dataset that refers to clients of a wholesale distributor. It includes the annual spending in monetary units (m.u.) on diverse product categories.

This project is done by using Data Science with python along with libraries such as Sklearn, Numpy, and Pandas. The given project is an unsupervised classification and clustering technique is applied to the data provided. Clustering is an automatic learning technique aimed at grouping a set of objects into subsets or clusters. In plain words, objects in the same cluster should be as similar as possible, whereas objects in one cluster should be as dissimilar as possible from objects in the other clusters. In this project kmean algorithm is implemented to create clusters that are coherent internally, but substantially different from each other.

PROBLEM STATEMENT

To analyze the customer's behavior (purchases made in the past by the clients) to define the marketing strategy by using various clustering techniques to segment customers. Clustering is an unsupervised learning algorithm that tries to cluster data based on their similarity. Thus, there is no outcome to be predicted, and the algorithm just tries to find patterns in the data.

INTRODUCTION

Company Profile

Tech Mahindra represents the connected world, offering innovative and customer-centric information technology experiences, enabling Enterprises, Associate and the Society to Rise™. **Tech Mahindra Limited** Indian multinational provider of Information Technology (IT), networking technology solutions, Integrated Engineering Solutions(IES) and Business Process Outsourcing (BPO) to various industry verticals and horizontals.

It is a subsidiary of the Mahindra Group; Tech Mahindra is a USD 4.9 billion company with 121,000+ professionals across 90 countries, helping 938 global customers including Fortune 500 companies. Tech Mahindra is the highest ranked Non-U.S. company in the Forbes Global Digital 100 list (2018) and in the Forbes Fab 50 companies in Asia (2018).

Data Science

Data science is the study of data. It involves developing methods of recording, storing, and analyzing data to effectively extract useful information. The goal of data science is to gain insights and knowledge from any type of data (structured or unstructured). Data Science can also be defined as extraction, preparation, analysis, visualization, and maintenance of information. It is a cross-disciplinary field that uses scientific methods and processes to draw insights from data.

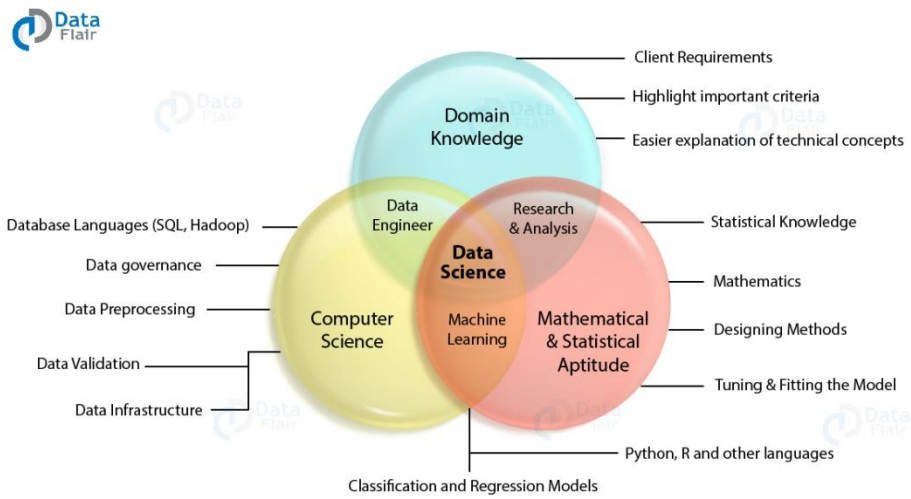

Figure 1: data science

Python

Python is a high-level language used for general-purpose programming. It is a dynamic language that supports both structured programmings as well as object-oriented programming.

Unlike C and Java, Python focuses on readability. Hence it attracts lots of developers and also has a very large developer community which in turn brings out large support to everyone. Also, Python has a very large library, which eases lots of tasks.

Why python is used in Data Science?

The below points show how Python helps in each step of data analyses:

- Assume data as a very huge Excel sheet with a large number of rows and columns. From these large data, we derive insights by performing some operations and searching for a particular type of data in each column and row. Performing such high computational tasks can be cumbersome and extremely time-consuming. Hence Python provides libraries like **Numpy** and **Pandas** which eases this task by the use of *parallel processing*.

- Not always we have data readily available to us. Sometimes we need to scrap the data from the web. Here Python has libraries like **beautifulsoup** and **scrapy** which help extract data from the internet.
- Seeing so many numbers all over screen might turn out to be a big headache sometimes and difficult to derive insights. The only way of doing this is to represent the data in the form of figures like bar graphs, histograms, and pie-charts. Python has libraries for this too. For this, we use libraries like **Matplotlib** and **Seaborn**.
- Machine learning is an **incredibly high computational** technique that involves heavy mathematics like calculus, probability and matrix operations over thousands of rows and columns. All this becomes very simple and efficient with the help of **scikit-learn**, a machine learning library in python.

PROJECT METHODOLOGY

Understanding of Python

Data science with python involves learning of basics such as types, expressions, and variables, string operations, lists, tuples, sets, dictionaries, conditions and branching, loops, functions, objects and classes, reading and writing files.

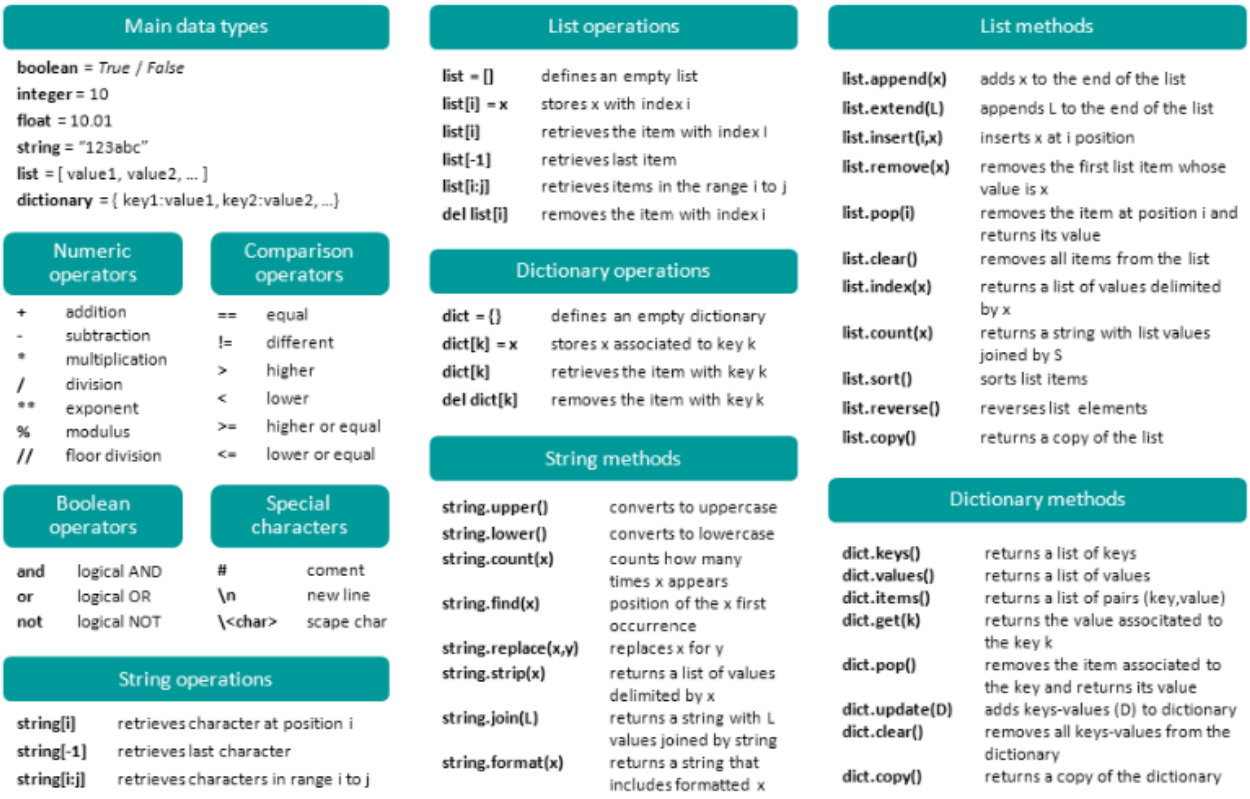

Figure 2: python basis

Libraries used

Pandas

Pandas is an open-source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Features of Pandas

- **Handling of data:** The Pandas library provides a really fast and efficient way to manage and explore data. It does that by providing us with Series and Data Frames, which help us not only to represent data efficiently but also manipulate it in various ways.
- **Alignment and indexing:** Organization and labeling of data are perfectly taken care of by the intelligent methods of alignment and indexing, which can be found within Pandas.
- **Handling missing data:** One of the many problems associated with data is the occurrence of missing data or value as they might adulterate study results. Some Pandas features have you covered on this end because handling missing values is integrated within the library.
- **Merging and joining of datasets**: Pandas can help to merge various datasets, with extreme efficiency so that we don't face any problems while analyzing the data.
- **Visualize:** Visualizing is what makes the results of the study understandable by human eyes. Pandas have an in-built ability which helps in plotting data and see the various kinds of graphs formed.
- **Grouping:** With the help of the features of Pandas like GroupBy, we can split data into categories, according to the criteria set. The GroupBy function splits the data, implements a function and then combines the results.
- **Perform mathematical operations on the data**

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

In NumPy, arrays allow a wide range of operations that can be performed on a particular array or a combination of Arrays. These operations include some basic Mathematical operation as well as Unary and Binary operations.

Every Numpy array is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. Every ndarray has an associated data type (dtype) object. This data type object (dtype) provides information about the layout of the array. The values of a ndarray are stored in a buffer that can be thought of as a contiguous block of memory bytes which can be interpreted by the dtype object. Numpy provides a large set of numeric data types that can be used to construct arrays. At the time of Array creation, Numpy tries to guess a data type, but functions that construct arrays usually also include an optional argument to explicitly specify the data type.

Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram, etc.

Scikit learn

Scikit-learn is an open-source Python library for machine learning. The library supports state-of-the-art algorithms such as KNN, XGBoost, random forest, SVM among others. It is built on top of Numpy. Scikit-learn is widely used in kaggle competition as well as prominent tech companies. Scikit-learn helps in preprocessing, dimensionality reduction (parameter selection), classification, regression, clustering, and model selection.

Seaborn

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.

Some of the functionalities offered by this library are:

- A dataset-oriented API for examining relationships between multiple variables
- Specialized support for using categorical variables to show observations or aggregate statistics
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
- Automatic estimation and plotting of linear regression models for different kinds of dependent variables
- High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
- Concise control over matplotlib figure styling with several built-in themes
- Tools for choosing color palettes that faithfully reveal patterns in your data

It aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

Data Science Methodology

Figure 3: the cycle of data science

a. Business Understanding

The very first step consists of business understanding. Whenever any requirement occurs, firstly we need to determine the business objective, assess the situation, determine data mining goals and then produce the project plan as per the requirement. Business objectives are defined in this phase.

b. Data Exploration

The second step consists of Data understanding. For further process, we need to gather initial data, describe and explore the data and verify data quality to ensure it contains the data we require. Data collected from the various sources is described in terms of its application and need for the project in this phase. This is also known as data exploration. This is necessary to verify the quality of data collected.

c. Data Preparation

Next, comes Data preparation. From the data collected in the last step, we need to select data as per the need, clean it, construct it to get useful information and then integrate it all. Finally, we need to format the data to get the appropriate data. Data is selected, cleaned, and integrated with the format finalized for the analysis in this phase.

d. Data Modeling

Once data is gathered, we need to do data modeling. For this, we need to select the modeling technique, generate test design, build a model and assess the model built. The data model is build to analyze relationships between various selected objects in the data, test cases are built for assessing the model and the model is tested and implemented on the data in this phase.

e. Data Evaluation

Next comes data evaluation where we evaluate the results generated in the last step, review the scope of error and determine the next steps that need to be performed. The results of the test cases are evaluated and reviewed for the scope of error in this phase.

f. Deployment

The final step in the analytic process is deployment. Here we need to plan the deployment and monitoring and maintenance, we need to produce the final reports and review the project. The results of the analysis are deployed in this phase. This is also known as reviewing the project.

PROJECT IMPLEMENTATION

Source and Data collection

Data for project (wholesale customer), the data dictionary, and the business problem statements were obtained from the organization. The dataset consists of 440 rows (instances) and 43 columns (features).

Importing libraries

The necessary libraries are imported.

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

#libraries used for plotting
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
```

Figure 4: importing libraries

Reading the Data set

The dataset is read by using the **read_csv** function by pandas, the data is read into a data frame '**data**' in the notebook.

Pandas Data Frame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

```
data = pd.read_csv("wcd.csv")
data.head()
```

Figure 5: reading the data

df.head() prints only the first five rows of all the columns.

Understanding dataset

The **shape** function is used to find out the number of columns and rows of data. By using the function, it is mentioned that the data consists of 440 rows and 8 columns.

```
data.shape
```

```
(440, 8)
```

Figure 6: The shape of the data

All the 8 features in the data are mentioned by using **columns function**.

```
data.columns
```

```
Index(['Channel', 'Region', 'Fresh', 'Milk', 'Grocery', 'Frozen',
       'Detergents_Paper', 'Delicassen'],
      dtype='object')
```

Figure 7: column names

info() method gives us information about every feature in the dataset like type and number of missing values. There are 8 integer columns in the data.

```
data.info()
```

```
Showing Meta Data :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
Channel             440 non-null int64
Region              440 non-null int64
Fresh               440 non-null int64
Milk                440 non-null int64
Grocery             440 non-null int64
Frozen              440 non-null int64
Detergents_Paper    440 non-null int64
Delicassen          440 non-null int64
dtypes: int64(8)
memory usage: 27.6 KB
```

Figure 8: info() function

describe() method gives us mean, standard deviation, min, max, count and other features of every feature in the data set. It is shown below:

```
data.describe()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

Figure 9: Description of the data

dtypes mentions the data types of all the features present in the data.

```
data.dtypes
```

```
Channel             int64
Region              int64
Fresh               int64
Milk                int64
Grocery             int64
Frozen              int64
Detergents_Paper    int64
Delicassen          int64
dtype: object
```

Figure 10: data types

IsNull finds out the present null values in the data. Since the data provided has no null values, handling of missing data need not be done.

```
pd.isnull(data).sum()
```

```
Channel             0
Region              0
Fresh               0
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
dtype: int64
```

Figure 11: number of null values

Preparing the data

Numerical and categorical features

Categorical data represents characteristics. Therefore it can represent things like a person's gender, language, etc. Categorical data can also take on numerical values (Example: 1 for female and 0 for male).

Numerical data is the numerical measurement expressed not through a natural language description, but rather in terms of numbers. These data have the meaning as a measurement, such as a person's height, weight, or IQ; or they're a count, such as the number of stock shares a person owns, etc.

According to the data, there are 6 **numerical** features. The names of the features are mentioned below:
Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicatessen

There are 2 **categorical** features. The names of the features are mentioned below:
Channel, Region

```
data.Region.value_counts()
```

```
3    316
1     77
2     47
Name: Region, dtype: int64
```

```
data.Channel.value_counts()
```

```
1    298
2    142
Name: Channel, dtype: int64
```

Figure 12: Categorical data

Independent and Dependent variables

Independent variables (also referred to as Features) are the input for a process that is being analyzed. Dependent variables are the output of the process. According to the problem statement provided by the organization, the data is based on unsupervised learning. Hence the data only consists of the independent variables and no dependent variable present.

Single valued features

A Feature having a value that covers more than 90% of the column is called a single-valued feature. All the features of the data are checked and no single-valued feature is present.

Detection of outliers and clipping of data

Outliers are extreme values that deviate from other observations on data; they may indicate variability in measurement, experimental errors or a novelty. The detection of outliers is done by using **quantiles** and **boxplot** on each column. The detection of outliers is only done on numerical features.

```
bf=data[['Fresh','Milk','Grocery','Frozen','Detergents_Paper','Delicassen']]
```

```
def quantile(s):
    data = bf[s]
    print(data.quantile(0.9))
    print(data.quantile(0.95))
    print(data.quantile(0.99))
    print(data.quantile(1))
```

Figure 13: Outlier

In the above figure, quantiles of every column at 0.90, 0.95, 0.99 and 1 are calculated.

```
for x in bf:
    print(x)
    quantile(x)
    print('\n\n')
```

```
Fresh
27090.500000000004
36818.5
56082.61
112151.0



Milk
12229.900000000001
16843.399999999947
37610.06000000003
73498.0
```

Figure 14: Quantiles

According to the above figure, there is a sudden increase in value at a particular quantile. At these particular quantiles, clipping is needed to be done on the entire column.

The other way of finding out outliers is by using a boxplot. Boxplots are a standardized way of displaying the distribution of data based on a five-number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum").

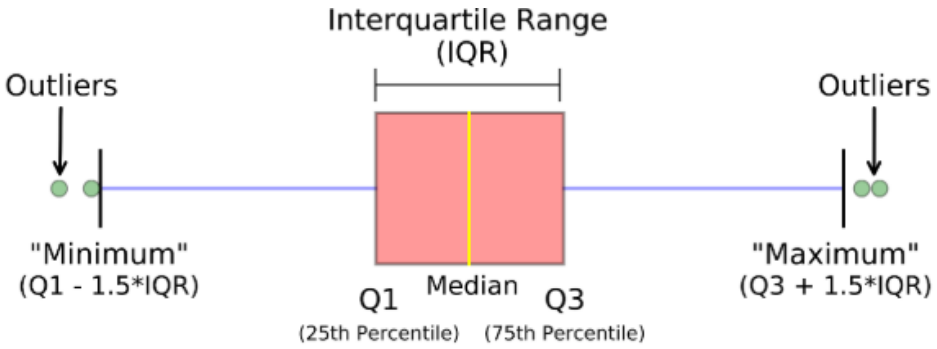

Figure 15: Boxplot

Median (Q2/50th Percentile): the middle value of the dataset.

The first quartile (Q1/25th Percentile): the middle number between the smallest number (not the “minimum”) and the median of the dataset.

Third quartile (Q3/75th Percentile): the middle value between the median and the highest value (not the “maximum”) of the dataset.

Interquartile range (IQR): 25th to the 75th percentile.

Maximum: Q3 + 1.5*IQR,

Minimum: Q1 -1.5*IQR.

Whiskers (shown in blue)

Outliers (shown as green circles)

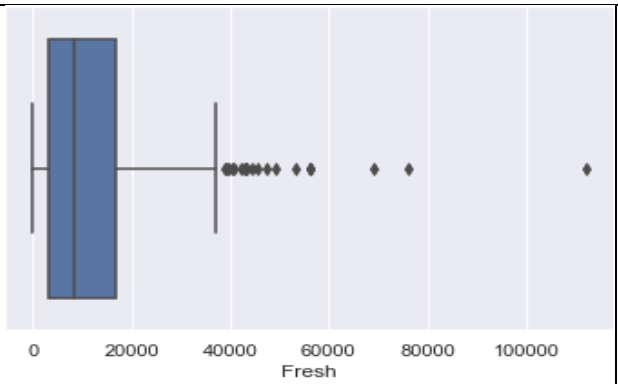

Figure 16: Boxplot of fresh feature

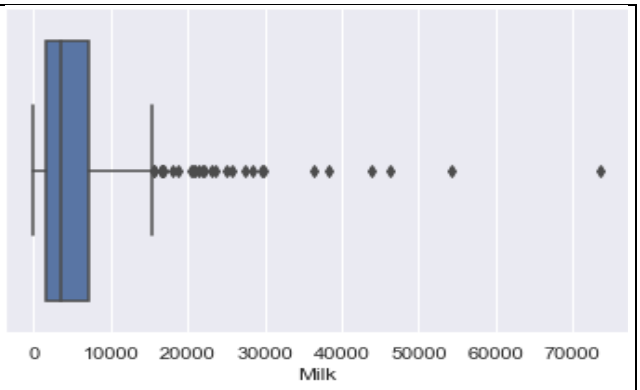

Figure 17: Boxplot of MIlk

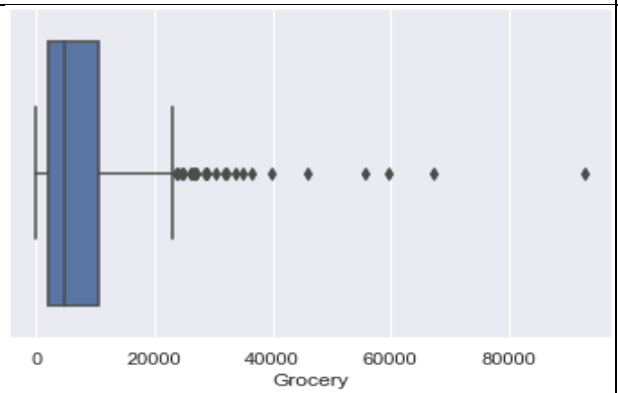

Figure 18: Boxplot of grocery

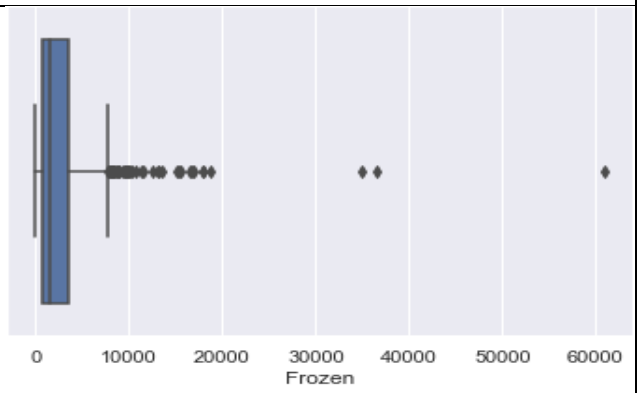

Figure 19: Boxplot of frozen

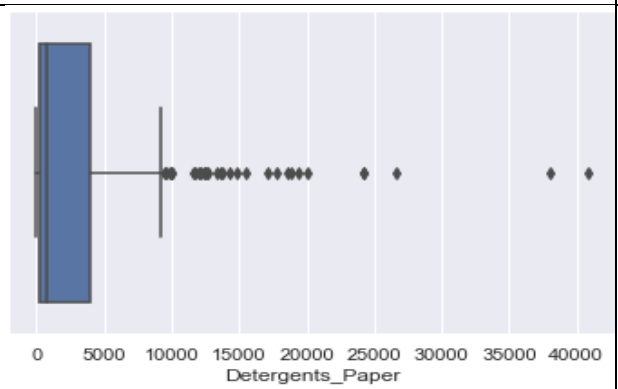

Figure 20: Boxplot of detergent paper

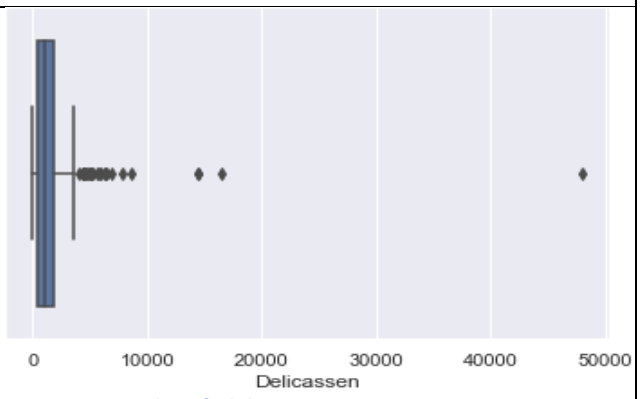

Figure 21: Boxplot of delicatessen

Before the treatment of outliers, the data is scaled using natural logarithm **np.log.**

```
log_data = np.log(dataset[['Fresh','Milk','Grocery','Frozen','Detergents_Paper','Delicassen']].copy())
```

```
log_data.head()
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	9.446913	9.175335	8.930759	5.365976	7.891331	7.198931
1	8.861775	9.191158	9.166179	7.474205	8.099554	7.482119
2	8.756682	9.083416	8.946896	7.785305	8.165079	7.988168
3	9.492884	7.086738	8.347827	8.764678	6.228511	7.488853
4	10.026369	8.596004	8.881558	8.272571	7.482682	7.988168

Figure 22: Natural Logarithm

Later the treatment of the outlier is done by clipping of data. Clipping is used to avoid using outliers in future calculations made.

The columns that are clipped are Fresh, Milk, Grocery, Frozen, Detergents_Paper, and Delicatessen.

```
log_data['Milk']=log_data['Milk'].clip(0,log_data['Milk'].quantile(0.90))
log_data['Fresh']=log_data['Fresh'].clip(0,log_data['Fresh'].quantile(0.95))
log_data['Grocery']=log_data['Grocery'].clip(0,log_data['Grocery'].quantile(0.90))
log_data['Frozen']=log_data['Frozen'].clip(0,log_data['Frozen'].quantile(0.90))
log_data['Detergents_Paper']=log_data['Detergents_Paper'].clip(0,log_data['Detergents_Paper'].quantile(0.90))
log_data['Delicassen']=log_data['Delicassen'].clip(0,log_data['Delicassen'].quantile(0.90))
```

Figure 23: Clipping of data

Dummification applied on categorical features

A Dummy variable or Indicator Variable is an artificial variable created to represent an attribute with two or more distinct categories/levels. A Dummy variable or Indicator Variable is an artificial variable created to represent an attribute with two or more distinct categories/levels.

For example:

Car	Fuel	...
A	gas	...
B	diesel	...
C	gas	...
D	gas	...

Car	Fuel	...	gas	diesel
A	gas	...	1	0
B	diesel	...	0	1
C	gas	...	1	0
D	gas	...	1	0

Figure 24: Dummification example

Categorical features which are to be dummified:

- Channel
- Region

Before dummification, the data frame which was scaled using logarithm is concatenated with Channel and Region features of the dataset data frame.

```
df  =  pd.concat([dataset[['Channel','Region']],log_data],axis=1)
df.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	Retail	other	9.446913	9.175335	8.930759	5.365976	7.891331	7.198931
1	Retail	other	8.861775	9.191158	9.166179	7.474205	8.099554	7.482119
2	Retail	other	8.756682	9.083416	8.946896	7.785305	8.165079	7.988168
3	Horeca	other	9.492884	7.086738	8.347827	8.764678	6.228511	7.488853
4	Retail	other	10.026369	8.596004	8.881558	8.272571	7.482682	7.988168

Figure 25 dummification on categorical features

The **pd.get_dummies** function is used to get the dummies for the respective column.

After dummification, the number of columns has increased from 8 to 9.

```
df = pd.get_dummies(df,columns=['Channel','Region'],drop_first=True)
df.head()
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Channel_Retail	Region_Oporto	Region_other
0	9.446913	9.175335	8.930759	5.365976	7.891331	7.198931	1	0	1
1	8.861775	9.191158	9.166179	7.474205	8.099554	7.482119	1	0	1
2	8.756682	9.083416	8.946896	7.785305	8.165079	7.988168	1	0	1
3	9.492884	7.086738	8.347827	8.764678	6.228511	7.488853	0	0	1
4	10.026369	8.596004	8.881558	8.272571	7.482682	7.988168	1	0	1

Figure 26: Dummification

Feature selection

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

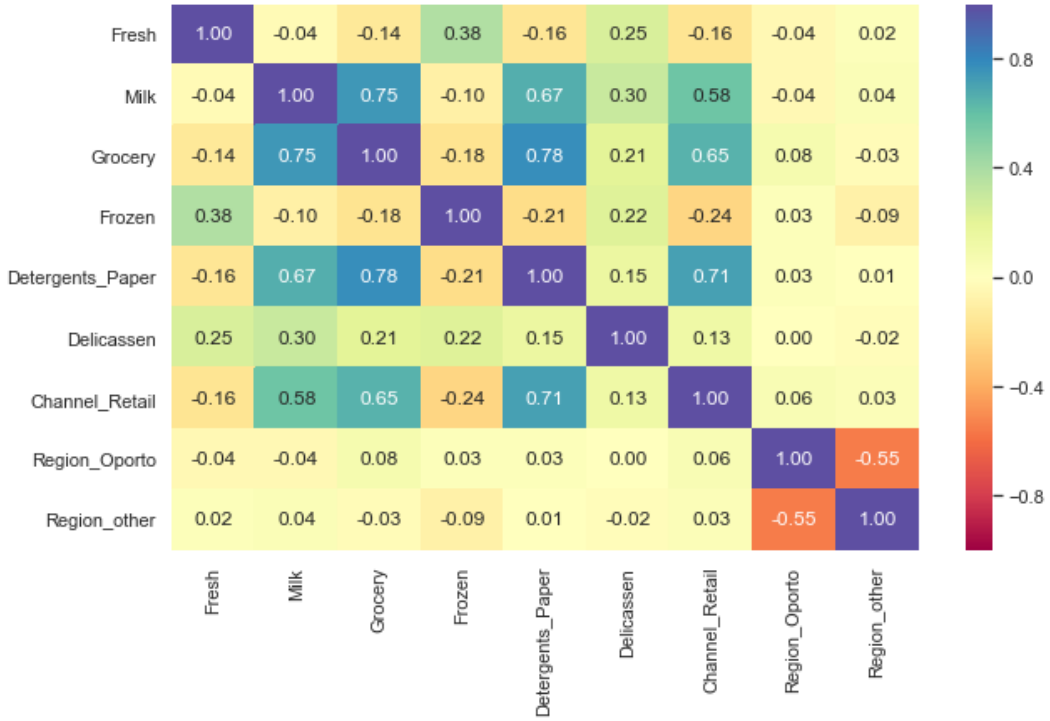

Figure 27: correlation matrix

The range taken is from 0.5 to 0.8 and below -0.5 to -0.8. According to the figure above there are 6 features that fall under the range mentioned and these features are the ones selected that will contribute to the model. The considered features are mentioned below:

- Milk
- Grocery
- Detergents_Paper
- Channel_Retail
- Region_Oporto
- Region_other

Feature Scaling

The dataset contains few features that highly vary in magnitudes, units, and range. Feature scaling should be performed when the scale of a feature is irrelevant or misleading. The algorithms which use **Euclidean Distance** measure are sensitive to Magnitudes. Here feature scaling helps to weigh all the features equally.

The **MinMaxScaler** is a feature scaling algorithm used on the data. It essentially shrinks the range such that the range is now between 0 and 1 (or -1 to 1 if there are negative values).

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df_std = scaler.fit_transform(df)
df_std = pd.DataFrame(df_std,columns=df.columns)
df_std.head()
```

```
D:\Anaconda\lib\site-packages\sklearn\preprocessing\data.py:323: DataConversionWarning: Data with
re all converted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Channel_Retail	Region_Oporto	Region_other
0	0.886689	0.956275	0.895222	0.376039	0.869104	0.885444	1.0	0.0	1.0
1	0.824540	0.959203	0.922131	0.745269	0.895746	0.926548	1.0	0.0	1.0
2	0.813378	0.939267	0.897067	0.799755	0.904129	1.000000	1.0	0.0	1.0
3	0.891571	0.569806	0.828593	0.971280	0.656352	0.927526	0.0	0.0	1.0
4	0.948234	0.849077	0.889599	0.885094	0.816819	1.000000	1.0	0.0	1.0

Figure 28: MinMaxScaler

Model Building

Building a model or set of models to solve the problem, test how well they perform and iterate until you have a model that gives satisfactory results. Stabilize and scale your model as well as your data collection and processing to produce useful outputs in your production environment.

The main aim of the project is to use clustering techniques to segment customers by purchase history and the algorithm implemented on the project is **K Mean**.

K Mean

K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are:

- The centroids of the K clusters, which can be used to label new data

- Labels for the training data (each data point is assigned to a single cluster)

K mean algorithm

The algorithm inputs are the number of clusters K and the data set. The data set is a collection of features for each data point. The algorithms start with initial estimates for the K centroids, which can either be randomly generated or randomly selected from the data set. The algorithm then iterates between two steps:

1. **Data assignment step:**
   - Each centroid defines one of the clusters.
   - In this step, each data point is assigned to its nearest centroid, based on the squared Euclidean distance.
   - More formally, if c_i is the collection of centroids in set C, then each data point x is assigned to a cluster based on

$$\underset{c_i \in C}{\operatorname{argmin}}\ dist(c_i,\ x)^2$$

   - Where $dist(\cdot)$ is the standard (L_2) Euclidean distance. Let the set of data point assignments for each i^{th} cluster centroid be S_i.

2. **Centroid update step:**
   - In this step, the centroids are recomputed. This is done by taking the mean of all data points assigned to that centroid's cluster.

$$c_i = \frac{1}{|S_i|}\sum_{x_i \in S_i} x_i$$

The algorithm iterates between steps one and two until a stopping criterion is met (i.e., no data points change clusters, the sum of the distances is minimized, or some maximum number of iterations is reached).

This algorithm is guaranteed to converge to a result. The result may be a local optimum (i.e. not necessarily the best possible outcome), meaning that assessing

more than one run of the algorithm with randomized starting centroids may give a better outcome.

Implementation of K Mean

Using feature selection process, the relevant features are selected and contributed to the model

```
X = kf[['Milk','Grocery','Detergents_Paper','Channel_Retail','Region_Oporto','Region_other']]
```

Figure 29: features selected

Choosing k

One method to validate the number of clusters is the **elbow method**. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of *k* (say, *k* from 1 to 10 in the examples above), and for each value of *k* calculate the sum of squared errors (SSE). Here inertia, an attribute of K Mean is used to calculate the sum of squared errors (SSE).

```
from sklearn.cluster import KMeans
cluster_range = range(1,20)
cluster_wss=[]
for cluster in cluster_range:
    model = KMeans(cluster)
    model.fit(X)
    cluster_wss.append(model.inertia_)
```

Figure 30: code for elbow method

Later on, using the above function in the figure, the Elbow method curve is plotted.

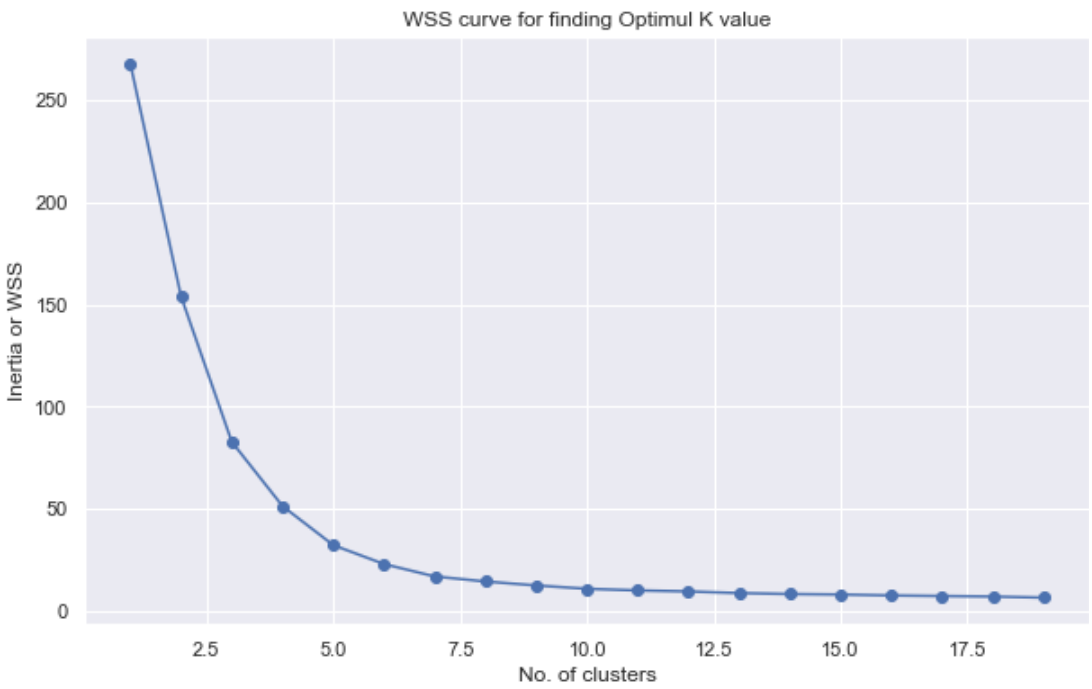

Figure 31: Elbow method

From the figure above, after k=5 the change in the value of inertia is no longer significant and most likely, neither is the variance of the rest of the data after the elbow point. Therefore we can discard everything after k=4. **The k value chosen for this model is 6**.

Parameters and Attributes of Kmean

- **n_clusters** (int, default: 8): The number of clusters to form as well as the number of centroids to generate.
- **N_job's:** It tells the engine how many processors it is allocated to use if it has a value of 1, it can only use one processor, a value of -1 means that there is no limit. After making the model learn let's try to predict the target variable of the test and train dataset.

- **Random state** (int, RandomState instance or None (default)): Determines random number generation for centroid initialization. Use an int to make the randomness deterministic.

The other unmentioned parameters without any particular values are considered in the model by their default value.

- **cluster_centers_** (array, [n_clusters, n_features]): Coordinates of cluster centers.
- **labels_** : Labels of each point
- **inertia_** (float): Sum of squared distances of samples to their closest cluster center.
- **n_iter_** (int): Number of iterations run.

Model fitting

The clustering of unlabeled data can be performed with the module **sklearn.cluster**.

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=6,random_state=100,n_jobs=1)
model.fit(X)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=6, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=100, tol=0.0001, verbose=0)
```

Figure 32: model fitting

The model is fit using the mentioned parameters. The other unmentioned parameters without any particular values are considered in the model by their default value.

Labels_ is an attribute of kmean which is used to generate the labels for the clusters. Since the k values is taken as 6, the labels produced are 0,1,2,3,4,5.

```
model.labels_
```

```
array([2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 0,
       0, 2, 2, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 0, 0, 0, 2, 2,
       2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 0, 0, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2,
       0, 2, 0, 0, 0, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 2, 2, 0, 2, 2, 2, 0,
       0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 0, 0, 2, 2, 2, 0, 0, 0, 2, 2, 2, 2,
       0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 2, 2, 0, 2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 0, 0, 2, 2, 0, 2, 0, 2,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 2, 0, 0, 1, 5,
       1, 1, 5, 5, 1, 1, 1, 5, 1, 5, 1, 5, 1, 5, 1, 1, 5, 1, 5, 1, 5, 1,
```

Figure 33: labels

cluster_centers_ is an attribute that produces coordinates for the centroids of each cluster.

```
model.cluster_centers_
```

```
array([[ 6.81405925e-01,  7.76230272e-01,  6.15390419e-01,
         4.99600361e-16,  2.08166817e-16,  1.00000000e+00],
       [ 6.90491157e-01,  7.84410806e-01,  6.34490081e-01,
         5.55111512e-17,  8.32667268e-17, -3.33066907e-16],
       [ 9.08432714e-01,  9.47494108e-01,  9.35807305e-01,
         1.00000000e+00, -6.93889390e-17,  1.00000000e+00],
       [ 8.75963102e-01,  9.42777927e-01,  9.40020315e-01,
         1.00000000e+00,  1.00000000e+00, -1.11022302e-16],
```

Figure 34: coordinates of centroids

Accuracy of the model

After fitting the model using particular parameters, the model built has obtained an accuracy of 74%.

```
from sklearn import metrics
metrics.silhouette_score(X, model.labels_)
```

```
0.7458207619142911
```

Figure 35: accuracy of the model

Since the model is properly fit, now the predicted clusters are stored into a new data frame under the column called clusters.

```
dataset_final['clusters']=model.predict(X)
dataset_final.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	clusters
0	2	3	12669	9656	7561	214	2674	1338	2
1	2	3	7057	9810	9568	1762	3293	1776	2
2	2	3	6353	8808	7684	2405	3516	7844	2
3	1	3	13265	1196	4221	6404	507	1788	0
4	2	3	22615	5410	7198	3915	1777	5185	2

Figure 36: cluster column

Using the cluster column groupby function is applied to the dataset. **Groupby** essentially splits the data into different groups depending on a variable of your choice.

```
clust_prof = dataset_final.groupby(['clusters'],as_index=False).mean()
clust_prof
```

	clusters	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	0	1.0	3.0	13878.052133	3486.981043	3886.734597	3656.900474	786.682464	1518.284360
1	1	1.0	1.0	12902.254237	3870.203390	4026.135593	3127.322034	950.525424	1197.152542
2	2	2.0	3.0	9831.504762	10981.009524	15953.809524	1513.200000	6899.238095	1826.209524
3	3	2.0	2.0	7289.789474	9190.789474	16326.315789	1540.578947	8410.263158	1239.000000
4	4	1.0	2.0	11650.535714	2304.250000	4395.500000	5745.035714	482.714286	1105.892857
5	5	2.0	1.0	5200.000000	10784.000000	18471.944444	2584.111111	8225.277778	1871.944444

Figure 37: groupby

PCA

The principal component analysis is a fast and flexible unsupervised method for dimensionality reduction in data. Here PCA is used for plotting the clusters obtained.

```
from sklearn.decomposition import PCA
pca2 = PCA(n_components=2)
pc = pca2.fit_transform(df_std)
pc_df = pd.DataFrame(pc)
pc_df.head()
```

	0	1
0	0.777331	-0.227159
1	0.753272	-0.213701
2	0.745141	-0.211639
3	-0.344032	-0.326709
4	0.676513	-0.216450

Figure 38: Implementation of PCA

Later the clusters column is concatenated with the PCA data frame.

```
pca = pd.concat([pc_df,dataset_final['clusters']],axis=1)
pca.columns = ['pc1','pc2','clusters']
print(pca.shape)
pca.head()
```

```
(440, 3)
```

	pc1	pc2	clusters
0	0.777331	-0.227159	2
1	0.753272	-0.213701	2
2	0.745141	-0.211639	2
3	-0.344032	-0.326709	0
4	0.676513	-0.216450	2

Figure 39: Concatenation of clusters and PCA DF

Plotting of clusters

In the mentioned below figure, the star-shaped structures indicate the centroids of each cluster. These centroids are obtained by using clusters_centers attribute.

The grey dots are the data points assigned to its nearest centroid, based on the squared Euclidean distance.

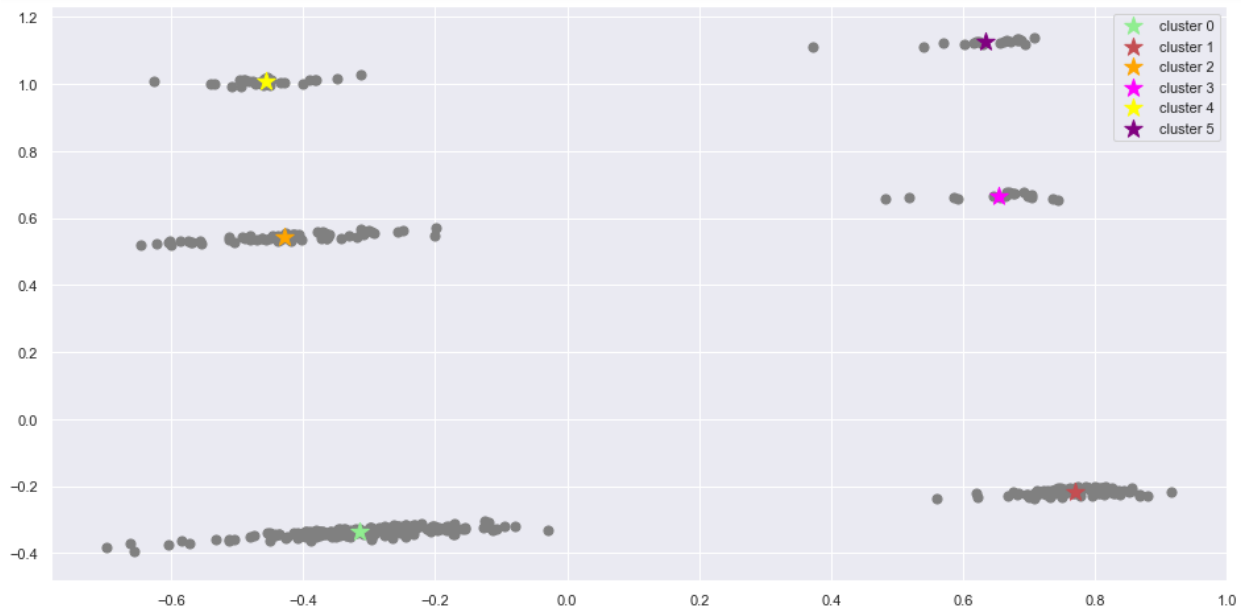

Figure 40: Plotting of clusters

RESULT

The main goal of the project was to implement customer segmentation on the bases of the purchase history of each client from a wholesale distributor. And this goal is achieved by implemented of **the K Mean algorithm**.

The K-means clustering algorithm is used to find groups that have not been explicitly labeled in the data. This can be used to confirm business assumptions about what types of groups exist or to identify unknown groups in complex data sets. Once the algorithm has been run and the groups are defined, any new data can be easily assigned to the correct group. Clustering is an unsupervised learning algorithm that tries to cluster data based on their similarity. Thus, there is no outcome to be predicted, and the algorithm just tries to find patterns in the data.

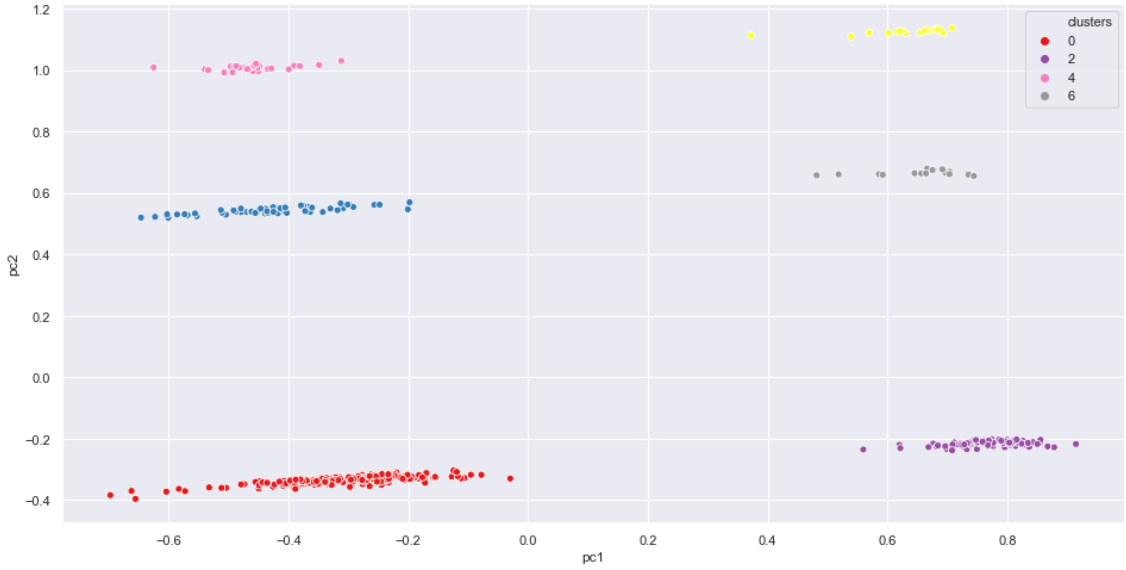

Figure 41: plotting of clusters

The figure above depicts that the K mean has created 6 groups or clusters which were not been explicitly labeled in the data. The model has attained an accuracy of 74%.

WEEKLY REPORT

TIME FRAME	PROPOSED SCHEDULE
WEEK 1 (22-05-19 TO 24-05-19)	-Introduction -Project Introduction
WEEK 2 (27-05-19 TO 31-05-19)	-learning of the basics of python -learning of the numpy and pandas package -practice the pandas and numpy commands in Jupyter notebook
WEEK 3 (03-06-19 TO 07-06-19)	-learning basics of data science -understanding different machine learning algorithms
WEEK 4 (10-06-19 TO 14-06-19)	-Understanding the data -Identifying the output variable -cleaning and preparing the dataset -Identifying which algorithm to be used
WEEK 5 (17-06-19 TO 21-06-19)	-Application of different classification model and trained the data -accuracy of performance of each algorithm obtained
WEEK 6 (24-06-19 TO 28-06-19)	-Learning about chatbot -Learning about the most commonly used chatbots -Implementation of Chatbot
WEEK 7 (01-07-19 TO 05-07-19)	-Worked on the report of project 1 and project 2
WEEK 8 (08-07-19 TO 12-07-19)	-Understanding the Wholesale customer data -cleaning and preparing of data -Application of Kmean algorithm
WEEK 9 (15-06-19 TO 19-06-19)	Worked on the report of project 3

CONCLUSION

After the completion of the project: Network anomaly detection, I got to learn about the network and different kinds of attacks faced by it. This project has helped obtain the basics of data science and how the data is analyzed in real-time.

The project aimed to predict the target variable by binomial classification. The data given was understood, prepared and analyzed. A statistical model was successfully built, generating a final result by successfully segmenting the customer's behavior using clusters. The model has attained an accuracy of 96.5%.

Undergoing this internship experience has made sure that a lot was learned about working within deadlines in a professional environment.

RESOURCES USED

The resources used in this project are

- IDE'S – Jupyter Notebook
- Software – Microsoft Excel
- Programming Languages -Python

REFERENCES

- https://www.kaggle.com/saipraneeth1996/whole-sale-customer-segmentation-using-clustering
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
- https://www.datascience.com/blog/k-means-clustering
- https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html
- http://benalexkeen.com/feature-scaling-with-scikit-learn/
- https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a