DISSERTATION

CUSTOMER SEGMENTATION FOR ONLINE RETAILERS USING RFM MODELING TO ADOPT PROFITABLE CUSTOMER-CENTRIC MARKETING STRATEGIES

By

Debendra Ray, M.Sc.

Presented To The

SWISS SCHOOL OF BUSINESS AND MANAGEMENT GENEVA

In Partial Fulfillment Of The Requirements For The Degree

DOCTOR OF BUSINESS ADMINISTRATION

Nov 2023

CUSTOMER SEGMENTATION FOR ONLINE RETAILERS USING RFM MODELING TO ADOPT PROFITABLE CUSTOMER-CENTRIC MARKETING STRATEGIES

by

Debendra Ray, M.Sc.

APPROVED BY

Aaron Nyanama, DBA, Chair

Anna Provodnikova, PhD,

Anna Provodnikova, PhD, Committee Member

SagarBansal

Sagar Bansal, DBA, Committee Member

RECEIVED/APPROVED BY:

Associate Dean

DEDICATION

This Research is dedicated wholeheartedly to my parents, wife, and daughter, whose unwavering support and boundless inspiration were instrumental in the completion of this thesis. Their love provided the moral, spiritual, and emotional foundation that made this achievement possible. My parents' wise counsel encouraged me to approach each day with mindfulness toward the ultimate goal. Finally, I am profoundly grateful to my wife, whose exceptional partnership made this journey not only possible but also immensely fulfilling.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the numerous individuals who played a pivotal role in the successful completion of this research.

Dr. Mario Silic, whose expertise guided me through the intricacies of doctoral research, deserves special recognition.

I am deeply indebted to my thesis supervisor, Dr. Sagar Bansal, for his invaluable insights and advice throughout the process. Dr. Bansal not only provided guidance in aligning the Research Proposal and Final-Thesis Report with best practices but also played a crucial role in the finalization of these documents. His support extended beyond academic matters as he assisted in maintaining a steady research pace and offered unwavering moral support, making this challenging journey achievable.

ABSTRACT

CUSTOMER SEGMENTATION FOR ONLINE RETAILERS USING RFM MODELING TO ADOPT PROFITABLE CUSTOMER-CENTRIC MARKETING STRATEGIES

Debendra Ray, M.Sc.
Nov 2023

Dissertation Chair: Aaron Nyanama, DBA

Customer Segmentation is the process of segregating customers based on some prominent features that could help online retailers sell more products with less marketing expenses. This process's rationale is the belief that customers exemplify differences in their attitude, behaviour, and demography. An unsupervised machine learning techniques like cluster modelling can develop groups or segments of a customer population. The goal is to create separate groups, but the groups themselves have closely related features, and this can be a powerful means to identify unsatisfied customer problems. In keeping with this data, companies can outperform the competition by developing customized marketing campaigns, designing an optimal distribution strategy, choosing specific product features for deployment, and prioritizing new product development efforts. Thereby, customer segmentation enables a company to customize its relationships and deliver personalized experiences to customers.

This research aims to develop a good understanding of the company's customer base and its behaviour, using the RFM (Recency, Frequency, Monetary) model to accurately classify existing customers' to underscore its most profitable customer groups.

The purpose of this research is to help the company invest in these customers' segments to generate revenue, reduce costs and be profitable.

This research will also predict the items customers will buy in the future using a collection of machine learning techniques: SVM, Logistic Regression, Decision Tree, Random Forest, Adaboost, Ensemble Classifier. This research will essentially categorize the selected customer base of a UK-based online retailer into appropriate customer segments and predict future purchases based on the customer segmentation.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER I:

INTRODUCTION

1.1 Introduction

Customer segmentation is a process to divide a company's consumer base into subgroups—the generation of subgroups by using some specific characteristics helps the company sell more products with less marketing expenditure. The consumer base of any company consists of two types of consumers: Existing consumers and Potential consumers. The consumer base is categorized into subgroups, and these subgroups are called segments. The subgroups are created in such a way that each subgroup of customers has some shared characteristics. (Sandhya and Hindol, 2018)

Different customer segments will help companies develop a customized marketing campaign in keeping with customer segments. This marketing strategy will help the company sell more products with lower marketing expenses and, in turn, will maximize the profit. This is the rationale for using customer segmentation analysis by the companies. (Sandhya and Hindol, 2018)

Companies use STP (Segmentation – Targetting – Positioning) approach to formulating their marketing strategies. First, the Segmentation stage entails creating customer segments in keeping with their profile characteristics: Demographic, Geographic, Geo-Demographic, Psychographics, Behavioral, Contextual, and Situational. Second, the Targetting stage entails segment evaluation to figure out product fitment for pertinent segments. Finally, the Positioning stage entails exploring the market opportunity and formulate a strategy to determine the best value proposition for the products. This approach helps the companies to achieve a multitude of benefits : (i) Determine appropriate product pricing, (ii) Develop customized marketing campaigns,(iii) Design an optimal distribution strategy,(iv) Choose specific product features for deployment,(v) Prioritize new product development efforts. (Sandhya and Hindol, 2018)

This research analysis will determine customer segments from an unlabelled, unsupervised transactional dataset. A prominent method to perform this segregation of

available data is unsupervised machine learning methods like clustering. In this method, we collect customer features and formulate different clusters that we can extract from the feature set. We can find a customer segment's traits by analyzing the cluster characteristics in the sequel to this.

Exploratory data analysis is yet another method to figure out the customer segments, including top decision points in the study. For instance, finding out the range of spending will provide insight into customer segments based on spending. Likewise, we could explore other essential features of customers until we get customer segments with noteworthy characteristics.

This research will use the clustering-based model to determine customer segments, which provides a mathematical framework to find segment boundaries in the data. The dataset we have consisted of only the customers' sales transactions, and it will be a challenge to work with this limited attribute dataset. As such, we will use an RFM – Recency, Frequency, and Monetary Value-based model and apply an unsupervised machine learning technique: K-means to identify different groups (clusters) for customers. (U Dinesh Kumar, 2017)

This Research will identify customer behavior patterns that increase or decrease business revenue to invest in appropriate customer segments to generate revenue. Customer Lifetime Value is an essential metric that helps the business determine these customer segments and helps companies take appropriate actions.RFM scores of customer clusters are perfect candidates for feature sets required to calculate the CLV.

In the sequel to the above steps for best customer segmentation and lifetime value prediction, the customer retention rate is also an important metric that helps customers determine the fitness of their products for the market(PMF-Product Market Fit) (Le and Ha, 2016). If the PMF is not satisfactory, the business will experience customer churn; therefore, Churn Prediction is a powerful technique to figure out the likely churn in a given period (Atteberry, Loeb, & Wyckoff, 2017).

This Research will use RFM Scores of the customer clusters to predict the customer's next purchase day. In keeping with these predictions, the business can formulate its

marketing actions like no promotional offer for customers who'll purchase anyway and design a campaign for customers who've not bought within a timeframe.

Finally, this Research will also predict the upcoming purchases using the above RFM scores of customer clusters. Using supervised machine learning techniques like SVM, we can predict the cluster to which the new customer will belong based on his first purchase in the online platform. For instance, if a customer has bought a cosmetic product, lives in India, is in the age group 40 – 55, and we have already generated a customer cluster that satisfies these characteristics, we can put this new customer in that particular customer segment and derive the list of items that the customer may buy in future.

The customer segmentation approach adopted in this research for the e-commerce domain applies to other domains as well, wherein the feature set will differ for each domain.

1.2 Research Problem

To segment the customer base efficiently to retain the existing customers and acquire new customers by analyzing online retail customers' buying patterns.

1.3 Purpose of Research

This research aims to develop a hybrid model to optimize the marketing strategy for a company's growth using ensemble ML algorithms.

Objectives of this research in keeping with the aims of this Research:

- To implement the basic model for customer segmentation using RFM, K-means, PCA techniques and measure the precision score.
- To optimize and implement this baseline approach for customer segmentation to increase the Classifier's Accuracy by using other ML algorithms: Logistic Regression, K-nearest neighbour, Decision Tree, Random Forest, Adaboost Classifier, Gradient boosting Classifier.
- To compare the precision scores of all the preceding algorithms to determine the best algorithm using ensemble methods.

1.4 Significance of the Study

This research will propose an optimal approach for customer segmentation using a combination of various machine learning algorithms to achieve high prediction Quality and Accuracy. The companies in the e-commerce domain will benefit from this Research. However, this Research will be equally beneficial for companies providing financial, travel, telecom, marketing, educational, entertainment services to build customer segmentation using datasets for the specific domain. For instance, customer segmentation in the financial domain will consider the data points like frequency of debit/credit card usage, personal/professional profile of customers, etc. Likewise, the customer segmentation in the travel domain will consider the data set about the frequency of flight booking, Demographic/professional profile of customers, etc

1.5 Research Questions

This research will strive to solve the following questions:

- Is the categorization of the customers in a specific segment feasible in keeping with their buying behaviors?
- Is the prediction for the type of items customers will purchase in the near future feasible in keeping with their segmentation?

The research will exclusively focus on the research problems and strive to evolve an approach through which the machine learning classifier's accuracy will be increased. To achieve this, we need to combine multiple machine learning algorithms and compare the results. Necessarily, this research will focus on building a classifier that will classify the customers into different customer segments, and the classifier should generate this classification result when the customer visits the platform for the first time.

This research will exclude the following aspects of customer segmentation, owing to the short span of the research:

- Analysis of Customer segmentation on future Sales.
- Cross-Selling approaches using Market Basket Analysis.
- Churn Analysis for Customer Segments.

CHAPTER II:

REVIEW OF LITERATURE

2.1 Introduction

Businesses cannot thrive without finding new customers (Kiseleva et al., 2017). We embark on the process of discovering new customers, by enhancing our knowledge of existing customers and their accurate classification. We unravel the buying patterns of the customers and the characteristics features that may be used to find those customers. The values on those characteristics features define market segments, which can be used to discover new customers.

A customer cluster is a group of consumers different from other groups of customers that matter to marketing campaigns and can be done through different techniques (Rajagopal, 2011). For target marketing, a customer's cluster membership can be among the most useful explanatory features (Callarisa Fiol et al., 2009).

We underline the customer segments using geographic, demographic, and behavioral characteristics. Features that are accessible and easily measured, are most useful for marketing campaigns (Farris et al., 2010).

The process of discovering customer groups is known as customer segmentation. Customer segmentation, when meticulously designed, helps in marketing campaigns (Clows, 2012). The Customer Segments are characterized by distinct features like what customers like, what they buy, where they buy, how often they buy, and how much they buy (Bhatnagar and Ghose, 2004). Marketing campaigns should be customized in keeping with these characteristics of specific clusters (Brito et al., 2015). Companies use customer segments in product development, advertising, promotion, pricing, and target marketing (Weinstein, 2013). Some customers are more responsive to advertisements, promotions than others. Companies can make decisions about marketing campaigns, by using insights from a customer's market segment (Revella, 2015).

2.2 Existing Literature Overview

This Study motivation stems from a Research Paper published by Chen, Sain & Guo (2012) which explains how data mining techniques of the RFM model could be used for customer segmentation. The dataset referred to in this paper is collected from a UK-based online retailer. They've segmented the above data set into five clusters of customers using the RFM model and K-means clustering algorithm. These clusters segregate Customer Profitability into High Value, Mid Value, and Low-Value. These clusters are further enhanced using a decision tree to gain an insight into the customers' sub-clusters. In the sequel to this Research, Chen, Sain & Guo (2012) could identify the most and the least profitable group of customers. Additionally, they've recommended exploring the synergy between different customer groups by examining the products they have purchased. This insight will help the online retailer determine with which group a new customer will affiliate. Another research recommended that exploring the customers' products to enrich the customer's website buying experience and predict the customer lifecycle value (Chaudhuri et al., 2021).

This Research Paper of Chen, Sain & Guo (2012) has few gaps : (i) In the data pre-processing, the removal of null and duplicate values is not covered. (ii) EDA for various data attributes is not covered. (iii) The Accuracy of the Classifier is not determined. (iv) PCA / Elbow-method / Silhouette analysis is not covered. (v) No metrics for Customer retention are considered either. (vi) Prediction of Future Purchases is not considered in the Research.

Zhen et al. (2014) published a research paper on A decision-making framework for precision marketing in 2014. This research paper proposes a framework to help the business determine customer segments' features and recommends proper marketing strategies to reduce the customer clusters' inventory. The data set referred to in this paper is collected from a company in China to implement the proposed framework. This Research achieves four Objectives: Firstly, a trend model to predict accurately monthly supply quantity; Secondly, segregate the customers using the RFM model; Thirdly, determine distinct customer clusters

using CHAID decision trees and Pareto Values; and finally, targeting customer clusters with distinct supply strategies. The trend model results proposed by this research paper outperform the prediction models like SARIMA and SVM. The RFM model is used to select the k-means algorithm's attributes, and the CHAID algorithm uses the resulting clusters as decision attributes (Rojlertjanya, 2019). To segregate the customers further, Pareto Ratio is used (Grosfeld-Nir, Ronen, and Kozlovsky, 2007). This Research has few gaps: First, the prediction model is simple; Second, it is hard to determine the number of Clusters; Third, the prediction model's scalability is subservient to the historical data size. Finally, this Research can be extended to developing an integrated prediction model combining all the models used in this paper.

Tsai, Hu, & Lu (2015) published a research paper in 2015 to argue the usage of two clustering techniques to solve Customer segmentation strategies for an automobile dealership. This Research explores the problem of customer segmentation about automotive customer relationship management in Taiwan. The clustering techniques: k-means and expectation-maximization(EM) are evaluated for accuracy The evaluation results underline the identification of four customer segments: loyal, potential, VIP, and churn customer groups. To improve the quality of services, several customized marketing campaigns for the four customer groups are suggested. Options include providing valuable presents during vehicle maintenance visits for loyal customers, designating a specific technician to offer maintenance services to potential customers, enhancing the quality of service for VIP customers, and improving relationships with churn customers. By validating the results of this work with different automobile dealers and by tracking the effectiveness of the marketing strategies in the case company, this work could be further extended. Similarly , another research explores the problem of customer segmentation about automotive customer relationship management in UK and Saudi Arabia and found that the clustering techniques are highly important (Ali, 2007)

Xixi He & Chen Li (2016) published their work on the impact of Customer Segmentation on E-commerce Websites. This Study entails a three-dimensional customer segmentation model in keeping with customer lifetime value, customer satisfaction, and customer activity, which classifies customers into different segments with high Accuracy. The RFM model, Kano model, and BG/NBD model are used to extract the variables (Belhadj, 2021). The customer segmentation model results in ten customer segments with corresponding marketing strategies to help enterprises maximize profits (Kim et al., 2006). Customer lifetime value reflects the profit that customers bring to the website; customer satisfaction reflects the extent of customers' recognition of website quality, products, and services; customer activity reflects loyalty and closes the relationship between customers and website. For all types of customer segments, a targeted marketing strategy is designed to retain the customers (Nasır, 2017).

Wen-Yu Chiang (2017) published a research paper: Discovering customer value for marketing systems in 2017 explains a method to identify valuable passengers in Taiwan airliners and retain them to increase the profits and growth rates. An innovative model FSLC (Frequency, high Season traveling, Locations, and Cancellation) is proposed based on the RFM model to identify valuable customers and generate useful association rules to find an optimized target market for dynamic marketing. The Supervised Apriori Algorithm is used based on the FSLC analysis to extract five useful association rules. Using this method, the airlines in Taiwan could extract customer values from their CRM system. This method could be applied to other retailers for analyzing customer values and association rules. An obvious pitfall of this method is that the collection of customer benefits Questionnaires is not easy.

Shohre Haghighatnia, Neda Abdolvand & Saeedeh Rajaee Harandi (2017) published in 2017 an innovative method to analyze customer behavior using discounts as a dimension. An innovative model: RdFdMd in keeping with RFM analysis, wherein d is the level of discount is used to analyze the impact of discounts on customers' purchasing behavior and profitability of the companies. To achieve clustering, the k-means algorithm is used. The

model evaluation results reflect better customer clustering, and discounts are identified as an important factor for customer purchase behavior. Retailers can achieve precise clustering for up-selling strategies using this method. A gap glaring in this method is that the correlation between the product and discounts is not considered. The method could be further extended to explore the time interval between discounts.

Morisada, Miwa, & Dahana published a research paper in 2018 to analyze valuable customer clusters in online fashion markets (Morisada, Miwa, & Dahana, 2018). This study explains the impact of behavioral characteristics of the customer on profitability in different clusters. The segmentation is achieved in keeping with the customer purchase rate, customer lifetime value, and average spending of the customers. Segment-level customer lifetime values are then calculated and combined with past purchase amounts to assign the segments to several tiers. A latent class model of purchase frequency, retention rate, and the purchase amount are developed and applied to a large-scale online transaction dataset of fashion products. The analysis reveals that innovators, brand-conscious customers, mobile device users, and loyal shop customers are more likely to belong to valuable segments. The model evaluation results underscore the impact of customer behavioral characteristics on the company's profitability and help the online fashion companies to design effective customer campaigns. Data from other fashion e-retailer companies could be used to improve the current Research's results. This work could further explore the impact of customer lifestyle on the profitability of online fashion retail companies.

Arash, Mohammad, and Mohammadreza (2018) published a research paper on A novel model for product bundling and direct marketing in e-commerce based on market segmentation in 2018. This paper aims to analyze customer loyalty by segmenting the customer base in keeping with their buying behavior in an online platform. The Dataset includes 541910 records received from 4340 customers from December 2014 to December 2015 in Golestan province in Iran. T Customers are segmented using the RFM model & K-means algorithm, which helps the company determine the customers' loyalty. Product bundles

are recommended to different customer segments accurately, which enhances the success of marketing campaigns.Apriori algorithm is used to establish the association rules between frequently bought products. The artificial neural network (ANN), Bayesian network, K-nearest neighbor (KNN), logistic regression (LR), C5.0 decision tree, and support vector machine (SVM) algorithms are applied to predict the product bundles for specific customer segments. There are many companies which struggle to get new users to their online platforms (Cusumano, Yoffie, & Gawer, 2020). This model will help companies retain the existing customer base and invite new customers to their online platform. The research results show high Accuracy if the RFM scores are factored-into the Product bundling. The Research Paper could be extended to develop a market response model for marketing campaigns, create a price bundling model to keep with the customer characteristics and create a real-time system for recommending product bundles to customer segments. A gap glaring in this Study is that no metrics are tracked to measure the retention rate of the profitable segment of customers.

Research Paper published by Sebastiaan et. al (2018): Profit-driven decision trees for churn prediction, has extensively used the decision trees for Churn Prediction to improve the revenue of the company. This paper proposes a customer churn prediction model: ProfTree based on a decision tree algorithm, to maximize the profit for the company. This technique shows marked improvement over classical churn prediction models, which fails to factor-in the benefits emanating from the profit maximization and misclassification costs. The Accuracy of these classical churn prediction models is determined by the areas under the ROC curve. Compared to this, the new classifier uses a metric called Expected Maximum Profit Measure for Customer Churn, to predict maximum profit. This model lends itself to good interpretation to figure out the cause of customer churn, owing to the usage of decision trees as an underlying classifier. This paper has used 9 real-life churn datasets, to measure the performance of the new model using cost, accuracy, and profit measures. The paper concludes that the new ProfTree technique surpasses other AUC accuracy-based models, in terms of high hit rate/recall rate in the prediction of churners. The paper also proposes, as a

future area of investigation, to develop a new model: ProfForest by combining ProfTree algorithm with the Random Forests with an expectation to further maximize the profit – the ProfForest will be an aggregation of ProfTrees.Gaps identified in this Study are : (i) No pre-processing steps are explained ; (ii) No metrics are tracked to measure the customer retension rate.

Eugene Wong and Yan Wei published a research paper on Customer online shopping experience data analytics in 2018. This research segments the 110840 customers of an online travel agency in keeping with their online purchasing behavior and predict their next purchases. The research approach entails the impact of competitor pricing on purchasing behavior of the company products. An integrated model of data mining of competitor pricing, customer segmentation, and prediction of future purchases helped the online travel agency tailor their offerings to keep with their respective customers' travel preferences. The Study segregates travel agencies ' high valued customers, RFM(Recency, Frequency, Monetary) and CLV (Customer Lifetime Value) methodologies are used for customer segmentation (Husnah and Novita, 2022). Regression methodology analyzes the impact of competitor ticket price changes on the customers' purchase decisions (Bimaruci et al., 2020). Over and above, using Association Rules analysis, a model is developed to predict the future purchases of the customers. Few gaps apparent in this Study are: No Pre-Processing steps are included, Customer Churn Rate is not tracked, and the Accuracy of the Prediction models is not reflected. This Research could be further extended by factoring-in demographic information and time-prediction of purchases.

Arno De Caignya et al. (2018) proposed, an innovative hybrid classification model for customer churn prediction in keeping with logistic regression and decision trees. A new e logit leaf model(LLM) model is recommended for customer churn prediction, which performs better than Decision trees and Logistic regression. The algorithm is executed in two parts: using decision rules are used to segment the customers and for every leaf of the tree, customer churn is predicted. The new LLM algorithm scores high on both: Performance and

Comprehensibility, vis-a-vis, the LR and the DT algorithms. The LLM model's predictive performance is measured by the area under the receiver operating characteristics curve (AUC). This work could be further extended in other settings like credit scoring or fraud detection where the same trade-off between predictive performance and comprehensibility is present.

Joy Christy et al. (2018) published their work on customer segmentation using RFM analysis. They have argued the usage of a new method of selecting the initial centroids for the K-means algorithm for clustering a UK online retailer's transactional data and comparing the accuracy of the new method with the classical k-means algorithm and fuzzy c-means algorithm. The cluster compactness, execution time, and the number of iterations of these algorithms are evaluated for comparison. The evaluation results reflect that lesser time, lesser number of iterations are required by the new approach and the centroids are also useful, as they are calculated based on RFM data distribution's effective medians. In keeping with RFM scores generated by the new clustering model, the company can customize its campaign strategies for its customers in the light of their buying behavior (Rahim et al., 2021). This work could be further extended to include market basket analysis of frequently bought products in each cluster. Eventually, this will help the company to provide better offerings to the customers for selected products.

Alireza Sheikha et al. (2019) proposed a B2B Customer Segmentation framework of Two-Stage Clustering for the Fintech industry. An innovative LRFMP model(Length, Recency, Frequency, Monetary, and Periodicity) model is developed for customer segmentation and behavior analysis. Dataset relates to the Iranian Fintech companies. The customer segmentation process combines the K-means clustering algorithm and LRFMP model. After initial clustering, additional clustering of segments is needed for a better understanding of valuable customers. In keeping with the characteristics of 23524 customers, Customer Clusters are analyzed. The first stage evaluation result underlines that customers are segmented into four groups. The first and fourth segments are segmented again, to

determine 11 segments of customers. This new clustering approach helps companies design effective marketing campaigns and improve customer relationships. The results underscore that the L and P variables can help companies tailor their marketing campaigns towards each cluster of customer segments. This framework helps improve the customers' clustering and enhances the efficiency of marketing campaigns for the B2B setting.

Jun et al. (2020) published a research paper on An Empirical Study on Customer Segmentation by Purchase Behaviors Using an RFM Model and K-Means Algorithm in 2020. This paper uses the K-means clustering algorithm and RFM model to segment the customers using a company's online sales data. Four Clusters are created to classify the customers in keeping with their purchase behaviors-RFM analysis determines the CRM strategies to provide a high degree of customer satisfaction monitored by KPIs like the growth of active customers, total purchase volume, and the total consumption amount. This research paper uses a dataset of a company in China from Nov 2017 to April 2019. This Research results in the segmentation of the customer base into four clusters, and in keeping with the RFM scores, appropriate CRM strategies are recommended for each group. These strategies' effectiveness led to an increased number of active customers by 529, total purchase volume by 279%, and the total consumption amount by 101.97%. Gaps identified in this Research are : (i) Necessary metrics for Customer retention rate is not tracked ; (ii) Accuracy of the Classifiers is not determined. This Research could be further extended by using more appropriate algorithms and embedding the algorithms in the CRM system for better decision making.

Maha Alkhayrat, Mohamad Aljnidi, and Kadan Aljoumaa (2020) published a research paper on A comparative dimensionality reduction study in telecom customer segmentation using deep learning and PCA. This Study explored the logs of telecom companies which entails customer behavior and actions. The key challenges in applying the analytics process to this data are (i) the Presence of a Large Number of features ; (ii) Sparsity in the Log Data. This Research aims to reduce the dimensions of the feature space using Principal Component Analysis(PCA) and Autoencoder Neural Network (ANN) machine learning algorithms,

apply the K-means algorithm to cluster the customers in the reduced space, and compare with the original space to achieve better quality clustering results. The original data set has 220 features for 100,000 customers. The silhouette analysis technique is used to compare the performance of Clusters created by the K-means algorithm on the original and the reduced data set. The best results are obtained by the autoencoder neural network approach which played a significant role in the dimensional reduction, and K-Mean Clustering Algorithm, where the clustering performance was enhanced with reduced dimensions. The original data set of 220 features is reduced in dimension to 20 features and the final results are very close or better than the original dataset clustering evaluation results. This Research could be further extended to measure metrics for customer retention rate viz; CLV, Churn Rate and predict future Sales of the customers.

2.2 Analysis of Key Literature

The above studies are succinctly summarized in the appended Table to underscore the comparison of approaches adopted in the research papers:

#	Studies	Dataset, Techniques	Purpose, Key Results, Future Study, Limitations
1	**Authors:** Chen, Sain, Guo **Title:** A case study of customer segmentation using data mining techniques of the RFM model **Year:** 2012 **Journal:** Database Marketing & Customer Strategy Management	· Dataset: 406 830 Transactions of a UK Online Retail Company for the period 1 January 2011 to 31 December 2011 and has 11 attributes. · Techniques: RFM analysis+ K-Means Algorithm + Decision Tree Algorithm	· Purpose: Aim to develop an algorithm for generating all RFM patterns from customers' purchasing data and generate valuable information on customer purchasing behavior for managerial decision-making. · Key Results: This model identifies the most and the least profitable group of customers. This insight will help the online retailer determine with which group a new customer will affiliate.

			Future Study: Explore the customer's buying experience to predict the Customer Lifecycle Value**.**
2	Authors: Zhen, Yain, Defu, Xiang, Stephen, Tao Title: A decision-making framework for precision marketing Year: 2014 Journal: ELSEVIER	**DataSet:** Transactions collected from a Company in China The data include six products: Smith, Howard, David, Edward, Yun, and Edward1. The production information such as monthly supply quantity, ordered quantity, ordered times, D-value, Lever of customers, commercial activities, scale of operations, commercial environment, and marketing type are known. Techniques: RFM analysis+ CHAID Decision Trees Algorithm + Pareto Analysis	**Purpose:** Develop a trend model to predict the material supply and segregate the customers to target the clusters with custom supply strategies. **Key Results:** The resulting trend model is better than the SARIMA / SVM algorithms. **Future Study:** Explore the Development of an integrated model using all the models used in this study. **Limitations:** The model is very simple and scalability is subservient to the Dataset size.

3	**Authors:** Chih-Fong, Ya-Han, and Yu-Hsin Title: Customer segmentation issues and strategies for an automobile dealership with two clustering techniques Year: 2015 Journal: Expert Systems	**Dataset:** Cars sold by an automobile dealer in Taiwan in the last 3 years. Techniques: PCA Algorithm + K-Means algorithm + Expectation-Maximization(EM) Algorithm	**Purpose:** Develop models using K-Means and Expectation-Maximization methods and compare the evaluation results. Key Results: The evaluation results underscore four customer segments: loyal, potential, VIP, and churn customer. To improve the quality of service customized marketing campaigns are designed in keeping with the resulting customer clusters. Future Study: Validating the results of this work with different automobile dealers and by tracking the effectiveness of the marketing strategies in the case company.
4	Authors: Xixi He, Chen Li Title: Application of Customer Segmentation on E-commerce Websites Year: 2016 Journal: IEEE	**Dataset:**45064 registered customers' purchasing data from daily necessities Chinese e-commerce website from August 1, 2014, to September 30, 2015, in total 14 months. **Techniques:** RFM Analysis + KANO Analysis + BG/NBD Analysis	**Purpose:** Develop a three-dimensional customer segmentation model based on customer lifetime value, customer satisfaction, and customer activity, which more accurately divides customers into different groups. Key Results: The customer segmentation model provides ten groups of customers with corresponding marketing strategies so that it can help enterprises maximize profits.

5	Authors: Wen-Yu Chiang Title: Discovering customer value for marketing systems Year:2017 Journal: International Journal of Production Research	**Dataset:** Departure and Arrival data of Airline Travellers in Taiwan for the period 2005 to 2015**.** **Techniques:** FSLC model + RFM model + Apriori Algorithm	**Purpose:** Develop a model (FSLC model, RFM model-based) via the data mining technologies to discover valuable travelers for airlines. **Key Results:** There are four markets found by the shopping benefit variables of air travelers in Taiwan. The proposed FSLC model and socio-economic variables are oriented into the markets via the supervised Apriori algorithm. Finally, there are five association rules created by this framework. Limitations: Questionnaires for customer benefits are not easy to be collected. Future Study: Researchers can focus future studies on the industry of the LCC or other businesses. In other business, the questionnaire items and FSLC model (RFM model-based) should be redesigned
6	Authors:ShohreHaghighatnia, Neda Abdolvand & Saeedeh RajaeeHarandi Title:Evaluating discounts as a dimension of customer behavior analysis	**DataSet:** Data related to an online retailer's six-month sales records in IRAN. The data of 130097 records were stored in databases, of which 64,038 records were related to the customers	**Purpose:** Develop a model based on RFM called RdFdMd, in which d is the level of discount used to analyze customer purchase behavior and the importance of discounts on customers' purchasing behavior and organizational profitability. **Key Results:** The results indicate that using the RdFdMd model achieves better customer clustering and

	Year:2017 Journal: Journal of Marketing Communications	who had received discounts. **Techniques:** RdFdMd analysis based on RFM, wherein d is the level of discount + K-Means Algorithm	valuation, and discounts were identified as an important criterion for customer purchases. **Future Study:** Explore the interval between discounts. Limitations: The relationship between the product and discounts is not considered.
7	Authors:Makoto Morisada, Yukihiro Miwa, Wirawan Dony Dahana Title: Identifying valuable customer segments in online fashion markets: An implication for customer tier programs Year: 2018. Journal: ELSEVIER	**Dataset:** An online shopping mall company provided the data, which encompass the purchase history records of customers over a one-year period. In total, 1,009,609 purchases were made by 101,480 customers during this period **Techniques:** RFM Analysis + K-Means algorithm + CLV Analysis	**Purpose:** Develop a latent class model to classify customers into several segments and rank those segments by their past and future revenue. Key Results: The results revealed that fashion innovativeness, brand consciousness, mobile device usage, and shop loyalty increase the probability of a customer becoming valuable. Limitations: The Study could be further extended for other fashion e-retailer to generalise the research and explore how customer profitability can be explained by lifestyle, perhaps by combining purchase history and survey data, which are currently sparse.
8	Authors:Arash, Mohammad, Mohammadreza Title: A novel model for product bundling and direct marketing in e-	**DataSet:** The data for this research were collected from the electronic transactions of the company with its customers. These	**Purpose:** Develop a novel model for product bundling and direct marketing in e-ecommerce based on market segmentation – RFM model is used to cluster customers into loyalty levels.Using K-means algorithm the product bundles were then determined.

	commerce based on market segmentation Year:2018 Journal:Growing Science	data include 541910 records received from 4340 customers from December 2014 to December 2015 in the cities of Golestan province in Iran. **Techniques:** RFM analysis + K-Means Algorithm +Silhouette Analysis + Apriori algorithm + SVM Algorithm	The association rules in each product bundle is determined by Apriori algorithm. Finally, SVM classification model is applied to recommend the product bundles to the customers. Key Results: By applying the proposed model, the product bundles are more precisely determined. Future Study: Explore development of a real-time system for recommending product bundles to customers, recommender systems can be implemented by using market segmentation and customer loyalty analysis.
9	Authors:Sebastiaan , Eugen, Bart, Seppe et. al Title:Profit driven decision trees for churn prediction Year: 2018 Journal: ELSEVIER	**DataSet:** The dataset of South Korean telecom operator contains a sample of 889 customers and 10 explanatory variables; 277 of these customers (i.e. 31.16%) were recorded as churners. **Techniques:** ProfTree Algorithm	**Purpose:** Develop a new churn classification method called ProfTree that uses an evolutionary algorithm to directly optimize the EMPC (Verbraken et al., 2013) in the model construction step of a decision tree. As a result, ProfTree aims to actively construct the most profitable model for a customer retention campaign. **Key Results:** The study consisted of applying the classifiers to 9 real-life churn datasets, and evaluated their out-of-sample classification performance using accuracy, cost, and profit related performance measures. In almost all cases, ProfTree outperforms its competitors, leading to significantly higher profits; whereas, in the worst

			case, its profit losses are relatively small compared to the respective best competitive model. Future Study: Explore development of a new model : ProfForest by combining ProfTree algorithm with the Random Forests with an expectation to further maximize the profit – the ProfForest will be aggregation of ProfTrees.
10	**Authors:**Eugene Wong and Yan Wei **Title:**Customer online shopping experience data analytics **Year:**2018 **Journal**: International Journal of Retail & Distribution Management	**DataSet:** Consists of 110840 customers of an online travel agency in Hong Kong **Techniques:** RFM Analysis + CLV Analysis + Regression Algorithm + Apriori Algorithm	**Purpose:** Develop a customer online behaviour analysis tool, segment high-value customers, analyse their online purchasing behaviour and predict their next purchases from an online air travel corporation **Key Results:** The relationship between the purchasing behaviour in an OTA and the price changes of different OTAs are analysed. There is a significant relationship between the flight duration time and the purchase lead time. The next travel destinations of segmented high-value customers are predicted with reference to their travel patterns and the significance of the relationships between destination pairs. **Future Study:** Explore factoring-in demographic information and time-prediction of purchases.
11	Authors:Arno De Caignya,Kristof	**DataSet:** xperiments are conducted on fourteen churn	**Purpose:** Develop a new hybrid algorithm, the logit leaf model (LLM), which is proposed to better classify

	Coussement , Koen W. De Bock Title:A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees Year: 2018. Journal: European Journal of Operational Research	datasets originating from European financial services provider **Techniques:** Decision tree + Logistic regression + Random forests + Logistic model tree + Logit leaf model	data, that enhances LR and DT Algorithms. **Key Results:** the LLM (1) provides more accurate models than using its building blocks, LR and DT, as standalone classification techniques. It performs at least evenly well as two homogenous ensemble methods RF and LMT, which are among the best performing classification techniques; (2) The LLM delivers a comprehensible method with benefits regarding the actionability of the model, which is its main advantage compared to the LMT and RF; (3) LLM can enrich both DT and LR . Future Study: Explore other settings where the same trade-off between predictive performance and comprehensibility is present such as credit scoring or fraud detection, where businesses want to predict suitable applicants.
12	Authors:Joy Christy, A. Umamakeswari, L. Priyatharsini, Neyaa	**DataSet:** The transactional data set of the customers of an online retail store for one year is obtained from the	**Purpose:** A novel idea for choosing the initial centroids in K- Means is proposed. **Key Results:** Segmentation is done using RFM analysis and then is extended to other algorithms like K –

	Title: RFM ranking – An effective approach to customer segmentation Year: 2018 Journal: Journal of King Saud University	University of California Irwin (UCI) repository. **Techniques:** RFM Analysis + K Means Algorithm + Fuzzy C-Means Algorithm + RM K-Means Algorithm	Means clustering, Fuzzy C – Means and a new algorithm RM K-Means by making a minor modification in the existing K – Means clustering. The working of these approaches is analyzed. The time taken by each algorithm to execute is analyzed, and it is observed that the proposed K – Means approach consumes lesser time and also reduces the number of iterations. The proposed algorithm is more effective because the centroids are more meaningful and are calculated at the beginning based on the effective medians of data distribution Future Study: Exploration of performance of the customers in each segment such as the products which are bought frequently by the members of each segment. This would help better in providing better promotional offers to specific
13	Authors: Alireza Sheikha, Tohid Ghanbarpourb, and Davoud Gholamiangonabadib Title: A Preliminary Study of Fintech Industry: A Two-Stage Clustering Analysis	**DataSet:** Includes the information on the value, the number, and the date of the first and last transactions for 23524 customers during a 45- month period ending on 13 June 2018, from the operational database	**Purpose:** Develop a two-stage clustering and LRFMP model(Length, Recency, Frequency, Monetary, and Periodicity) simultaneously for customer segmentation and behavior analysis. **Key Results:** The results indicate that the L and P variables can significantly help with the interpretation of and tailoring marketing strategies towards each cluster of customer groups.

	for Customer Segmentation in the B2B Setting Year: 2019 Journal: Journal of Business-to-Business Marketing	of one of the online payment processor companies in the Iranian Fintech industry. **Techniques:** RFM Analysis + K-Means Algorithm + Davies–bouldin index Analysis	Limitations: The data should be drawn from the customers who were active in the last 12-month period and one product only. Future Study: Exploration of building upon a more comprehensive body of data with more products and time frames extended.
14	Authors: Maha Alkhayrat, Mohamad Aljnidi, and Kadan Aljoumaa Title: A comparative dimensionality reduction study in telecom customer segmentation using deep learning and PCA Year:2020 Journal: Journal of Big Data	**Dataset:** The dataset contains 220 features that belonging to 100,000 customers. **Techniques:** Principal Component Analysis(PCA) Algorithm + Autoencoder Neural Network(ANN) Algorithm + K-Means Clustering Algorithm + Silhouette analysis + Davies–Bouldin index Analysis	**Purpose:** Aims to explore dimensionality reduction on a real telecom dataset and evaluate customers' clustering in reduced and latent space, compared to original space to achieve better quality clustering results. **Key Results:** ANN approach played a significant role in the dimensional reduction, and K-Mean Clustering Algorithm, where the clustering performance was enhanced with reduced dimensions. For the original data set of 220 features, we were able to reduce its dimension to 20 features and obtain results very close or better than the original dataset clustering evaluation results.

15	Authors: Jun, Wen, Sang, Yuanyuan, Liping, and Guangshu Title: An Empirical Study on Customer Segmentation by Purchase Behaviors Using an RFM Model and K-Means Algorithm Year: 2020. Journal: ELSEVIER	**Dataset:** Dataset consists of 10,248 purchase data entries created at a community shopping platform from November 1, 2017, to April 15, 2019, involving 1,013 customers. /is platform sells 134 types of commodities, mostly including cooked food and pasta **Techniques:** RFM analysis + K-Means Algorithm	**Purpose:** Customer purchase behaviors are analyzed systematically based on the online transaction data of a company by using RFM and K-means clustering algorithm **Key Results:** Customers are classified into four groups based on their purchase behaviors. On this basis, different CRM strategies are brought forward to gain a high level of customer satisfaction. The improvement results of some key performance indices are given in brief as follows. The number of active customers has grown by 529. The total purchase volume has increased by 279%, and the total consumption amount has increased by 101.97%. Future Study: Explore embedding algorithm into the CRM system to support managers' decision making is a good way to help for performance improvement

2.4 Summary

The review of research work in customer segmentation underscores the significance of the RFM analysis and the K-Means Clustering unsupervised machine learning algorithm. In most of the studies, either these vanilla methodologies are utilized or their variants are productively deployed to improve the performance of the customer clustering. Silhouette Analysis is used to measure the cohesiveness of the clusters and the Elbow method is used to predict the initial value of K in the K-means algorithm.PCA & ANN Algorithms are used to reduce the dimensions of the large datasets, before applying the K-means algorithm. To track customer retention, metrics like CLV and Churn Rate are used as well. All these methodologies help the companies to design their campaign strategies for customers segmented by the Clusters and increase their profits.

CHAPTER III:

METHODOLOGY

3.1 Description of Dataset

This analysis uses a dataset from Online Retail platforms, mostly from UK, and contains actual transactions between Dec 1, 2010, and Dec 9, 2011.These transactions are collected from registered non-store online retail platforms selling unique all-occasion gifts and many customers of these platforms are wholesalers.

This dataset has 532610 records and is downloaded from the UCI link: http://archive.ics.uci.edu/ml/datasets/online+retail.

Description of the Dataset Attributes are as follows :

- Invoice No: This data attribute indicates the invoice numbers. It is a six-digit integer number. The records are uniquely assigned for each transaction. If the invoice number starts with the letter ‘c’, then it indicates a cancellation.
- StockCode: This data attribute indicates the product (item) code. It is a five-digit integer number. All the item codes are uniquely assigned to each distinct product.
- Description: This data attribute contains the description of the item.
- Quantity: This data attribute contains the quantities for each product per transaction. The data is in a numeric format.
- InvoiceDate: The data attribute contains the invoice date and time. It indicates the day and time when each transaction was generated.
- UnitPrice: The price indicates the product price per unit in sterling.
- CustomerID: This column has the customer identification number. It is a five-digit integer number uniquely assigned to each customer.

- Country: This column contains the geographic information about the customer. It records the country name of the customers.

To begin with, a baseline approach is built to implement a basic model for customer segmentation application. Eventually, this baseline approach will be improved.
To implement the customer segmentation model, the implementation methodology will have the following steps:

- Data preparation
- Exploratory data analysis (EDA)
- Generating customer categories
- Classifying customers

3.2 Data Preparation

This is a basic step to build any analytics application. First, it's important to validate that the format of the data is in an appropriate form. If it is not, then there's a need to prepare the dataset in such a way that the application can be easily built. In this step, it's important to figure out whether the dataset is of good quality. Some basic facts about the dataset could also be established. Luckily, the format of the e-commerce dataset doesn't need any change, but further exploration of the dataset is needed in such a way that the quality of the dataset could be ascertained. If the format of the dataset is not proper then there's a need to decide the format of the dataset in such a way that any kind of analysis can be performed using the dataset. Data records can be converted to either CSV or JSON or XML format. Besides, general facts about the dataset can be derived, such as whether the dataset is biased or not, whether the dataset contains any null values, the mapping of the customers with Customer_ID is proper or not, whether their purchases are properly recorded in the dataset or not, and so on. To Load the dataset, the pandas read_csv API is used (refer Figure 1)

** Data Preparation **

```
In [2]: # read the datafile
df_initial = pd.read_csv('D:\\Personal\\ML\\LJMUProject\\Mid Thessis Report\\Code\\Customer_segmentation-master\\input_data\\data
                         dtype={'CustomerID': str,'InvoiceID': str})
print('Dataframe dimensions:', df_initial.shape)
df_initial['InvoiceDate'] = pd.to_datetime(df_initial['InvoiceDate'])
```

```
Dataframe dimensions: (541909, 8)
```

```
In [3]: # show first lines
display(df_initial[:5])
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850	United Kingdom

Figure 1 Load the dataset, the pandas read_csv API

The dimensions of the dataset are (541909,8). This means that there are 541,909 records in the dataset and eight data attributes, explained in the previous section.

3.3 Exploratory Data Analysis (EDA)

In EDA, validation of the statistical properties of the dataset is made, and the following preprocessing steps are performed:

- Removing null data entries
- Removing duplicate data entries
- EDA for various data attributes

3.3.1 Removing null data entries

First, there's a need to validate the data type of each of the attributes as well as find out which column has a null value. (refer Figure 2)

** Exploratory Data Analysis ** ¶

Identify null values

In [4]:
```
# gives some infomation on columns types and number of null values
tab_info=pd.DataFrame(df_initial.dtypes).T.rename(index={0:'column type'})
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()).T.rename(index={0:'null values (nb)'}))
tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()/df_initial.shape[0]*100).T.
                         rename(index={0:'null values (%)'}))
print ('-' * 10 + " Display information about column types and number of null values " + '-' * 10 )
print
display(tab_info)
```

```
---------- Display information about column types and number of null values ----------
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
column type	object	object	object	int64	datetime64[ns]	float64	object	object
null values (nb)	0	0	1454	0	0	0	135080	0
null values (%)	0	0	0.268311	0	0	0	24.9267	0

Figure 2 Validate the attributes and remove the null values

The above code generates the total number of null values for each data attribute. Also, the code generates the percentage of null values for each data attribute. Apparently, for the CustomerID column, there are ~25% of data entries that are null. That means that there is no CustomerID value available for ~25% of the dataset. This indicates that many entries do not belong to any customer. They are abended data entries and cannot be mapped to the existing CustomerIDs. In the sequel to this, these data entries have to be deleted. For code snippet on deleting null data entries from the dataset (refer Figure 3):

```
In [5]: df_initial.dropna(axis = 0, subset = ['CustomerID'], inplace = True)
        print('Dataframe dimensions:', df_initial.shape)
        # gives some information on columns types and number of null values
        tab_info=pd.DataFrame(df_initial.dtypes).T.rename(index={0:'column type'})
        tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()).T.rename(index={0:'null values (nb)'}))
        tab_info=tab_info.append(pd.DataFrame(df_initial.isnull().sum()/df_initial.shape[0]*100).T.
                                 rename(index={0:'null values (%)'}))
        display(tab_info)
```

```
Dataframe dimensions: (406829, 8)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
column type	object	object	object	int64	datetime64[ns]	float64	object	object
null values (nb)	0	0	0	0	0	0	0	0
null values (%)	0	0	0	0	0	0	0	0

Figure 3 Deleting null entries

3.3.2 Removing duplicate data entries

The presence of duplicate data entries in the dataset is validated and the pandas duplicate() function will be used to verify this (refer Figure 4). There are 5,225 duplicate data entries in the dataset. Therefore, we have removed them.

```
In [6]: print('Duplicate data entries: {}'.format(df_initial.duplicated().sum()))
        df_initial.drop_duplicates(inplace = True)
```

```
Duplicate data entries: 5225
```

Figure 4 Validating null entries

3.3.3 EDA for various data attributes

EDA for each data attribute will provide more insight into the dataset. Later on, these facts will be used to build an accurate customer segmentation application.

Exploration of the data attributes will be done in the following order :

- Country
- Customer and Products
- Product Categories
- Defining product categories

3.3.4 Analysis of the Country

There's a need to find out facts such as how many countries there are in our dataset. To answer this question, the following code is executed (refer Figure 5):

Figure 5 Exploring data attribute: country

Further, there's also a need to find the country from which we receive the maximum number of orders, which could be retrieved by using the pandas groupby() and count() functions and sort the number of orders in descending order (refer Figure 6).

```
In [8]: temp_no_of_order_per_count = df_initial[['CustomerID','Country']].groupby(['Country']).count()
        temp_no_of_order_per_count = temp_no_of_order_per_count.reset_index(drop = False)

        print('-' * 10 + " Contry-wise order calculation "+ '-' * 10)
        print
        print (temp_no_of_order_per_count.sort_values(
            by='CustomerID', ascending=False).rename(index=str,
                                                     columns={"CustomerID": "Country wise number of order"}))
```

```
---------- Contry-wise order calculation ----------
                Country  Country wise number of order
35       United Kingdom                        356728
14              Germany                          9480
13               France                          8475
10                 EIRE                          7475
30                Spain                          2528
23          Netherlands                          2371
3               Belgium                          2069
32          Switzerland                          1877
26             Portugal                          1471
0              Australia                          1258
24                Norway                          1086
18                 Italy                           803
6       Channel Islands                            757
12              Finland                            695
7                Cyprus                            611
31               Sweden                            461
1               Austria                            401
```

Figure 6 Exploring data attribute: country for the orders received

Preceding code snippet underscores that there is a majority of orders from UK-based customers. Next, I need to explore the customer and product variables.

3.3.5 Analysis of the Customer and products

There are approximately 400,000 data items and there's a need to find the count of users & products that are present in these records. The value_counts() function will be used from the pandas library (refer Figure 7).

** Exploring the data attribute : Customers and products **

```
In [9]: pd.DataFrame([{'products': len(df_initial['StockCode'].value_counts()),
                       'transactions': len(df_initial['InvoiceNo'].value_counts()),
                       'customers': len(df_initial['CustomerID'].value_counts()),
                      }], columns = ['products', 'transactions', 'customers'],
                      index = ['quantity'])
```

Out[9]:

	products	transactions	customers
quantity	3684	22190	4372

Figure 7 Exploring data attribute: customers and products

This dataset contains the records of 4372 customers who purchased 3684 different products and the total number of records is 22,190. Closely related to this is the question of how many products have been purchased for each transaction. To answer this question, the

InvoiceNo and InvoiceDate data attributes are used to compute the number of products purchased for every transaction (refer Figure 8).

```
In [10]: temp = df_initial.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['InvoiceDate'].count()
         nb_products_per_basket = temp.rename(columns = {'InvoiceDate':'Number of products'})
         nb_products_per_basket[:10].sort_values('CustomerID')
```

Out[10]:

	CustomerID	InvoiceNo	Number of products
0	12346	541431	1
1	12346	C541433	1
2	12347	537626	31
3	12347	542237	29
4	12347	549222	24
5	12347	556201	18
6	12347	562032	22
7	12347	573511	47
8	12347	581180	11
9	12348	539318	17

Figure 8 Exploring data attribute: invoice number and invoice date

Some users have made a purchase only once on the e-commerce platform and bought one item. An example of this kind of user is CustomerID 12346. Some users frequently buy a large number of items per order. An example of this kind of user is CustomerID 12347. In the InvoiceNo data attribute, there is a prefix ‘C’ for one invoice, which indicates that the particular transaction has been canceled.

In keeping with the 3rd observation, the count of the number of records pertinent to the canceled orders is made using a lambda expression and compute the percentage of canceled orders (refer Figure 9).

** Analysis of cancelled orders **

```
In [11]: nb_products_per_basket['order_cancelled'] = nb_products_per_basket['InvoiceNo'].apply(
             lambda x:int('C' in x))
         display(nb_products_per_basket[:5])


         n1 = nb_products_per_basket['order_cancelled'].sum()
         n2 = nb_products_per_basket.shape[0]
         percentage = (n1/n2)*100
         print('Number of orders cancelled: {}/{} ({:.2f}%) '.format(n1, n2, percentage))
```

	CustomerID	InvoiceNo	Number of products	order_cancelled
0	12346	541431	1	0
1	12346	C541433	1	1
2	12347	537626	31	0
3	12347	542237	29	0
4	12347	549222	24	0

```
Number of orders cancelled: 3654/22190 (16.47%)
```

Figure 9 Analysis of cancelled orders

The following list of some of the canceled order entries helps to find out how to handle them (refer Figure 10).

```
In [12]: display(df_initial.sort_values('CustomerID')[:5])
```

Out[ ]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
61619	541431	23166	MEDIUM CERAMIC TOP STORAGE JAR	74215	2011-01-18 10:01:00	1.04	12346	United Kingdom
61624	C541433	23166	MEDIUM CERAMIC TOP STORAGE JAR	-74215	2011-01-18 10:17:00	1.04	12346	United Kingdom
286623	562032	22375	AIRLINE BAG VINTAGE JET SET BROWN	4	2011-08-02 08:48:00	4.25	12347	Iceland
72260	542237	84991	60 TEATIME FAIRY CAKE CASES	24	2011-01-26 14:30:00	0.55	12347	Iceland
14943	537626	22772	PINK DRAWER KNOB ACRYLIC EDWARDIAN	12	2010-12-07 14:57:00	1.25	12347	Iceland

Figure 10 Analysis of cancelled orders

If the order is canceled, then there is another transaction that will mostly have an identical transaction except for the quantity and invoice date. First, validation of all the entries in the dataset should be made. The checking operation can be done using a simple logic – the canceled order has a negative quantity, so validation should be made on whether there is an order entry with the same description and the same quantity. There are some discount entries as well, which are to be handled by discarding the discount entries (refer Figure 11).

```
In [13]: df_check = df_initial[df_initial['Quantity'] < 0][['CustomerID','Quantity',
                                                    'StockCode','Description','UnitPrice']]
         for index, col in  df_check.iterrows():
             if df_initial[(df_initial['CustomerID'] == col[0]) & (df_initial['Quantity'] == -col[1])
                         & (df_initial['Description'] == col[2])].shape[0] == 0:
                 print(df_check.loc[index])
                 print(15*'-'+'>'+' HYPOTHESIS NOT FULFILLED')
                 break

CustomerID        14527
Quantity             -1
StockCode             D
Description    Discount
UnitPrice          27.5
Name: 141, dtype: object
---------------> HYPOTHESIS NOT FULFILLED
```

Figure 11 Analysis of discount entries

The presence of a '*Discount*' entry leads to the failure of the initial hypothesis – Therefore, the hypothesis is checked again, by dropping the '*Discount*' entries (refer Figure 12).

```
In [14]: df_check = df_initial[(df_initial['Quantity'] < 0) & (df_initial['Description'] != 'Discount')][
                                ['CustomerID','Quantity','StockCode',
                                 'Description','UnitPrice']]

         for index, col in  df_check.iterrows():
             if df_initial[(df_initial['CustomerID'] == col[0]) & (df_initial['Quantity'] == -col[1])
                         & (df_initial['Description'] == col[2])].shape[0] == 0:
                 print(index, df_check.loc[index])
                 print(15*'-'+'>'+' HYPOTHESIS NOT FULFILLED')
                 break

154 CustomerID                          15311
Quantity                               -1
StockCode                           35004C
Description    SET OF 3 COLOURED  FLYING DUCKS
UnitPrice                             4.65
Name: 154, dtype: object
---------------> HYPOTHESIS NOT FULFILLED
```

Figure 12 Dropping discount entries

The above results underscore that the initial hypothesis is not verified. Therefore, cancellations may not have orders made beforehand. The preceding code confirms that there are no similar entries present in our dataset for all canceled transactions. To circumvent this situation, a new entity is created in the dataframe, which indicates whether the transaction is canceled or not. There are three possibilities for canceled orders :

Some transactions were canceled without corresponding orders, probably because the orders were made before December 2010. The available dataset is from December 2010 to December 2011.

Some orders were canceled with exactly one counterpart and there's a need to consider them as well. Some entries are doubtful and validation should be made to check whether there is at least one counterpart with the same quantity available. If available, then those entries are marked as doubtful.

```
In [15]: df_cleaned = df_initial.copy(deep = True)
         df_cleaned['QuantityCanceled'] = 0

         entry_to_remove = [] ; doubtfull_entry = []

         for index, col in  df_initial.iterrows():
             if (col['Quantity'] > 0) or col['Description'] == 'Discount': continue
             df_test = df_initial[(df_initial['CustomerID'] == col['CustomerID']) &
                                  (df_initial['StockCode']  == col['StockCode']) &
                                  (df_initial['InvoiceDate'] < col['InvoiceDate']) &
                                  (df_initial['Quantity']   > 0)].copy()

             # Cancelation WITHOUT counterpart
             if (df_test.shape[0] == 0):
                 doubtfull_entry.append(index)

             # Cancelation WITH a counterpart
             elif (df_test.shape[0] == 1):
                 index_order = df_test.index[0]
                 df_cleaned.loc[index_order, 'QuantityCanceled'] = -col['Quantity']
                 entry_to_remove.append(index)

             # Various counterparts exist in orders: we delete the last one
             elif (df_test.shape[0] > 1):
                 df_test.sort_index(axis=0 ,ascending=False, inplace = True)
                 for ind, val in df_test.iterrows():
                     if val['Quantity'] < -col['Quantity']: continue
                     df_cleaned.loc[ind, 'QuantityCanceled'] = -col['Quantity']
                     entry_to_remove.append(index)
                     break
```

Figure 13 Analysis of cancelled entries (1)

```
In [16]: print("entry_to_remove: {}".format(len(entry_to_remove)))
         print("doubtfull_entry: {}".format(len(doubtfull_entry)))

         entry_to_remove: 7521
         doubtfull_entry: 1226
```

Figure 14 Analysis of cancelled entries (2)

The preceding figure underscores that 7,521 entries show the canceled orders with their counterpart. 1,226 entries show canceled orders without their counterpart. For the sake of simplicity, all the entries related to the canceled orders are deleted. Code snippet for deleting these entries is appended herewith (refer Figure 15) :

```
In [17]: df_cleaned.drop(entry_to_remove, axis = 0, inplace = True)
         df_cleaned.drop(doubtfull_entry, axis = 0, inplace = True)
         remaining_entries = df_cleaned[(df_cleaned['Quantity'] < 0) & (df_cleaned['StockCode'] != 'D')]
         print("nb of entries to delete: {}".format(remaining_entries.shape[0]))
         remaining_entries[:5]

         nb of entries to delete: 48
```

Out[17]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	QuantityCanceled
77598	C542742	84535B	FAIRY CAKES NOTEBOOK A6 SIZE	-94	2011-01-31 16:26:00	0.65	15358	United Kingdom	0
90444	C544038	22784	LANTERN CREAM GAZEBO	-4	2011-02-15 11:32:00	4.95	14659	United Kingdom	0
111968	C545852	22464	HANGING METAL HEART LANTERN	-5	2011-03-07 13:49:00	1.65	14048	United Kingdom	0
116064	C546191	47566B	TEA TIME PARTY BUNTING	-35	2011-03-10 10:57:00	0.70	16422	United Kingdom	0
132642	C547675	22263	FELT EGG COSY LADYBIRD	-49	2011-03-24 14:07:00	0.66	17754	United Kingdom	0

Figure 15 Deletion of cancelled entries

3.3.6 Analysis of the Stock Code

Now analysis of the entries based on the stock code is required because, during the identification of the canceled order, discount items based on the stock code D are discovered. So first of all, stock codes and their meaning are listed (refer Figure 16).

```
In [19]: list_special_codes = df_cleaned[df_cleaned['StockCode'].str.contains('^[a-zA-Z]+', regex=True)]['StockCode'].unique()
         list_special_codes

Out[19]: array(['POST', 'D', 'C2', 'M', 'BANK CHARGES', 'PADS', 'DOT'],
               dtype=object)

In [20]: for code in list_special_codes:
             print("{:<15} -> {:<30}".format(code, df_cleaned[df_cleaned['StockCode'] == code]['Description'].unique()[0]))

         POST            -> POSTAGE
         D               -> Discount
         C2              -> CARRIAGE
         M               -> Manual
         BANK CHARGES    -> Bank Charges
         PADS            -> PADS TO MATCH ALL CUSHIONS
         DOT             -> DOTCOM POSTAGE
```

Figure 16 Analysis of stock codes

Let's focus on the pricing of the individual order. In the given dataset, the order from a single customer has been split into several lines. Each entry in our dataset indicates the price for a single kind of product. If the order including different products is placed by a

single customer, then there are multiple entries for that particular order. The number of data entries depends on how many different products that order has. I need to obtain the total price for each order. To achieve that, I will add a column named TotalPrice, which gives us the total value of the order or the basket price for a single order. The main logic for deriving TotalPrice is that we are multiplying UnitPrice with the net quantity. We obtain the net quantity by deducting the canceled quantity from the total quantity (refer Figure 17).

```
In [21]: df_cleaned['TotalPrice'] = df_cleaned['UnitPrice'] * (df_cleaned['Quantity'] - df_cleaned['QuantityCanceled'])
         df_cleaned.sort_values('CustomerID')[:5]
```

Out[21]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	QuantityCanceled	TotalPrice
61619	541431	23166	MEDIUM CERAMIC TOP STORAGE JAR	74215	2011-01-18 10:01:00	1.04	12346	United Kingdom	74215	0.0
148288	549222	22375	AIRLINE BAG VINTAGE JET SET BROWN	4	2011-04-07 10:43:00	4.25	12347	Iceland	0	17.0
428971	573511	22698	PINK REGENCY TEACUP AND SAUCER	12	2011-10-31 12:25:00	2.95	12347	Iceland	0	35.4
428970	573511	47559B	TEA TIME OVEN GLOVE	10	2011-10-31 12:25:00	1.25	12347	Iceland	0	12.5
428969	573511	47567B	TEA TIME KITCHEN APRON	6	2011-10-31 12:25:00	5.95	12347	Iceland	0	35.7

Figure 17 Derivation of the column: TotalPrice

Once the total price is obtained, the sum for individual orders is generated, and then the entries based on the invoice data are grouped. Only those data entries that have a basket price greater than 0 are listed (refer Figure 18).

In [22]:

```
# sum of purchases / user & order
temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['TotalPrice'].sum()
basket_price = temp.rename(columns = {'TotalPrice':'Basket Price'})

# date of the order
df_cleaned['InvoiceDate_int'] = df_cleaned['InvoiceDate'].astype('int64')
temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['InvoiceDate_int'].mean()
df_cleaned.drop('InvoiceDate_int', axis = 1, inplace = True)
basket_price.loc[:, 'InvoiceDate'] = pd.to_datetime(temp['InvoiceDate_int'])

# selection of significant entries
basket_price = basket_price[basket_price['Basket Price'] > 0]
basket_price.sort_values('CustomerID')[:6]
```

Out[22]:

	CustomerID	InvoiceNo	Basket Price	InvoiceDate
1	12347	537626	711.79	2010-12-07 14:57:00.000001024
2	12347	542237	475.39	2011-01-26 14:29:59.999999744
3	12347	549222	636.25	2011-04-07 10:42:59.999999232
4	12347	556201	382.52	2011-06-09 13:01:00.000000256
5	12347	562032	584.91	2011-08-02 08:48:00.000000000
6	12347	573511	1294.32	2011-10-31 12:25:00.000001280

Figure 18 Derivation of the Basket Price

In the sequel to this, an idea emanates about the distribution of the orders' amounts for the given dataset. In keeping with the prices for all the orders present in the dataset, the ranges are determined. This helps to derive the number of orders in the dataset that are above 200 GBP. This also helps to identify the number of orders that are below 100 GBP. This kind of information helps to derive the data distribution based on the number of orders. This helps to visualize a basic picture of the sales on the e-commerce platform. The code snippet for generating the data distribution based on the orders' amounts is displayed in the following (refer Figure 19):

```
In [23]: # Purchase count
price_range = [0, 50, 100, 200, 500, 1000, 5000, 50000]
count_price = []
for i, price in enumerate(price_range):
    if i == 0: continue
    val = basket_price[(basket_price['Basket Price'] < price) &
                       (basket_price['Basket Price'] > price_range[i-1])]['Basket Price'].count()
    count_price.append(val)

# Representation of the number of purchases / amount
plt.rc('font', weight='bold')
f, ax = plt.subplots(figsize=(11, 6))
colors = ['yellowgreen', 'gold', 'wheat', 'c', 'violet', 'royalblue','firebrick']
labels = [ '{}<.<{}'.format(price_range[i-1], s) for i,s in enumerate(price_range) if i != 0]
sizes  = count_price
explode = [0.0 if sizes[i] < 100 else 0.0 for i in range(len(sizes))]
ax.pie(sizes, explode = explode, labels=labels, colors = colors,
       autopct = lambda x:'{:1.0f}%'.format(x) if x > 1 else '',
       shadow = False, startangle=0)
ax.axis('equal')
f.text(0.5, 1.01, "Distribution of order amounts", ha='center', fontsize = 18);
```

Figure 19 Distribution of order amounts (1)

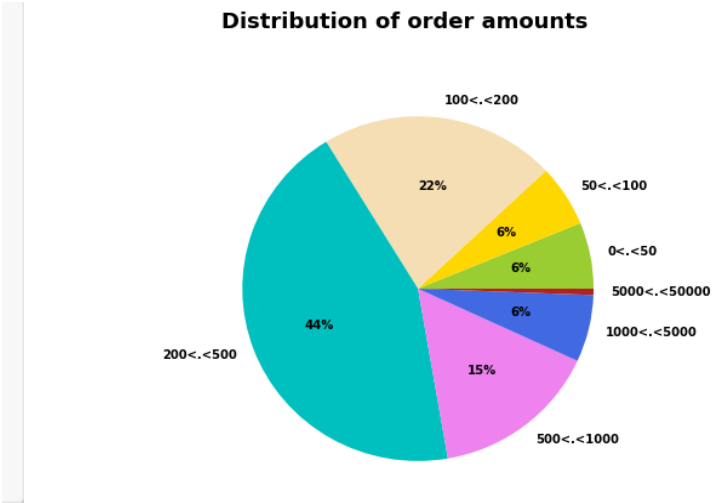

Figure 20 Distribution of order amounts (2)

Apparently, approximately 65% of the orders are above 200 GBP.

3.3.7 Product Categories

The EDA of the product-related data attribute is initiated, which will entail the following kinds of analysis :

- Analyzing the product description
- Defining the product categories
- Characterizing the content clusters

Two data attributes are used: StockCode which contains a Unique ID for each product and a Description to group the products into different categories. To begin with, the product description is considered.

First, the function will take the dataframe as an input, and then, the following operations are performed :

- Extract names(nouns) from the product description.
- Generate the root form of the extracted names - store it as the key and all associated names as its value. A stemmer from the NLTK library will be used for this step, which generates the root form of the words by removing the suffixes and prefixes.
- Count the frequency of the roots of the names

If various names have the same root, then the root form will be considered as the keyword tag (refer Figure 21).

```
        if pd.isnull(s): continue
        lines = s.lower()
        tokenized = nltk.word_tokenize(lines)
        nouns = [word for (word, pos) in nltk.pos_tag(tokenized) if is_noun(pos)]

        for t in nouns:
            t = t.lower() ; racine = stemmer.stem(t)
            if racine in keywords_roots:
                keywords_roots[racine].add(t)
                count_keywords[racine] += 1
            else:
                keywords_roots[racine] = {t}
                count_keywords[racine] = 1

    for s in keywords_roots.keys():
        if len(keywords_roots[s]) > 1:
            min_length = 1000
            for k in keywords_roots[s]:
                if len(k) < min_length:
                    clef = k ; min_length = len(k)
            category_keys.append(clef)
            keywords_select[s] = clef
        else:
            category_keys.append(list(keywords_roots[s])[0])
            keywords_select[s] = list(keywords_roots[s])[0]

    print("number of keywords in variable '{}': {}".format(colonne,len(category_keys)))
    return category_keys, keywords_roots, keywords_select, count_keywords
```

Figure 21 Extract names(nouns) from the product description

This function is called to feed the dataframe as an input and generate the keywords – Appended code snippet explains this step :

```
In [25]: df_produits = pd.DataFrame(df_initial['Description'].unique()).rename(columns = {0:'Description'})
```

Figure 22 Generation of keywords (1)

```
In [26]: keywords, keywords_roots, keywords_select, count_keywords = keywords_inventory(df_produits)

         number of keywords in variable 'Description': 1482
```

Figure 23 Generation of keywords (2)

Three variables are returned:

- Keyword: This is the list of extracted names
- Keywords_roots: This is a repository where the keys are the root of the name and values are the list of names pertinent to the root name
- Count_keywords: This is a dictionary that keeps track of the frequency of each name. The count indicates the number of times a particular name appeared in the description. This dictionary will be converted to a list eventually.

The following code snippet is used to plot the keywords versus their frequency graphs:

```
In [28]: liste = sorted(list_products, key = lambda x:x[1], reverse = True)

plt.rc('font', weight='normal')
fig, ax = plt.subplots(figsize=(7, 25))
y_axis = [i[1] for i in liste[:125]]
x_axis = [k for k,i in enumerate(liste[:125])]
x_label = [i[0] for i in liste[:125]]
plt.xticks(fontsize = 15)
plt.yticks(fontsize = 13)
plt.yticks(x_axis, x_label)
plt.xlabel("Number of occurences", fontsize = 18, labelpad = 10)
ax.barh(x_axis, y_axis, align = 'center')
ax = plt.gca()
ax.invert_yaxis()

plt.title("Words occurence",bbox={'facecolor':'k', 'pad':5}, color='w',fontsize = 25)
plt.show()
```

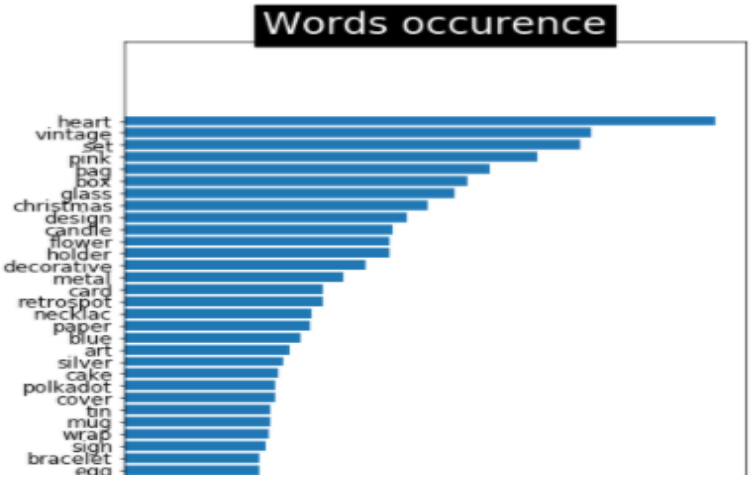

Figure 24 Graph of keywords and frequency

The preceding figure underscores that the word heart has appeared the maximum number of times in the product description – this word frequency is used to categorize products.

3.3.8 Defining product categories

The most recurring names have appeared in more than 200 products and more than 1400 words are obtained – However, less important words such as names of colors will be discarded and only those words that appear in the dataset more than 13 times will be retained. The following code snippet achieves this effect:

```
In [29]: list_products = []
         for k,v in count_keywords.items():
             word = keywords_select[k]
             if word in ['pink', 'blue', 'tag', 'green', 'orange']: continue
             if len(word) < 3 or v < 13: continue
             if ('+' in word) or ('/' in word): continue
             list_products.append([word, v])

         list_products.sort(key = lambda x:x[1], reverse = True)
         print('Preserved words:', len(list_products))

         Preserved words: 193
```

Figure 25 Derivation of product categories (1)

These keywords are used to create groups of products – To achieve this, a matrix X is defined as:

	word 1	...	word j	...	word N
product 1	$a_{1,1}$				$a_{1,N}$
...			...		
product i	...		$a_{i,j}$		...
...			...		
product M	$a_{M,1}$				$a_{M,N}$

where the $a_{i,j}$ coefficient is 1 if the description of the product *i* contains the word *j*, and 0 otherwise.

Figure 26 Derivation of product categories (2)

```
In [30]: liste_produits = df_cleaned['Description'].unique()
         #print(liste_produits[0:2])
         X = pd.DataFrame()
         for key, occurence in list_products:
             X.loc[:, key] = list(map(lambda x:int(key.upper() in x), liste_produits))
         #print(X[0:1])
```

Figure 27 Derivation of product categories (3)

The matrix X is developed using the one-hot-encoding principle and implies the presence of words in the description of the products.

In the sequel to this, groups for the products are created based on the price range – To this end, the keyword list that is generated will be used, validate if the product description has the words that are present in the keywords, and take the mean value of the UnitPrice (refer Figure 28).

```
In [31]: threshold = [0, 1, 2, 3, 5, 10]
label_col = []
for i in range(len(threshold)):
    if i == len(threshold)-1:
        col = '.>{}'.format(threshold[i])
    else:
        col = '{}<.<{}'.format(threshold[i],threshold[i+1])
    #print(i)
    #print(col)
    label_col.append(col)
    X.loc[:, col] = 0

for i, prod in enumerate(liste_produits):
    prix = df_cleaned[ df_cleaned['Description'] == prod]['UnitPrice'].mean()
    #print (prix)
    j = 0
    while prix > threshold[j]:
        j+=1
        if j == len(threshold): break
    X.loc[i, label_col[j-1]] = 1
```

```
In [32]: print("{:<8} {:<20} \n".format('range', 'number of products') + 20*'-')
for i in range(len(threshold)):
    if i == len(threshold)-1:
        col = '.>{}'.format(threshold[i])
    else:
        col = '{}<.<{}'.format(threshold[i],threshold[i+1])
    print("{:<10}  {:<20}".format(col, X.loc[:, col].sum()))
```

```
range    number of products
--------------------
0<.<1       964
1<.<2       1009
2<.<3       673
3<.<5       606
5<.<10      470
.>10        156
```

Figure 28 Price ranges for products

Using k-means clustering algorithm, the clusters of the products are created. The scikit-learn library is used to implement the K-means clustering algorithm. The algorithm from scikit-learn uses Euclidean distance, which is not the best choice in the case of categorical variables. Instead, Hamming distance should be used and the most suitable library

for that is kmods, but this library is not available for all operating systems, so the only fallback option is to use the scikit-learn library. The number of clusters is defined that can represent the data perfectly and then silhouette score is used. The silhouette coefficient is computed using two parameters: The first is the mean intra-cluster distance (a) and the second is the mean nearest-cluster distance (b) for each sample in the dataset. The equation is : (b – a) / max(a , b). The b indicates the distance between a sample and the nearest cluster that the sample is not a part of. This score works if the number of labels is 2<= n_labels <= n_samples – 1. The best possible value for this score is 1, and the worst value is -1. Value 0 shows that clusters have overlapped. Negative values indicate that sample has been assigned to the wrong cluster. The following code snippet achieves the choice of an ideal number of clusters using silhouette score:

```
In [33]: matrix = X.as_matrix()
         for n_clusters in range(3,10):
             kmeans = KMeans(init='k-means++', n_clusters = n_clusters, n_init=30)
             kmeans.fit(matrix)
             clusters = kmeans.predict(matrix)
             silhouette_avg = silhouette_score(matrix, clusters)
             print("For n_clusters =", n_clusters, "The average silhouette_score is :", s

         For n_clusters = 3 The average silhouette_score is : 0.09751688498995637
         For n_clusters = 4 The average silhouette_score is : 0.12609893747265383
         For n_clusters = 5 The average silhouette_score is : 0.1466257603527048
         For n_clusters = 6 The average silhouette_score is : 0.14861342533737415
         For n_clusters = 7 The average silhouette_score is : 0.15083317916457992
         For n_clusters = 8 The average silhouette_score is : 0.15054954899418788
         For n_clusters = 9 The average silhouette_score is : 0.1480858139817441
```

Figure 29 Analysis of cluster count using silhouette score

The code is implemented using scikit-learn API. Apparently, beyond five clusters, a cluster may contain minimal products, so the products are categorized into five clusters. An attempt is made to increase the value of the silhouette score by iterating through the dataset and the appended code snippet achieves this effect:

```
In [34]: n_clusters = 5
         silhouette_avg = -1
         while silhouette_avg < 0.145:
             kmeans = KMeans(init='k-means++', n_clusters = n_clusters, n_init=30)
             kmeans.fit(matrix)
             clusters = kmeans.predict(matrix)
             silhouette_avg = silhouette_score(matrix, clusters)

             #km = kmodes.KModes(n_clusters = n_clusters, init='Huang', n_init=2, verbose=0)
             #clusters = km.fit_predict(matrix)
             #silhouette_avg = silhouette_score(matrix, clusters)
             print("For n_clusters =", n_clusters, "The average silhouette_score is :", silhouette_avg)

         For n_clusters = 5 The average silhouette_score is : 0.1452148389646187
```

Figure 30 Characterizing the content of clusters

Further analysis of the properties of the product clusters entails the following steps :

- Silhouette intra-cluster score analysis
- Analysis using a word cloud
- Principal component analysis (PCA)

Before embarking on this process, I need to verify the number of products in each cluster and the following code snippet achieves this effect:

```
In [35]: pd.Series(clusters).value_counts()

Out[35]: 0    1159
         1     964
         2     673
         4     606
         3     476
         dtype: int64
```

Figure 31 Analysis of number of products in each cluster

3.3.9 Silhouette intra-cluster score analysis

An in-depth analysis of these five clusters and their elements is made. First, the silhouette intra-cluster score analysis is initiated, wherein the intra-cluster score for each element will be examined and sort the scores. After sorting, a graph will be drawn to

represent the silhouette coefficient value on the x-axis and the cluster label on the y-axis. Silhouette intra-cluster scores are generated for all samples to build this graph so that an optimal value for n_clusters could be chosen. The Clusters are represented pictorially for an ideal choice of n_clusters = 5 in keeping with the intra-cluster score generated earlier.

```
In [36]: def graph_component_silhouette(n_clusters, lim_x, mat_size, sample_silhouette_values, clusters):
             #plt.rcParams["patch.force_edgecolor"] = True
             plt.style.use('fivethirtyeight')
             mpl.rc('patch', edgecolor = 'dimgray', linewidth=1)

             fig, ax1 = plt.subplots(1, 1)
             fig.set_size_inches(8, 8)
             ax1.set_xlim([lim_x[0], lim_x[1]])
             ax1.set_ylim([0, mat_size + (n_clusters + 1) * 10])
             y_lower = 10
             for i in range(n_clusters):

                 # Aggregate the silhouette scores for samples belonging to cluster i, and sort them
                 ith_cluster_silhouette_values = sample_silhouette_values[clusters == i]
                 ith_cluster_silhouette_values.sort()
                 size_cluster_i = ith_cluster_silhouette_values.shape[0]
                 y_upper = y_lower + size_cluster_i
                 #color = cm.spectral(float(i) / n_clusters) facecolor=color, edgecolor=color,
                 ax1.fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values, alpha=0.8)

                 # Label the silhouette plots with their cluster numbers at the middle
                 ax1.text(-0.03, y_lower + 0.5 * size_cluster_i, str(i), color = 'red', fontweight = 'bold',
                         bbox=dict(facecolor='white', edgecolor='black', boxstyle='round, pad=0.3'))

                 # Compute the new y_lower for next plot
                 y_lower = y_upper + 10
```

Figure 32 Silhouette intra-cluster score analysis (1)

After executing and calling this function, the graph displayed is shown in the appended screenshot:

```
In [37]:
# define individual silouhette scores
sample_silhouette_values = silhouette_samples(matrix, clusters)

# and do the graph
graph_component_silhouette(n_clusters, [-0.07, 0.33], len(X), sample_silhouette_values, clusters)
```

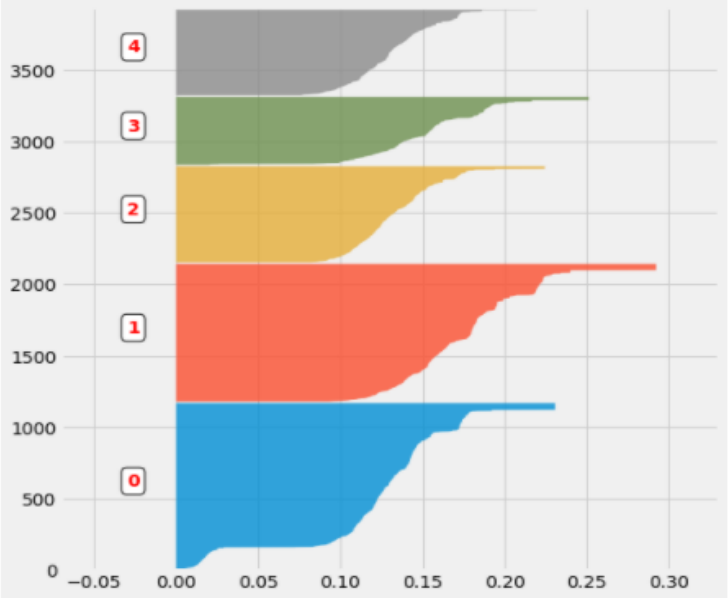

Figure 33 Silhouette intra-cluster score analysis (2)

3.3.10 Analysis using a word cloud

Analysis of the clusters based on the keywords is made and check what words each cluster has. For this analysis, the word cloud library will be used. In these clusters, a similar kind of product belongs to one cluster. Visualizing the words for the entire cluster will help in concluding if the clusters have similar kinds of products or not. Graphs are generated that are intuitive enough to judge the accuracy of clustering. To get an overall view of the cluster contents, the most frequently occurring keywords in each of them is determined in the following code snippet:

```
In [38]:
liste = pd.DataFrame(liste_produits)
liste_words = [word for (word, occurence) in list_products]

occurence = [dict() for _ in range(n_clusters)]

for i in range(n_clusters):
    liste_cluster = liste.loc[clusters == i]
    for word in liste_words:
        if word in ['art', 'set', 'heart', 'pink', 'blue', 'tag']: continue
        occurence[i][word] = sum(liste_cluster.loc[:, 0].str.contains(word.upper()))
```

Figure 34 Analysis using a word cloud (1)

The appended screenshot has the code snippet for generating the word cloud graphs:

In [39]:

```
def random_color_func(word=None, font_size=None, position=None,
                      orientation=None, font_path=None, random_state=None):
    h = int(360.0 * tone / 255.0)
    s = int(100.0 * 255.0 / 255.0)
    l = int(100.0 * float(random_state.randint(70, 120)) / 255.0)
    return "hsl({}, {}%, {}%)".format(h, s, l)

def make_wordcloud(liste, increment):
    ax1 = fig.add_subplot(4,2,increment)
    words = dict()
    trunc_occurences = liste[0:150]
    for s in trunc_occurences:
        words[s[0]] = s[1]

    wordcloud = WordCloud(width=1000,height=400, background_color='lightgrey',
                          max_words=1628,relative_scaling=1,
                          color_func = random_color_func,
                          normalize_plurals=False)
    wordcloud.generate_from_frequencies(words)
    ax1.imshow(wordcloud, interpolation="bilinear")
    ax1.axis('off')
    plt.title('cluster n{}'.format(increment-1))

fig = plt.figure(1, figsize=(14,14))
color = [0, 160, 130, 95, 280, 40, 330, 110, 25]
for i in range(n_clusters):
    list_cluster_occurences = occurence[i]

    tone = color[i] # define the color of the words
    liste = []
    for key, value in list_cluster_occurences.items():
        liste.append([key, value])
    liste.sort(key = lambda x:x[1], reverse = True)
    make_wordcloud(liste, i+1)
```

Figure 35 Analysis using a word cloud (2)

Figure 36 Analysis using a word cloud (3)

From the preceding graphs, the conclusion of the following points can be drawn:

- Cluster number 2 contains all the words related to gifts, such as Christmas, packaging, gift, cards, and so on.
- Cluster number 4 contains all the words related to items of luxury. So, words such as silver, lace, and so on are present in this cluster.
- Some words are present in every cluster, so it is difficult to differentiate them.

3.3.11 Principal component analysis (PCA)

To validate whether all the clusters have truly distinct values, the composition of the clusters should be considered. The one-hot encoded matrix of the keywords has a large number of dimensions or a large number of variables. There may be a situation where because of the large number of variables, our clustering algorithm may over-fit the dataset. First of all, there's a need to reduce the number of variables, but this cannot be done randomly. The most important variables that can represent most of the characteristics of the dataset must be selected. The procedure for reducing the number of variables logically is called dimensionality reduction. To achieve this, PCA is used, which is a statistical technique in which we will perform the orthogonal transformation to convert a highly correlated set of data samples into a set of linearly uncorrelated variables, and these variables are referred to as principal components. So basically, the PCA technique is used to reduce the number of variables that are considered so far. PCA technique will help reduce the dimensions and avoid the over-fitting issue.

```
In [40]: pca = PCA()
         pca.fit(matrix)
         pca_samples = pca.transform(matrix)
```

```
In [41]: fig, ax = plt.subplots(figsize=(14, 5))
         sns.set(font_scale=1)
         plt.step(range(matrix.shape[1]), pca.explained_variance_ratio_.cumsum(), where='mid',
                 label='cumulative explained variance')
         sns.barplot(np.arange(1,matrix.shape[1]+1), pca.explained_variance_ratio_, alpha=0.5, color = 'g',
                    label='individual explained variance')
         plt.xlim(0, 100)

         ax.set_xticklabels([s if int(s.get_text())%2 == 0 else '' for s in ax.get_xticklabels()])

         plt.ylabel('Explained variance', fontsize = 14)
         plt.xlabel('Principal components', fontsize = 14)
         plt.legend(loc='upper left', fontsize = 13);
```

Figure 37 Principal component analysis (1)

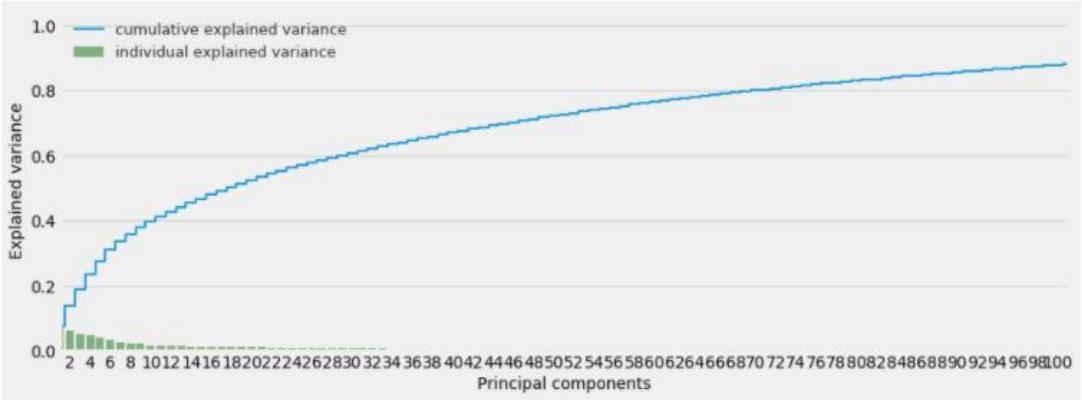

Figure 38 Principal component analysis (2)

In the preceding code, validation is done for the amount of variance explained by each component. To explain 90% of the variance in our dataset, more than 100 components are required. However, to visualize the data, only a limited number of components are needed.

```
In [42]: pca = PCA(n_components=50)
         matrix_9D = pca.fit_transform(matrix)
         mat = pd.DataFrame(matrix_9D)
         mat['cluster'] = pd.Series(clusters)
```

Figure 39 Principal component analysis (3)

```
In [43]: import matplotlib.patches as mpatches

         sns.set_style("white")
         sns.set_context("notebook", font_scale=1, rc={"lines.linewidth": 2.5})

         LABEL_COLOR_MAP = {0:'r', 1:'gold', 2:'b', 3:'k', 4:'c', 5:'g'}
         label_color = [LABEL_COLOR_MAP[l] for l in mat['cluster']]

         fig = plt.figure(figsize = (12,10))
         increment = 0
         for ix in range(4):
             for iy in range(ix+1, 4):
                 increment += 1
                 ax = fig.add_subplot(3,3,increment)
                 ax.scatter(mat[ix], mat[iy], c= label_color, alpha=0.4)
                 plt.ylabel('PCA {}'.format(iy+1), fontsize = 12)
                 plt.xlabel('PCA {}'.format(ix+1), fontsize = 12)
                 ax.yaxis.grid(color='lightgray', linestyle=':')
                 ax.xaxis.grid(color='lightgray', linestyle=':')
                 ax.spines['right'].set_visible(False)
                 ax.spines['top'].set_visible(False)

                 if increment == 9: break
             if increment == 9: break


         comp_handler = []
         for i in range(5):
             comp_handler.append(mpatches.Patch(color = LABEL_COLOR_MAP[i], label = i))

         plt.legend(handles=comp_handler, bbox_to_anchor=(1.1, 0.97),
                    title='Cluster',
                    shadow = True, frameon = True, framealpha = 1,fontsize = 13,
                    bbox_transform = plt.gcf().transFigure) #facecolor = 'lightgrey',

         plt.tight_layout()
```

Figure 40 Principal component analysis (4)

The output of the preceding code is in form of graphs appended herewith:

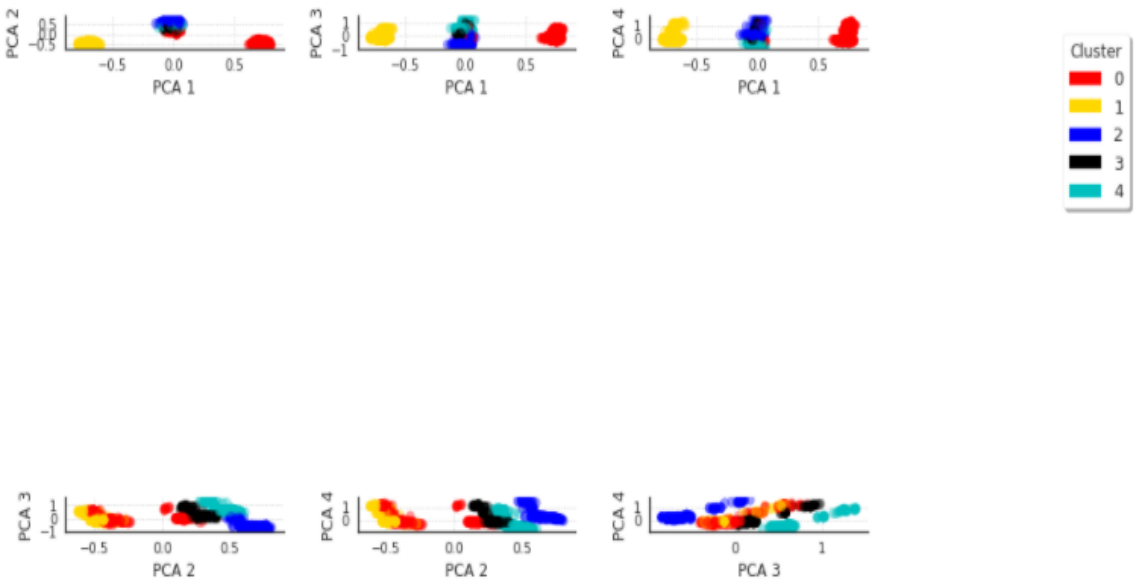

Figure 41 Principal component analysis (5)

This completes the EDA to generate a basic insight into the dataset. In the sequel to this, the process of building customer categories or customer segmentation will be embarked upon.

3.4 Achieving Research Goals

The first goal is to develop customer segmentation and from this point onward, the focus will be mainly on how we can come up with customer segmentation. So far, analysis of orders, products, prices, and so on have been done, and going forward, the main focus would be on generating customer categories based on the insights from the EDA process.

Following are the steps for developing the customer categories:

Formatting data:

- Grouping products
- Splitting the dataset
- Grouping orders

Creating customer categories:

- Data encoding
- Generating customer categories

Primarily, to achieve the first goal of customer segmentation, the RFM Analysis (Recency, Frequency, Monetary) will be performed using the K-Means, PCA, and Silhouette algorithms. In the sequel to this, the behavior of the customer of each segment will be obtained, and items in keeping with these characteristics could be recommended. A marketing campaign based on these generated facts could be designed.

The second goal of this research is to build a Classifier that will segregate the customers into different customer segments established earlier. On the first visit of the customer to the platform, the Classifier should generate this classification result. To implement this kind of functionality, various supervised machine learning algorithms will be used. The scikit-learn API will be utilized in this implementation.

To develop the baseline classifier, the following steps will be performed:

- Defining the helper functions
- Splitting the data into training and testing
- Implementing the Machine Learning Algorithm

For the baseline approach, the Support Vector Machine (SVM) classifier will be implemented, using the helper functions. The baseline model will be tested using the following approaches:

- Generating the accuracy score for the classifier
- Generating the confusion matrix for the classifier
- Generating the learning curve for the classifier

Finally, this research will attempt to optimize the baseline approach using the following strategies:

Explore the scope for improvement by increasing the Accuracy of the Classifier

Try other ML algorithms to compare their results –Build a voting mechanism, should there be a need. Basically, in the revised approach, various ML algorithms will be tried out to finalize the algorithm to be used. Primarily, six algorithms will be implemented in the revised approach :

- Logistic regression
- K-nearest neighbor

- Decision tree
- Random Forest
- Adaboost classifier
- Gradient boosting classifier

In keeping with the precision score of all the preceding algorithms, a decision will be made which algorithm can be used and which can't be used. The classifier model that will be generated in this revised approach should provide the best possible accuracy.

3.5 Summary

This Research will help develop a customer segmentation model based on the buying behavior of the customers. To achieve this, various ML algorithms are used, such as SVM, linear regression, decision tree, random forest, gradient boosting, etc. The accuracy of the classifier is further improved in the sequel to the usage of multiple algorithms.

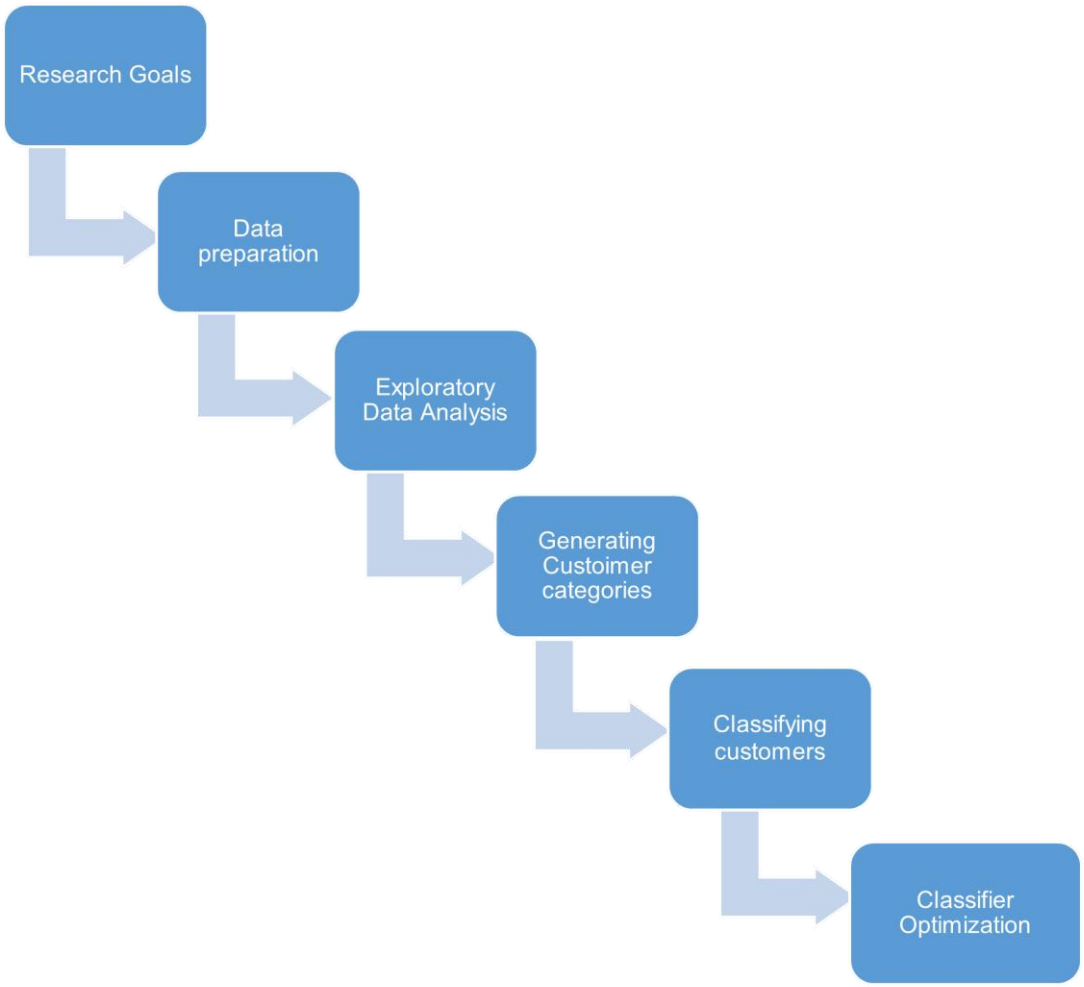

Figure 42 Pictorial Representation of Methodology

CHAPTER IV:

ANALYSIS

4.1 Introduction

The focus of this analysis is to develop customer segmentation using the RFM model, in keeping with the insights derived during the EDA for the Orders, Products, Prices, etc. This helps to fulfill the first goal of this thesis. To fulfill the second goal of the thesis, this analysis will be further progressed to build a classifier that will classify customers into different customer segments that were established in the first part of the analysis. Over and above, the classifier should accurately generate the classification result when the customer visits the platform for the first time.

4.2 Generating customer categories

The following process will be adopted to develop the customer categories :

- Formatting data:
- Grouping products
- Splitting the dataset
- Grouping orders
- Creating customer categories:
- Data encoding
- Generating customer categories

4.2.1 Formatting Data

Using the five clusters, list of keywords, matrices of products generated during the EDA, a new categorical variable named categ_product will be generated to indicate the cluster of each product (refer Figure 43).

In [44]:
```
corresp = dict()
for key, val in zip (liste_produits, clusters):
    corresp[key] = val

df_cleaned['categ_product'] = df_cleaned.loc[:, 'Description'].map(corresp)
df_cleaned[['InvoiceNo', 'Description',
            'categ_product']][:10]
```

Out[44]:

	InvoiceNo	Description	categ_product
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	2
1	536365	WHITE METAL LANTERN	4
2	536365	CREAM CUPID HEARTS COAT HANGER	4
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	4
4	536365	RED WOOLLY HOTTIE WHITE HEART.	4
5	536365	SET 7 BABUSHKA NESTING BOXES	3
6	536365	GLASS STAR FROSTED T-LIGHT HOLDER	4
7	536366	HAND WARMER UNION JACK	2
8	536366	HAND WARMER RED POLKA DOT	0
9	536367	ASSORTED COLOUR BIRD ORNAMENT	0

Figure 43 Defining Product Categories

The new variable indicates the cluster number for each data entry, using which products will be grouped. Product grouping helps to reveal the amount spent on each product category. To achieve this effect, five new variables are added: categ_0,categ_1, categ_2,categ_3, categ_4 , and the Figure 44 illustrates the code snippet :

In [45]:
```
for i in range(5):
    col = 'categ_{}'.format(i)
    df_temp = df_cleaned[df_cleaned['categ_product'] == i]
    price_temp = df_temp['UnitPrice'] * (df_temp['Quantity'] - df_temp['QuantityCanceled'])
    price_temp = price_temp.apply(lambda x:x if x > 0 else 0)
    df_cleaned.loc[:, col] = price_temp
    df_cleaned[col].fillna(0, inplace = True)


df_cleaned[['InvoiceNo', 'Description',
            'categ_product', 'categ_0', 'categ_1', 'categ_2', 'categ_3','categ_4']][:10]
```

Out[45]:

	InvoiceNo	Description	categ_product	categ_0	categ_1	categ_2	categ_3	categ_4
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	2	0.00	0.0	15.3	0.0	0.00
1	536365	WHITE METAL LANTERN	4	0.00	0.0	0.0	0.0	20.34
2	536365	CREAM CUPID HEARTS COAT HANGER	4	0.00	0.0	0.0	0.0	22.00
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	4	0.00	0.0	0.0	0.0	20.34
4	536365	RED WOOLLY HOTTIE WHITE HEART.	4	0.00	0.0	0.0	0.0	20.34
5	536365	SET 7 BABUSHKA NESTING BOXES	3	0.00	0.0	0.0	15.3	0.00
6	536365	GLASS STAR FROSTED T-LIGHT HOLDER	4	0.00	0.0	0.0	0.0	25.50
7	536366	HAND WARMER UNION JACK	2	0.00	0.0	11.1	0.0	0.00
8	536366	HAND WARMER RED POLKA DOT	0	11.10	0.0	0.0	0.0	0.00
9	536367	ASSORTED COLOUR BIRD ORNAMENT	0	54.08	0.0	0.0	0.0	0.00

Figure 44 Defining Product Categories for amount spent

A couple of additional variables: FirstPurchase, LastPurchase are generated to signify the number of elapsed days since the last purchase and the first purchase (refer Figure 45).

In [46]:

```
# sum of purchases / user & order
temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['TotalPrice'].sum()
basket_price = temp.rename(columns = {'TotalPrice':'Basket Price'})

# percentage of the price of the order / product category
for i in range(5):
    col = 'categ_{}'.format(i)
    temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)[col].sum()
    basket_price.loc[:, col] = temp

# date of the order

df_cleaned['InvoiceDate_int'] = df_cleaned['InvoiceDate'].astype('int64')
temp = df_cleaned.groupby(by=['CustomerID', 'InvoiceNo'], as_index=False)['InvoiceDate_int'].mean()
df_cleaned.drop('InvoiceDate_int', axis = 1, inplace = True)
basket_price.loc[:, 'InvoiceDate'] = pd.to_datetime(temp['InvoiceDate_int'])

# selection of significant entries:
basket_price = basket_price[basket_price['Basket Price'] > 0]
basket_price.sort_values('CustomerID', ascending = True)[:5]
```

Out[46]:

	CustomerID	InvoiceNo	Basket Price	categ_0	categ_1	categ_2	categ_3	categ_4	InvoiceDate
1	12347	537626	711.79	187.20	23.40	83.40	124.44	293.35	2010-12-07 14:57:00.000001024
2	12347	542237	475.39	168.75	84.34	53.10	0.00	169.20	2011-01-26 14:29:59.999999744
3	12347	549222	636.25	369.15	81.00	71.10	0.00	115.00	2011-04-07 10:42:59.999999232
4	12347	556201	382.52	74.40	41.40	78.06	19.90	168.76	2011-06-09 13:01:00.000000256

Figure 45 Deriving Last and Fisrt Purchase

In this step, the new dataframe: basket_price provides much-needed insight into the basket price for each order and the price distribution over the five clusters is also apparent. This step helps establish the monetary aspect of the RFM model.

The next steps in this analysis will entail splitting the dataset basket_price, which has data entries for the last 12 months, into the training dataset having 10 month's data entries and the testing dataset having 2 month's data entries. The following code snippet illustrates this step :

```
In [47]: print(basket_price['InvoiceDate'].min(), '->',  basket_price['InvoiceDate'].max())

         2010-12-01 08:26:00 -> 2011-12-09 12:50:00

In [48]: set_entrainement = basket_price[basket_price['InvoiceDate'] < datetime.date(2011,10,1)]
         set_test         = basket_price[basket_price['InvoiceDate'] >= datetime.date(2011,10,1)]
         basket_price = set_entrainement.copy(deep = True)
```

Figure 46 Splitting the dataset into Training and Testing

Splitting the dataframe at this stage helps the usage of the new dataframe, while we can use the train and test dataset later.

This analysis will further merge the customers and their orders to provide insights into the number of orders placed by the customers and this helps establish the frequency aspect of the RFM model (refer Figure 47).

```
In [49]: # of visits and stats on cart amount / users
         transactions_per_user=basket_price.groupby(by=['CustomerID'])['Basket Price'].agg(['count','min',
                                                                                            'max','mean','sum'])
         for i in range(5):
             col = 'categ_{}'.format(i)
             transactions_per_user.loc[:,col] = basket_price.groupby(by=['CustomerID'])[col].sum() /\
                                                transactions_per_user['sum']*100

         transactions_per_user.reset_index(drop = False, inplace = True)
         basket_price.groupby(by=['CustomerID'])['categ_0'].sum()
         transactions_per_user.sort_values('CustomerID', ascending = True)[:5]
```

Out[49]:

	CustomerID	count	min	max	mean	sum	categ_0	categ_1	categ_2	categ_3	categ_4
0	12347	5	382.52	711.79	558.172000	2790.86	33.948317	10.442659	14.524555	8.676179	32.408290
1	12348	4	227.44	892.80	449.310000	1797.24	61.983931	38.016069	0.000000	0.000000	0.000000
2	12350	1	334.40	334.40	334.400000	334.40	60.406699	11.692584	27.900718	0.000000	0.000000
3	12352	6	144.35	840.30	345.663333	2073.98	66.125517	0.491808	3.370331	14.301006	15.711338
4	12353	1	89.00	89.00	89.000000	89.00	57.752809	0.000000	19.887640	22.359551	0.000000

Figure 47 Deriving the frequency of orders placed by customers

This code invariably generates the minimum order amount, the maximum order amount, and the mean order amount.

Finally, to establish the recency aspect of the RFM model, a couple of new variables are generated to signify the number of elapsed days since the last purchase and the first

purchase. These variables are FirstPurchase and LastPurchase – Following code snippet underscores the usage of these variables:

```
In [50]: last_date = basket_price['InvoiceDate'].max().date()

         first_registration = pd.DataFrame(basket_price.groupby(by=['CustomerID'])['InvoiceDate'].min())
         last_purchase      = pd.DataFrame(basket_price.groupby(by=['CustomerID'])['InvoiceDate'].max())

         test  = first_registration.applymap(lambda x:(last_date - x.date()).days)
         test2 = last_purchase.applymap(lambda x:(last_date - x.date()).days)

         transactions_per_user.loc[:, 'LastPurchase'] = test2.reset_index(drop = False)['InvoiceDate']
         transactions_per_user.loc[:, 'FirstPurchase'] = test.reset_index(drop = False)['InvoiceDate']

         transactions_per_user[:5]
```

Out[50]:

	CustomerID	count	min	max	mean	sum	categ_0	categ_1	categ_2	categ_3	categ_4	LastPurchase	FirstPurchase
0	12347	5	382.52	711.79	558.172000	2790.86	33.948317	10.442659	14.524555	8.676179	32.408290	59	297
1	12348	4	227.44	892.80	449.310000	1797.24	61.983931	38.016069	0.000000	0.000000	0.000000	5	288
2	12350	1	334.40	334.40	334.400000	334.40	60.406699	11.692584	27.900718	0.000000	0.000000	240	240
3	12352	6	144.35	840.30	345.663333	2073.98	66.125517	0.491808	3.370331	14.301006	15.711338	2	226
4	12353	1	89.00	89.00	89.000000	89.00	57.752809	0.000000	19.887640	22.359551	0.000000	134	134

Figure 48 Deriving the Fist and Last Purchase

The customer categories which are of interest are the ones that make only one order so that these customers could be targetted appropriately to retain them. The following code snippet helps reveal the data for the number of customers that belong to this category:

```
In [51]: n1 = transactions_per_user[transactions_per_user['count'] == 1].shape[0]
         n2 = transactions_per_user.shape[0]
         print("No. customers with single purchase: {:<2}/{:<5} ({:<2.2f}%)".format(n1,n2,n1/n2*100))

         No. customers with single purchase: 1445/3608  (40.05%)
```

Figure 49 Deriving Customer Category with only one order

It's apparent from the above code snippet that 40% of the customer base has placed only one order, and the marketing strategies should be designed to retain them on the online platform.

4.2.2 Creating customer categories

This section of the analysis will entail the generation of the customer segmentation, in keeping with the RFM analysis of the customer's purchase pattern. Essentially, this analysis will entail a couple of steps: Data Encoding and Generating customer categories or customer segmentation.

A Dataframe will be generated to contain the summary of all operations performed so far. Each record of this dataframe is associated with a single client, which will be used to characterize various types of customers.

The generated Dataframe has different variables having different ranges and variations – As such, a matrix will be generated to standardize these data entries. The appended code snippet explains this step:

```
In [52]: list_cols = ['count','min','max','mean','categ_0','categ_1','categ_2','categ_3','categ_4']
         #_____________________________________________________________
         selected_customers = transactions_per_user.copy(deep = True)
         matrix = selected_customers[list_cols].as_matrix()
```

```
In [53]: scaler = StandardScaler()
         scaler.fit(matrix)
         print('variables mean values: \n' + 90*'-' + '\n' , scaler.mean_)
         scaled_matrix = scaler.transform(matrix)

         variables mean values:
         ------------------------------------------------------------------------------------------
          [  3.62305987 259.93189634 556.26687999 377.06036244  32.75310053
           13.98907929  21.19884856  15.6945421   16.37327913]
```

Figure 50 Standardization of Data Entries

Before creating customer segmentation, a base of important variables should be created. To achieve this, the principal component analysis will be used to describe the customer segmentation accurately. The following code snippet explains the usage of PCA to this end:

```
In [54]: pca = PCA()
         pca.fit(scaled_matrix)
         pca_samples = pca.transform(scaled_matrix)
```

```
In [55]: fig, ax = plt.subplots(figsize=(14, 5))
         sns.set(font_scale=1)
         plt.step(range(matrix.shape[1]), pca.explained_variance_ratio_.cumsum(), where='mid',
                  label='cumulative explained variance')
         sns.barplot(np.arange(1,matrix.shape[1]+1), pca.explained_variance_ratio_, alpha=0.5, color = 'g',
                     label='individual explained variance')
         plt.xlim(0, 10)

         ax.set_xticklabels([s if int(s.get_text())%2 == 0 else '' for s in ax.get_xticklabels()])

         plt.ylabel('Explained variance', fontsize = 14)
         plt.xlabel('Principal components', fontsize = 14)
         plt.legend(loc='best', fontsize = 13);
```

Figure 51 Deriving the Principal Components(1)

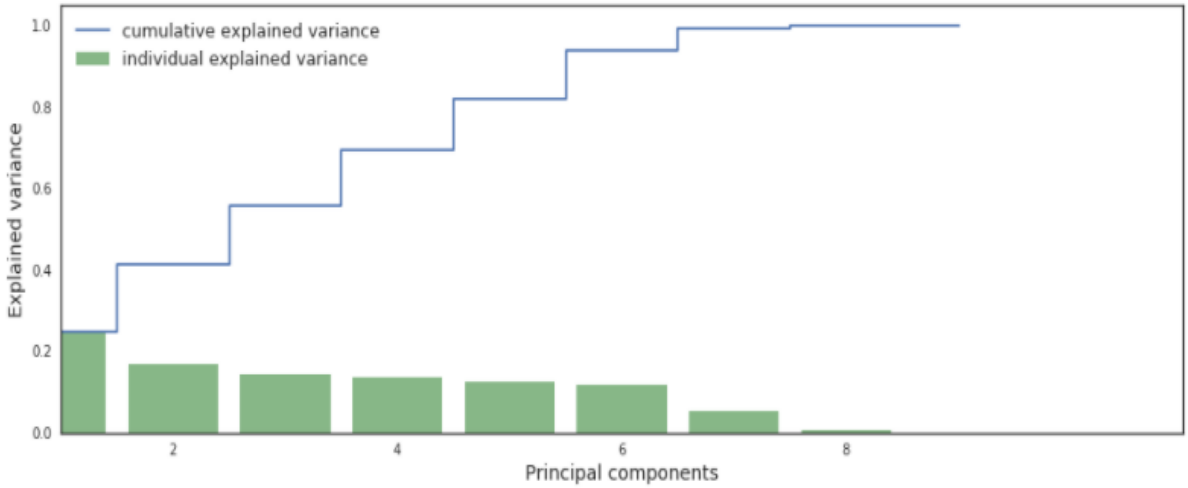

Figure 52 Deriving the Principal Components(2)

This underscores eight principal components. The next steps entail the generation of customer segmentation using the k-means clustering algorithm. The number of clusters will be derived using the Silhouette score. The appended code snippet underscores the derivation of 11 Clusters using the Silhouette Score method:

```
In [56]: n_clusters = 11
         kmeans = KMeans(init='k-means++', n_clusters = n_clusters, n_init=100)
         kmeans.fit(scaled_matrix)
         clusters_clients = kmeans.predict(scaled_matrix)
         silhouette_avg = silhouette_score(scaled_matrix, clusters_clients)
         print('silhouette score: {:<.3f}'.format(silhouette_avg))

         silhouette score: 0.218
```

```
In [57]: pd.DataFrame(pd.Series(clusters_clients).value_counts(), columns = ['number of clients']).T
```

Out[57]:

	0	2	6	9	3	1	7	5	8	10	4
number of clients	1553	502	334	294	291	265	191	151	13	7	7

Figure 53 Deriving the Clusters using Silhoutte score

The above code snippet reflects the number of customers in each cluster and also underscores the fact that there is a large difference in the size of the segmentation – As such, there's a need for further analysis of the cluster components using PCA. The following code snippet illustrates this step and the graphical representation:

```
In [58]: pca = PCA(n_components=6)
         matrix_3D = pca.fit_transform(scaled_matrix)
         mat = pd.DataFrame(matrix_3D)
         mat['cluster'] = pd.Series(clusters_clients)
```

Figure 54 Analysis of Cluster Components (1)

```
In [59]: import matplotlib.patches as mpatches

         sns.set_style("white")
         sns.set_context("notebook", font_scale=1, rc={"lines.linewidth": 2.5})

         LABEL_COLOR_MAP = {0:'r', 1:'tan', 2:'b', 3:'k', 4:'c', 5:'g', 6:'deeppink', 7:'skyblue', 8:'darkcyan',
                            9:'orange',
                            10:'yellow', 11:'tomato', 12:'seagreen'}
         label_color = [LABEL_COLOR_MAP[l] for l in mat['cluster']]

         fig = plt.figure(figsize = (12,10))
         increment = 0
         for ix in range(6):
             for iy in range(ix+1, 6):
                 increment += 1
                 ax = fig.add_subplot(4,3,increment)
                 ax.scatter(mat[ix], mat[iy], c= label_color, alpha=0.5)
                 plt.ylabel('PCA {}'.format(iy+1), fontsize = 12)
                 plt.xlabel('PCA {}'.format(ix+1), fontsize = 12)
                 ax.yaxis.grid(color='lightgray', linestyle=':')
                 ax.xaxis.grid(color='lightgray', linestyle=':')
                 ax.spines['right'].set_visible(False)
                 ax.spines['top'].set_visible(False)

                 if increment == 12: break
             if increment == 12: break

         #_____________________________________
         # I set the legend: abreviation -> airline name
         comp_handler = []
         for i in range(n_clusters):
             comp_handler.append(mpatches.Patch(color = LABEL_COLOR_MAP[i], label = i))

         plt.legend(handles=comp_handler, bbox_to_anchor=(1.1, 0.9),
                    title='Cluster',
                    shadow = True, frameon = True, framealpha = 1,
                    fontsize = 13, bbox_transform = plt.gcf().transFigure) #facecolor = 'lightgrey',

         plt.tight_layout()
```

Figure 55 Analysis of Cluster Components (2)

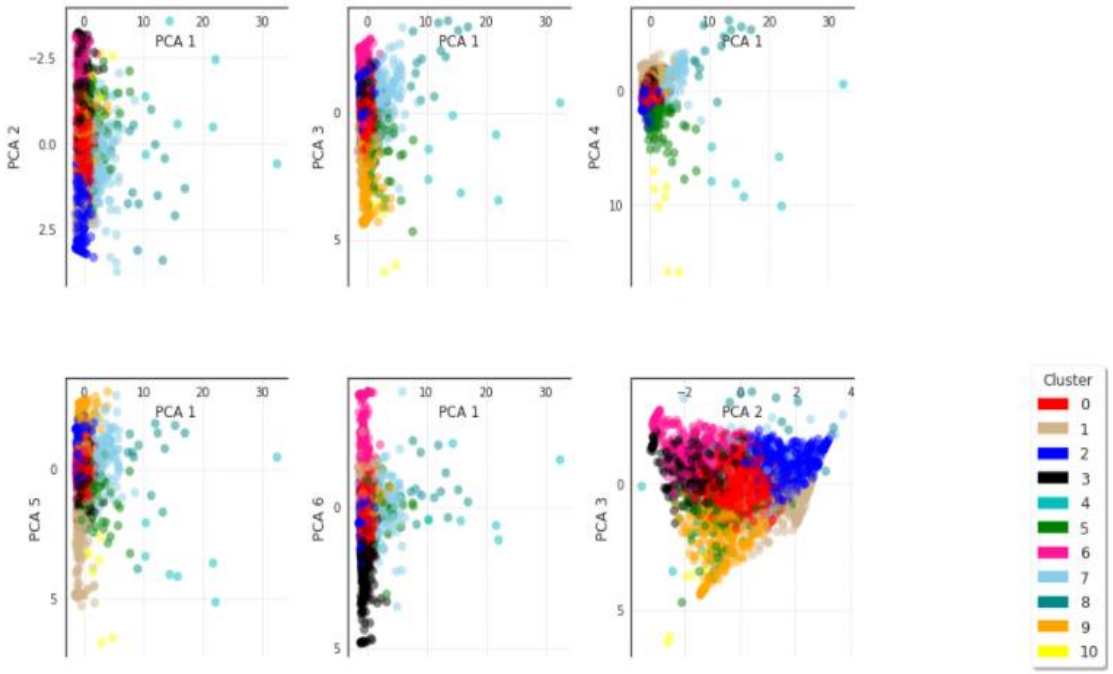

Figure 56 Analysis of Cluster Components (3)

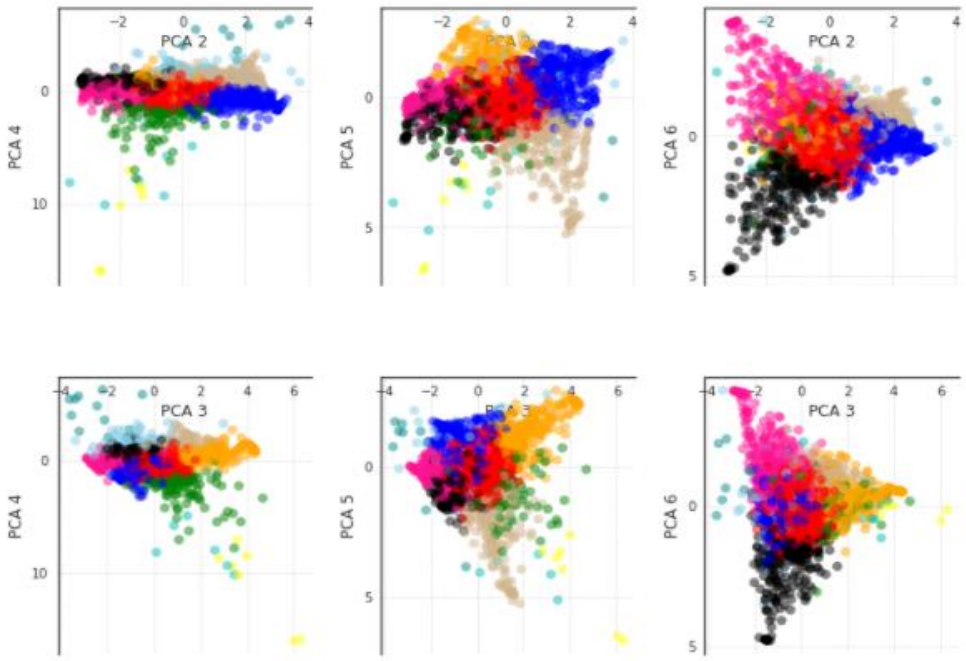

Figure 57 Analysis of Cluster Components (4)

It's worthwhile to note that, the first component separates the tinniest cluster from the rest – As such, for this dataset, we can say that there will always be a representation in which two segments will appear to be distinct.

To further this analysis, a silhouette score for each cluster will be generated, which will underscore the quality of the separation of data samples. The code snippet is appended to this end:

```
In [60]: sample_silhouette_values = silhouette_samples(scaled_matrix, clusters_clients)
         #_____________________________
         # define individual silhouette scores
         sample_silhouette_values = silhouette_samples(scaled_matrix, clusters_clients)
         #_____________________
         # and do the graph
         graph_component_silhouette(n_clusters, [-0.15, 0.55], len(scaled_matrix), sample_silhouette_values,
                                    clusters_clients)
```

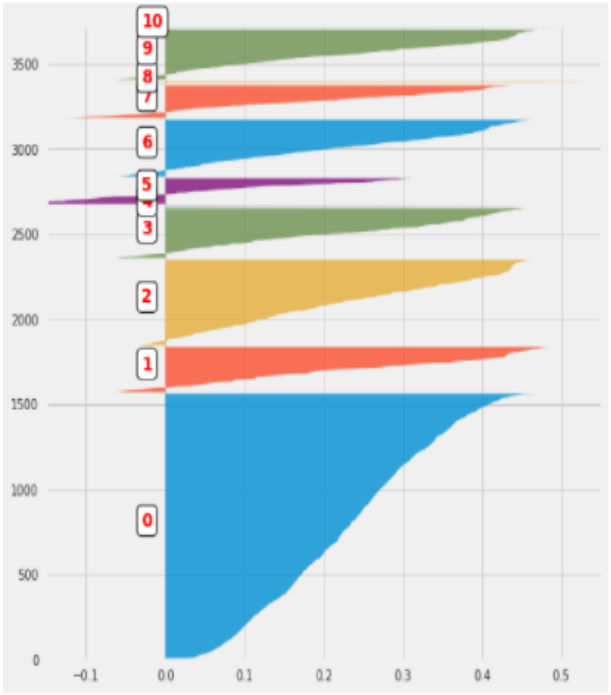

Figure 58 Silhoutte Scores of each cluster

Preceding graphs underscores that the clusters are disjointed. However, to figure out more about the behavior of the customers of each cluster, new variables will be added to define the cluster to which each customer belongs. To achieve this, a new dataframe : selected_customers will be generated, and its content will be averaged to provide the average basket price, total visits, and so on.

The appended code snippet illustrates this step:

```
In [61]: selected_customers.loc[:, 'cluster'] = clusters_clients
```

Figure 59 Generation of the new dataframe : selected_customers (1)

```
In [62]: merged_df = pd.DataFrame()
         for i in range(n_clusters):
             test = pd.DataFrame(selected_customers[selected_customers['cluster'] == i].mean())
             test = test.T.set_index('cluster', drop = True)
             test['size'] = selected_customers[selected_customers['cluster'] == i].shape[0]
             merged_df = pd.concat([merged_df, test])
         #_____________________________________________
         merged_df.drop('CustomerID', axis = 1, inplace = True)
         print('number of customers:', merged_df['size'].sum())

         merged_df = merged_df.sort_values('sum')

         number of customers: 3608
```

Figure 60 Generation of the new dataframe : selected_customers (2)

Now, the content of the dataframe will be reorganized, mindful of the following aspects:

- Data reorganization should be done in keeping with the amount spent in each product category.
- In the sequel to step 1 above, Data reorganization should be done in keeping with the total amount spent.

Below code snippet explains the above implementation :

```
In [63]: liste_index = []
         for i in range(5):
             column = 'categ_{}'.format(i)
             liste_index.append(merged_df[merged_df[column] > 45].index.values[0])

         liste_index_reordered = liste_index
         liste_index_reordered += [ s for s in merged_df.index if s not in liste_index]

         merged_df = merged_df.reindex(index = liste_index_reordered)
         merged_df = merged_df.reset_index(drop = False)
         display(merged_df[['cluster', 'count', 'min', 'max', 'mean', 'sum', 'categ_0',
                            'categ_1', 'categ_2', 'categ_3', 'categ_4', 'size']])
```

Figure 61 Data reorganization based on amount spent on each product category (1)

Out[ ]:

	cluster	count	min	max	mean	sum	categ_0	categ_1	categ_2	categ_3	categ_4	size
0	2.0	2.432271	198.114363	326.923946	255.382523	661.292114	66.451745	8.875318	10.694135	7.387286	6.594482	502
1	1.0	2.260377	193.966755	316.250415	247.336663	593.788566	22.081663	54.848316	11.317774	5.467600	6.284647	265
2	9.0	2.591837	209.375646	382.809660	292.227928	823.159728	17.664849	6.971079	60.857427	7.381182	7.125462	294
3	6.0	2.482036	191.252725	307.080689	243.361162	620.928204	17.160023	5.214240	11.495649	53.155612	12.991371	334
4	3.0	2.144330	202.483505	339.846014	264.699716	665.022405	17.008286	6.695432	14.265775	10.810850	51.272885	291
5	0.0	3.259498	223.028050	457.820812	331.429539	1085.060413	32.743319	13.613680	21.774264	14.751059	17.121494	1553
6	7.0	1.712042	1033.485759	1395.821209	1197.460012	2175.600634	35.608706	12.208932	21.297989	13.714542	17.170184	191
7	8.0	1.692308	3253.388462	4380.010000	3794.797051	6250.506154	29.560482	21.744522	13.782335	19.088117	15.824545	13
8	5.0	18.503311	85.567682	1699.433576	585.954054	10333.611457	30.517202	12.181422	25.393523	15.685715	16.242889	151
9	10.0	92.000000	10.985714	1858.250000	374.601553	34845.105714	33.256402	13.117583	22.527857	17.721038	13.402971	7
10	4.0	26.857143	510.302857	20131.802857	5514.816882	113654.117143	30.232736	7.873243	25.738996	18.567149	17.587876	7

Figure 62 Data reorganization based on amount spent on each product category (2)

This helps to establish the behavior of the customer for each segment – in keeping with these characteristics, items could be recommended and desired marketing campaign could be designed as well. For instance, a particular marketing strategy could be applied to the customers who belong to clusters 4 and 8. Premium products could be recommended to the cluster 1 clients. It's also worthwhile to observe that the first 5 clusters correspond to a strong dominance of purchases in a particular category of products. Other clusters are differentiated by either basket averages (mean **),** or the total sum spent by the clients (sum), or the total number of visits made (count).

Until this stage, the first goal of this thesis is achieved, and hereafter, the necessary analysis will be done to achieve the second goal of the thesis.

4.3 Classifying Customers

To fulfill the second goal of the thesis, a classifier will be build to classify the customers into different customer segments established in section 4.2. Further, this classifier should be in a position to generate this classification result when the customer visits the online platform for the first time. To implement such functionality, several supervised

machine learning algorithms and scikit-learn API will be used. The following steps will be performed to develop a baseline classifier :

- Defining the helper functions
- Splitting the data into training and testing
- Implementing the Machine Learning algorithm.

4.3.1 Helper Function Definition

A class: class_fit is defined to implement the following functions :

- The train function helps to train the model.
- The predict function helps to predict the result for the test dataset or the new data sample.
- The grid_search function helps to figure out appropriate hyperparameters and the value of cross-validation(CV) folds.
- The grid_fit function helps to train the model using cross-validation and generate the optimal hyperparameters.
- The grid_predict function helps to generate prediction as well as the accuracy score.

The appended is the code snippet to this end :

In [66]:
```
class Class_Fit(object):
    def __init__(self, clf, params=None):
        if params:
            self.clf = clf(**params)
        else:
            self.clf = clf()

    def train(self, x_train, y_train):
        self.clf.fit(x_train, y_train)

    def predict(self, x):
        return self.clf.predict(x)

    def grid_search(self, parameters, Kfold):
        self.grid = GridSearchCV(estimator = self.clf, param_grid = parameters, cv = Kfold)

    def grid_fit(self, X, Y):
        self.grid.fit(X, Y)

    def grid_predict(self, X, Y):
        self.predictions = self.grid.predict(X)
        print("Precision: {:.2f} % ".format(100*metrics.accuracy_score(Y, self.predictions)))
```

Figure 63 Definition of the Class_Fit (1)

4.3.2 Splitting the data into training and testing

Data stored in the dataset selected_customers will be split into train and test data, as underscored in the appended code snippet :

In [67]:
```
selected_customers.head()
```

Out[67]:

	CustomerID	count	min	max	mean	sum	categ_0	categ_1	categ_2	categ_3	categ_4	LastPurchase	FirstPurchase	cluster
0	12347	5	382.52	711.79	558.172000	2790.86	33.948317	10.442659	14.524555	8.676179	32.408290	59	297	0
1	12348	4	227.44	892.80	449.310000	1797.24	61.983931	38.016069	0.000000	0.000000	0.000000	5	288	2
2	12350	1	334.40	334.40	334.400000	334.40	60.406699	11.692584	27.900718	0.000000	0.000000	240	240	2
3	12352	6	144.35	840.30	345.663333	2073.98	66.125517	0.491808	3.370331	14.301006	15.711338	2	226	2
4	12353	1	89.00	89.00	89.000000	89.00	57.752809	0.000000	19.887640	22.359551	0.000000	134	134	2

In [68]:
```
columns = ['mean', 'categ_0', 'categ_1', 'categ_2', 'categ_3', 'categ_4' ]
X = selected_customers[columns]
Y = selected_customers['cluster']
```

Figure 64 Definition of the Class_Fit (2)

The Prediction will be made for the Cluster Number for the New Customer, which is stored in the variable Y, and the columns such as mean, categ_0 to categ_4 which are input features for the ML model are stored in the variable x. This data is split into training and testing in the proportion 80% and 20% respectively, which is explained by the appended code snippet :

```
In [69]: X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, train_size = 0.8)
```

Figure 65 Splitting the dataset into training and testing subsets

These training and testing datasets will be used to implement the ML algorithms.

4.3.3 Implementing the Machine Learning (ML) algorithm

For the baseline approach, the Support Vector Machine(SVM) classifier will be implemented, using the helper functions defined in the class class_fit. This is illustrated through the code snippet appended herewith:

```
In [70]: svc = Class_Fit(clf = svm.LinearSVC)
         svc.grid_search(parameters = [{'C':np.logspace(-2,2,10)}], Kfold = 5)

In [71]: svc.grid_fit(X = X_train, Y = Y_train)
```

Figure 66 Defining the SVM Classifier

It is apparent from this code snippet that, svc is an instance of the class class_fit and linear SVC is used. The method grid_search of the class class_fit is used to search optimal hyperparameters as well as obtain the number of CV folds. Further, the method grid_fit of the class class_fit is used to train the ML model using the training dataset.

This completes the implementation of the baseline approach and hereafter, the result will be tested. The confusion matrix and the learning curve will be used to evaluate the ML models.

4.4.4 Testing the result of the baseline approach

The baseline model will be tested using the following approaches:

- Generating the accuracy score for the classifier
- Generating the confusion matrix for the classifier
- Generating the learning curve for the classifier

4.3.5 Generating the accuracy score for the classifier

The method grid_predict is used to generate the accuracy score for testing the dataset. The code snippet appended herewith will check the accuracy of the SVM algorithm:

```
In [72]: svc.grid_predict(X_test, Y_test)

         Precision: 70.78 %
```

Figure 67 Deriving the accuracy score

The precision score for the baseline approach is 79.50% and the quality of the prediction will be evaluated using a confusion matrix.

4.3.6 Generating the confusion matrix for the classifier

The generated confusion matrix provides an insight into which class is classified correctly and which classes have been misclassified the data most of the time. The following code snippet explains this implementation, wherein the confusion_matrix API for sklearn is used:

```
In [73]: def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix', cmap=plt.cm.Blues):
             if normalize:
                 cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
                 print("Normalized confusion matrix")
             else:
                 print('Confusion matrix, without normalization')

             plt.imshow(cm, interpolation='nearest', cmap=cmap)
             plt.title(title)
             plt.colorbar()
             tick_marks = np.arange(len(classes))
             plt.xticks(tick_marks, classes, rotation=0)
             plt.yticks(tick_marks, classes)

             fmt = '.2f' if normalize else 'd'
             thresh = cm.max() / 2.
             for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
                 plt.text(j, i, format(cm[i, j], fmt),
                          horizontalalignment="center",
                          color="white" if cm[i, j] > thresh else "black")
             #_____________________________________________
             plt.tight_layout()
             plt.ylabel('True label')
             plt.xlabel('Predicted label')
```

Figure 68 Defining the function for confusion matrix

Using the above method: plot_confusion_matrix, the confusion matrix is generated :

```
In [74]: class_names = [i for i in range(11)]
         cnf_matrix = confusion_matrix(Y_test, svc.predictions)
         np.set_printoptions(precision=2)
         plt.figure(figsize = (8,8))
         plot_confusion_matrix(cnf_matrix, classes=class_names, normalize = False, title='Confusion matrix')
```

Figure 69 Generating the confusion matrix (1)

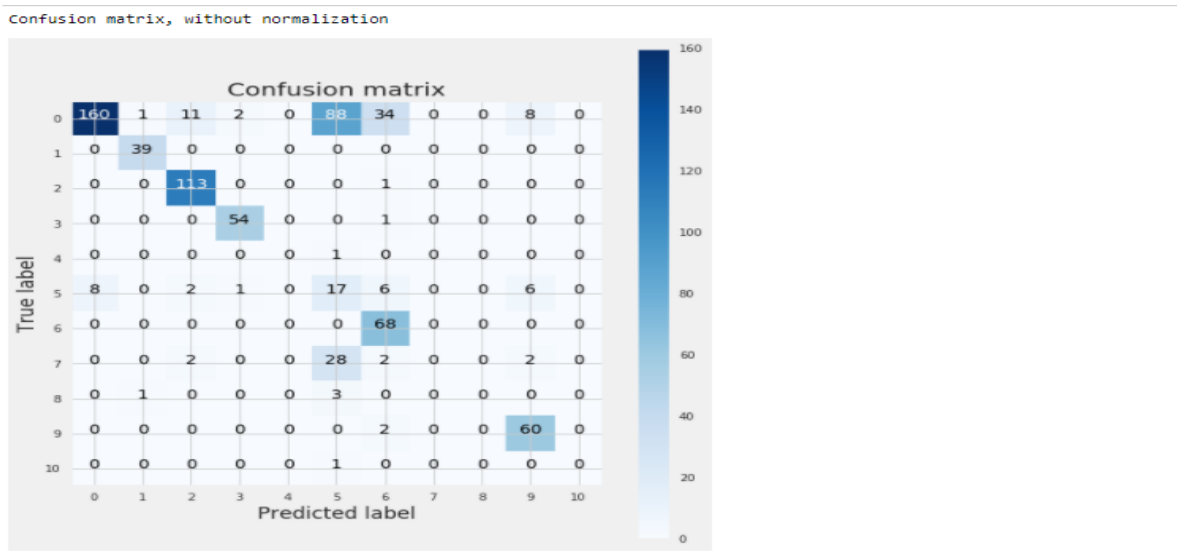

Figure 70 Generating the confusion matrix (2)

It's apparent from the confusion matrix that, the classifier could classify the data into the class labels 0, 2, 3, 6, 9 accurately, whereas for the class labels 1, 4, 5, 7, 8, 10, the classifier is not performing so well.

4.3.7 Generating the learning curve for the classifier

A learning curve signifies whether the classifier is facing the over-fitting or under-fitting issue. The method plot_learning_curve is used to draw the learning curve for the classifier. The following is the code snippet to this end:

```
In [75]: def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                                 n_jobs=-1, train_sizes=np.linspace(.1, 1.0, 10)):
             """Generate a simple plot of the test and training learning curve"""
             plt.figure()
             plt.title(title)
             if ylim is not None:
                 plt.ylim(*ylim)
             plt.xlabel("Training examples")
             plt.ylabel("Score")
             train_sizes, train_scores, test_scores = learning_curve(
                 estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
             train_scores_mean = np.mean(train_scores, axis=1)
             train_scores_std = np.std(train_scores, axis=1)
             test_scores_mean = np.mean(test_scores, axis=1)
             test_scores_std = np.std(test_scores, axis=1)
             plt.grid()

             plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                              train_scores_mean + train_scores_std, alpha=0.1, color="r")
             plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                              test_scores_mean + test_scores_std, alpha=0.1, color="g")
             plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
             plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation score")

             plt.legend(loc="best")
             return plt
```

Figure 71 Defining the function for the learning curve

Using the above method, the learning curve of the SVC classifier is represented below:

```
In [76]: g = plot_learning_curve(svc.grid.best_estimator_,
                                 "SVC learning curves", X_train, Y_train, ylim = [1.01, 0.6],
                                 cv = 5,  train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5,
                                                         0.6, 0.7, 0.8, 0.9, 1])
```

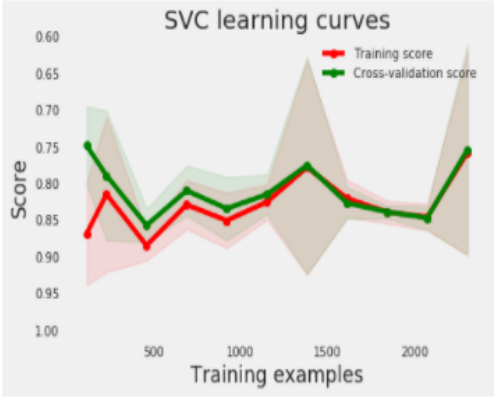

Figure 72 Generating the learning curve

It's apparent from this visual representation that, the CV curve converges at the same limit as the sample size is increased. This means that the classifier has low variance and not suffering from over-fitting. Variance is the value that indicates how much the target function will change if a different training dataset is provided. Ideally, the value of the target function is derived from the training dataset by Machine Learning Algorithm, however, the value of

the estimated function should not change too much if another training dataset is used. Minor change (minor variance) in the estimated function is expected. Here, the accuracy score has a low bias, which means the model is not facing the under-fitting issue either.

4.4 Summary

The baseline approach has few inherent problems and there's a need to optimize the current approach. The problems are as follows :

- The precision score is low and there's scope for improvement. A Voting mechanism could be built, should there be a need.
- There's a need to try other ML algorithms so that the results can be compared.
- In the revised approach, there's a need to try out various ML algorithms so that relevant algorithm could be selected.

CHAPTER V:

RESULTS AND EVALUATION

5.1 Introduction

In this section, all the problems will be taken into consideration and a new approach will be adopted to increase the accuracy of the classifier. As underscored in section 4.4, there's a need to implement other ML algorithms. In the revised approach, the following algorithms will be implemented:

- Logistic regression
- K-nearest neighbor
- Decision tree
- Random forest
- Adaboost classifier
- Gradient boosting classifier

In keeping with the precision score of all the preceding algorithms, a decision could be made on the algorithm best suited towards the fulfillment of the thesis goals.

5.2 Building the revised approach

In this section, various ML algorithms will be implemented, check their precision score, and monitor their learning curve. There is a total of six ML algorithms that will be used to identify which one is best suited for the current application.

To implement the ML algorithms: logistic regression, K-nearest neighbor, decision tree, random forest, Adaboost, and gradient descent, the helper class developed earlier will be used.

The following is the code snippet to this end:

```
In [77]: lr = Class_Fit(clf = linear_model.LogisticRegression)
         lr.grid_search(parameters = [{'C':np.logspace(-2,2,20)}], Kfold = 5)
         lr.grid_fit(X = X_train, Y = Y_train)
         lr.grid_predict(X_test, Y_test)

         Precision: 86.84 %
```

```
In [78]: g = plot_learning_curve(lr.grid.best_estimator_, "Logistic Regression learning curves", X_train, Y_train,
                                 ylim = [1.01, 0.7], cv = 5,
                                 train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```

Figure 73 Logistic Regression learning curves

```
In [79]: knn = Class_Fit(clf = neighbors.KNeighborsClassifier)
         knn.grid_search(parameters = [{'n_neighbors': np.arange(1,50,1)}], Kfold = 5)
         knn.grid_fit(X = X_train, Y = Y_train)
         knn.grid_predict(X_test, Y_test)

         Precision: 81.72 %
```

```
In [80]: g = plot_learning_curve(knn.grid.best_estimator_, "Nearest Neighbors learning curves", X_train, Y_train,
                                 ylim = [1.01, 0.7], cv = 5,
                                 train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```

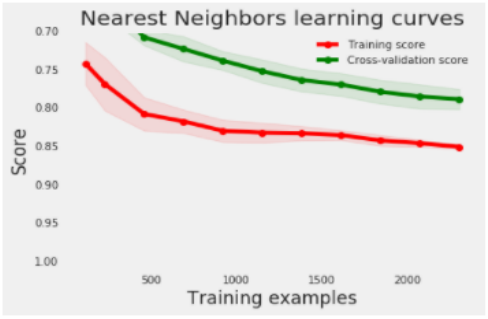

Figure 74 Nearest K-Neighbours learning curves

```
In [81]: tr = Class_Fit(clf = tree.DecisionTreeClassifier)
         tr.grid_search(parameters = [{'criterion' : ['entropy', 'gini'], 'max_features' :['sqrt', 'log2']}], Kfold = 5)
         tr.grid_fit(X = X_train, Y = Y_train)
         tr.grid_predict(X_test, Y_test)

         Precision: 85.73 %

In [82]: g = plot_learning_curve(tr.grid.best_estimator_, "Decision tree learning curves", X_train, Y_train,
                                 ylim = [1.01, 0.7], cv = 5,
                                 train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```

Figure 75 Decision Tree learning curves

```
In [83]: rf = Class_Fit(clf = ensemble.RandomForestClassifier)
         param_grid = {'criterion' : ['entropy', 'gini'], 'n_estimators' : [20, 40, 60, 80, 100],
                       'max_features' :['sqrt', 'log2']}
         rf.grid_search(parameters = param_grid, Kfold = 5)
         rf.grid_fit(X = X_train, Y = Y_train)
         rf.grid_predict(X_test, Y_test)

         Precision: 89.34 %

In [84]: g = plot_learning_curve(rf.grid.best_estimator_, "Random Forest learning curves", X_train, Y_train,
                                 ylim = [1.01, 0.7], cv = 5,
                                 train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```

Figure 76 Random Forest learning curves

```
In [85]: ada = Class_Fit(clf = AdaBoostClassifier)
         param_grid = {'n_estimators' : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]}
         ada.grid_search(parameters = param_grid, Kfold = 5)
         ada.grid_fit(X = X_train, Y = Y_train)
         ada.grid_predict(X_test, Y_test)

         Precision: 55.96 %

In [86]: g = plot_learning_curve(ada.grid.best_estimator_, "AdaBoost learning curves", X_train, Y_train,
                               ylim = [1.01, 0.4], cv = 5,
                               train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```

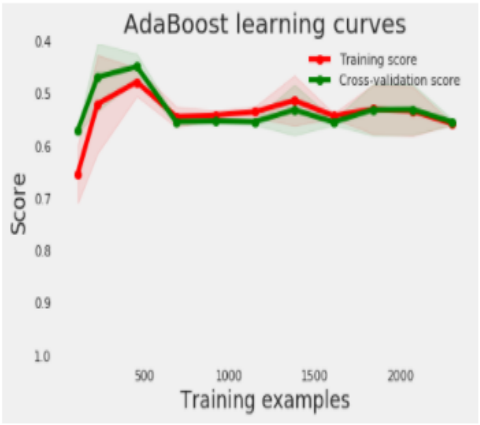

Figure 77 AdaBoost learning curves

```
In [87]: gb = Class_Fit(clf = ensemble.GradientBoostingClassifier)
         param_grid = {'n_estimators' : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]}
         gb.grid_search(parameters = param_grid, Kfold = 5)
         gb.grid_fit(X = X_train, Y = Y_train)
         gb.grid_predict(X_test, Y_test)

         Precision: 89.47 %

In [88]: g = plot_learning_curve(gb.grid.best_estimator_, "Gradient Boosting learning curves", X_train, Y_train,
                               ylim = [1.01, 0.7], cv = 5,
                               train_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```

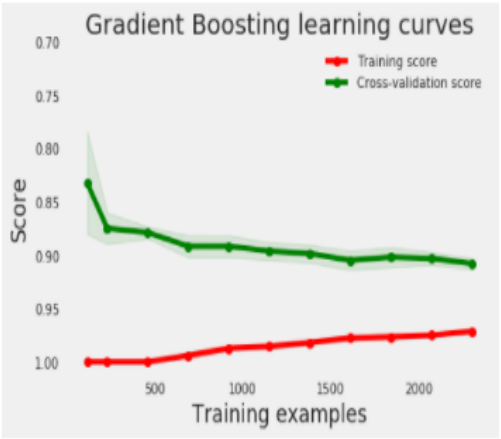

Figure 78 Gradient Boosting learning curves

It's apparent from the generated precision scores of the classifiers that, random forest and gradient-boosting classifiers have the greatest precision. However, the learning curves of

the classifiers underscore that all the classifiers are trained appropriately and there's no under-fitting / over-fitting issue, as the scores are improving with the increase in the data size.

5.3 Problems with the revised approach

The major problem with this approach is that a decision has to be made concerning the algorithm to be used for this application – To this end, the Adaboost classifier will be discarded as its precision score is too low.

Over and above, there's no single classifier that works well for all class labels. There may be a classifier that works well for class label 0, whereas another may work well for class label 8. I believe, no other classifier should be discarded and a voting mechanism should be developed instead. More precisely, there's a need to develop an ensemble model to improve the quality and accuracy of the prediction.

5.4 The Best approach

To improve the revised approach, a voting mechanism will be used using the voting classifier APIs of scikit-learn. To this end, the grid-searching will be used to generate appropriate hyperparameters for each classifier. Thereafter, voting-classifier APIs of scikit-learn will be used to train the model. The classifier model that will be generated in this approach should provide the best possible accuracy. The benefit of an ensemble classifier like a voting classifier is that no single classifier can perfectly classify all the samples and a combination of classifiers provide more accuracy because the problem with one classifier can be overcome by another classifier.

5.4.1 Implementing the best approach

The grid search is used to obtain the best possible hyperparameters, followed by usage of voting classifier API. The appended is the code snippet to this end:

```
In [89]: rf_best  = ensemble.RandomForestClassifier(**rf.grid.best_params_)
         gb_best  = ensemble.GradientBoostingClassifier(**gb.grid.best_params_)
         svc_best = svm.LinearSVC(**svc.grid.best_params_)
         tr_best  = tree.DecisionTreeClassifier(**tr.grid.best_params_)
         knn_best = neighbors.KNeighborsClassifier(**knn.grid.best_params_)
         lr_best  = linear_model.LogisticRegression(**lr.grid.best_params_)
```

Figure 79 Adjusting classifier parameter

```
In [90]: votingC = ensemble.VotingClassifier(estimators=[('rf', rf_best),('gb', gb_best),
                                                        ('knn', knn_best)], voting='soft')
```

Figure 80 Merging the results of various classifiers

```
In [91]: votingC = votingC.fit(X_train, Y_train)
```

Figure 81 Training the classifier

```
In [92]: predictions = votingC.predict(X_test)
         print("Precision: {:.2f} %".format(100*metrics.accuracy_score(Y_test, predictions)))

         Precision: 90.03 %
```

Figure 82 Creating prediction for the model

This approach provides 90% precision and we need to test this approach to hold out a corpus of two months to figure out the performance of the voting classifier on the unseen dataset.

5.4.2 Testing the best approach

The ML model is tested on 20% of the dataset, which is put aside before even starting the training. For training, 10 months dataset is considered and it's time to test the model on the hold-out corpus. The hold-out corpus consists of 2 months of data entries. The following are the steps in implementation:

- Transforming the hold-out corpus in the form of the training dataset
- Converting the transformed dataset into a matrix form
- Generating the predictions

In keeping with the above steps, there's a need to convert the data that resides in the set_test dataframe in the form of the training dataset. To achieve this, a copy of the new dataframe named basket_price will be stored. Finally, the user characteristics are generated with the help of the same operation performed for the baseline approach. The transformed dataset will be stored in the dataframe : transactions_per_user.

```
In [93]: basket_price = set_test.copy(deep = True)
```

```
In [94]: transactions_per_user=basket_price.groupby(by=['CustomerID'])['Basket Price'].agg(['count','min','max','mean','sum'])
         for i in range(5):
             col = 'categ_{}'.format(i)
             transactions_per_user.loc[:,col] = basket_price.groupby(by=['CustomerID'])[col].sum() /\
                                                transactions_per_user['sum']*100

         transactions_per_user.reset_index(drop = False, inplace = True)
         basket_price.groupby(by=['CustomerID'])['categ_0'].sum()

         #_____________________
         # Correcting time range
         transactions_per_user['count'] = 5 * transactions_per_user['count']
         transactions_per_user['sum']   = transactions_per_user['count'] * transactions_per_user['mean']

         transactions_per_user.sort_values('CustomerID', ascending = True)[:5]
```

Out[94]:

	CustomerID	count	min	max	mean	sum	categ_0	categ_1	categ_2	categ_3	categ_4
0	12347	10	224.82	1294.32	759.57	7595.70	25.053649	12.696657	32.343299	5.634767	24.271627
1	12349	5	1757.55	1757.55	1757.55	8787.75	52.138488	4.513101	12.245455	20.389178	10.713778
2	12352	5	311.73	311.73	311.73	1558.65	60.084047	6.672441	8.735123	17.290604	7.217785
3	12356	5	58.35	58.35	58.35	291.75	100.000000	0.000000	0.000000	0.000000	0.000000
4	12357	5	6207.67	6207.67	6207.67	31038.35	26.686341	5.089832	14.684737	25.189000	28.350089

Figure 83 Transforming the hold-out corpus

There's a need to convert the transformed dataset into the matrix format, as the classifiers take the matrix as an input. The following is the code snippet to this end:

```
In [95]: list_cols = ['count','min','max','mean','categ_0','categ_1','categ_2','categ_3','categ_4']
         #_____________________________________________________
         matrix_test = transactions_per_user[list_cols].as_matrix()
         scaled_test_matrix = scaler.transform(matrix_test)
```

Figure 84 Converting the transformed dataset

The precision score will be generated using the voting classifiers. The following code snippet helps generate the prediction for the test dataset:

```
In [99]: predictions = votingC.predict(X)
         print("Precision: {:.2f} % ".format(100*metrics.accuracy_score(Y, predictions)))

         Precision: 76.83 %
```

Figure 85 Generating predictions

5.5 Summary

Using the best approach method, an accuracy of 76% could be achieved on the hold-out corpus. This is an improvement over the baseline approach and it's worthwhile to note that, this is achieved using just 10 months of data to build this model. By using 10 months dataset, the best possible accuracy could be achieved for this domain. The results could be further improved if more data records are considered.

CHAPTER VI:

SUMMARY, IMPLICATIONS, AND RECOMMENDATIONS

6.1 Introduction

In this thesis, customer segmentation is developed in keeping with the behavior of the customers. To this end, various algorithms are used viz; SVM, linear regression, decision tree, random forest, gradient boosting, voting-based models. The best accuracy could be achieved by using the voting-based model.

6.2 Discussion & Conclusion

This thesis has twofold goals:

- To segregate the customers into distinct segments using RFM analysis
- To build a classifier to classify the customers into different customer segments established in the first goal and generate this classification result when the customer visits the online platform for the first time.

The first goal is achieved by creating clusters of customers using the K-means clustering algorithm and the contents of the clusters are further characterized using Silhouette intra-cluster-score analysis, Analysis using word cloud, Principal component analysis(PCA).In keeping with this EDA, the RFM behavior of the customers is derived for each cluster, using which items could be recommended and the marketing campaigns could be designed.

The second goal is achieved through the usage of supervised machine learning algorithms. A baseline approach to build the classifier using the SVM algorithm resulted in a low precision score. A revised approach is designed to improve the accuracy score by comparing the precision scores of the machine learning algorithms: Logistic regression, K-nearest neighbor, Decision tree, Random forest, Adaboost classifier, Gradient boosting classifier. Albeit, none of these classifiers have under-fitting or over-fitting issues, the major

problem in the revised approach is to decide which algorithm should be used and there's no single classifier that seems to work well for all class labels. This mandates usage of a voting classifier which is an ensemble machine learning model which helps to combine various classifiers to generate higher accuracy. The best approach emanated with the usage of the voting classifier which generated an accuracy of 76% using 10 months of data to build this model. The accuracy could be further improved by using more data records.

6.3 Contribution

An attempt is made in this research to evolve a methodology using an ensemble machine learning model to achieve a higher accuracy of the classifier, which is an improvement over the techniques underscored by the research papers cited in this report.

Customer segmentation achieved in this research will help small and midsized organizations to optimize their marketing strategy as well as significantly improve the customer acquisition cost. The research will be equally beneficial for companies providing financial, travel, telecom, marketing, educational, entertainment services to build customer segmentation using datasets for the specific domain. For instance, customer segmentation in the financial domain will consider the data points viz; the transaction history of the account holder concerning the frequency of using a debit card or credit card, per-month income, per-month expenditure, the average balance the customer is maintaining in their bank account(s), the type of account user have, professional information of the customer, and so on. Likewise, the customer segmentation in the travel domain will consider the data set about the frequency of flight booking, demographic/professional profile of customers, etc.

6.4 Future work

This thesis could further be extended in the following vistas:

6.4.1 Customer Lifetime Value Analysis

This metric is calculated by evaluating: Total Gross Revenue – Total Cost. This analysis underscores the impact the investments made on customers have on profitability. Investments like acquisition costs, offline ads, promotion discounts, etc, made in customers to generate revenue, make some customers valuable while there are customers who pull down the profitability of the company which mandates appropriate actions.

6.4.2 Customer Churn Analysis

Market fitment of a product is determined by the retention rate of the customers. If the product fitment in a market is low, then the company could experience a quick customer churn. Therefore, Churn Prediction is a technique to improve customer retention, using which a company can easily figure out who is likely to churn in a given period.

6.4.3 Predicting Sales

Knowing future Sales helps the company to achieve the following objectives:

- This could be used as a business at the usual level that a company can expect.
- This could be used by the company for planing the demand and supply actions.
- This could be used to plan the budgets and targets.

Deep learning techniques like LSTM or Time Series Forecasting methods like ARIMA / SARIMA / VAR could be used to build models to predict future sales.

6.4.4 Cross-Selling

Effective cross-selling for online retail customers can be achieved by analyzing the Market Baskets of online transaction data and predicting future purchases. To this end, products frequently purchased together are identified and recommendation systems are developed based on these findings, using Association Rules Mining methods. The whole concept of association rule-mining is based on the fact that customer purchase behavior has a pattern that can be exploited for selling more items to the customer in the future – A couple of algorithms that could be used for associative rule mining are FP Growth and Apriori.

REFERENCES

1. Ali, I. (2007). ‘Customer relationship management: a qualitative cross-case analysis in the UK and Saudi Arabia.’
2. Sheikha, A., Ghanbarpourb, T., & Gholamiangonabadib, D. (2019). ‘A Preliminary Study of Fintech Industry: A Two-Stage Clustering Analysis for Customer Segmentation in the B2B Setting.’ *Journal of Business-to-Business Marketing*.
3. Beheshtian-Ardakani, A., Fathian, M. & Gholamian, M. (2018). ‘A novel model for product bundling and direct marketing in e-commerce based on market segmentation.’ *Decision Science Letters*, 7(1), pp.39-54.
4. Caignya, A., Coussement, K. & Bock, W. (2018). ‘A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees.’ *European Journal of Operational Research*.
5. Atteberry, A., Loeb, S. and Wyckoff, J. (2017). ‘Teacher churning: Reassignment rates and implications for student achievement.’ *Educational Evaluation and Policy Analysis,* 39(1), pp.3-30.
6. Belhadj, T. (2021). ‘Customer Value Analysis Using Weighted RFM model: Empirical Case Study.’ *Al Bashaer Economic Journal*, 7(3).
7. Bhatnagar, A. & Ghose, S. (2004). ‘A latent class segmentation analysis of e-shoppers.’ *Journal of business research*, 57(7), pp.758-767.
8. Bimaruci, H., Hudaya, A. & Ali, H. (2020). ‘Model of consumer trust on travel agent online: analysis of perceived usefulness and security on re-purchase interests.’ *Dinasti International Journal of Economics, Finance & Accounting*, 1(1), pp.110-124.
9. Brito, P., Soares, C., Almeida, S., Monte, A. & Byvoet, M. (2015). ‘Customer segmentation in a large database of an online customized fashion business.’ *Robotics and Computer-Integrated Manufacturing*, 36, pp.93-100.

10. Callarisa Fiol, L., Alcaniz, E., Tena, M. and Garcia, J. (2009). 'Customer loyalty in clusters: perceived value and satisfaction as antecedents.' *Journal of Business-to-Business Marketing*, 16(3), pp.276-316.
11. Chaudhuri, N., Gupta, G., Vamsi, V. and Bose, I. (2021). 'On the platform but will they buy? Predicting customers' purchase behavior using deep learning.' *Decision Support Systems*, 149, p.113622.
12. Chen, Sain & Guo. (2012). 'A case study of customer segmentation using data mining techniques of the RFM model.' *Database Marketing & Customer Strategy Management.*
13. Chih-Fong, T., Ya-Han, H., and Yu-Hsin, L. (2015) 'Customer segmentation issues and strategies for an automobile dealership with two clustering techniques.' *ELSEVIER Expert Systems*.
14. Clow, K. (2012). 'Integrated advertising, promotion and marketing communications.' *Pearson Education India*.
15. Cusumano, M., Yoffie, D. & Gawer, A. (2020). 'The future of platforms.' *MIT Sloan Management Review*.
16. Eugene, W. & Yan, W. (2018). 'Customer online shopping experience data analytics.' *International Journal of Retail & Distribution Management*.
17. Farris, P., Bendle, N., Pfeifer, P. & Reibstein, D. (2010). 'Marketing metrics: The definitive guide to measuring marketing performance.' *Pearson Education.*
18. Grosfeld-Nir, A., Ronen, B. & Kozlovsky, N. (2007). 'The Pareto managerial principle: when does it apply?.' *International Journal of Production Research*, 45(10), pp.2317-2325.
19. Husnah, M. & Novita, R., 2022. 'Clustering of Customer Lifetime Value with Length Recency Frequency and Monetary Model Using Fuzzy C-Means Algorithm.' *2022 International Conference on Informatics Electrical and Electronics (ICIEE)*, pp. 1-4. IEEE.

20. Joy Christy, A. Umamakeswari, L. Priyatharsini, Neyaa. (2018) ‘"RFM ranking – An effective approach to customer segmentation.’ *Journal of King Saud University.*
21. Jun, S., Yuanyuan, L., & Guangshu. (2020) ‘An Empirical Study on Customer Segmentation by Purchase Behaviors Using an RFM Model and K-Means Algorithm.’ *ELSEVIER Expert Systems*.
22. Kim, S., Jung, T., Suh, E. & Hwang, H. (2006). ‘Customer segmentation and strategy development based on customer lifetime value: A case study.’ *Expert Systems With Applications*, 31(1), pp.101-107.
23. Kiseleva, E., Berkalov, S., Doroshenko, S., Khmelkova, N., Petrova, G., Krukova, E. & Karelina, A. (2017). ‘The importance of customers’ character accentuations.’ *The European Proceedings of Social & Behavioural Sciences (EpSBS).* Vol. 19.
24. Le, M. & Ha, K. (2016). ‘Attaining success through growth hacking.’ *MySQUAR*.
25. Maha, A., Mohamad, A. & Kadan, A. (2020). ‘A comparative dimensionality reduction study in telecom customer segmentation using deep learning and PCA.’ *Journal of Big Data*.
26. Makot, M., Yukihiro, M. & Wirawan, D. (2018) ‘Identifying valuable customer segments in online fashion markets - An implication for customer tier programs.’ *ELSEVIER Expert Systems.*
27. Nasır, S. (2017). ‘Customer retention strategies and customer loyalty in Advertising and Branding: Concepts, Methodologies, Tools, and Applications’ *IGI Global.* pp.1178-1201.
28. Rahim, M., Mushafiq, M., Khan, S. & Arain, Z.A. (2021). ‘RFM-based repurchase behavior for customer classification and segmentation.’ *Journal of Retailing and Consumer Services,* 61.
29. Rajagopal, D. (2011). ‘Customer data clustering using data mining technique.’ *arXiv.* preprint arXiv:1112.2663.

30. Revella, A. (2015). ‘Buyer personas: how to gain insight into your customer's expectations, align your marketing strategies, and win more business.’ *John Wiley & Sons.*
31. Rojlertjanya, P. (2019). ‘Customer segmentation based on the rfm analysis model using k-means clustering technique: a case of it solution and service provider in thailand.’
32. Kuruganti, S. & Basu, H. (2018). ‘2nd Edition Business Analytics – Applications to consumer marketing.’ *McGraw Hill.*
33. Sebastiaan, Eugen, Bart & Seppe. (2018). ‘Profit-driven decision trees for churn prediction.’ *ELSEVIER Expert Systems.*
34. Shohre, H., Neda, A. & Saeedeh, R. (2017). ‘Evaluating discounts as a dimension of customer.’ *Journal of Marketing Communications.*
35. Kumar, D. (2017). ‘1st Edition Business Analytics.’ *WILEY*.
36. Weinstein, A. (2013). ‘Handbook of market segmentation: Strategic targeting for business and technology firms.’ *Routledge.*
37. Wen-Yu, C. (2017). ‘Discovering customer value for marketing systems.’ *International Journal of Production Research.*
38. Xixi, H., Chen, L. (2016). ‘Application of Customer Segmentation on E-commerce Websites.’ *IEEE.*
39. Zhen, Y., Defu, X. & Stephen, T. (2014). ‘A decision-making framework for precision marketing*.’ ELSEVIER Expert Systems*.