Anticipating New customer's purchases MSC PROJECT REPORT

Student Name: Muhammad Bilal Azeem

Student ID: 19055756

Supervisor: Dr Thiago Matos Pinto

Module leader: Dr Helen Xiang

7COM1075: Data Science and Analytics Masters Project

MSc Data Science and Analytics with Advance Research

School of Physics, Engineering, and Computer Science

University of Hertfordshire

MSc Final Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in 7COM1075 Data Science and Analytics at the University of Hertfordshire (UH).

It is my own work except where indicated in the report.

I did not use human participants in my MSc Project.

I hereby permit the report to be made available on the university website, Provided the source is acknowledged

ABSTRACT

As more and more people are inclined towards purchasing from e-commerce stores, causing an increase in e-commerce platforms, it is becoming essential for eCommerce organisations to build up strategies to survive the competition. Effective Marketing plays a key role for companies in retaining customers and increasing overall sales volume. However, there's no one-size-fits-all regarding effective marketing, and different strategies will appeal to different customers. Companies segment customers into similar subsets based on their purchasing habits to efficiently develop effective marketing strategies that appeal to different types of customers to deal with this heterogeneity. However, since clustering algorithms tend to be non-parametric, they need to be re-run for every new customer unless a parametric model is developed to learn the mapping between customer purchasing habits and their respective clusters. This study aims to segment customers into clusters using non-parametric k-means clustering and train and validate multiple parametric machine learning models to predict the segments for future customers of an e-commerce company in Pakistan. Validation results show that the Random Forest classifier produced the highest weighted average precision, followed by gradient boosting and decision trees. An ensemble model based on the results of the three best models was also implemented; it was predicted with a precision of 98.75%, which is greater than any of the three classifiers and is very satisfactory. Since forward runs of a trained classifier are much faster than re-segmenting customers, this method also allows companies to reap the benefits of customer segmentation for future customers.

ACKNOWLEDGEMENT

First and foremost, I would like to thank Almighty Allah (SWT), the most beneficent and the most merciful and my humblest gratitude to his messenger Holy Prophet Muhammad (Peace be upon him).

Words cannot express my gratitude to Dr Thiago Matos Pinto, my project supervisor; his continuous guidance and support in this project are highly appreciated.

I want to give special gratitude to Dr Helen Xiang, this module's leader, who continuously encouraged and guided us with the sample reports and project ideas.

I would also like to thank Mr Sohaib Imran, Data Scientist and PhD Researcher at Lancaster University, who helped me organise my thoughts and ideas for this project and helped me streamline my code.

Last but not least, major thanks to my family, who have supported me in all the years of my study and this MSc. Their belief in me made me motivated during this journey.

Table of Contents

MSc Final Project Declaration i
ABSTRACT ii
ACKNOWLEDGEMENT iii
Table of Contents iv
1 INTRODUCTION 1
1.1 Background 1
1.2 Problem Statement 1
1.3 Research Aim 2
1.4 Objectives 2
1.5 Research Questions 2
1.6 Project Plan 2
1.7 Thesis Overview 3
1.7.1 Section 1: Introduction 3
1.7.2 Section 2: Literature Review 3
1.7.3 Section 3: Methods 3
1.7.4 Section 4: Methodology 3
1.7.5 Section 5: Results and Discussion 3
1.7.6 Section 6: Conclusion and Recommendations 3
2 LITERATURE REVIEW 4
2.1 Introduction to Data Science 4
2.2 Introduction to Ecommerce 4
2.3 Introduction to Customer to Segmentation 5
2.4 Related work on Customer Segmentation 6
3 Methods 8
3.1 Data Cleaning 8
3.2 Principal Component Analysis 8
3.3 Silhouette score 10
3.4 K-Means clustering 10
3.5 Support Vector Machine 12
3.6 Confusion Matrix 13
3.7 Learning Curves 14
3.8 Logistic Regression 15
3.9 K-nearest neighbour 16
3.10 Decision Trees 17

3.10.1 Gini 18
3.10.2 Entropy 18
3.11 Random Forest 19
3.12 AdaBoost Classifier 20
3.13 Gradient boosting classifier 22
3.14 Voting Classifier 22
4 Methodology 24
4.1 Business Understanding 24
4.2 Data Understanding 24
4.3 Data Preparation 25
4.3.1 Data Setup 25
4.3.2 Data Cleaning 25
4.3.3 Data Insights 25
4.4 Modelling 26
4.4.1 Defining Customer Categories 26
4.4.2 Classification of Customers 26
4.5 Evaluation 27
5 Results and Discussion 28
5.1 Koe heading ..... **Error! Bookmark not defined.**
6 CONCLUSION AND FUTURE RECOMMENDATIONS 31
REFERENCES 32
APPENDIX - A I
7 Appendix B: Code XXX

Table of Figures

Figure 1 Data Cleaning steps 8
Figure 2 Silhouette Score cluster separation Source: (Ashutosh Bhardwaj, 2020) 10
Figure 3 K means Source: (Zhang et al., 2017) 11
Figure 4 Non-Linear SVM process Source: (Khan and Jain, 2016) 12
Figure 5 Linear SVM process, a line dividing support vectors into two different classes Source: (Atul Agarwal, 2019) 12
Figure 6 Choosing the best plane for classification Source: (Neha, Neha K.S and Prof Santhosh Rebello, 2016) 12
Figure 7 SVM classifier hyperplanes summary Source: (Neha, Neha K.S and Prof Santhosh Rebello, 2016) 12
Figure 8 Simple image of the confusion matrix Source: (Eugenia Anello, 2021) 13
Figure 9 Linear Regression vs Logistic Regression Source: (Maithili Joshi, 2019) 16
Figure 10 Logistic Regression graph Source: (Ayush Pant, 2019) 16
Figure 11 Logistic Regression graph Source: (Ayush Pant, 2019) 16
Figure 12 knn Image showing how similar data points typically exist close to each other Source: (Onel Harrison, 2018b) 17
Figure 13 Decision Tree Source: (Nagesh Singh Chauhan, 2022) 18
Figure 14 Random Forest Classifier general figure Source: (Karan Kashyap, 2019) 19
Figure 15 Training and classification phases of Random Forest classifier Source: (Belgiu and Drăguţ, 2016) 20
Figure 16 Adaboost Classifier Source: (Anshul Saini, 2021b) 21
Figure 17 Gradient Boosting Classifier Source: (Anjani Kumar, 2020) 22
Figure 18 Phases of the CRISP-DM reference model Source: (Niaksu, 2015) 24
Figure 19 Output result of all classifiers 28
Figure 20 Output of voting classifiers 28
Figure 21 Output of classifiers used on the testing dataset 29
Figure 22 Output of voting classifier 29
Figure 23 Data Cleaning detailed structure I
Figure 24 Output: Comparison of customers with single order vs multiple orders II
Figure 25 Output: Comparison of customers who bought from one category vs from multiple categories II
Figure 26 Output: total order cancelled vs customers whose orders got cancelled III
Figure 27 Output: Categories of cancelled orders IV
Figure 28 output: Total completed and pending orders V
Figure 29 Output: Category name vs encoded number V
Figure 30 Output: Elbow Method to find the optimal number of clusters VI
Figure 31 Output: Silhouette Score analysis VI
Figure 32 Output: Silhouette Score for n number of clusters VII
Figure 33 Output: Silhouette Intra cluster Score VIII
Figure 34 Output of clusters vs number of customers in each cluster IX
Figure 35 Output: PCA to find a variance IX
Figure 36 Output: PCA performed on six components X
Figure 37 output clusters XI
Figure 38 Output Cluster radar chart XIII
Figure 39 K-Fold structure Source: (Pedregosa et al., 2011) XIII
Figure 40 Output: SVC Confusion Matrix XIV

Figure 41 Output: SVC Learning Curves XV
Figure 42 Output: Logistic Regression Confusion Matrix XVI
Figure 43 Output: Logistic Regression Learning Curves XVII
Figure 44 Output: Nearest Neighbour Confusion Matrix XVIII
Figure 45 Output: Nearest Neighbour Learning Curves XIX
Figure 46 Output: Decision Tree Confusion Matrix XX
Figure 47 Output: Decision Tree Learning Curves XXI
Figure 48 Output: Random Forest Confusion Matrix XXII
Figure 49 Output: Random Forest Learning Curves XXIII
Figure 50 Output: Ada Boost Confusion Matrix XXIV
Figure 51 Output: Ada Boost Learning Curve XXV
Figure 52 Output: Gradient Boosting Confusion Matrix XXVI
Figure 53 Output: Gradient Boosting Learning Curve XXVII
Figure 54 Output: Voting Classifier Confusion Matrix XXVIII
Figure 55 Output: Voting Classifier Learning Curve XXIX
Figure 56 Number of orders in each status XXIX
Figure 57 Orders with negative grand total XXX
Figure 58 Missing Values in each column XXX

1 INTRODUCTION

1.1 Background

According to Statista, Retail e-commerce sales amounted to around £3.9 trillion in 2021, forecasted that in the next four years it will grow by 50 per cent, which will be around £6 trillion. This shows how much e-commerce is on the boom these years and becoming an indispensable part of the retail industry (Daniela Coppola, 2022).

The number of online buyers is climbing each year, so E-commerce is continuously evolving. Businesses are trying hard to improve their product quality, design, and rates than their competitors to attract more customers. Companies are analysing and predicting customer activities to improve their marketing strategies. Overall, business strategies are customer-centric to find and target potential customers.

The main challenge businesses face is, failing to connect with customers due to a large and diverse customer base. They must play the divide and rule policy to break customers into groups with distinct needs and characteristics. Every business-related handbook explains and justifies segmentation as a great marketing strategy (Smith, 1956; Claycamp and Massy, 1968; Moorthy, 1984). Customer segmentation means dividing the existing customers into groups based on similar characteristics. According to the Pareto effect described by (Xixi He and Li, 2016), even if the number of customers is in the minority, it can bring huge profits for the companies. Using customer segmentation, companies can differentiate the value and needs of different customer types; this will help companies recommend appropriate products to their customers.

According to (Thomas, 2007), the main purpose of segmentation is to concentrate marketing strategy and force on a particular segment or subset of customers to achieve a competitive advantage in that segment.

According to (Smeureanu, Ruxanda and Badea, 2013), good customer segmentation will result in profitable business development strategies. It helps businesses identify particular characteristics of their products and services in demand, making business plans efficient and economically viable.

1.2 Problem Statement

Every customer has a different character and needs, and an E-commerce website needs to know its customers' behaviour and needs. Having the same strategies, marketing, and promotions for every customer is not optimal.

E-commerce companies need to segment their customers and find their differences. Companies need to know what type of customer they are dealing with, how many are their loyal customers, jackpot customers, how many have stopped doing business with them and how their customers have changed in the past few years.

Once existing customers have been divided into different segments, a method is needed to map new customers to these segments to reap the benefits of segmentation for new customers without having to repeat the segmentation process.

This project will make it easier for companies to meet customers' needs, take care of important ones and convince those they are losing. The project will also help companies better understand customer purchasing patterns to recommend another viable purchasable product with the product the customer is buying.

1.3 Research Aim

This project aims to create a tool to analyse customer behaviour and segment those customers based on relevant characteristics such as purchasing habits. It also aims to create a model to map new customers to one of these segments. Businesses will benefit from this tool by analysing current and new customers' purchase behaviour and predicting their next purchases.

1.4 Objectives

The objective of this project is to determine whether it is possible to predict a segment where new customers should belong.

Segment customers based on their purchasing habits via k-means clustering and evaluate the silhouette score of these clusters over a different number of clusters.

Train various machine learning models to map customers to their corresponding k-means clusters and evaluate relevant metrics on the quality of these models.

1.5 Research Questions

This project aims to answer the following research questions:

How can distinct clusters be obtained by segmenting customers based on their purchasing habits via k-means clustering, and how does the quality of these clusters vary as the number of clusters is increased?

To what precision and recall, on average, can machine learning models be trained to map customers to their corresponding k-means clusters?

1.6 Project Plan

S.No	Task Name	Task Details	Completion Date / tentative date
1	Literature Review	Researched the previous and related work and found gaps in required work	02/06/2022
2	Project Proposal	Created a proposal for my project Submitted to Dr Thiago and got his approval on email	18/05/2022
3	Development of Aims and Objectives	Analysed the gap found in the literature review and defined the aims and objectives of the project	17/06/2022
4	Found and Build Data Set	Found the data set relevant to my project and modified it as per the needs	05/06/2022
5	Pre-processing of data	Cleaned data, performed feature extraction	21/06/2022
6	Explore Data Set	relationship or dominance of product/city Used map and graph to demonstrate what data looks like	05/07/2022

		Created charts to define categories and the number of orders in each	
		Analysed the data	
7	Customer Segmentation	Created customer categories	20/07/2022
		classify customers	
		Performed customer segmentation	
		Finalised the coding	
		Tested the code	
8	Obtaining and interpreting the result		25/07/2022
9	Wrote the dissertation	Completed and finalised the project report	03/08/2022
		Completed formatting of the report	
		Performed grammar and spelling check	
		Inserted and updated the bibliography	
10	Submitted the final Project Report	Submitted the final Project Report	04/08/2022

1.7 Thesis Overview

The thesis structure is as follows:

1.7.1 Section 1: Introduction

This section has covered the background on the topic. This section also covers this project's aims, objectives and the timeline and plan followed while doing this project. The problem statement is defined in this section, and research questions that have centred this thesis are also mentioned.

1.7.2 Section 2: Literature Review

In this section, theories on data science and e-commerce are discussed and have also covered the previous and related work about customer segmentation and predicting customer behaviour.

1.7.3 Section 3: Methods

This section consists of the definition of tools used in this project.

1.7.4 Section 4: Methodology

This section consists of steps performed in this project like data cleaning, implementation of Machine learning techniques and further. A detailed description of the approaches is also provided.

1.7.5 Section 5: Results and Discussion

This section record and interpret the obtained results.

1.7.6 Section 6: Conclusion and Recommendations

This section covers the summary of the whole report, the conclusion obtained from the project and the future scope of this project.

2 LITERATURE REVIEW

2.1 Introduction to Data Science

This era of data science, data analytics and big data; has created an enormous hype and buzz in the industry, bringing innovative and economic opportunities (LONGBING CAO, 2017). (Douglas Laney, 2001; McKinsey, 2011; CSC, 2012; Dan Vesset *et al.*, 2012; Mark A. Beyer and Douglas Laney, 2012) identifies this era as "big data growth", while (Tony Hey and Anne Trefethen, 2003) identify this age as a "Data Deluge". Initially, Data Science was hyped by private data-oriented companies but now has expanded to governmental organisations and academic institutions.

According to (Chen et al., 2012; Chris Yiu, 2012; Khan et al., 2014; Labrinidis & Jagadish, 2012), the recognition of challenges, opportunities and value of data science and related technologies are reshaping the scientific and engineering fields, social science, business and management. This reshaping and shifting of paradigm are not only because of data but also of all the uses of data by understanding, exploring, and utilising it.

(Nielsen, 2017) believes that data is the most valuable asset for the organisation, and authors think that it can be only truly a unique asset in future. Organisations currently have huge data from multiple sources; if this data is processed reliably and quickly, it will increase the chances of extracting actionable insights to make data-driven decisions (H. Chen, Chiang and Storey, 2012; Waller and Fawcett, 2013). (Chin *et al.*, 2017) added that researchers and professionals think data-driven decision-making is a powerful way to increase a business's value. (Roy and Seitz, 2018) an interview says that the data first mentality is indeed a vital step, but businesses have to have processes and capabilities to use that data. (Brynjolfsson, Hitt and Kim, 2011) comments that organisation that uses Big data and Data science technologies can achieve 5-6 per cent higher productivity. (Müller, Fay and vom Brocke, 2018) have analysed multiple companies' data sets and have concluded that investment in data science has resulted in the improvement of business productivity.

2.2 Introduction to Ecommerce

Since the beginning of the internet, the options for buying and selling goods and/or services have changed. Businesses used to buy and sell goods through physical stores and now can do from online virtual stores as well (van den Poel and Buckinx, 2005). It doesn't end with just adding a new channel to e-commerce; businesses use data to improve their businesses. It helps eCommerce companies manage their inventory effectively and help them forecast future requirements. Tools like sentiment analysis help them understand the customer experience, and Machine learning helps them to make better IVR and chatbots which can resolve customer issues timely and effectively.

Ecommerce is a form of doing business by using computers and the internet. (Vladimir, 1996) comments that E-commerce helps businesses store transactions and dealings with the help of telecommunication networks. According to (Treese and Stewart, 2003), e-commerce is a worldwide phenomenon for ordering and selling products and services using the internet. A range of factors influences this phenomenon of online shopping, like website atmosphere, availability of information, and convenience of transactions (Morganosky and Cude, 2000; Wolfinbarger and Gilly, 2003; Prashar, Vijay and Parsad, 2015). The tendency to shop online as compared to physical stores gets recognised because of its convenience. (Donthu and Garcia, 1999). (Gehrt, Yale and Lawson, 1996; Morganosky and Cude, 2000; Constantinides, 2004) defined convenience as fast and simple information browsing, shopping and completing the transaction, while (Eastlick and Feinberg, 1999; Jiang, Yang and Jun, 2013) say that shopping from home is the major convenience of E-commerce.

(Kim and Stoel, 2004) stated that the customers are not satisfied with just the website's usability; instead, the content on a website and the quality of their money transactions in purchasing goods have a major impact on them. However (Aren *et al.*, 2013; Gao and Bai, 2014) argues that for a customer, repurchasing intention from the same online store is positively impacted by the usefulness and ease of use of the website.

2.3 Introduction to Customer to Segmentation

Customer Segmentation divides the customers into smaller, identical, distinct and specific groups based on their relevant characteristics and purchasing habits. It is a very popular marketing activity, and many companies use it to understand better their customer's needs and characteristics (Peker, Kocyigit and Eren, 2017). Customer Segmentation is being used in multiple industries like retail, as discussed by (Peker, Kocyigit and Eren, 2017; Dogan, Ayçin and Bulut, 2018; Haghighatnia, Abdolvand and Rajaee Harandi, 2018; Yoseph and Heikkila, 2018), electronic businesses (Daoud *et al.*, 2015; X He and Li, 2016; Beheshtian-Ardakani, Fathian and Gholamian, 2018; Li and Li, 2018; Pondel and Korczak, 2018), wholesale industry (Rezaeinia and Rahmani, 2016), SMEs: Small and Medium Enterprise (Marisa *et al.*, 2019), finance (Hamdi and Zamiri, 2016; Moeini and Alizadeh, 2016; Patel, Agrawal and Josyula, 2016; Aryuni, Madyatmadja and Miranda, 2018; Ravasan and Mansouri, 2018; Sheikh, Ghanbarpour and Gholamiangonabadi, 2019), health care (Hosseini and Mohammadzadeh, 2016; Tarokh and EsmaeiliGookeh, 2019) and Information Technology (Akhondzadeh-Noughabi and Albadvi, 2015; Panuš *et al.*, 2016; Safari, Safari and Montazer, 2016; Chen, Zhang and Zhao, 2017). It is also applied to non-commercial institutions, as discussed by (Weng, 2016).

(Peacock, 1998) comments that Retailers and direct vendors use the Market base analysis to find the natural liking of customers for products. They can focus their promotional strategies using the position segmentation technique, which is the relationship between customers and the product they purchase at that point of sale.

(Paul *et al.*, 2010) says that although customer segmentation is measured on the individual level, it is always reported on an aggregate level. Customer satisfaction levels provide important indicators of their purchasing purposes and loyalty to the brand.

According to (http://www.celerant.com/, no date), mining or analysing datasets enables understanding of the market trends and helps in market analysis and price optimisation. (Berson and Thearling, 1999) discussed how the eCommerce website data can be transformed into knowledgeable facts that can improve the business. This can build the relationship between the customer and business as the business will be able to fulfil customers' needs.

According to (Kim, 2000), three things increase customer value; up-selling, cross-selling, and customer retention. The author describes customer retention as an effort to keep customers at the business and stop them from changing their minds. (Massnick, 1997) stated that the cost of gaining new customers is five times more than keeping existing customers, and it is ten times more to get a dissatisfied customer back. A study by (Reichheld and Teal, 1996) states that a five-point increase in customer retention can increase profits by more than 25 per cent. Businesses want to determine customers' wishes to plan their marketing strategies (Baer, no date). Communications are built according to the characteristics of the customers, and it cannot be easy on personal approaches, so if divided into groups with the same characteristics, it would be much easier (Schneider, 2016). (*Magento. An Introduction to Customer Segmentation*, 2014) describes the benefits of using customer segmentation on how it enables them to match with the customers to offer similar products. It changed their communication with the customer using customer data, helped them identify the most profitable customers, and helped them update their products to meet customer needs; according to (Baer, no date; Collica, 2017, customer segmentation can categorise items or subjects into groups with a commonality.

(Steenkamp and ter Hofstede, 2002) highlights the main objective of customer segmentation, which is to develop a bucket of customers based on similarities; this can be done with the help of multiple parameters or attributes related to business, for example, market, psychographics, region or lifestyle. They discussed that it would help the businesses make advertisements or do marketing, optimise sales or retain customers.

(Hawkins, Best and Coney, 1998) states that the customer's value classifies customer segmentation, demands, preference and factors the organisation decides. So customers in the same group will have certain similarities, but different segmented groups will have distinct characteristics.

2.4 Related work on Customer Segmentation

(Hwang, Jung and Suh, 2004) suggested a new lifetime value model (LTV), and they did the customer segmentation with customer defection and cross-selling opportunities. Customer values are valued from three perspectives; current value, potential value and customer loyalty. This assists in identifying customer segmentation with balanced opinions.

(Marcus, 1998) introduced an approach for customer segmentation and customer value matrix. The technique identified key customer segments and recommended marketing strategies and tactics.

(Deng and Gao, 2020) proposed new features in customer segmentation in e-commerce; they combined the traditional K-Means algorithm with the semi-supervised AP algorithm naming it the SAPK+K-means algorithm. The output had a low error rate in obtaining clusters from the two data sets, but the acquisition time was long; they defended it by considering it a guarantee of the proposed algorithm's high validity and accuracy.

(X He and Li, 2016) proposed a three-dimensional model that combined three variables to divide customer groups, which improves extensive customer segmentation and makes it more accurate according to their study. It will provide a targeted marketing strategy and differentiated service for all types of customer groups to retain customers.

(Ballestar, Grau-Carles and Sainz, 2018) The cashback website's customer segmentation is based on customers' commercial activity and role within the website's social network. The concept of loyalty, social networks, and customer engagement and evolution was applied to display the customer role depends on customer positioning in the network. The study showed that the more engaged customer is, the more they are transactional.

(Punhani *et al.*, 2021) in their paper analysed the database based on the parameters like date, customer ide, product category, the value of the item and payment method, time spent by the customer on-site and the clicks done by them. The authors used data mining techniques to find hidden patterns and specific behaviours in the data set and then applied the K-Mean clustering algorithm to analyse the products with the highest sale and which payment method is highly used.

Few other researchers have also done customer segmentation using clustering methods and compared the outcomes, like (Monil *et al.*, 2020) used and compared K-Means Clustering, Affinity Propagation Algorithm, Density-Based Clustering and Hierarchical Clustering. (Kansal *et al.*, 2018) used and compared k-Means, Agglomerative, and Meanshift clustering algorithms. (Ezenkwu, Ozuomba and Kalu, 2015) just used K-means, but on MATLAB, (Kashwan and Velu, 2013) also used just K-Means for segmentation, but they did the stability check on their clusters using analysis of

Variance (ANOVA) test.
(Ozan, 2018) uses the Normal Equation Method, Multivariate Linear Regression Method, and Logistic Regression Method for customer segmentation. They all were implemented on the existed manually segmented data to check their efficiency and processing time. (Smeureanu, Ruxanda and Badea, 2013) have used Neural Networks and Support Vector Machines for customer segmentation, tuned multiple parameters, shared the performances and discussed the best outcomes. They all performed the customer segmentation, which can help companies in the strategic planning but they are all limited on the customers existed at the moment of data capture, for future customer companies had to do the analysis again.

(Dullaghan and Rozaki, 2017) did customer segmentation using C.5 algorithm, using naïve Bayesian modelling, it was for telecommunication business, so customer's profile behaviour like billing and socio-demographic were considered. In the study, they only used one algorithm, which did show approvable outcomes on their current dataset, but in future, there is a possibility that this algorithm won't be the most effective.

3 Methods

3.1 Data Cleaning

Many businesses are storing and acquiring massive amounts of data. These data sets facilitate improved decision-making and richer analytics and increasingly provide training data for Machine learning. However, data quality is one issue, as dirty data leads to incorrect decisions and unreliable analysis (Chu *et al.*, 2016). (Rahm and Do, 2000; Johnson and Dasu, 2003) suggests that analysts should consider the effects of dirty data before making any decisions from that data. Figure 22 describes this in full detail.

Steps to clean the data by (Rahm and Do, 2000) have been summarised in the figure below (figure 1):

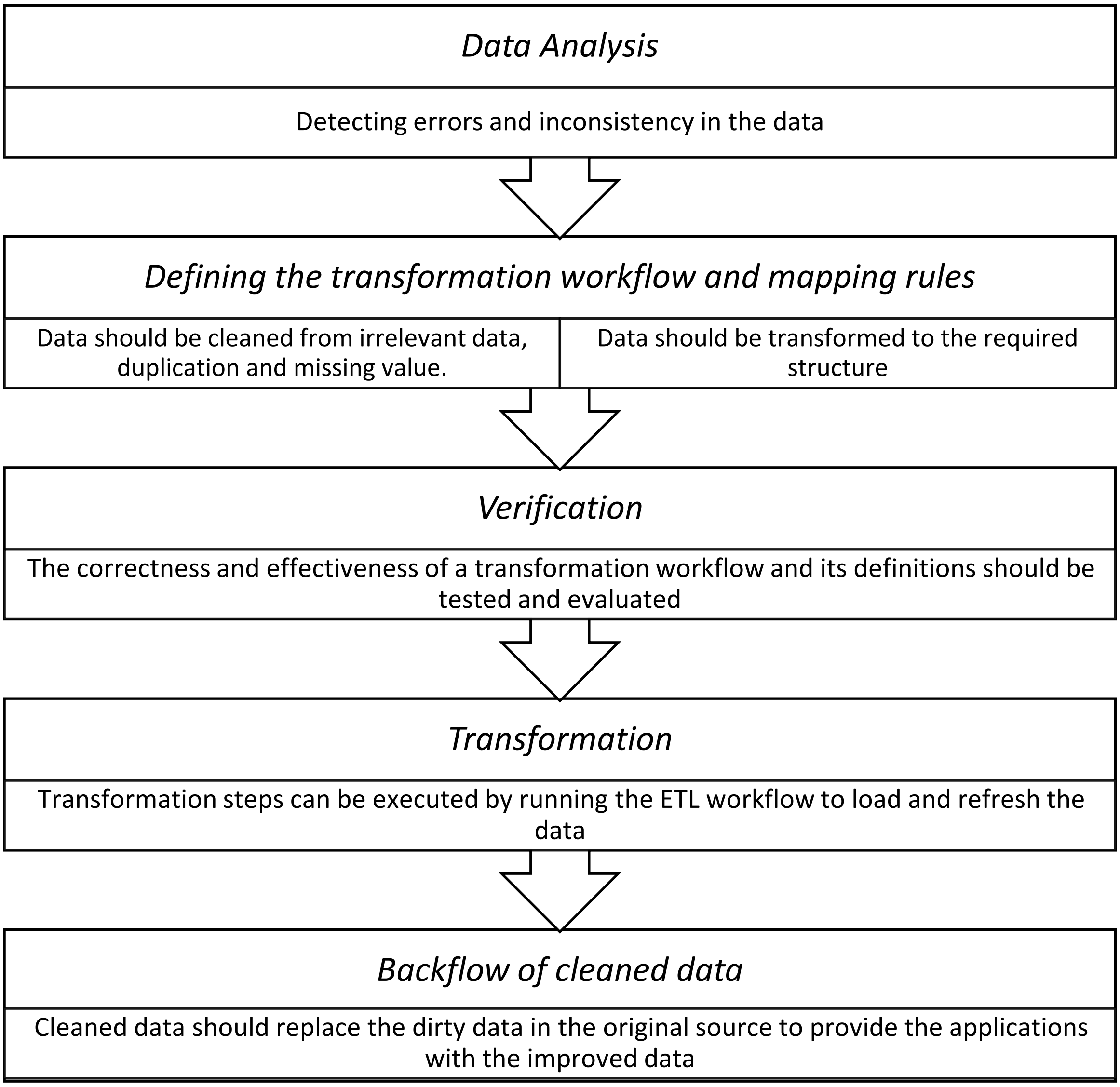

Figure 1 Data Cleaning steps

3.2 Principal Component Analysis

Several measurement techniques gather data for many more variables. The high dimensionality of data makes its visualisation difficult and limits data exploration. Principal component analysis, or

PCA, is a mathematical algorithm that reduces the dimension of data but retains most of the variations in the dataset (Jolliffe, 2002). The reduction is achieved by identifying directions known as principal components, along which the variation in the data is maximal. Using a few components, every sample represents fewer numbers instead of thousands of variables. Then they can be plotted, making it possible to visually analyse and assess differences and similarities between samples and determine if the samples can be grouped. (Zakaria Jaadi, 2022) has broken down PCA in five steps:

1. Standardisation: Some variables have large ranges while others have small to avoid large ranges dominating, so data is transformed to a comparable scale to prevent this problem. Formula for standardisation $z = \frac{value = mean}{standard\ deviation}$
2. Covariance matrix computation: To understand how variables of a data set vary from the mean and to find a relationship between them, a covariance matrix is made; it is also helpful to identify correlations to avoid redundancy. The covariance matrix is a n x n symmetric matrix; n represents several dimensions. Below is the example of 3-dimensional where variables are $x, y\ and\ z$.

$$\begin{matrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{matrix}$$

   Since the covariance of the variable with itself is its variance $Cov(x,x) = Var(x)$, all diagonals will have a variance of each variable. The covariance matrix is commutative, so $Cov(x,y) = Cov(y,x)$ So entries would be symmetric from the main diagonal, so upper and lower triangular from the diagonal are the same.

   If the sign of covariance is

   - Positive: both variables increase or decrease together and so are correlated
   - Negative: one increases and the other decreases and so inversely correlated

3. Computing Eigenvalues and Eigenvectors: Eigenvectors and Eigenvalues are always paired with each other, so every eigenvector has an eigenvalue, and they are equal to several dimensions of the data. The covariance matrix's Eigenvectors are the axes' directions with the most information or variance, so we call them principal components. Eigenvalues are the amount of variance carried in each principal component. The rank of eigenvectors defines the order of significance of principal components in their eigenvalues from highest to lowest.
   Principal Components: They are the new variables constructed as linear combinations of initial variables. The combinations to make Principal combinations make the uncorrelated, and most information from initial variables is compressed into first components. So even if the data is 18-dimensional, PCA puts the maximum information in the first component, the second has the remaining, the third has the remaining left from the first two, and so on. This can be seen in the scree plot below (figure 34)
4. Feature Vector: According to the significance of eigenvalues, either they all can be selected, or a few can be discarded. The final one's eigenvector is put as a column of a vector. This is known as a feature vector.
5. Analysis: In this step, a feature vector is used to reorient the data from the original axes to the axes represented by the principal components. It is done by multiplying the original dataset's transpose by the feature vector's transpose.

$$Final\ Dataset = Feature\ Vector^{T} \times Standarised\ Original\ Dataset^{T}$$

3.3 Silhouette score

A major problem in the clustering algorithm like k-means used in this project is finding an optimal number of clusters (Forgy, 1965). One value of k (number of clusters) cannot be used for every dataset; it depends on the methods used for measuring similarities, initial seed values in the partitioning and more. One solution that can be used is to develop a dendrogram from the hierarchical clustering, but it is expensive, subjective, and slower than k-means.

The more direct method of finding an optimal number of clusters is the silhouette scores, which measure clusters' quality(Rousseeuw, 1987). The higher the Silhouette coefficient value, the better the clustering, which helps decide the optimal value of the k (number of clusters) (Kassambara, 2017). (Ogbuabor and Ugwoke, 2018) added that Silhouette score values only lie between -1 and +1. If the value is near -1, it shows that objects are not clustered properly; if it is near +1, it proves the clustering is correct.

Score $s_{(i)} = \frac{b_{(i)}-a_{(i)}}{max\{b_{(i)},a_{(i)}\}}$ Where a is an average intra-cluster distance (distance between each point within a cluster) and b is the average inter-cluster distance (distance between all clusters), as seen in figure 2.

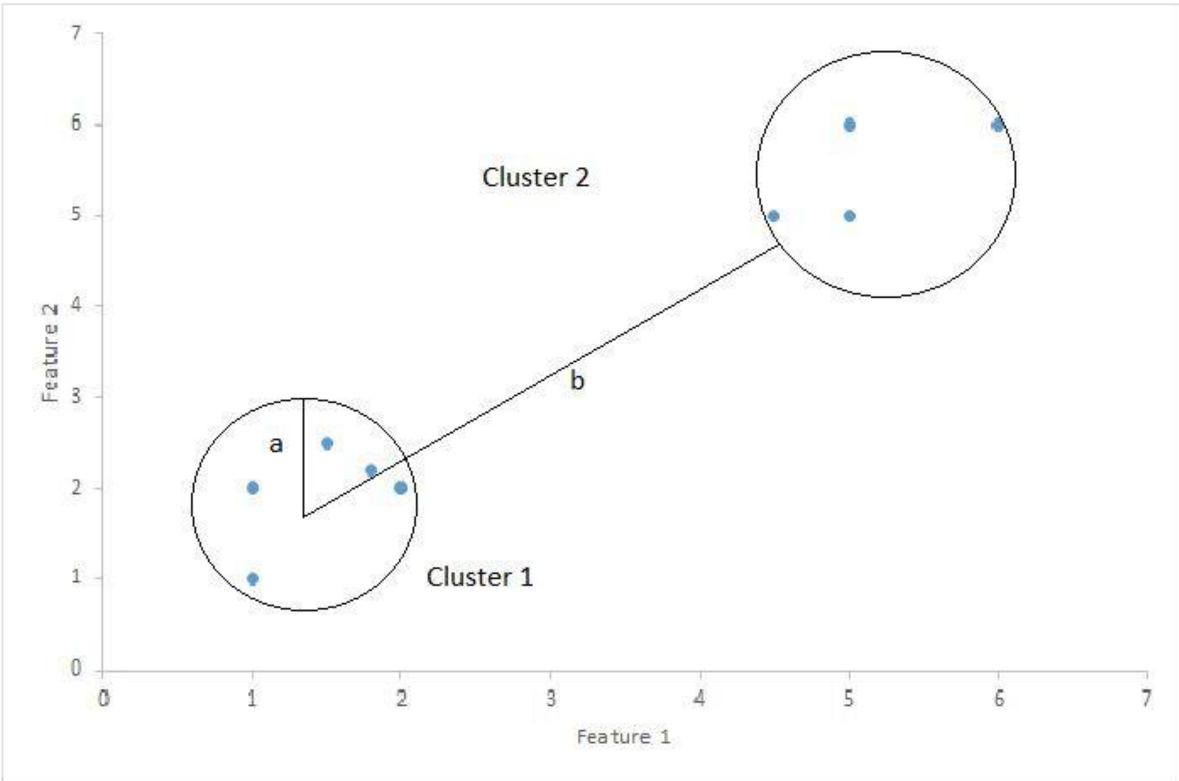

Figure 2 Silhouette Score cluster separation
Source: (Ashutosh Bhardwaj, 2020)

3.4 K-Means clustering

Clustering finds homogeneous groups of data points from the dataset. Each of these groups is called a cluster. These clusters can be known as a region where the density of objects is locally higher than in other regions. Clustering aims at portioning a dataset into clusters or disjoint subsets so that that specific clustering criterion can be optimised. The most popular clustering method is the k-means algorithm which minimises the clustering error (Likas, Vlassis and Verbeek, 2003). (Andrey Bulezyuk, 2017) describes K-means goal as to group the data points of the same characteristics together and discover the models. It uses a fixed number of clusters known as k. K is the number of centroids, so the K-means algorithm allocates every data point to its nearest clusters while keeping the centroids as small as possible. Figure 3 below explains how the clusters are separated in K-Means.

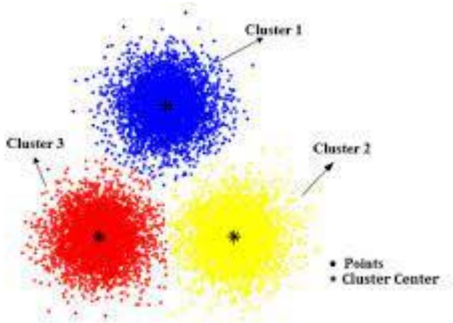

Figure 3 K means
Source: (Zhang et al.*, 2017)*

3.5 Support Vector Machine

A support vector machine (SVM) is a computational algorithm that learns by example to assign labels to objects (Boser, Guyon and Vapnik, 1992). SVM provides a classification learning model and an algorithm instead of a regression model or an algorithm (Hearst *et al.*, 1998). (Neha, Neha K.S and Prof Santhosh Rebello, 2016) states that a hyperplane is constructed in SVM to separate the two classes, and the algorithm tries to attain the maximum separation between the classes, as shown in Figure 6. The large margin of separation reduces the generalised error, so the best classifier would be the one that will achieve maximum separation between the classes. Bounding planes are drawn parallel to the classifier and pass through one or more than one point in the dataset. Margin is the distance between the planes, and finding the hyper-central plane to maximise the margin is defined as SVM learning; this information is summarised in figure 7.

SVM used a mathematical model $y = \omega^T x + b$ where w = vectors, b= biased term (w_0) and x are variables. This equation is manipulated to allow linear domain divisions. According to (Hastie et al., 2009), SVM can be divided into two models; linear and non-linear. It is linear if the data domain is divided linearly; by a hyperplane or straight line, as displayed below in figure 5. If the data domain cant is divided linearly but can be transformed to space (feature space) where it can be divided linearly to separate the classes, it is known as a non-linear support vector machine, as displayed in figure 4.

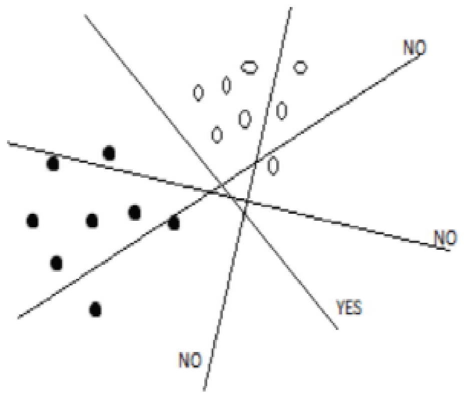

Figure 6 Choosing the best plane for classification
Source: (Neha, Neha K.S and Prof Santhosh Rebello, 2016)

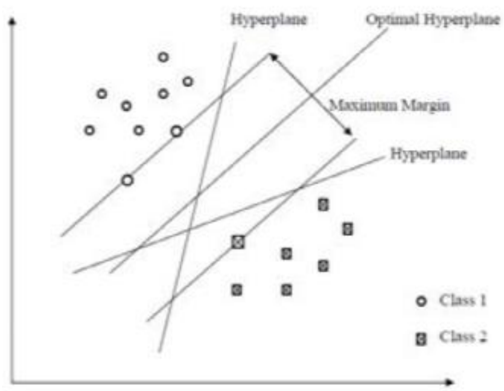

Figure 7 SVM classifier hyperplanes summary
Source: (Neha, Neha K.S and Prof Santhosh Rebello, 2016)

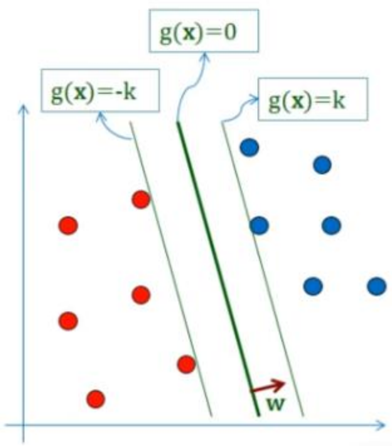

Figure 5 Linear SVM process, a line dividing support vectors into two different classes
Source: (Atul Agarwal, 2019)

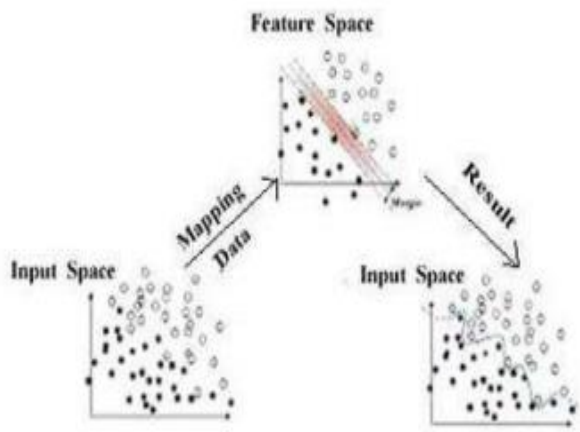

Figure 4 Non-Linear SVM process
Source: (Khan and Jain, 2016)

3.6 Confusion Matrix

(Beauxis-Aussalet and Hardman, 2014) describes the confusion matrix as a major resource to evaluate the errors in classification problems. According to (Maria Navin & Pankaja, 2016), evaluation of Supervised machine learning can be obtained by computing statistical measures, namely True Negatives, False Negatives, True Positives and False Positives. These components form a confusion matrix, as shown in the figure. It is a table generated for a classifier on a binary dataset and can be used to describe the classifier's performance. The terms in the matrix are:

- TP or True Positives: This means that actual and prediction values both are Yes and correct
- TN or True Negatives: This means that actual and prediction values both are No and correct
- FP or False Positives: This means that the actual was No, but the prediction is Yes and so incorrect
- FN or False Negatives: This means that the actual was Yes, but the prediction is No and so incorrect

Figure 6 below clears what the confusion matrix is and what the terms in the matrix described above are,

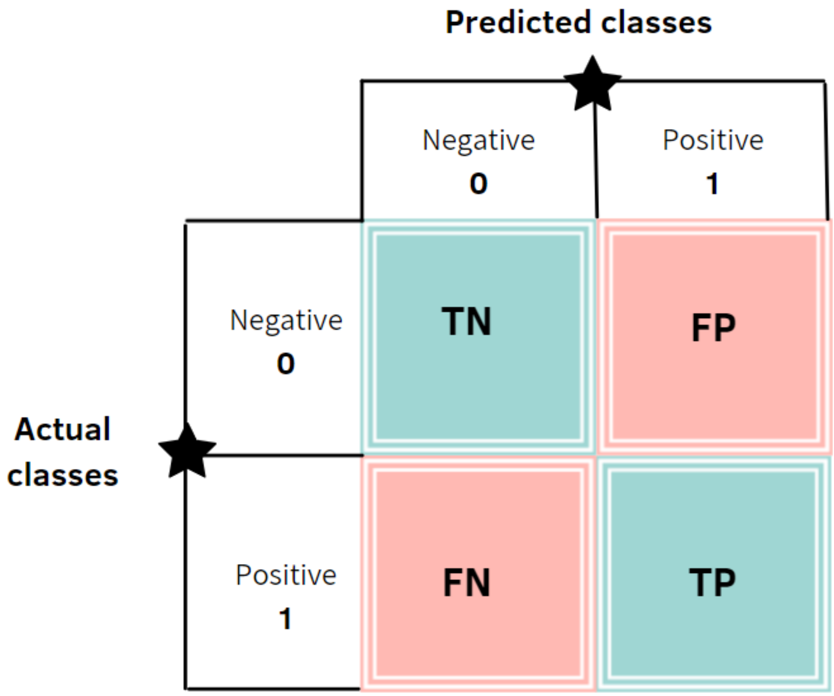

Figure 8 Simple image of the confusion matrix
Source: (Eugenia Anello, 2021)

According to (Vapnik, 1999; Saito and Rehmsmeier, 2015) following performance measures were calculated using a confusion matrix:

- Error rate (ERR): It is calculated as the number of all incorrect predictions divided by the total number of points in the dataset. It is known to be best if 0.0 and worst if 1.0.

$$\frac{FP + FN}{P + N}$$

- Accuracy: It is calculated as the number of all corrected predictions divided by the total number of points in the dataset. 1.0 is the best accuracy, whereas 0.0 is the worst accuracy.

$$\frac{TP + TN}{P + N}$$

- Sensitivity (Recall or True positive rate): It is calculated as the number of correct positive predictions divided by the total number of positives. 1.0 is the best sensitivity, while 0.0 is the worst.

$$\frac{TP}{P}$$

- Specificity (True negative rate): It is calculated as the number of correct negative predictions divided by the total number of negatives. 1.0 is the best Specificity, while 0.0 is the worst.

$$\frac{TN}{N}$$

- Precision (Positive predictive value): It is calculated as the number of correct positive predictions divided by a total number of positive predictions.

$$\frac{TP}{TP + FP}$$

- False positive rate: It is calculated as the number of incorrect positive predictions divided by the total number of negatives. It is best when 0.0 while worst when 1.0.

$$\frac{FP}{N}$$

- F – Score: It is a harmonic mean of precision and recall.

$$F_\beta = \frac{(1 + \beta^2)(Precision \cdot Recall)}{(\beta^2 \cdot Precision + Recall)}$$

As β is either 0.5, 1 or 2:

$$F_{0.5}\ Score\ = \frac{(1.25)(Precision \cdot Recall)}{(0.25 \cdot Precision + Recall)}$$

$$F_1\ Score\ = \frac{(2)(Precision \cdot Recall)}{(Precision + Recall)}$$

$$F_2\ Score\ = \frac{(5)(Precision \cdot Recall)}{(4 \cdot Precision + Recall)}$$

Key: FP: False Positive, FN: False Negative, P: All Positive, N: All Negative

3.7 Learning Curves

According to (Anzanello & Fogliatto, 2011), learning curves are considered effective tools to monitor the performance of workers exposed to a new task. As task repetition occurs, a mathematical representation of the learning process is provided by learning curves.

Machine learning algorithm that learns and optimise their internal parameters extensively use learning curves. Learning curves can evaluate how well the model is learning if they are used during the training of a machine learning model. They can also validate the model using them on the testing dataset (Brownlee, 2018).

- Train Learning Curve: It is used when the learning curve is calculated from the training dataset, which shows how well the model is learning

- Validation Learning Curve: It is used on a test dataset which shows how well the model is generalising.

Mostly dual learning curves are constructed that give the idea of learning and generalising the model.

Sometimes learning curves are also created for multiple metrics where one plot is created for the learning curves of each metric and the second is created for training and validation datasets

- Optimisation Learning Curves: These are calculated on the metric by which the parameters of each model are being optimised.
- Performance Learning Curves: They are calculated on the metric by which the model will be evaluated and selected.

3.8 Logistic Regression

(Oswal, 2019) describes Logistic Regression as a classification algorithm, which, when provided with a set of independent variables, can predict binary outcomes like; True/False, 1/0, Yes/No. Dummy variables are used to represent the outcomes. Oswal simplified it by saying that fitting data to a logit function predicts the probability of occurrence of an event. (Gladence, Karthi and Anu, 2015) comments that Logistic Regression is widely used because of the relationship between the discrete variable. Although regression gives better results on numerical data values, it also allows the prediction of discrete variables by the mixed values of continuous and discrete predictors.

Logistic Regression is similar to the Linear Regression model but uses a more complex cost function, defined as the "sigmoid function" or "logistic function". A comparison of both can be seen in figure 9.

The logistic regression's hypothesis limits the cost function between 0 and 1.
$0 \leq h_\theta(x) \leq 1$

To map predicted values to probabilities, the sigmoid function is used. It maps the real value into another value between 0 and 1. Sigmoid functions can be seen in the figure.

The formula of Sigmoid functions: $f(x) = \frac{1}{1+e^{-x}}$

The hypothesis of logistic regression: $h\theta(x) = \frac{1}{1+e^{-(\beta_O+\beta_1 x)}}$

The logistic regression cost function represents the optimisation objective; the cost function is created and minimised so the accurate model can be developed with minimum error. The cost function for y =0 and y =1 can be seen in figure 10 and figure 11, respectively.

The cost function of Logistic Regression $Cost(h\theta(x), y) = \begin{cases} -\log\big(h\theta(x)\big) & if\ \ y = 1 \\ -\log\big(1 - h\theta(x)\big) & if\ \ y = 0 \end{cases}$

The above function can be compressed into a single function:

$$J(\theta) = -\frac{1}{m}\sum[y^{(i)}\log\left(h\theta(\chi(i))\right) + \left(1 - y^{(i)}\right)\log(1 - h\theta(x(i)))]$$

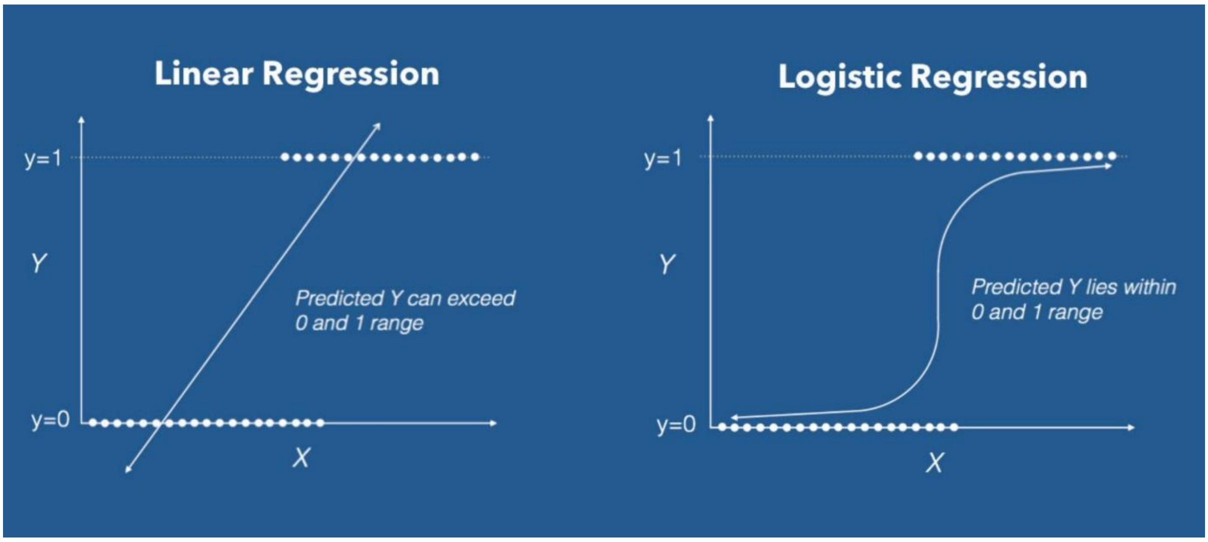

Figure 9 Linear Regression vs Logistic Regression
Source: (Maithili Joshi, 2019)

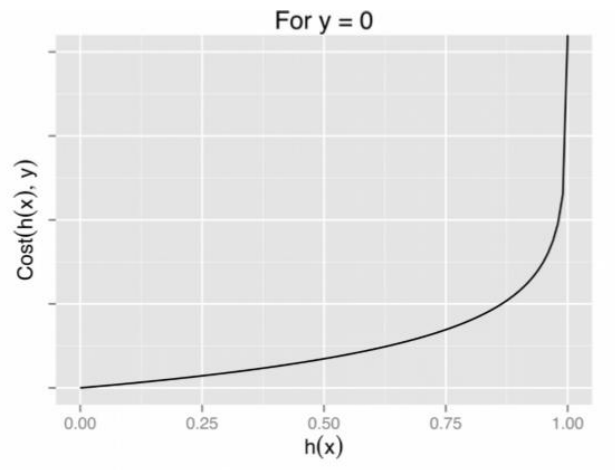

Figure 10 Logistic Regression graph
Source: (Ayush Pant, 2019)

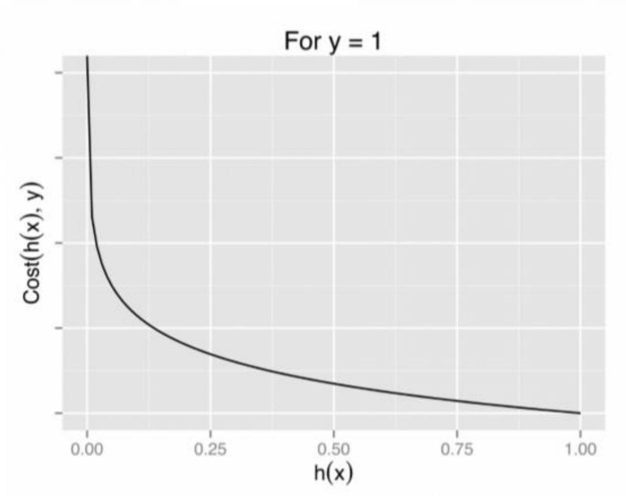

Figure 11 Logistic Regression graph
Source: (Ayush Pant, 2019)

3.9 K-nearest neighbour

It is a decision rule used as a classification algorithm in statistical pattern recognition (Devijver and Kittler, 1982). Every class is provided with a set of sample prototypes and a training set of pattern vectors from that class. To classify an unknown vector, its closest k neighbour is found from all the prototype vectors, and a majority rule decides a class label. K value is kept as an odd number to avoid ties on the overlap class regions (Laaksonen and Oja, 1996). K-Nearest Neighbour (K-NN) is a very simple and elegant rule, but the error rate is still low in practice. (Duda and Hart, 1973) displayed that, in theory, the asymptotic error rate occurs when the number of prototype samples becomes very large and close to optimal Bayes. K-NN rule is the standard comparison method against every new classifier.

The k-NN algorithm assumes similar things exist nearby. It pivots on this assumption being true enough to make the algorithm useful. It captures the idea of proximity, similarity, closeness or distance. Figure 12 shows that most of the similar data points are close to each other distance can be calculated mathematically; Euclidean (straight line distance) (Onel Harrison, 2018a).

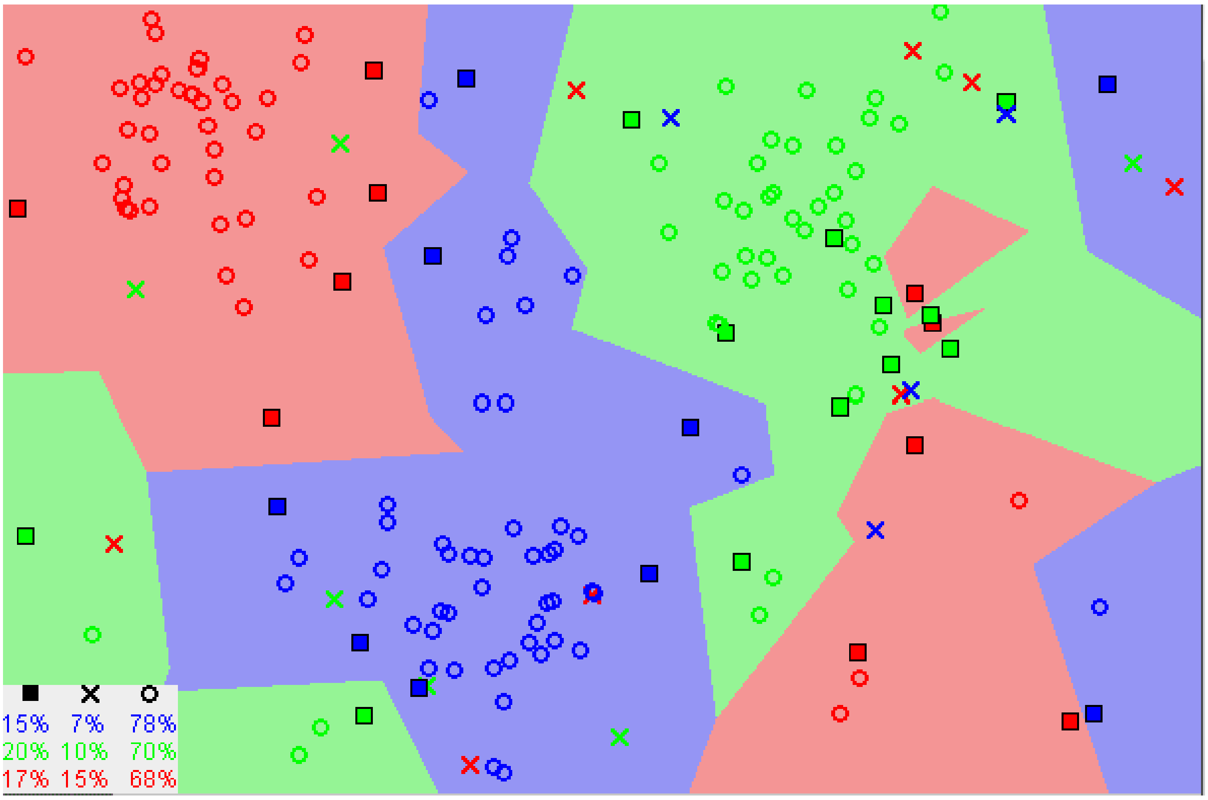

Figure 12 knn Image showing how similar data points typically exist close to each other
Source: (Onel Harrison, 2018b)

3.10 Decision Trees

These supervised learning techniques learn from decision rules from data's feature prediction. They can be used in classification and regression, sometimes called CART (Classification And Regression Trees). Decision Trees (DT) models are adaptive basis function models; they learn features directly from data instead of asking for pre-specification (quantstart, 2020). (Murphy, 2012) comments that these models are not linear in parameters, so they can only computer a locally optimal MLE (maximum likelihood estimate) for the parameters.

Decision Trees models partition the feature space into multiple simple rectangular regions, divided by axis parallel splits. When constructed, DT produces a graphical flowchart type image which is interpretable by if-then-else decision rulesets. The mean and mode of the training observations are used to predict a particular prediction.

probabilistic adaptive basis function $f(x) = E(y|x) = \sum_{m=1}^{M} \omega_m \varphi(x; v_m)$

- ω_m = mean response in a particular region
- v_m = variable split at a particular threshold

The decision tree starts with the root node, which has two homogenous child nodes. The tree growing process continues until it is impossible to split into different nodes. This binary partitioning means that the tree will only be divided into two child nodes. When the tree stops growing, all observations within each child node have fallen into an identical class, or some predefined constraint is achieved (Zhou *et al.*, 2019). Figure 13 represents the general structure of the Decision Tree.

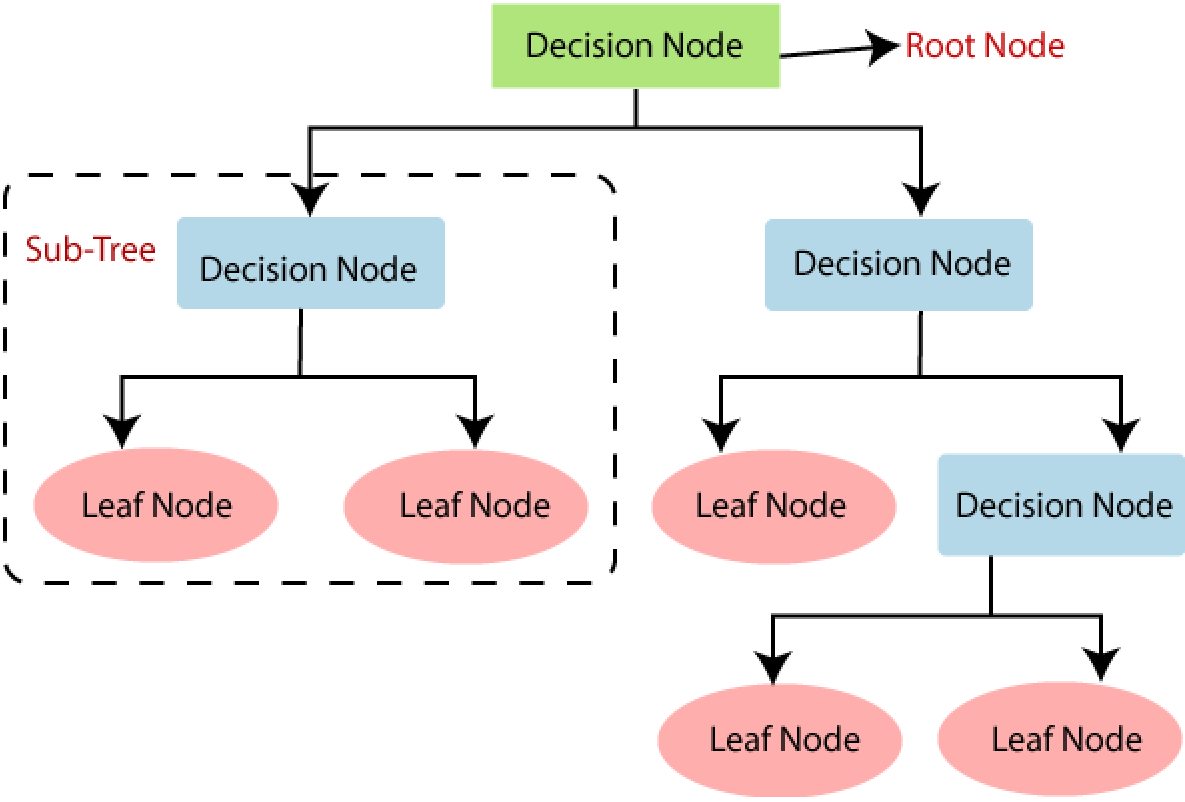

Figure 13 Decision Tree
Source: (Nagesh Singh Chauhan, 2022)

In the scikit-learn library of python, the quality of the split of the features is measured by the parameter named "criterion". Users had a choice of choosing "Gini" or "entropy".

3.10.1 Gini

According to (Tangirala, 2020), the purity of a particular class is determined by the GINI index, which does it by splitting along a particular attribute. Better the split, the higher the purity of the sets. (Sharda *et al.*, 2014) defines GINI as $L = 1 - \sum_j p_j^2$ where L is the dataset, j is the different class label and P_j is relative frequency. When the dataset is randomly labelled, some elements can be mislabeled; the measure of that frequency of mislabelling is known as Gini impurity (Paluszynska, Biecek and Jiang, 2017). (Sundhari, 2011) states that the Gini index is minimised at each split to make the tree less diverse and progress. The minimum value of the Gini index is 0, which means that the node is pure and all the elements the node contains are of unique class so that the node won't be split again. The features with less Gini Index are chosen for the optimum split. If the probability of two classes is the same, the Gini Index is maximum.

3.10.2 Entropy

Information gain is based on entropy. It measures the extent of randomness or impurity in the dataset (Shannon, 1948). The entropy of the dataset will be 0 if all the elements in the node belong to one class, which means that all are homogenous and there is no randomness and impurity in the node (Tangirala, 2020).

Entropy is defined by the following formula; $L = -\sum_{i=1}^{j} p_i \, log(p_i)$ where p_i Is the class label. If that label is 1, $log(p_i)$ Will result in 0, making entropy's output zero, which means homogenous

(Wang *et al.*, 2017). If the impurity of the dataset is higher, the entropy value will be higher (Fan *et al.*, 2011).

3.11 Random Forest

Breiman was invented in the early 2000s (Breiman, 2001a), and now it is known to be one of the most successful approaches to dealing with data. It is a supervised learning algorithm influenced by the works of (Amit and Geman, 1997; Ho, 1998; Dietterich, 2000). The concept of random forest is simple to divide and conquer, resulting in very effective; it samples the data fractions, grows a randomised tree predictor on every small piece and then sums these predictors together (Biau and Scornet, 2016) (figure 14). It is an ensemble classifier which uses a set of Classification And Regression Trees (CART) to make a prediction (Breiman, 2001b). The trees in RF are made by taking a subset of training samples through replacement known as the bagging approach, which indicates that one sample can be selected multiple times while others may not. In-bag or two-thirds of samples are used for the training of the trees, while the remaining one-third, also known as out-of-the-bag samples, are used for internal cross-validation to estimate how well the RF model is performing (Breiman, 2001b). Figure 15 shows the whole process (i = samples, j = variables, p = probability, c = class, s = data, t = number of trees, d = new data to be classified, and value = the different values that the variable j can have).

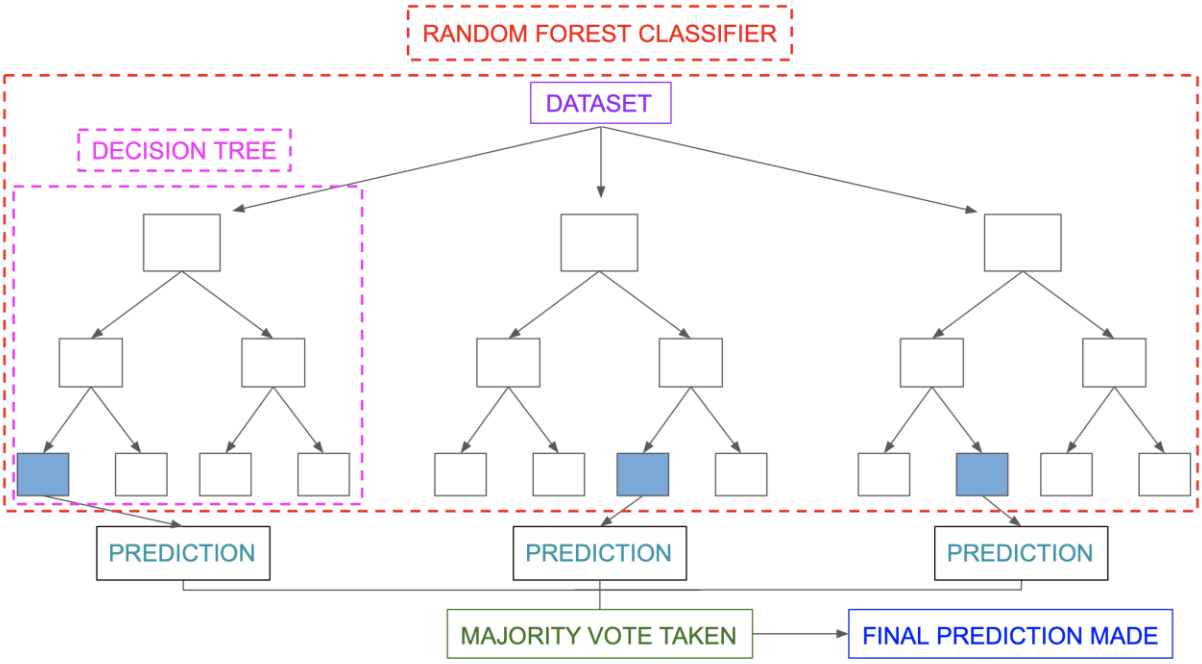

Figure 14 Random Forest Classifier general figure
Source: (Karan Kashyap, 2019)

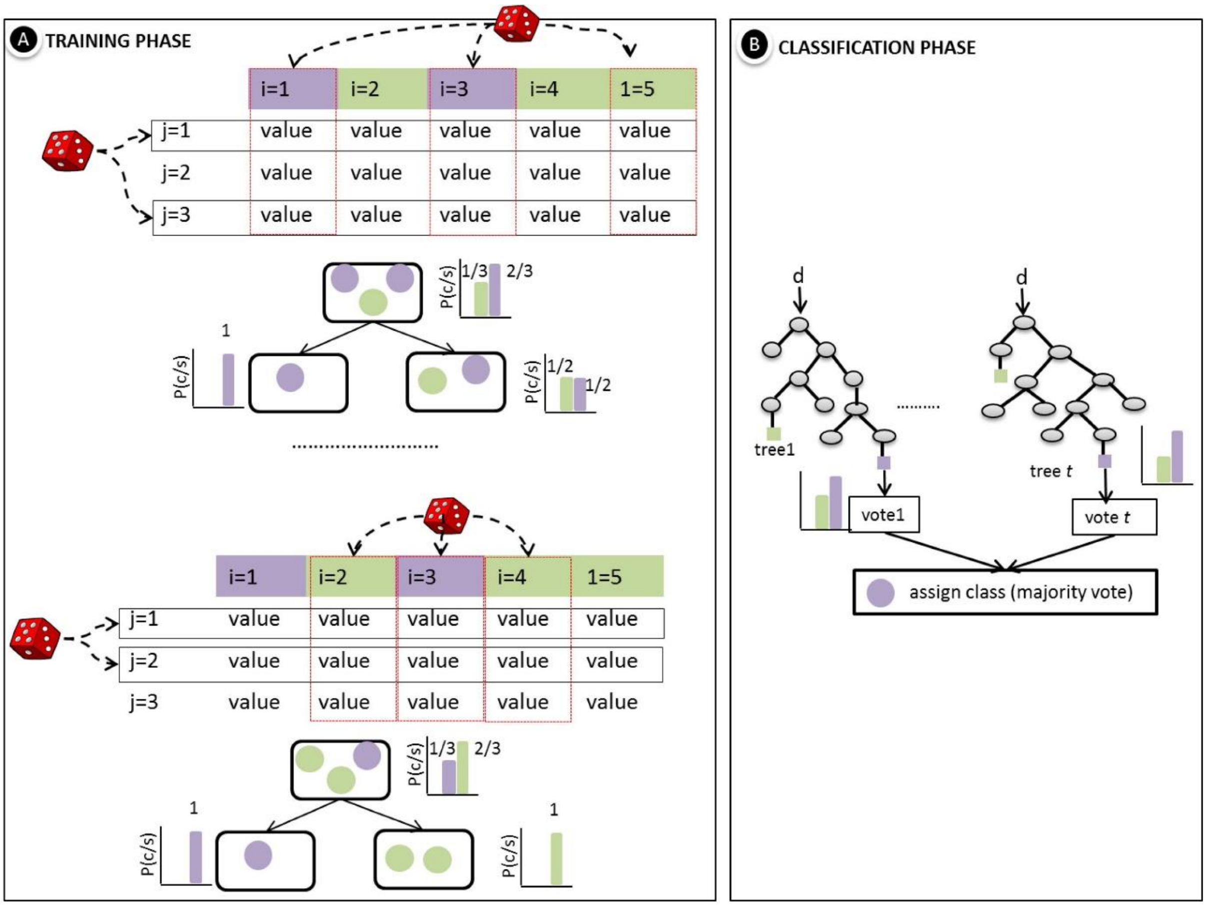

Figure 15 Training and classification phases of Random Forest classifier
Source: (Belgiu and Drăguţ, 2016)

3.12 AdaBoost Classifier

The diversity among weak classifiers is the reason for AdaBoost good performance (An and Kim, 2010). It is one of the famous algorithms for ensemble classifiers by selecting weak member classifiers. It generates a strong classifier combined with several weak classifiers and adjusts weight through the repetition process, and the original training dataset is not compromised.

These trees are also called Decision Stumps. (Anshul Saini, 2021a) have displayed the steps for the AdaBoost algorithm. First, all datasets are assigned with weights; initially, they all are equal.

Formula to calculate weight: $\omega(x_i y_1^1) = \frac{1}{N}, i = 1,2,..n$, where N is the total number of data points.

Then a decision stump is created for each feature, the Gini Index is calculated for each tree, and the one with the lowest Gini Index is selected as the first stump.

Next, influence/importance/amount of say is calculated for the classifier; the formula for this is: $\alpha = \frac{1}{2} log_e \frac{1-Total\ Error}{Total\ Error}$

The error in this algorithm is the summation of all sample weights misclassified. The total error is always between 0 and 1.

Now new sample weight is calculated and updated in the table

$$New\ Sample\ Weight = old\ weight * e^{\pm Amount\ of\ say(\alpha)}$$

Updated weights are normalised to make their sum equal to 1, which is done by dividing all weights by their sum. Figure 16 shows all processes.

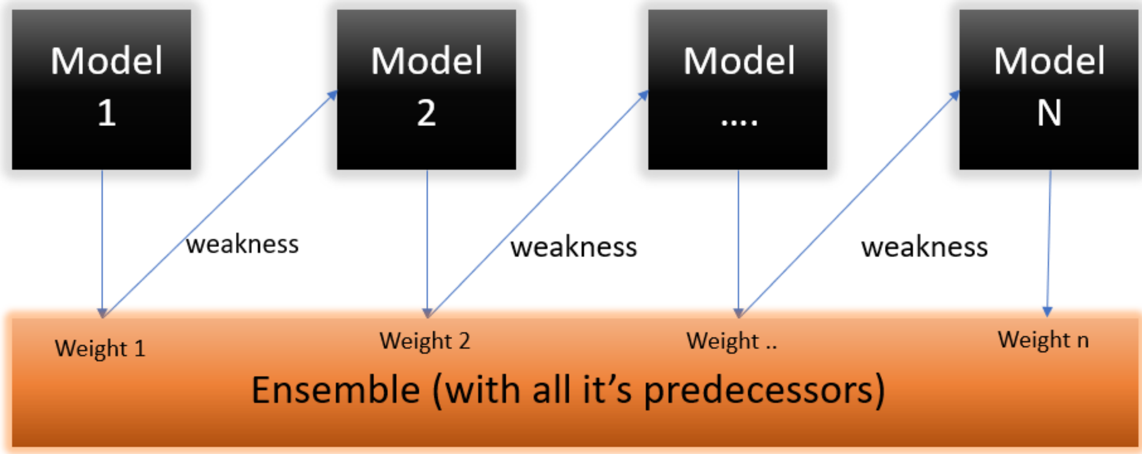

Figure 16 Adaboost Classifier
Source: (Anshul Saini, 2021b)

3.13 Gradient boosting classifier

It is a gradient-based approach which learns for a boosting classifier incrementally. It estimates a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ based on a linear combination of weak learners $h: \mathbb{R}^n \rightarrow \mathbb{R}$ as $f(x) = \sum_{J=1}^{M} \alpha_j h_j(x; \theta_j)$ where $x \in \mathbb{R}^n$ is an input vector $\alpha_j \in \mathbb{R}^n$ Is a real-valued weight and M is the number of weak learners (Friedman, 2001; Becker *et al.*, 2013).

(Son *et al.*, 2015) displayed a training example $\{(x_i y_i) | x_i \in \mathbb{R}\ and\ y_i \in \mathbb{R}\}_{i=1:N}$, f(·) is constructed greedily where θ_j and α_j are selected of a weak learner and iterated to minimise an augmented loss function which is: $L = \sum_{i=1}^{N} l(y_i, f(x_i)) = \sum_{i=1}^{N} exp(-y_i, f(x_i))$ Where an exponential loss function is adopted.

A classifier that classifies data poorly, which seems random guesses, and so has a high error rate, is known as a Weak Learner. The approach of adding the weak learners iteratively or sequentially, step by step, is known as the additive model. The loss function in the gradient boosting classifier estimates how good the model is at making the predictions on the dataset provided. Figure 17 below summarises the Gradient Boosting classifier.

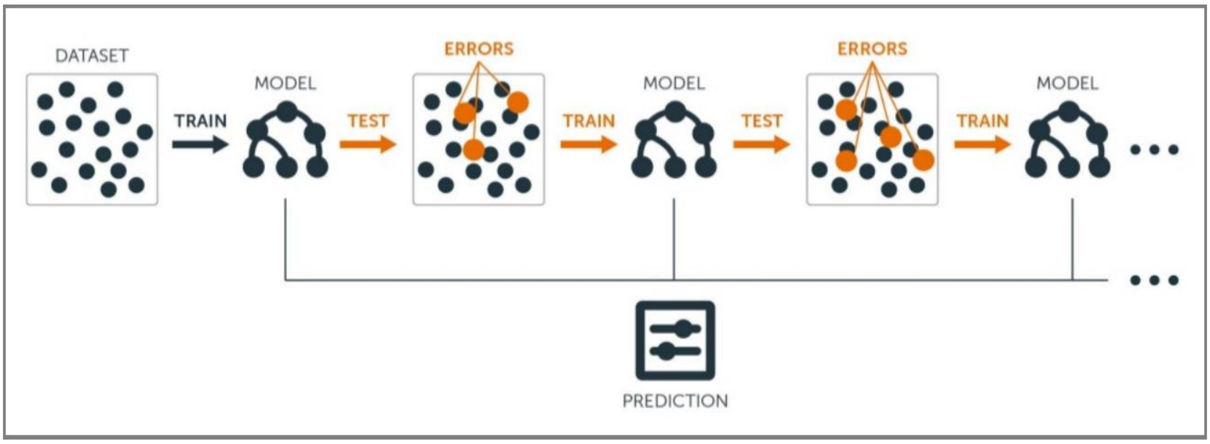

Figure 17 Gradient Boosting Classifier
Source: (Anjani Kumar, 2020)

3.14 Voting Classifier

(Jason Brownlee, 2021) Describes a Voting classifier as an ensemble machine learning model that combines the predictions from multiple models. It improves model performance by achieving better performance than a single model. For classification, it sums the each label's predictions and then predicts the label with the majority vote.

According to (Kumar et al., 2017), every individual algorithm has its strengths and drawbacks, so an approach which combines more than one algorithm and gives accurate classifications is very effective, and voting is one of the simplest ways in ensemble learning techniques which does the combining and output the increased accuracy of the prediction models. The voting classifier wraps multiple models into one and averages the sub-model predictions to make predictions for new data. (Naib and Chhabra, 2014) says that voting uses the majority voting as a combining rule, which applies to the inputted classifiers to increase the accuracy percentage.

There are two types of voting (Raihan-Al-Masud and Mustafa, 2019):

- Hard voting or max voting: A class is predicted with the largest sum of votes from model

- Soft voting: A class is predicted with the largest summed probability from models

4 Methodology

In this project, CRISP-DM (CRoss-Industry Standard Process for Data Mining) reference model is used as an approach to develop and evaluate the proposed model. CRISP-DM has past phases of projects related to each other, and important relationships could exist between any task depending on the project's goal (Chapman *et al.*, 2000), which will be needed in a project like this. Figure 18 below highlights the relationships.

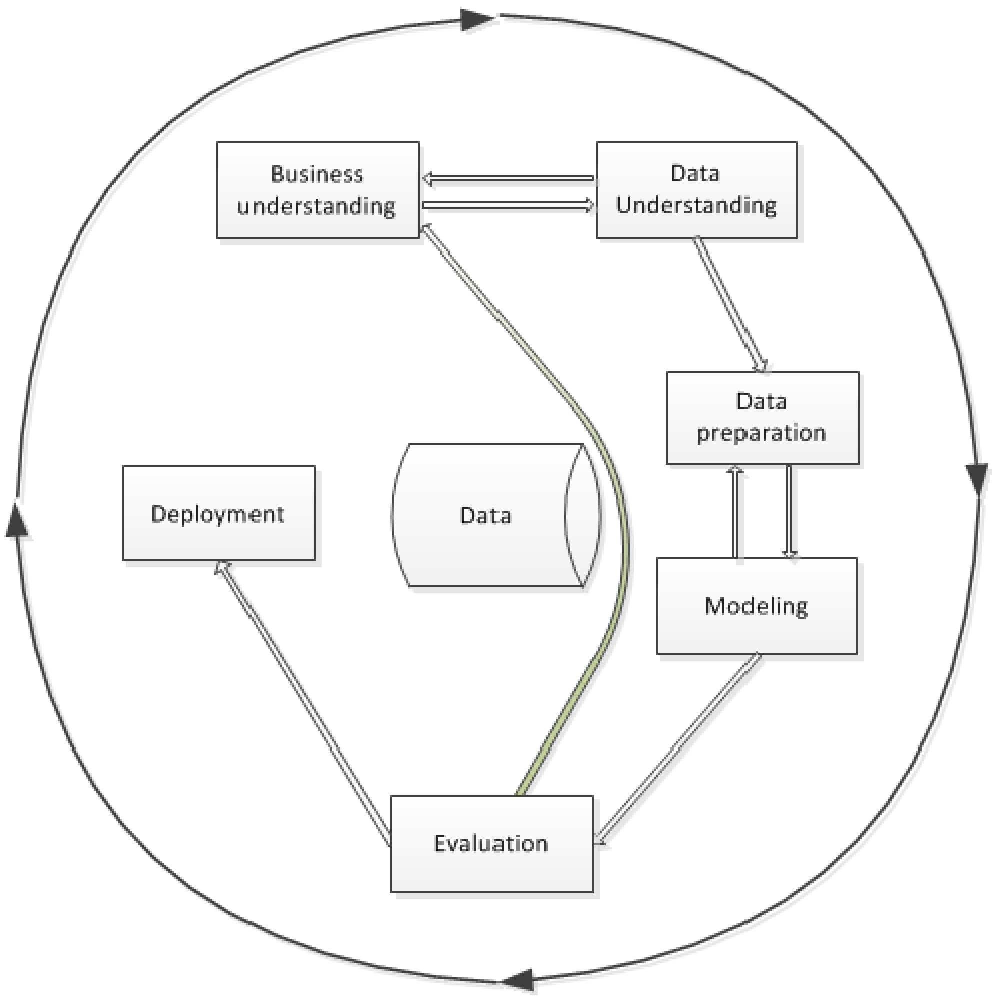

Figure 18 Phases of the CRISP-DM reference model
Source: (Niaksu, 2015)

4.1 Business Understanding

Business understanding has already been discussed in the previous chapters.

4.2 Data Understanding

Data required for this project was from an e-commerce website with different category products, the terminating status of that order was either completed, refunded or cancelled, and it should have details like prices, discounts if any, date of purchase and date of customer's first purchase.

The required dataset was searched in open-source platforms like Data.World, practicaldatascience.co.uk, imerit.net, and www.kaggle.com. The original plan was to source data from a UK-based e-commerce store, but the available dataset was either specific to a particular category or had very few entries, so the next best choice was to expand the search outside the UK.

Datasets for multiple countries were available, but the interests were inclined toward Pakistan's Dataset as it was easier to understand, was very extensive, and matched all the requirements.

The shortlisted Data Set was from Kaggle, "Pakistan's Largest E-Commerce Dataset" (Zeeshan-Ul-Hassan Usman and Sana Rasheed, 2021). The dataset is about the E-commerce company of Pakistan for orders from March 2016 to August 2018. It has around 580,000 orders in detail. The dataset contains information on orders, like item detail, shipping method, payment method, product categories, date of order, stock-keeping unit (SKU), price and quantity of an Item. The customer details are encrypted as customer ID, and the date on which the user became a customer is also stored.

4.3 Data Preparation

4.3.1 Data Setup

Relevant libraries and packages were imported, and data was loaded in the jupyter notebook using the panda's library.

4.3.2 Data Cleaning

After data was uploaded, it was analysed briefly, and unnecessary data was found. Blank columns and rows with Null Customer ID were removed (figure 58). Few columns: BI status and Sales commission code which were unnecessary for further use in the project, were also removed, along with repetitive columns of dates. Types of some of the columns were also corrected.

The statistical analysis of the data was performed, highlighting a few values resulting in the negative grand total (figure 57); they were supposedly typing errors in the dataset and were removed after further confirmation through analysis.

Next, all the order statuses were observed (figure 56), and it was found that multiple order statuses meant the same. Orders with statuses like

- Complete, received, cod, paid, closed, exchange **meant** completed
- Cancelled, order_refunded, refund, fraud **meant** cancelled
- Payment_review, pending, processing, held, pending_paypal **meant** pending

So similar statuses were merged.

A new dataset was created just for the cancelled orders to investigate them further for insight, but they were removed from the main dataset along with the orders with NA status.

4.3.3 Data Insights

After the data was cleaned, the number of products was analysed in each transaction, and looking at just the first few products; it was clear that in the dataset, there were customers who had bought multiple times, customers who had more than one product in one transaction and customers who only had one transaction and one product per transaction per user as well. This insight was a relief, showing that the dataset was worthy of further exploration and processing.

Columns whose names were inappropriate in a dataset were renamed to a better names.

The cancelled orders dataset was analysed thoroughly to get the idea of how many customers cancelled and how many orders from total orders were cancelled; the categories whose orders were cancelled were also analysed.

Next, the product categories were analysed; "others" and "/N" categories were merged as they meant the same for the project.

Category names in the dataset were in the text string, and to reduce the computational complexity of the text, a label encoder was used, assigning unique numbers to each category.

In the dataset, columns were created with the category names and a new variable was defined, which multiplies the item's price with its quantity to determine the total price. Dataset was categorised with the amount of each transaction with each category, so the columns of each category have the price percentage of the total transaction.

The anomaly was also removed from the data.

The dataset was then divided into two. One for the training and one for the testing; the testing one was saved separately. The dataset for processing is updated with just the training dataset part.

4.4 Modelling

4.4.1 Defining Customer Categories

Now the part was to define the customer categories, so the dataset was used to collect insight into customer behaviour. All transactions were grouped based on each customer's ID, and their transaction history was analysed. For each customer, the number of days passed from their first and last purchase was calculated along with the count of customers who bought multiple and single times. A matrix from the updated dataset was created, consisting of only relevant features for customer behaviour, like the mean and the sum of all purchases by each customer, the total number of purchases they made and the ratio of purchases from each product category. The vast differences in the ranges would have resulted in the bias of future processes toward extreme values, so the matrix was standardised. A PCA was performed to find the separation quality between all the columns (figure 34,35). A scatter plot was constructed for the PCA (figure 37).

As the data was prepared, K-Means were needed to find the clusters. Two approaches were used to find the optimal value of K or the number of clusters: the Elbow method and the Silhouette score. K-Means were performed, and the number of customers for each cluster was calculated (figure 33).

Based on the K-means clustering results, A dataset was created mapping each customer ID and their relevant features to the cluster they belong to. This dataset would later train machine learning models to map customers to their respective clusters. A radar chart was plotted (figure 37) to analyse what the clusters looked like.

4.4.1.1 Classification of Customers

Customers were classified using machine learning algorithms; the train set defined previously was again broken into 2, train and validation data with a ratio of 80 and 20, respectively. A class was defined to avoid the repetition of code for each classifier used.

The following machine learning model was trained:

1. Support vector classifier; a linear scale was used to perform SVC, and multiple parameters were tested to optimise it. For regularisation, ten values were tested from 0.01 to 100 to check the effect of avoiding the misclassification of points and to a larger margin of separation where misclassifying is accepted. To avoid the risk of overfitting, the cross-validation method is used in the SVM; it splits the training dataset into five smaller sets in the project. So a model is trained on four sets, and 1 set is used to validate results. These

sets are computed so that all five sets are tested once. (figure 38) summarises the K-fold process.

2. Logistic regression classifier; the parameters were set similar to SVC; for regularisation from 0.01 to 100, 10 equidistant values were tested. The same K-fold value of five was selected so the model can be trained on four sets, and 1 set can be used to validate results.
3. K-Nearest classifier; the parameters included the number of neighbours tested from 1 to 100, with step value = 1 and the K-fold whose value remains the same 5, as previous classifiers.
4. Decision tree; in the parameters, the quality of the split was tested by entropy and Gini. To control the overfitting, the max feature parameter was tested on sqrt and $log_{2,}$ the K-fold whose value remains the same 5, as previous classifiers.
5. Random forest classifier: Most parameters like criterion, max features and k-fold remain the same as the decision tree. But n estimator (the number of trees) was added; it was tested on values from 20 to 100, with a difference of 20 steps.
6. Ada boost classifier; the parameter tested is the n estimator, which is the number of models to iteratively trained; the values tested were 10 to 100 with the steps of 10. K-fold parameters were set to 5 like before.
7. Gradient boosting classifier: the parameters were chosen the same as the Ada boost classifier.

Learning curves were constructed for all the classifiers for the training and cross-validation scores (figure 40,42,44,46,48,50). And to check their results, a confusion matrix was constructed (figure 39,41,43,45,47,49), which displayed the predictions performed by each classifier and how its predicted value related to the actual values of the data.

Lastly, the voting classifier is used to find the best parameters for all the classifiers. They were saved on the separate variables, and then soft voting was performed using random forest, gradient boosting and decision tree. The learning curve was constructed to check the training score, cross-validation (figure 54), and confusion matrix (figure 53) to find evaluation metrics.

4.5 Evaluation

A data frame which was kept hidden from the processes was selected. Grouping and modification on the data frame were done like before on the training set. Then it was converted to a matrix which was then standardised. A k-mean analysis was performed to find the clusters; the result was set on the Y axis, and the X axis dataset with the customer information was selected. All the classifiers were used to predict, and in the end, the voting classifier was used to predict the outcome.

5 Results and Discussion

For this project, we separated the orders cancelled from the main dataset(figure 28). The dataset with processed orders had data for 79,309 unique customers, which bought 62,531 unique products in 202,314 transactions. Two third of the customers had only ordered once, while one-third have placed orders more than once (figure 23). Nearly 72% of the customers ordered from only one category, while 28% ordered from multiple categories (figure 24).

Of 402440 transactions, 49.73% were cancelled, and of 146293 customers, 45.79% of them cancelled their orders (figure 25). Categories were also observed for the cancelled orders (figure 26); mobiles and Tablets were cancelled the most; 65,102. Followed by Appliances; 26662 and Women's fashion; 25,007.

Then the dataset was divided into two sets, one was trained, and one was separated for testing purposes. To make clusters K-Means algorithm was selected, but to find the optimal number of clusters in which the data can be clustered, the two most popular methods Elbow method and the Silhouette score, were used. From the elbow method, a plot was generated, and it was observed that the elbow was made on cluster number 24 (figure 29). The Silhouette score also gave the maximum value of 0.64566 at 24 clusters (figure 30,31,32).

The classifiers were used to train the data and were cross-validated; the figure below displays how they performed; their weighted precision, weighted recall, weighted F1-Score and Accuracy. The support was 7484 for all classifiers.

	Weighted				
Classifiers	Precision	Recall	F1-Score	Accuracy	Accuracy %
Support Vector Classifier	0.87510	0.87614	0.86420	0.87614	87.61
Logistic Regression	0.72890	0.80211	0.75277	0.80211	80.21
K-Nearest Algorithm	0.92538	0.92584	0.92517	0.92584	92.58
Decision Tree	0.97625	0.97662	0.97635	0.97662	97.66
Random Forest	0.98102	0.98263	0.98120	0.98263	98.26
Ada Boost Classifier	0.21874	0.32403	0.23338	0.32403	32.40
Gradient Boosting Classifier	0.98301	0.98263	0.98255	0.98263	98.26

Figure 19 Output result of all classifiers

The above results (figure 19) show that the best performing classifier is Gradient Boosting Classifier, followed by Random Forest and then Decision Tree, so these three were selected in the Voting Ensemble Classifier. The below figure shows the output from the Voting Ensemble classifier (figure 20).

	Weighted				
Classifiers	Precision	Recall	F1-Score	Accuracy	Accuracy %
Voting Ensemble Classifier	0.98131	0.98236	0.98159	0.98236	98.24

Figure 20 Output of voting classifiers

After training classifiers, a testing dataset which was kept hidden from the training was then processed. Ada Boost classifiers showed poor output while processing the training data's cross-validation, so it wasn't included in the testing dataset. Other classifiers were tested, and the following results were obtained (figure 21).

Classifiers	Accuracy %
Random Forest	96.25
Gradient Boosting Classifier	96.11
Decision Tree	95.08
K-Nearest Algorithm	86.21
Support Vector Classifier	82.32
Logistic Regression	76.76

Figure 21 Output of classifiers used on the testing dataset

After that, the voting classifier was used on the testing dataset, and the following accuracy was obtained (figure 22).

Classifiers	Accuracy %
Voting Classifier	95.8

Figure 22 Output of voting classifier

Customer segmentation can help businesses with the following advantages:

- Businesses can send promotions and new product features. They can customise marketing strategies and programs suitable for particular customer segments, which could differentiate them from competitors and improve their business market position. For example, if they want to boost the sales on Mobiles and Tablets, they can send a personalised email about new sales and products, especially to the customers in clusters 22,19 and 15, as all customers in these customers have purchased from the Mobiles and Tablets categories every time.
- Sales can be increased as more tailored content can be shared with the respective segment, increasing turnover and making the business more profitable. For example, If the sales of Mobiles and Tablets are going down, they can approach customers of cluster 18, who have spent a lot in those categories before.
- They can identify products associated with what segments to manage the supply and demand.
- They can find the hidden dependencies and associations between customers, products or between customers and products.
- It enables businesses to identify which customers will stop purchasing and consider different business strategies to find solutions to stop them from leaving or building their interest.

Furthermore, project findings also highlighted that company has 68.81% of customers had just bought once from them, which can be concerning for them; from the further analysis, they can check why the majority of the customers are not returning after the one-time purchase and can implement strategies to bring them back and retain future customers.

There is a need to find a fruitful basis for the other causes and reasons to explain the similar behaviour of customers in each segment. The company can further analyse why nearly half of the orders were cancelled, was there too much delivery time? Or, after they placed an order, they found competitor advertisements showing the same product at a discounted price. Another finding was that 72% of customers only bought from a single category, meaning that customers either find better quality or cheaper rates at different sites or don't trust the company for those categories. If the business had to grow, it needed to ensure that it fulfilled the customer's all purchasing needs for higher revenue generation.

6 CONCLUSION AND FUTURE RECOMMENDATIONS

This project proposed a different approach to customer segmentation, where customer segmentation is predicted even for upcoming customers. The clustering for customers was done based on their purchasing behaviour. This project presents the promising effects of using machine learning on customer segmentation. Overall, 24 clusters were discovered on the dataset using K-Means clustering; as can be seen on the radar chart (figure 37), they all are distinct from each other, while some represent the customers who just bought from one single category, some who bought from multiple categories but didn't spend a large amount in those, some who bought few times but spent much money, some who bought very frequently.

Multiple machine learning classifiers were trained and tested to predict clusters for new customers, like Support Vector Classifier, Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, AdaBoost Classifier and Ensemble voting classifier. The classification methods were evaluated by several evaluation metrics like precision, recall, f1 score and accuracy (figure 19,20) to account for unequal class sizes; all metrics were calculated with the weighted data. The project results showed that Random Forest, Gradient Boosting, and Decision Tree classifiers predict better customer behaviours concerning the categories they purchased.

In future, customer behaviour can be analysed with the geographical location of orders and customer demographics. Additional analysis can be performed for the customer who ordered more than once to identify the relationship between future orders from previous orders. Market Basket Analysis can also be performed, showing the customers what other customers have bought along with the selected product.

REFERENCES

Akhondzadeh-Noughabi, E. and Albadvi, A. (2015) "Mining the dominant patterns of customer shifts between segments by using top-k and distinguishing sequential rules," *Management decision* [Preprint].

Amit, Y. and Geman, D. (1997) "Shape quantization and recognition with randomized trees," *Neural computation*, 9(7), pp. 1545–1588.

Andrey Bulezyuk (2017) *Machine learning model: Python sklearn & keras*.

Anjani Kumar (2020) "Introduction to the Gradient Boosting Algorithm," *Analytics Vidhya*, 20 May.

Anshul Saini (2021a) "AdaBoost Algorithm – A Complete Guide for Beginners," *analyticsvidhya*, 15 September.

Anshul Saini (2021b) "AdaBoost Algorithm – A Complete Guide for Beginners," *analytics vidhya*, 15 September.

Anzanello, M.J. and Fogliatto, F.S. (2011) "Learning curve models and applications: Literature review and research directions," *International Journal of Industrial Ergonomics*, 41(5), pp. 573–583. Available at: https://doi.org/https://doi.org/10.1016/j.ergon.2011.05.001.

Aren, S. *et al.* (2013) "Factors affecting repurchase intention to shop at the same website," *Procedia-Social and Behavioral Sciences*, 99, pp. 536–544.

Aryuni, M., Madyatmadja, E.D. and Miranda, E. (2018) "Customer segmentation in XYZ bank using K-means and K-medoids clustering," in *2018 International Conference on Information Management and Technology (ICIMTech)*. IEEE, pp. 412–416.

Ashutosh Bhardwaj (2020) "Silhouette Coefficient," 26 May.

Atul Agarwal (2019) "Support Vector Machine — Formulation and Derivation," *Towards Data Science*, 24 September.

Ayush Pant (2019) "Introduction to Logistic Regression," *Towards Data Science* [Preprint].

Baer, D. (no date) "CSI: Customer Segmentation Intelligence for Increasing Profits. SAS Glob Forum. 2012: 1-13."

Ballestar, M.T., Grau-Carles, P. and Sainz, J. (2018) "Customer segmentation in e-commerce: Applications to the cashback business model," *Journal of Business Research*, 88, pp. 407–414. Available at: https://doi.org/https://doi.org/10.1016/j.jbusres.2017.11.047.

Beauxis-Aussalet, E. and Hardman, L. (2014) "Visualization of confusion matrix for non-expert users," in *IEEE Conference on Visual Analytics Science and Technology (VAST)-Poster Proceedings*.

Becker, C. *et al.* (2013) "Supervised feature learning for curvilinear structure segmentation," in *International conference on medical image computing and computer-assisted intervention*. Springer, pp. 526–533.

Beheshtian-Ardakani, A., Fathian, M. and 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.

Belgiu, M. and Drăguţ, L. (2016) “Random forest in remote sensing: A review of applications and future directions,” *ISPRS journal of photogrammetry and remote sensing*, 114, pp. 24–31.

Berson, A. and Thearling, K. (1999) *Building data mining applications for CRM*. McGraw-Hill, Inc.

Biau, G. and Scornet, E. (2016) “A random forest guided tour,” *TEST*, 25(2), pp. 197–227. Available at: https://doi.org/10.1007/s11749-016-0481-7.

Boser, B.E., Guyon, I.M. and Vapnik, V.N. (1992) “A training algorithm for optimal margin classifiers,” in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152.

Breiman, L. (2001a) “Random forests,” *Machine learning*, 45(1), pp. 5–32.

Breiman, L. (2001b) “Random forests,” *Machine learning*, 45(1), pp. 5–32.

Brownlee, J. (2018) *Better deep learning: train faster, reduce overfitting, and make better predictions*. Machine Learning Mastery.

Brynjolfsson, E., Hitt, L.M. and Kim, H.H. (2011) “Strength in numbers: How does data-driven decisionmaking affect firm performance?,” *Available at SSRN 1819486* [Preprint].

Chapman, P. *et al.* (2000) “CRISP-DM 1.0: Step-by-step data mining guide,” in.

Chen, Chiang and Storey (2012) “Business Intelligence and Analytics: From Big Data to Big Impact,” *MIS Quarterly*, 36(4), p. 1165. Available at: https://doi.org/10.2307/41703503.

Chen, H., Chiang, R.H.L. and Storey, V.C. (2012) “Business intelligence and analytics: From big data to big impact,” *MIS quarterly*, pp. 1165–1188.

Chen, Q., Zhang, M. and Zhao, X. (2017) “Analysing customer behaviour in mobile app usage,” *Industrial Management & Data Systems* [Preprint].

Chin, J.K. *et al.* (2017) “Advanced analytics: Nine insights from the C-suite,” *McKinsey Analytics, July* [Preprint].

Chris Yiu (2012) *The Big Data Opportunity: Making government faster, smarter and more personal*, *policyexchange.org.uk*.

Chu, X. *et al.* (2016) “Data cleaning: Overview and emerging challenges,” in *Proceedings of the 2016 international conference on management of data*, pp. 2201–2206.

Claycamp, H.J. and Massy, W.F. (1968) “A theory of market segmentation,” *Journal of Marketing Research*, 5(4), pp. 388–394.

Collica, R.S. (2017) *Customer segmentation and clustering using SAS Enterprise Miner*. Sas Institute.

Constantinides, E. (2004) “Influencing the online consumer’s behavior: the Web experience,” *Internet research* [Preprint].

CSC (2012) *Big data universe beginning to explode*, *http://www.csc.com/insights/flxwd/ 78931-big_data_growth_just_beginning_to_explode*.

Dan Vesset *et al.* (2012) “ Worldwide Big Data Technology and Services.”

Daniela Coppola (2022) *E-commerce worldwide*, *statista*.

Daoud, R.A. *et al.* (2015) “Combining RFM model and clustering techniques for customer value analysis of a company selling online,” in *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*. IEEE, pp. 1–6.

Deng, Y. and Gao, Q. (2020) “A study on e-commerce customer segmentation management based on improved K-means algorithm,” *Information Systems and e-Business Management*, 18(4), pp. 497–510. Available at: https://doi.org/10.1007/s10257-018-0381-3.

Devijver, P.A. and Kittler, J. (1982) *Pattern recognition: A statistical approach*. Prentice hall.

Dietterich, T.G. (2000) “Ensemble methods in machine learning,” in *International workshop on multiple classifier systems*. Springer, pp. 1–15.

Dogan, O., Ayçin, E. and Bulut, Z. (2018) “Customer segmentation by using RFM model and clustering methods: a case study in retail industry,” *International Journal of Contemporary Economics and Administrative Sciences*, 8.

Donthu, N. and Garcia, A. (1999) “The internet shopper,” *Journal of advertising research*, 39(3), p. 52.

Douglas Laney (2001) *3D Data Management: Controlling Data Volume, Velocity and Variety*.

Duda, R.O. and Hart, P.E. (1973) “Pattern recognition and scene analysis.” Wiley, New York.

Dullaghan, C. and Rozaki, E. (2017) “Integration of machine learning techniques to evaluate dynamic customer segmentation analysis for mobile customers,” *arXiv preprint arXiv:1702.02215* [Preprint].

Eastlick, M.A. and Feinberg, R.A. (1999) “Shopping motives for mail catalog shopping,” *Journal of Business Research*, 45(3), pp. 281–290.

Eugenia Anello (2021) “How to evaluate you model using the Confusion Matrix,” *Towards AI*, 23 February.

Ezenkwu, C.P., Ozuomba, S. and Kalu, C. (2015) “Application of K-Means algorithm for efficient customer segmentation: a strategy for targeted customer services.”

Fan, R. *et al.* (2011) “Entropy-based information gain approaches to detect and to characterize gene-gene and gene-environment interactions/correlations of complex diseases,” *Genetic epidemiology*, 35(7), pp. 706–721.

Forgy, E.W. (1965) “Cluster analysis of multivariate data: efficiency versus interpretability of classifications,” *biometrics*, 21, pp. 768–769.

Friedman, J.H. (2001) “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232.

Gao, L. and Bai, X. (2014) “Online consumer behaviour and its relationship to website atmospheric induced flow: Insights into online travel agencies in China,” *Journal of Retailing and Consumer Services*, 21(4), pp. 653–665.

Gehrt, K.C., Yale, L.J. and Lawson, D.A. (1996) “The convenience of catalog shopping: Is there more to it than time?,” *Journal of Direct Marketing*, 10(4), pp. 19–28.

Gladence, L.M., Karthi, M. and Anu, V.M. (2015) “A statistical comparison of logistic regression and different Bayes classification methods for machine learning,” *ARPN Journal of Engineering and Applied Sciences*, 10(14), pp. 5947–5953.

Haghighatnia, S., Abdolvand, N. and Rajaee Harandi, S. (2018) “Evaluating discounts as a dimension of customer behavior analysis,” *Journal of Marketing Communications*, 24(4), pp. 321–336.

Hamdi, K. and Zamiri, A. (2016) “Identifying and segmenting customers of pasargad insurance company through RFM model (RFM),” *International Business Management*, 10(18), pp. 4209–4214.

Hastie, T. *et al.* (2009) *The elements of statistical learning: data mining, inference, and prediction*. Springer.

Hawkins, D.I., Best, R.J. and Coney, K.A. (1998) *Consumer behavior : building marketing strategy*. 7th edn.

He, Xixi and Li, C. (2016) “The research and application of customer segmentation on e-commerce websites,” in *2016 6th International Conference on Digital Home (ICDH)*. IEEE, pp. 203–208.

He, X and Li, C. (2016) “The Research and Application of Customer Segmentation on E-Commerce Websites,” in *2016 6th International Conference on Digital Home (ICDH)*, pp. 203–208. Available at: https://doi.org/10.1109/ICDH.2016.050.

Hearst, M.A. *et al.* (1998) “Support vector machines,” *IEEE Intelligent Systems and their applications*, 13(4), pp. 18–28.

Ho, T.K. (1998) “The random subspace method for constructing decision forests,” *IEEE transactions on pattern analysis and machine intelligence*, 20(8), pp. 832–844.

Hosseini, Z.Z. and Mohammadzadeh, M. (2016) “Knowledge discovery from patients’ behavior via clustering-classification algorithms based on weighted eRFM and CLV model: An empirical study in public health care services,” *Iranian journal of pharmaceutical research: IJPR*, 15(1), p. 355.

http://www.celerant.com/ (2016) *Data Mining: “Big opportunities for small to mid-sized retailers.”*

Hwang, H., Jung, T. and Suh, E. (2004) “An LTV model and customer segmentation based on customer value: a case study on the wireless telecommunication industry,” *Expert Systems with Applications*, 26(2), pp. 181–188. Available at: https://doi.org/https://doi.org/10.1016/S0957-4174(03)00133-7.

Jason Brownlee (2021) “How to Develop Voting Ensembles With Python,” *machine learning mastery*, 27 April.

Jiang, L.A., Yang, Z. and Jun, M. (2013) “Measuring consumer perceptions of online shopping convenience,” *Journal of Service management* [Preprint].

Johnson, T. and Dasu, T. (2003) “Data quality and data cleaning,” in *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data - SIGMOD ’03*. New York, New York, USA: ACM Press, p. 681. Available at: https://doi.org/10.1145/872757.872875.

Jolliffe, I.T. (2002) *Principal component analysis for special types of data*. Springer.

Kansal, T. *et al.* (2018) “Customer segmentation using K-means clustering,” in *2018 international conference on computational techniques, electronics and mechanical systems (CTEMS)*. IEEE, pp. 135–139.

Karan Kashyap (2019) “Machine Learning- Decision Trees and Random Forest Classifiers,” 21 October.

Kashwan, K.R. and Velu, C.M. (2013) “Customer segmentation using clustering and data mining techniques,” *International Journal of Computer Theory and Engineering*, 5(6), p. 856.

Kassambara, A. (2017) “Determining the optimal number of clusters: 3 must know methods,” *Available onli ne: https://www. datanovia. com/en/lessons/determiningthe-optimal-number-of-clusters-3-must-know-methods/.(accessed on 31 April 2018)* [Preprint].

Khan, J.A. and Jain, N. (2016) “A survey on intrusion detection systems and classification techniques,” *Int J Sci Res Sci Eng Technol*, 2(5), pp. 202–208.

Khan, N. *et al.* (2014) “Big Data: Survey, Technologies, Opportunities, and Challenges,” *The Scientific World Journal*, 2014, pp. 1–18. Available at: https://doi.org/10.1155/2014/712826.

Kim, J. (2000) “e-CRM for e-Business.” Gurm.

Kim, S. and Stoel, L. (2004) “Apparel retailers: website quality dimensions and satisfaction,” *Journal of retailing and consumer services*, 11(2), pp. 109–117.

Kumar, U.K., Nikhil, M.B.S. and Sumangali, K. (2017) “Prediction of breast cancer using voting classifier technique,” in *2017 IEEE international conference on smart technologies and management for computing, communication, controls, energy and materials (ICSTM)*. IEEE, pp. 108–114.

Laaksonen, J. and Oja, E. (1996) “Classification with learning k-nearest neighbors,” in *Proceedings of international conference on neural networks (ICNN’96)*. IEEE, pp. 1480–1483.

Labrinidis, A. and Jagadish, H. v. (2012) “Challenges and opportunities with big data,” *Proceedings of the VLDB Endowment*, 5(12), pp. 2032–2033. Available at: https://doi.org/10.14778/2367502.2367572.

Li, X. and Li, C. (2018) “The research on customer classification of B2C platform based on k-means algorithm,” in *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE, pp. 1871–1874.

Likas, A., Vlassis, N. and Verbeek, J.J. (2003) “The global k-means clustering algorithm,” *Pattern recognition*, 36(2), pp. 451–461.

LONGBING CAO (2017) “Data Science: A Comprehensive Overview,” *ACM Computing Surveys*, 50(3).

Magento. An Introduction to Customer Segmentation (2014). info2.magento.com/.../ An_Introduction_to_Customer_Segmentation.

maithili joshi (2019) “A Comparison between Linear and Logistic Regression,” *medium.com*, 4 April.

Marcus, C. (1998) “A practical yet meaningful approach to customer segmentation,” *Journal of Consumer Marketing*, 15(5), pp. 494–504. Available at: https://doi.org/10.1108/07363769810235974.

Maria Navin, J.R. and Pankaja, R. (2016) “Performance analysis of text classification algorithms using confusion matrix,” *International Journal of Engineering and Technical Research (IJETR)*, 6(4), pp. 75–78.

Marisa, F. *et al.* (2019) “Segmentation model of customer lifetime value in small and medium enterprise (SMEs) using K-means clustering and LRFM model,” *International Journal of Integrated Engineering*, 11(3).

Mark A. Beyer and Douglas Laney (2012) “The Importance of ‘Big Data’: A Definition.”

Massnick, F. (1997) *The customer is CEO: How to measure what your customers want--and make sure they get it*. Amacom Books.

McKinsey (2011) “Big Data: The Next Frontier for Innovation, Competition, and Productivity.”

Moeini, M. and Alizadeh, S.H. (2016) “Proposing a new model for determining the customer value using RFM model and its developments (case study on the Alborz insurance company),” *J. Eng. Appl. Sci*, 100(4), pp. 828–836.

Monil, P. *et al.* (2020) “Customer Segmentation Using Machine Learning,” *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*, 8(6), pp. 2104–2108.

Moorthy, K.S. (1984) “Market segmentation, self-selection, and product line design,” *Marketing Science*, 3(4), pp. 288–307.

Morganosky, M.A. and Cude, B.J. (2000) “Consumer response to online grocery shopping,” *International Journal of Retail & Distribution Management* [Preprint].

Müller, O., Fay, M. and vom Brocke, J. (2018) “The effect of big data and analytics on firm performance: An econometric analysis considering industry characteristics,” *Journal of Management Information Systems*, 35(2), pp. 488–509.

Murphy, K.P. (2012) *Machine learning: a probabilistic perspective*. MIT press.

Nagesh Singh Chauhan (2022) “Decision Tree Algorithm, Explained,” *KDnuggets* , 9 February.

Naib, M. and Chhabra, A. (2014) “Ensemble vote approach for predicting primary tumors using data mining,” in *2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence)*. IEEE, pp. 97–102.

Neha, Neha K.S and Prof Santhosh Rebello (2016) “OPINION MINING ON BRAND AIMIT USING SUPPORT VECTOR MACHINE,” *International Journal of Latest Trends in Engineering and Technology*, Special Edition.

Niaksu, O. (2015) “CRISP Data Mining Methodology Extension for Medical Domain,” *Baltic J. Modern Computing*, 3, pp. 92–109.

Nielsen, O.B. (2017) “A comprehensive review of data governance literature,” *Selected Papers IRIS*, 8, pp. 120–133.

Ogbuabor, G. and Ugwoke, F.N. (2018) “Clustering algorithm for a healthcare dataset using silhouette score value,” *International Journal of Computer Science & Information Technology*, 102(2018), pp. 27–37.

Onel Harrison (2018a) “Machine Learning Basics with the K-Nearest Neighbors Algorithm,” *Towards Data Science*, 10 September.

Onel Harrison (2018b) “Machine Learning Basics with the K-Nearest Neighbors Algorithm.”

Oswal, N. (2019) “Predicting rainfall using machine learning techniques,” *arXiv preprint arXiv:1910.13827* [Preprint].

Ozan, Ş. (2018) “A case study on customer segmentation by using machine learning methods,” in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE, pp. 1–6.

Paluszynska, A., Biecek, P. and Jiang, Y. (2017) “randomForestExplainer: Explaining and visualizing random forests in terms of variable importance,” *R package version 0.9* [Preprint].

Panuš, J. *et al.* (2016) “Customer segmentation utilization for differentiated approach,” in *2016 International Conference on Information and Digital Technologies (IDT)*. IEEE, pp. 227–233.

Patel, Y.S., Agrawal, D. and Josyula, L.S. (2016) “The RFM-based ubiquitous framework for secure and efficient banking,” in *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*. IEEE, pp. 283–288.

Paul, F. *et al.* (2010) “The Definitive Guide to Measuring Marketing Performance,” *Marketing Metrics* [Preprint].

Peacock, P.R. (1998) “Data mining in marketing: Part 1,” *Marketing Management*, 6(4), p. 8.

Pedregosa, F. *et al.* (2011) “Scikit-learn: Machine learning in Python,” *the Journal of machine Learning research*, 12, pp. 2825–2830.

Peker, S., Kocyigit, A. and Eren, P.E. (2017) “LRFMP model for customer segmentation in the grocery retail industry: a case study,” *Marketing Intelligence & Planning* [Preprint].

van den Poel, D. and Buckinx, W. (2005) “Predicting online-purchasing behaviour,” *European Journal of Operational Research*, 166(2), pp. 557–575. Available at: https://doi.org/10.1016/j.ejor.2004.04.022.

Pondel, M. and Korczak, J. (2018) “Collective clustering of marketing data-recommendation system Upsaily,” in *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, pp. 801–810.

Prashar, S., Vijay, T.S. and Parsad, C. (2015) “Antecedents to Online Shopping: Factors Influencing the Selection of Web Portal,” *International Journal of E-Business Research*, 11. Available at: https://link.gale.com/apps/doc/A416501429/AONE?u=nysl_oweb&sid=googleScholar&xid=976c530a.

Punhani, R. *et al.* (2021) “Application of Clustering Algorithm for Effective Customer Segmentation in E-Commerce,” in *2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, pp. 149–154. Available at: https://doi.org/10.1109/ICCIKE51210.2021.9410713.

quantstart (2020) *Beginner’s Guide to Decision Trees for Supervised Machine Learning*, *quantstart.com*.

Rahm, E. and Do, H.H. (2000) “Data cleaning: Problems and current approaches,” *IEEE Data Eng. Bull.*, 23(4), pp. 3–13.

Raihan-Al-Masud, M. and Mustafa, H.A. (2019) “Network Intrusion Detection System Using Voting Ensemble Machine Learning,” in *2019 IEEE International Conference on Telecommunications and Photonics (ICTP)*, pp. 1–4. Available at: https://doi.org/10.1109/ICTP48844.2019.9041736.

Ravasan, A.Z. and Mansouri, T. (2018) “A fuzzy ANP based weighted RFM model for customer segmentation in auto insurance sector,” in *Intelligent Systems: Concepts, Methodologies, Tools, and Applications*. IGI Global, pp. 1050–1067.

Reichheld, F.F. and Teal, T. (1996) “The loyalty effect, harvard business school press,” *Boston, MA* [Preprint].

Rezaeinia, S.M. and Rahmani, R. (2016) “Recommender system based on customer segmentation (RSCS),” *Kybernetes* [Preprint].

Rousseeuw, P.J. (1987) “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of computational and applied mathematics*, 20, pp. 53–65.

Roy, R. and Seitz, B. (2018) “How to build a data-first culture for a digital transformation,” *Digital McKinsey* [Preprint].

Safari, F., Safari, N. and Montazer, G.A. (2016) “Customer lifetime value determination based on RFM model,” *Marketing Intelligence & Planning* [Preprint].

Saito, T. and Rehmsmeier, M. (2015) “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” *PloS one*, 10(3), p. e0118432.

Schneider, G. (2016) *Electronic commerce*. Cengage Learning.

Shannon, C.E. (1948) “A note on the concept of entropy,” *Bell System Tech. J*, 27(3), pp. 379–423.

Sharda, R. *et al.* (2014) “Business intelligence and analytics,” *System for Decesion Support* [Preprint].

Sheikh, A., Ghanbarpour, T. and Gholamiangonabadi, 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*, 26(2), pp. 197–207.

Smeureanu, I., Ruxanda, G. and Badea, L.M. (2013) “Customer segmentation in private banking sector using machine learning techniques,” *Journal of Business Economics and Management*, 14(5), pp. 923–939. Available at: https://doi.org/10.3846/16111699.2012.749807.

Smith, W.R. (1956) “Product differentiation and market segmentation as alternative marketing strategies,” *Journal of marketing*, 21(1), pp. 3–8.

Son, J. *et al.* (2015) “Tracking-by-segmentation with online gradient boosting decision tree,” in *Proceedings of the IEEE international conference on computer vision*, pp. 3056–3064.

Steenkamp, J.-B.E.M. and ter Hofstede, F. (2002) “International market segmentation: issues and perspectives,” *International journal of research in marketing*, 19(3), pp. 185–213.

Sundhari, S.S. (2011) “A knowledge discovery using decision tree by Gini coefficient,” in *2011 International Conference on Business, Engineering and Industrial Applications*. IEEE, pp. 232–235.

Tangirala, S. (2020) “Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm,” *International Journal of Advanced Computer Science and Applications*, 11(2), pp. 612–619.

Tarokh, M.J. and EsmaeiliGookeh, M. (2019) “Modeling patient’s value using a stochastic approach: An empirical study in the medical industry,” *Computer methods and programs in biomedicine*, 176, pp. 51–59.

Thomas, J.W. (2007) “Market segmentation,” *Decision analyst*, pp. 1–4.

Tony Hey and Anne Trefethen (2003) “The Data Deluge: An e-Science Perspective,” pp. 809–824.

Treese, G.W. and Stewart, L.C. (2003) *Designing systems for Internet commerce*. Addison-Wesley Professional.

Vapnik, V.N. (1999) “An overview of statistical learning theory,” *IEEE transactions on neural networks*, 10(5), pp. 988–999.

Vladimir, Z. (1996) “Electronic commerce: structures and issues,” *International journal of electronic commerce*, 1(1), pp. 3–23.

Waller, M.A. and Fawcett, S.E. (2013) “Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management,” *Journal of Business Logistics*. Wiley Online Library, pp. 77–84.

Wang, Y. *et al.* (2017) “Improvement of ID3 algorithm based on simplified information entropy and coordination degree,” *Algorithms*, 10(4), p. 124.

Weng, C.-H. (2016) “Knowledge discovery of digital library subscription by RFC itemsets,” *The Electronic Library* [Preprint].

Wolfinbarger, M. and Gilly, M.C. (2003) “eTailQ: dimensionalizing, measuring and predicting etail quality,” *Journal of retailing*, 79(3), pp. 183–198.

Yoseph, F. and Heikkila, M. (2018) “Segmenting retail customers with an enhanced RFM and a hybrid regression/clustering method,” in *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*. IEEE, pp. 108–116.

Zakaria Jaadi (2022) *A Step-by-Step Explanation of Principal Component Analysis (PCA)*, *builtin*.

Zeeshan-Ul-Hassan Usman and Sana Rasheed (2021) “Pakistan’s Largest E-Commerce Dataset,” *https://www.kaggle.com/* [Preprint].

Zhang, J. *et al.* (2017) “K-means-clustering-based fiber nonlinearity equalization techniques for 64-QAM coherent optical communication system,” *Optics Express*, 25(22), pp. 27570–27580. Available at: https://doi.org/10.1364/OE.25.027570.

Zhou, B. *et al.* (2019) “Analysis of factors affecting hit-and-run and non-hit-and-run in vehicle-bicycle crashes: a non-parametric approach incorporating data imbalance treatment,” *Sustainability*, 11(5), p. 1327.

APPENDIX A - Figures

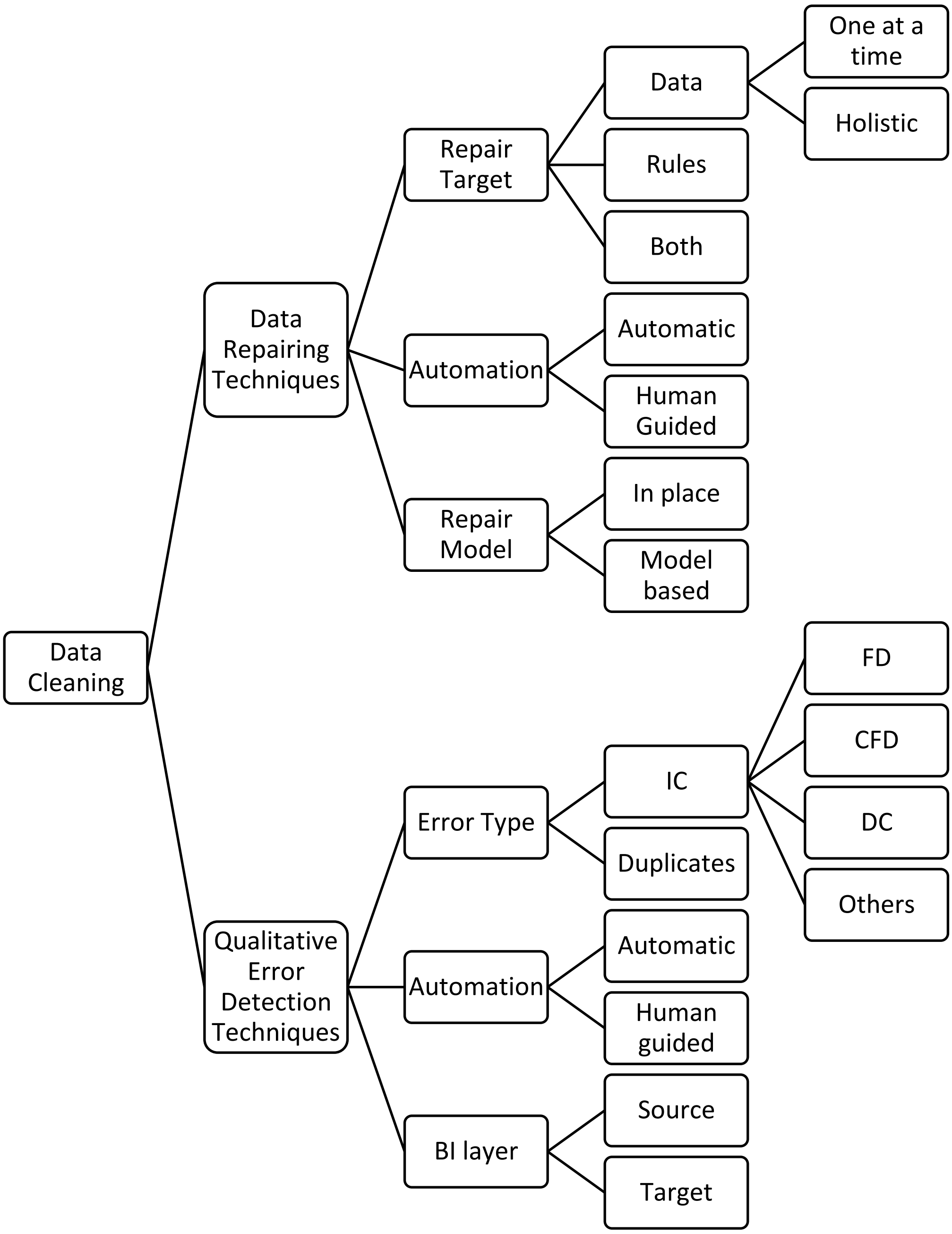

Figure 23 Data Cleaning detailed structure

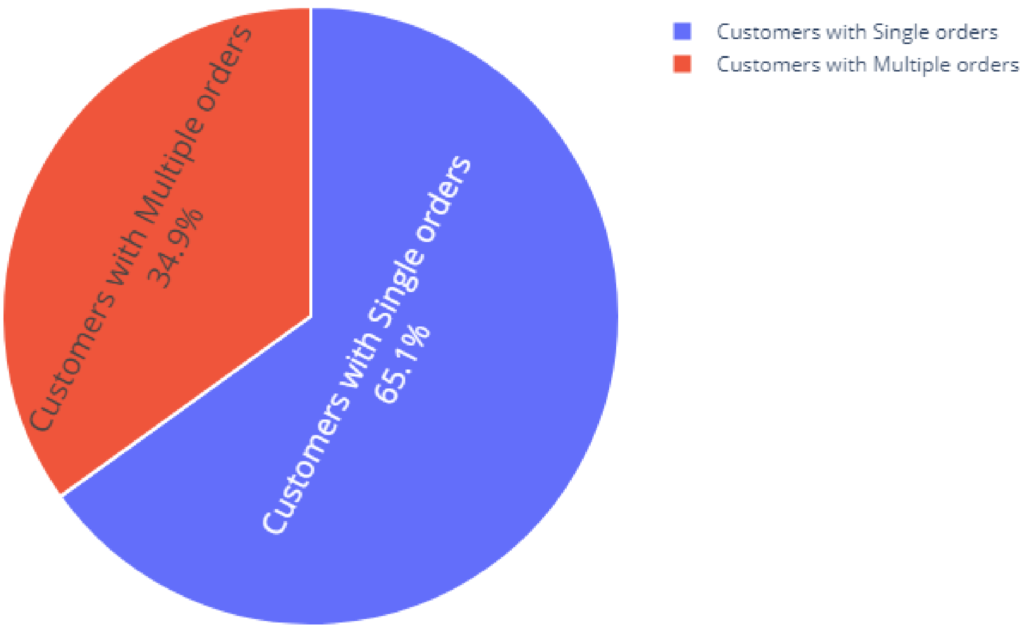

Figure 24 Output: Comparison of customers with single order vs multiple orders

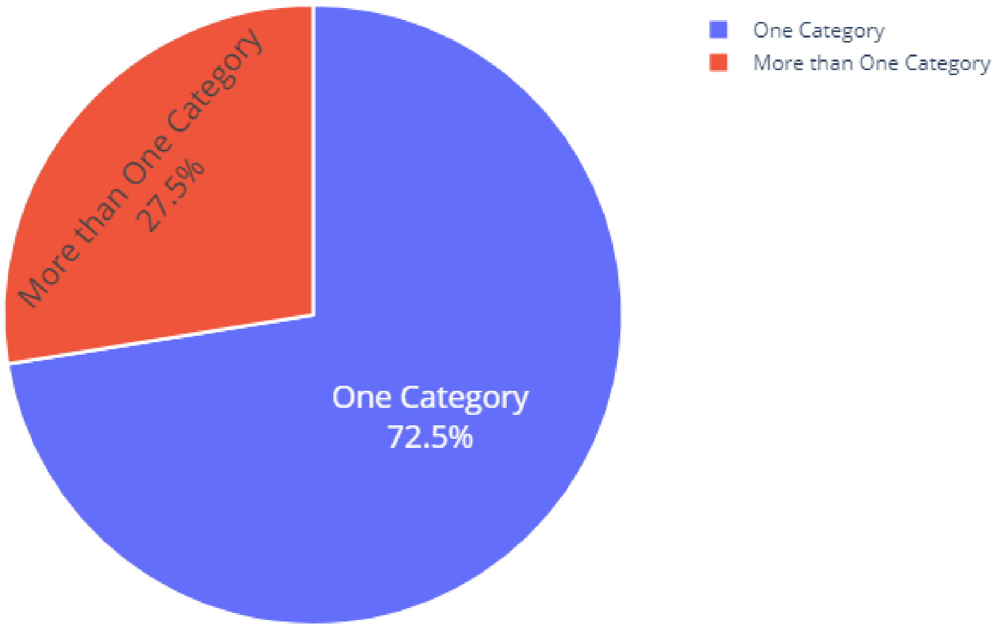

Figure 25 Output: Comparison of customers who bought from one category vs from multiple categories

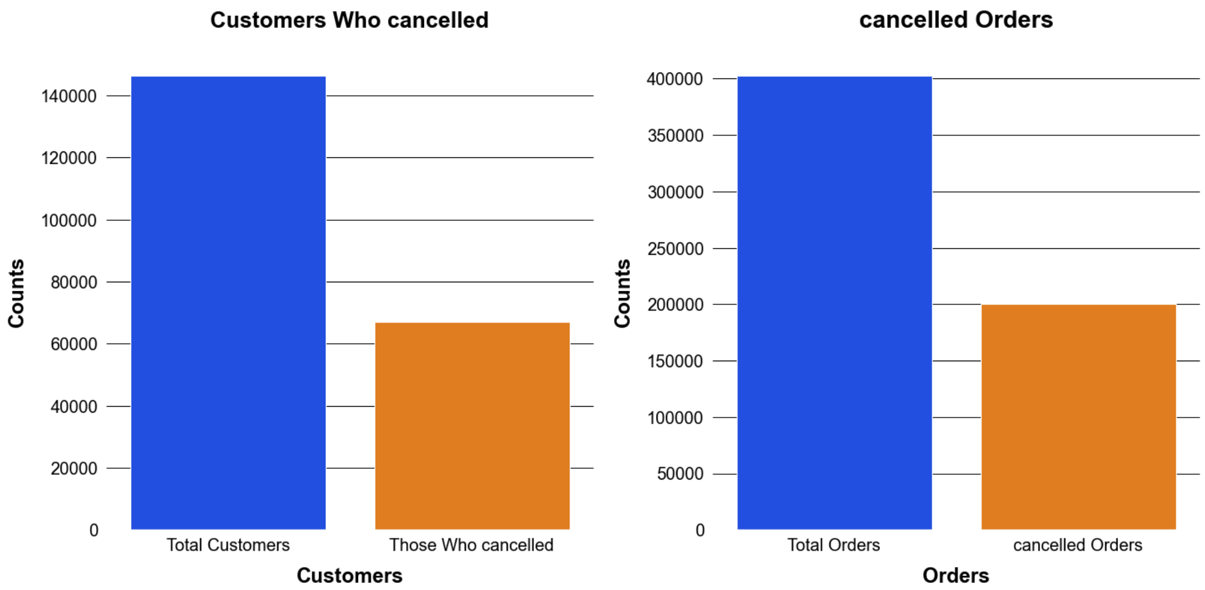

Figure 26 Output: total order cancelled vs customers whose orders got cancelled

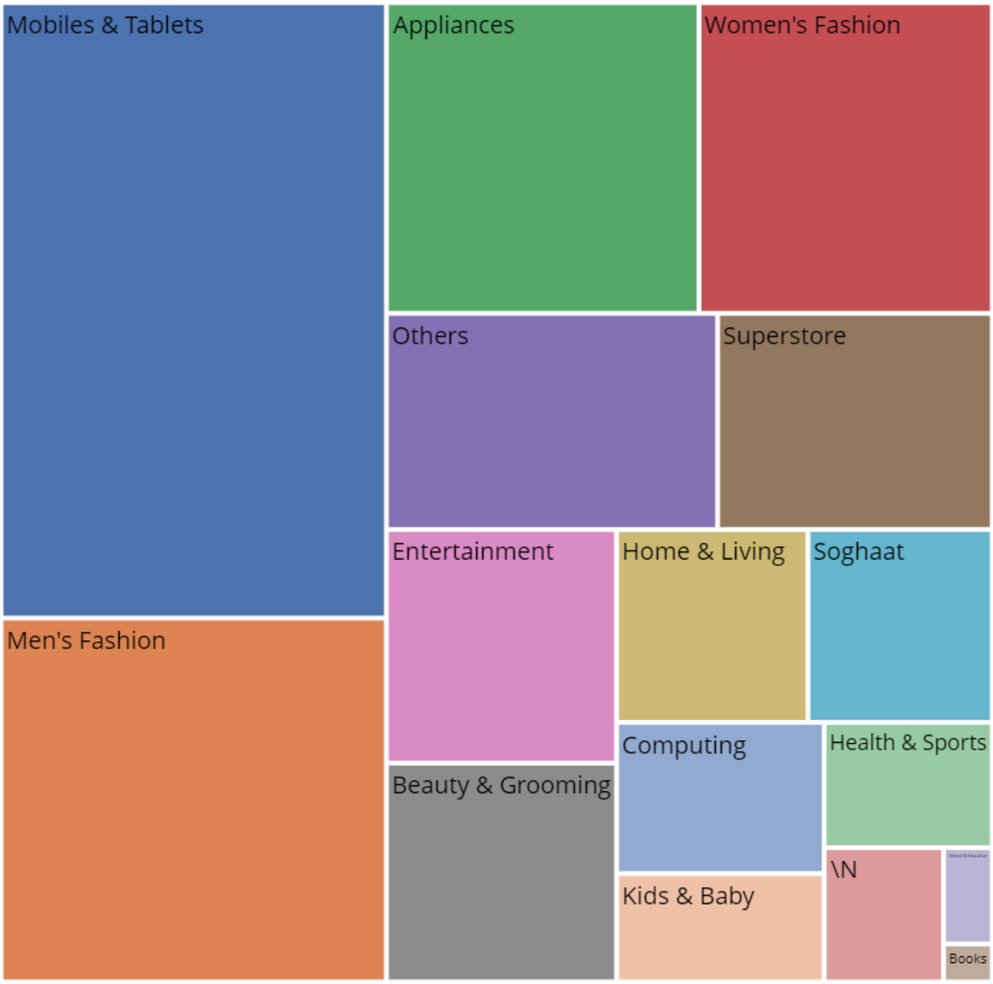

Figure 27 Output: Categories of cancelled orders

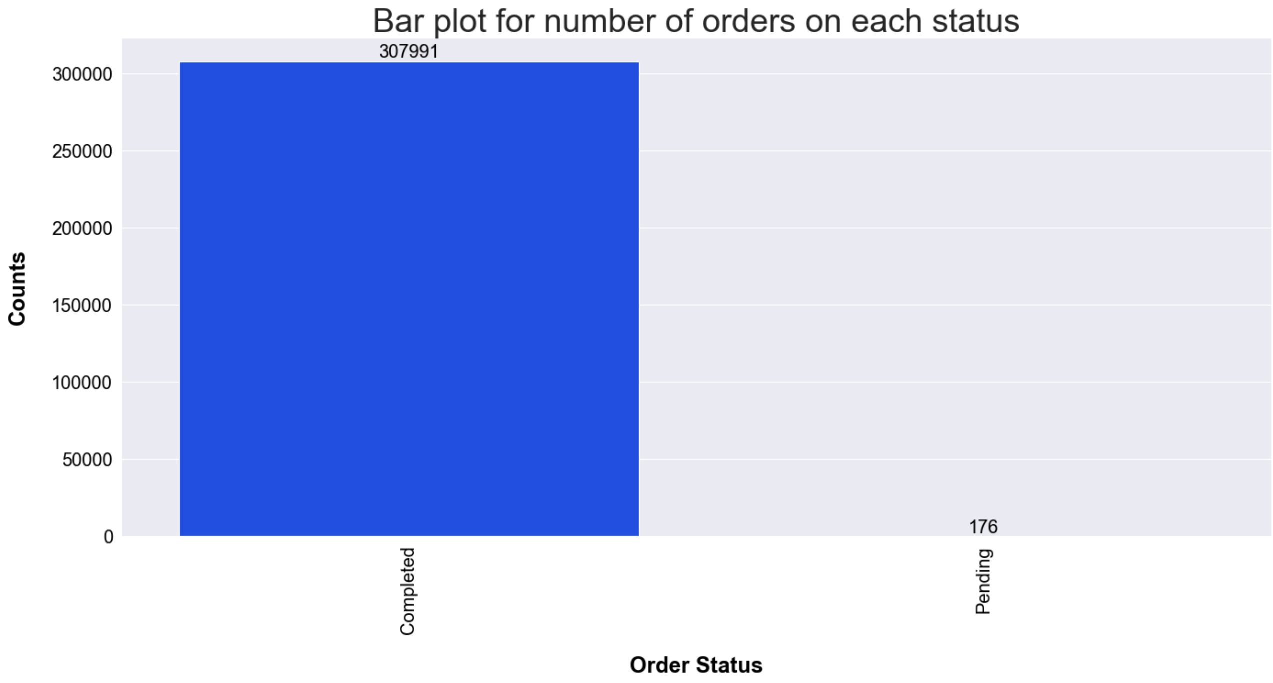

Figure 28 output: Total completed and pending orders

Category name	Category Number
Appliances	0
Beauty & Grooming	1
Books	2
Computing	3
Entertainment	4
Health & Sports	5
Home & Living	6
Kids & Baby	7
Men's Fashion	8
Mobiles & Tablets	9
Others	10
School & Education	11
Soghaat	12
Superstore	13
Women's Fashion	14

Figure 29 Output: Category name vs encoded number

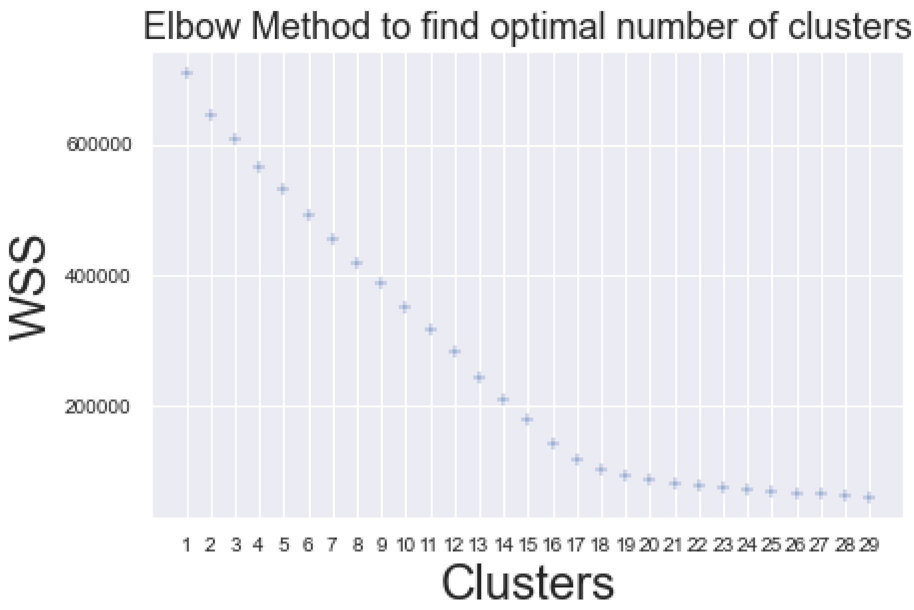

Figure 30 Output: Elbow Method to find the optimal number of clusters

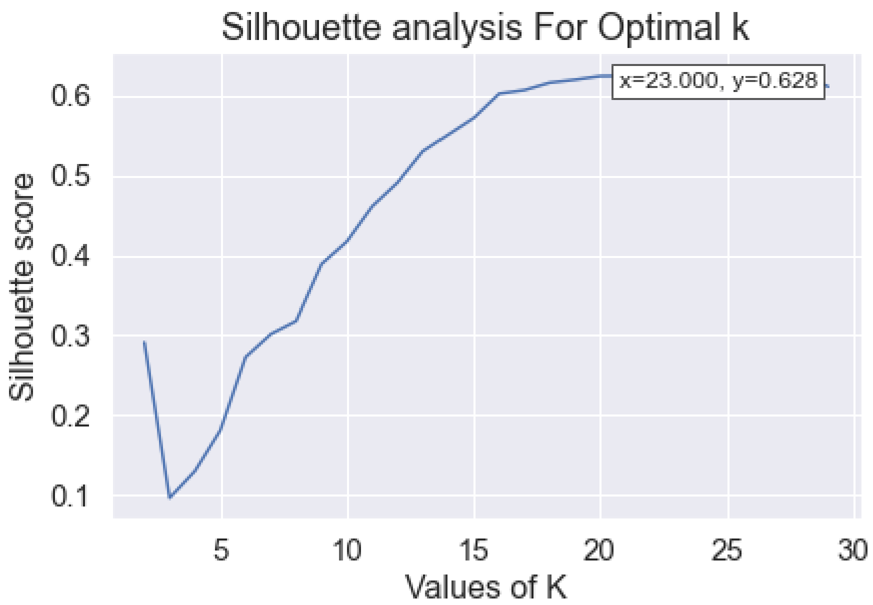

Figure 31 Output: Silhouette Score analysis

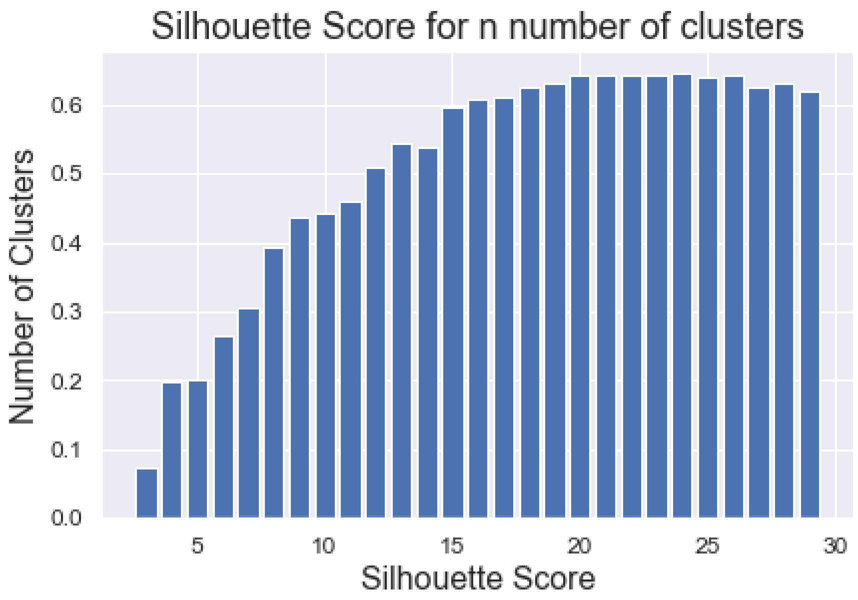

Figure 32 Output: Silhouette Score for n number of clusters

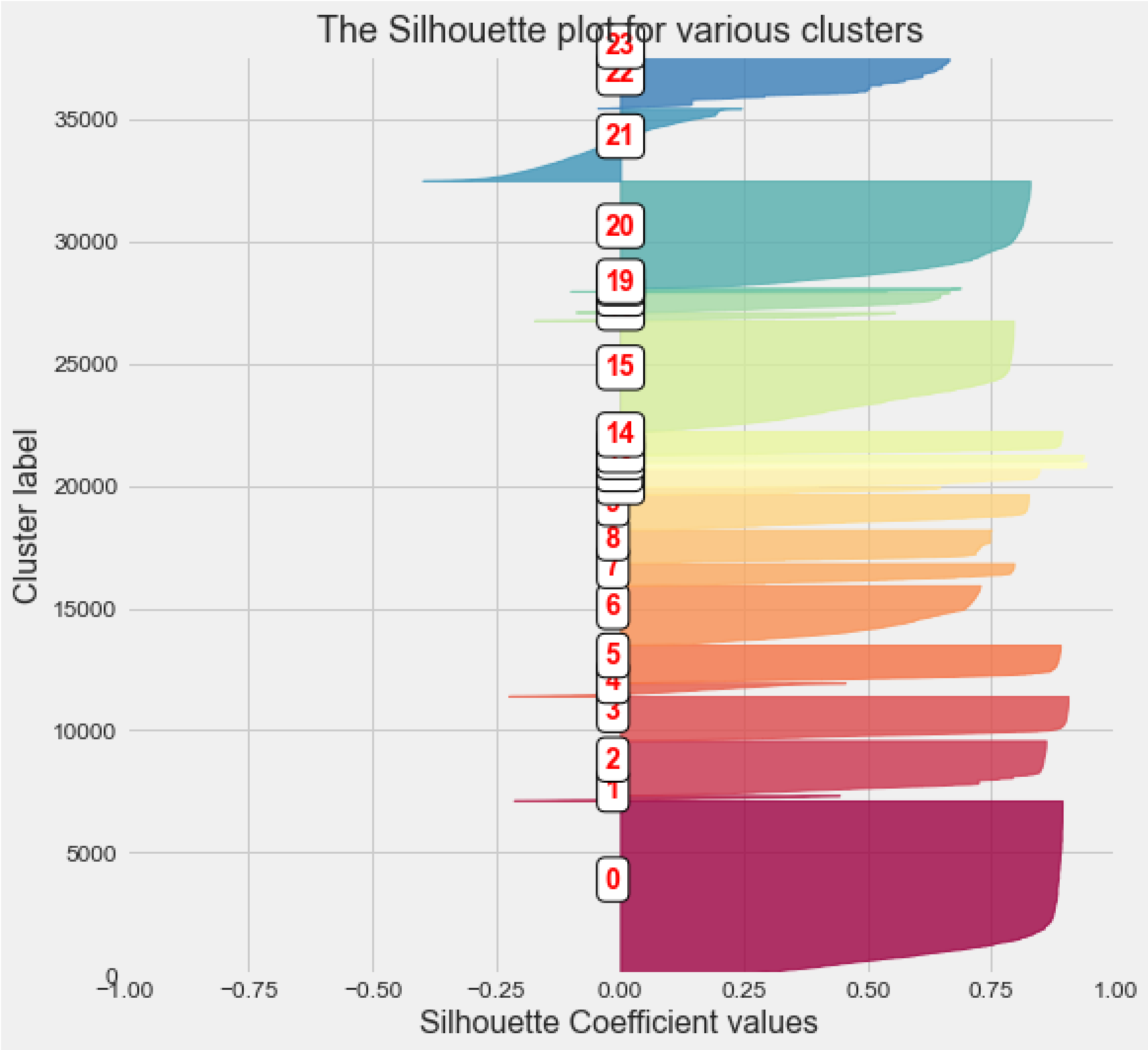

Figure 33 Output: Silhouette Intra cluster Score

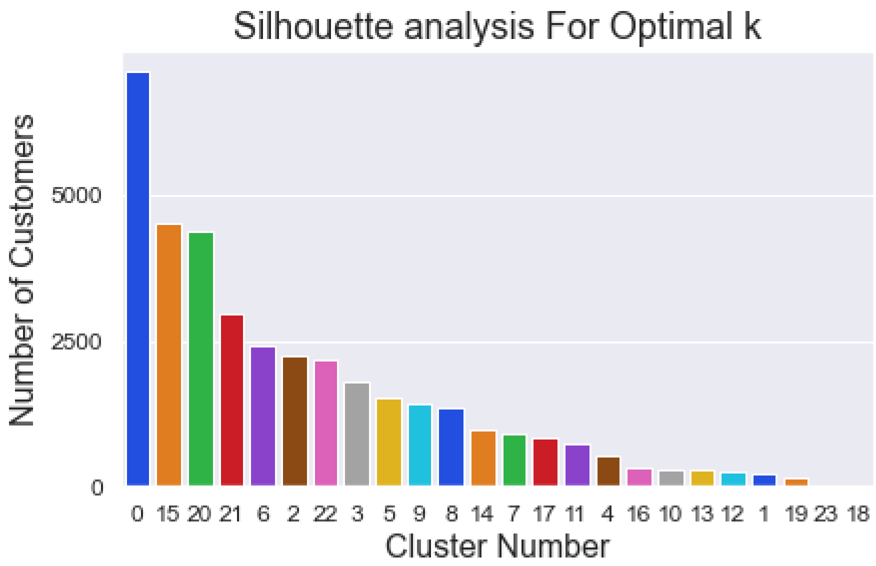

Figure 34 Output of clusters vs number of customers in each cluster

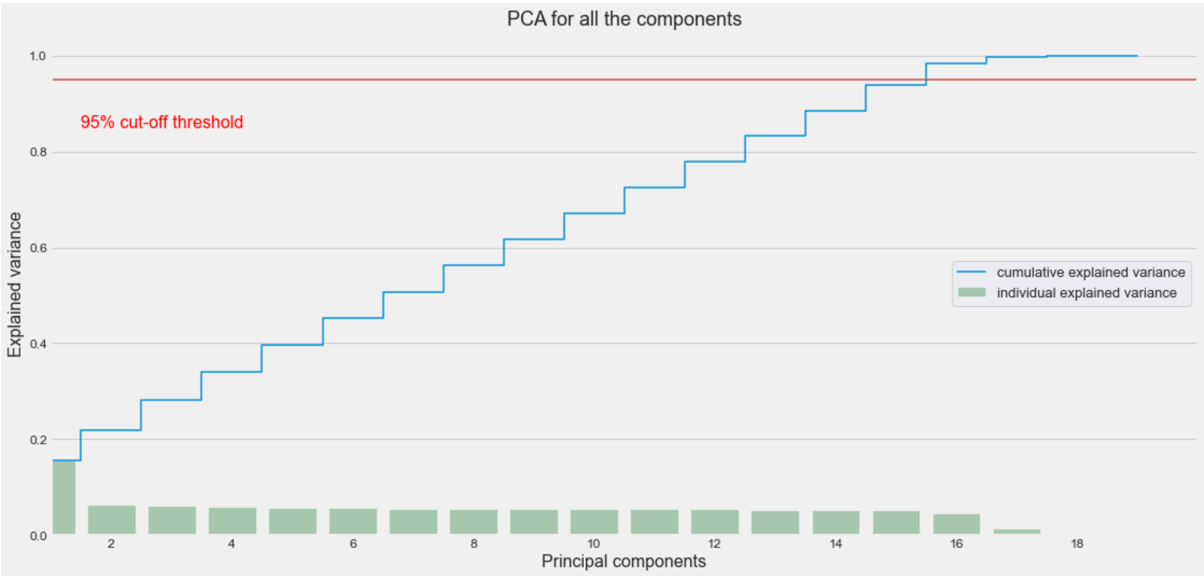

Figure 35 Output: PCA to find a variance

Figure 36 Output: PCA performed on six components

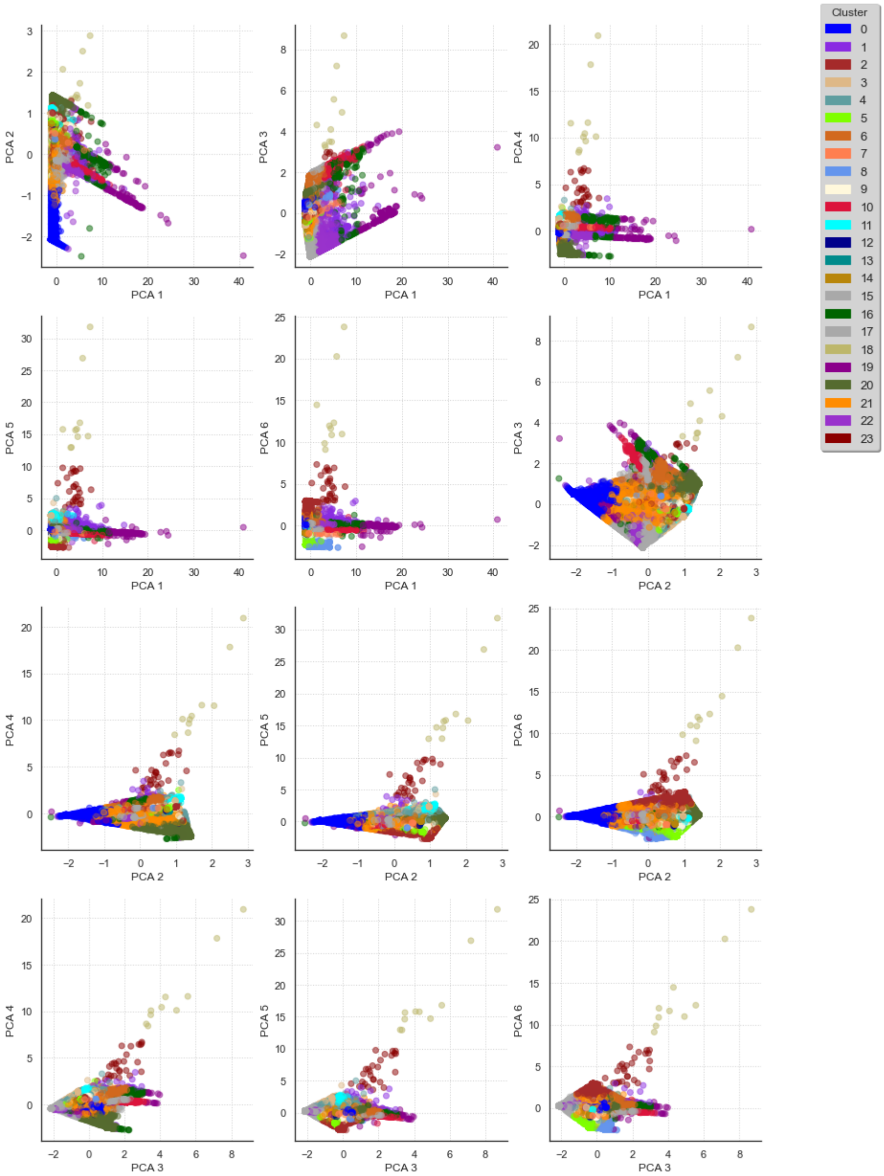

Figure 37 output clusters

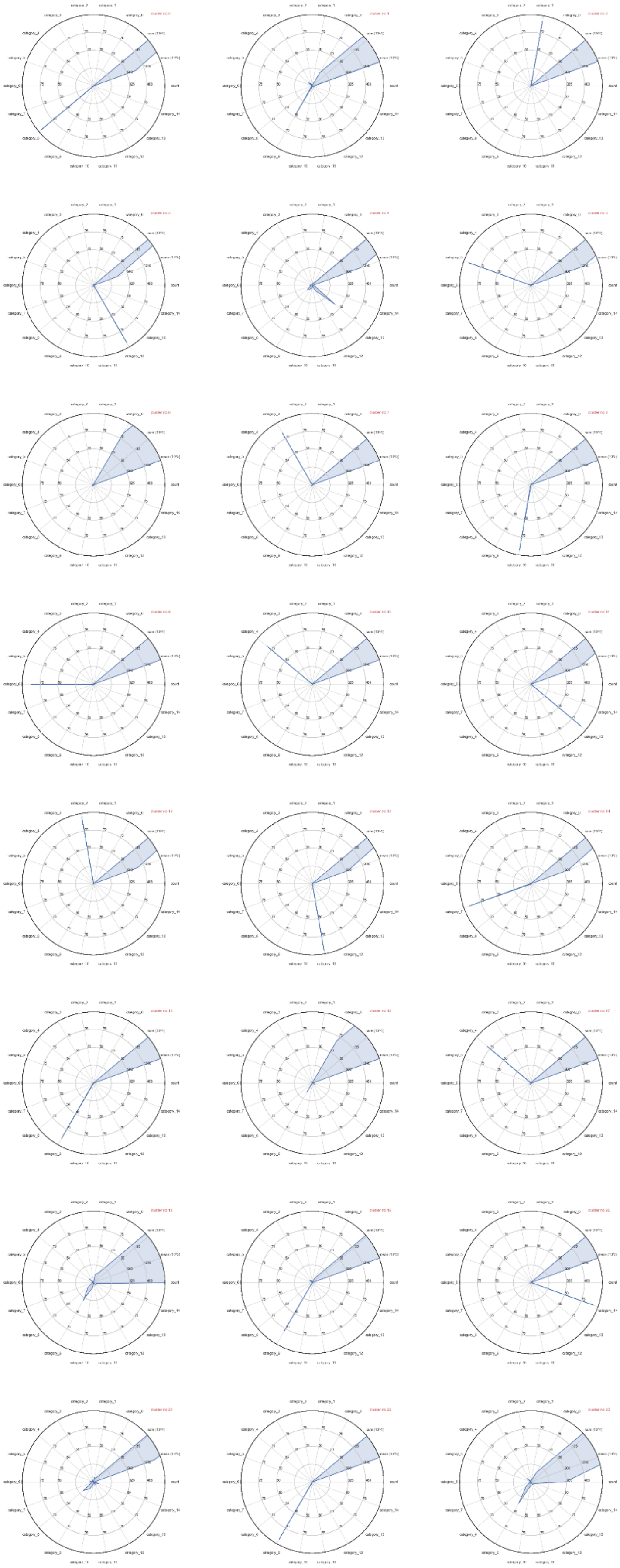

Figure 38 Output Cluster radar chart

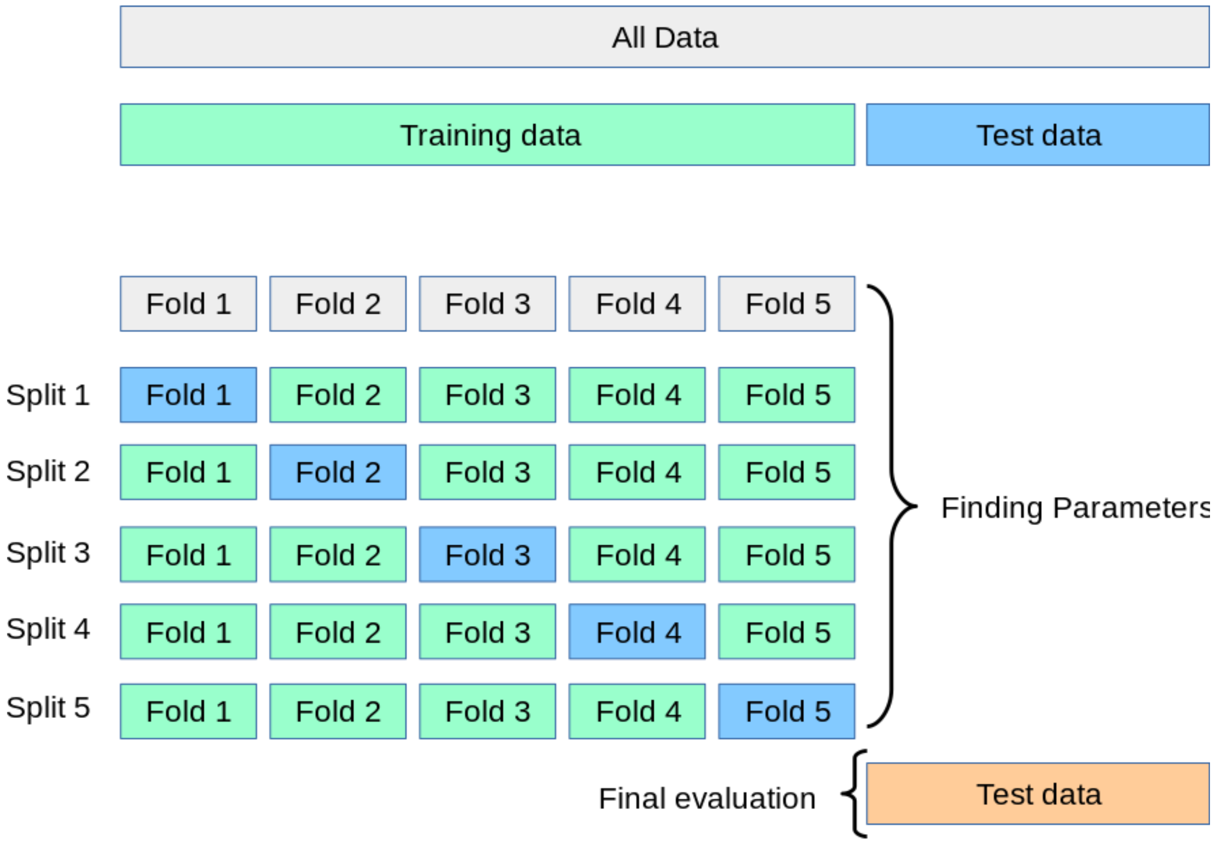

Figure 39 K-Fold structure
Source: (Pedregosa et al.*, 2011)*

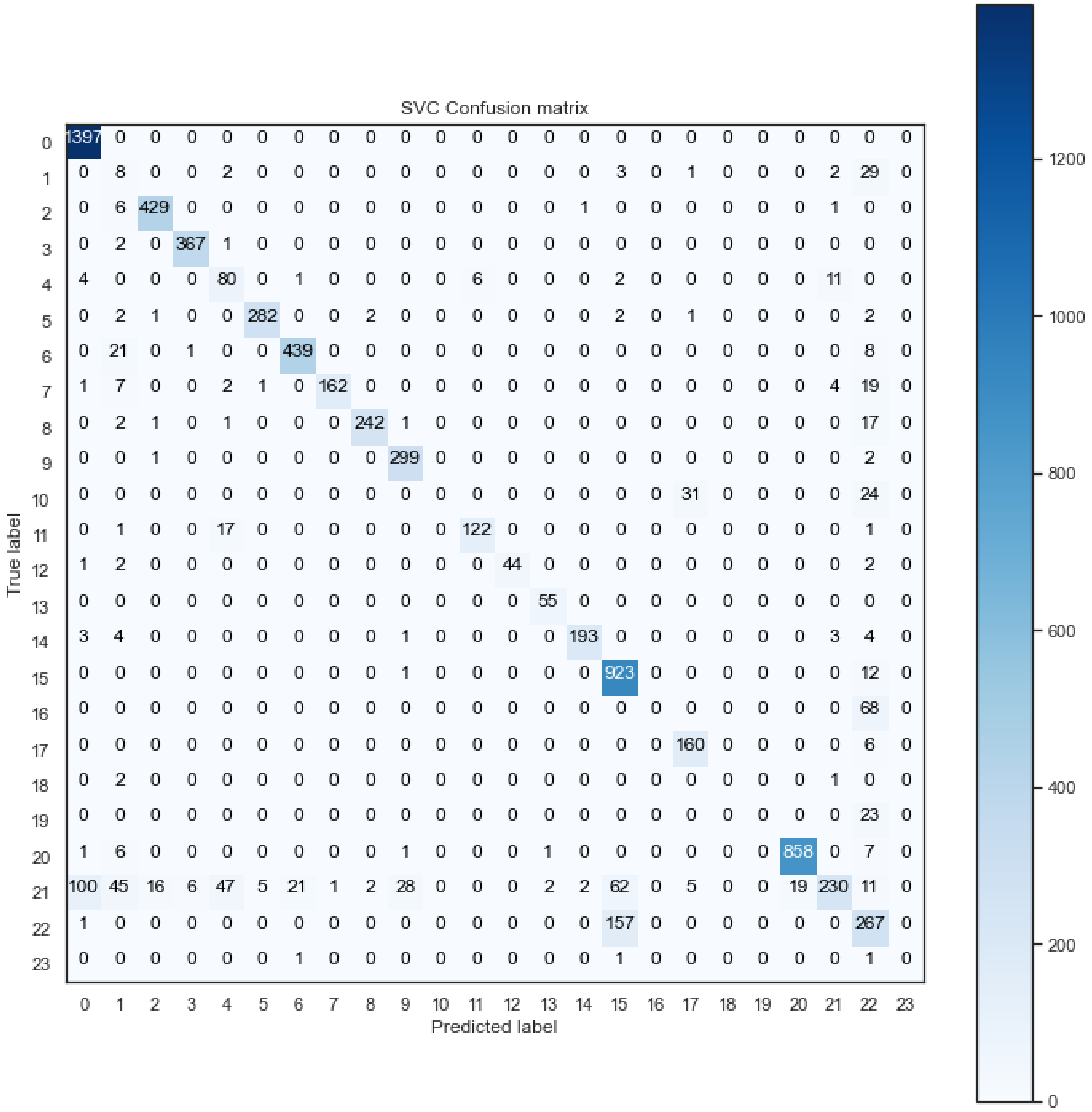

Figure 40 Output: SVC Confusion Matrix

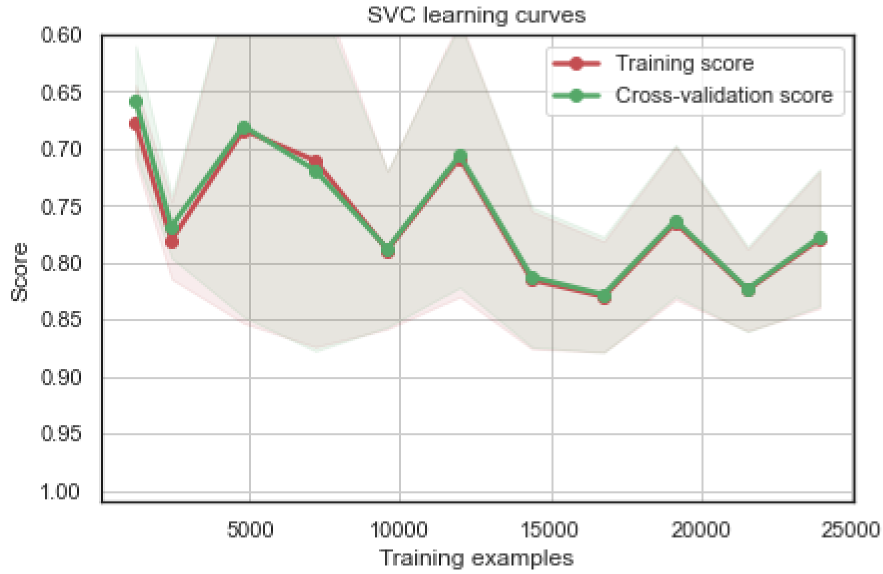

Figure 41 Output: SVC Learning Curves

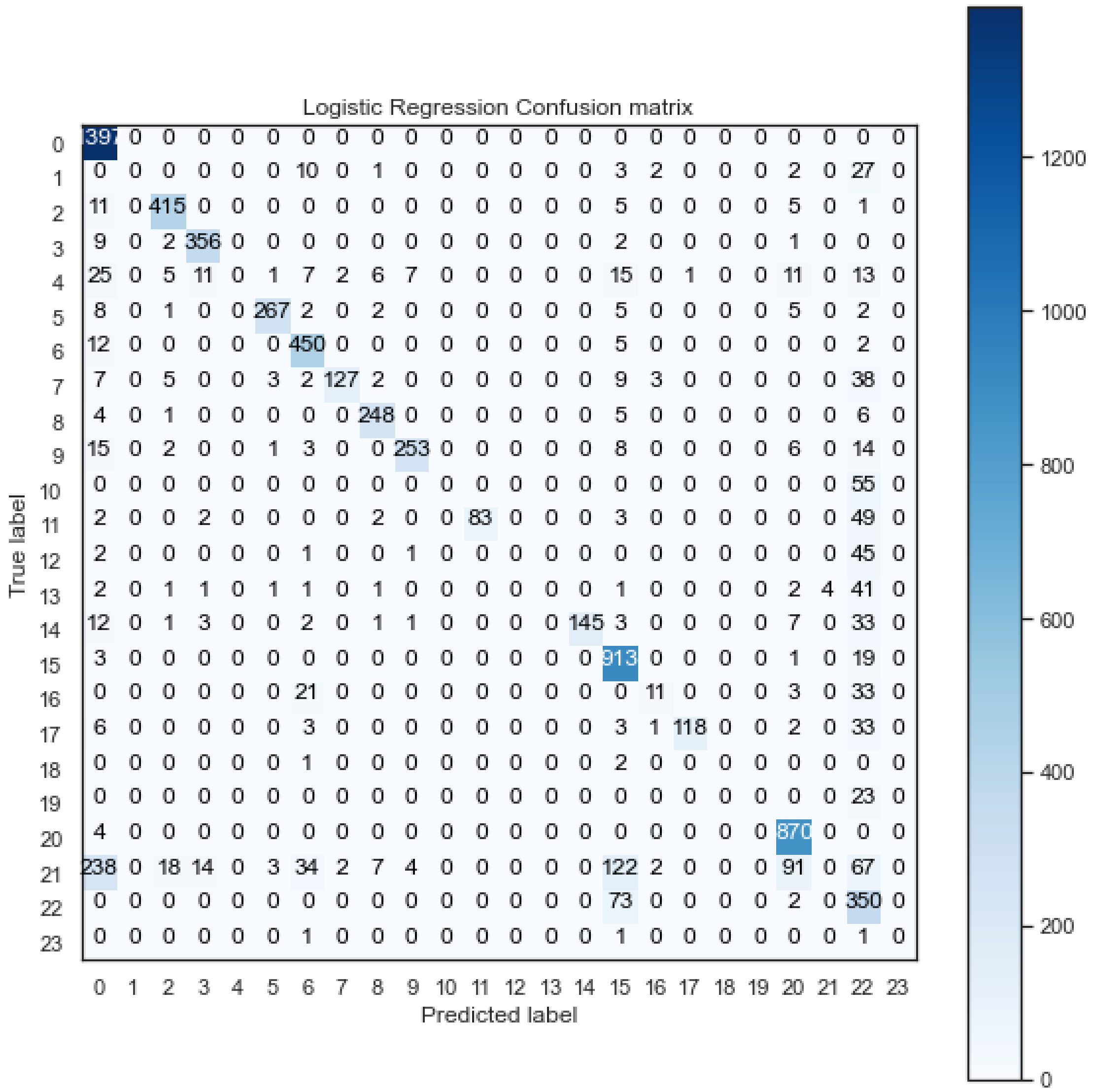

Figure 42 Output: Logistic Regression Confusion Matrix

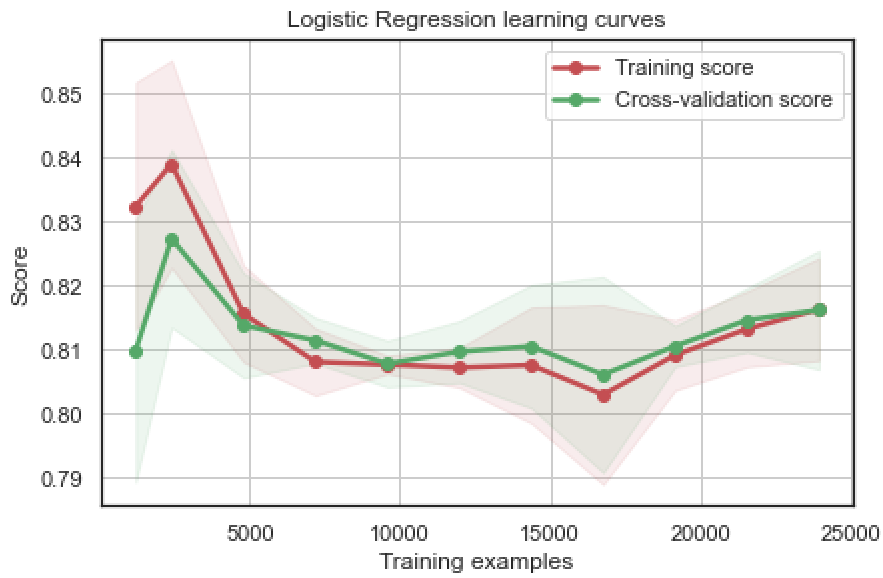

Figure 43 Output: Logistic Regression Learning Curves

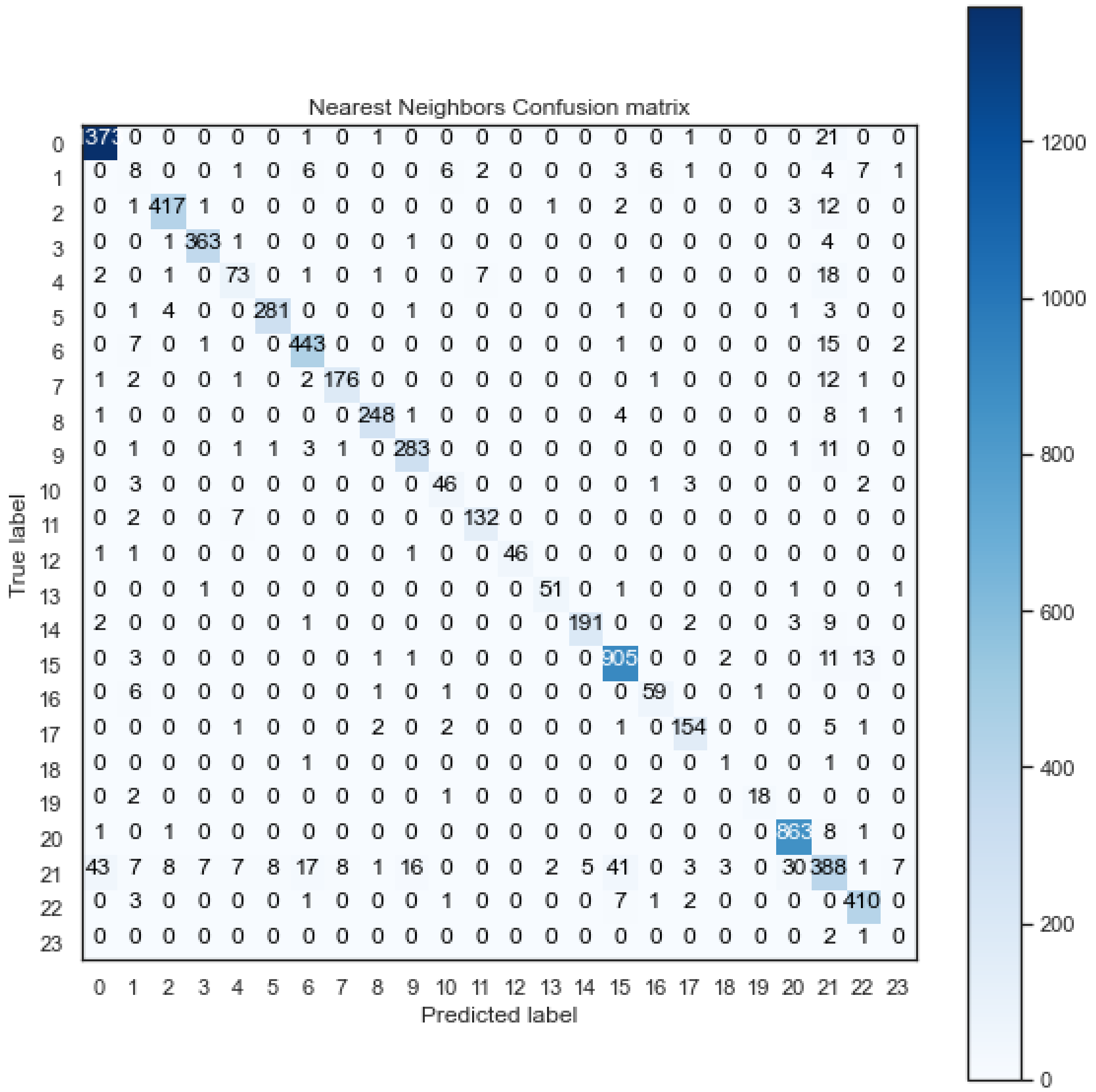

Figure 44 Output: Nearest Neighbour Confusion Matrix

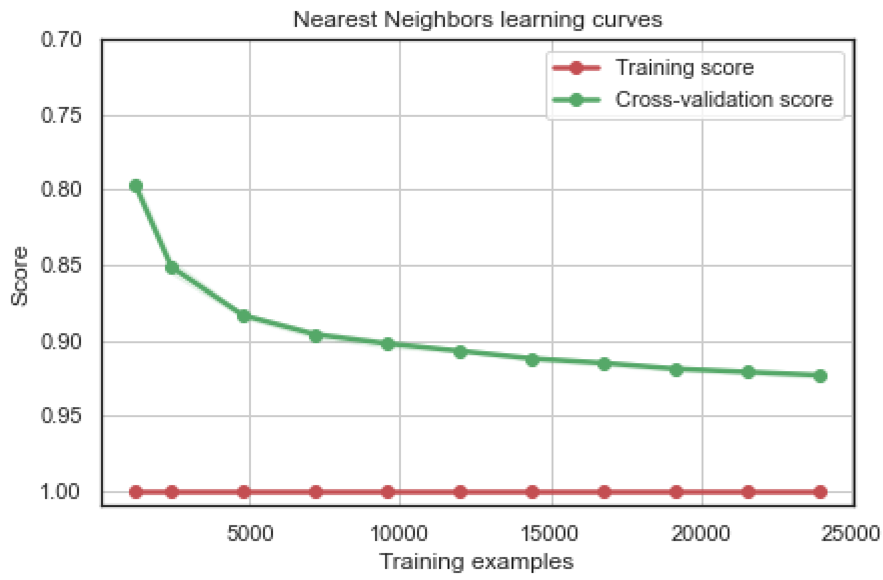

Figure 45 Output: Nearest Neighbour Learning Curves

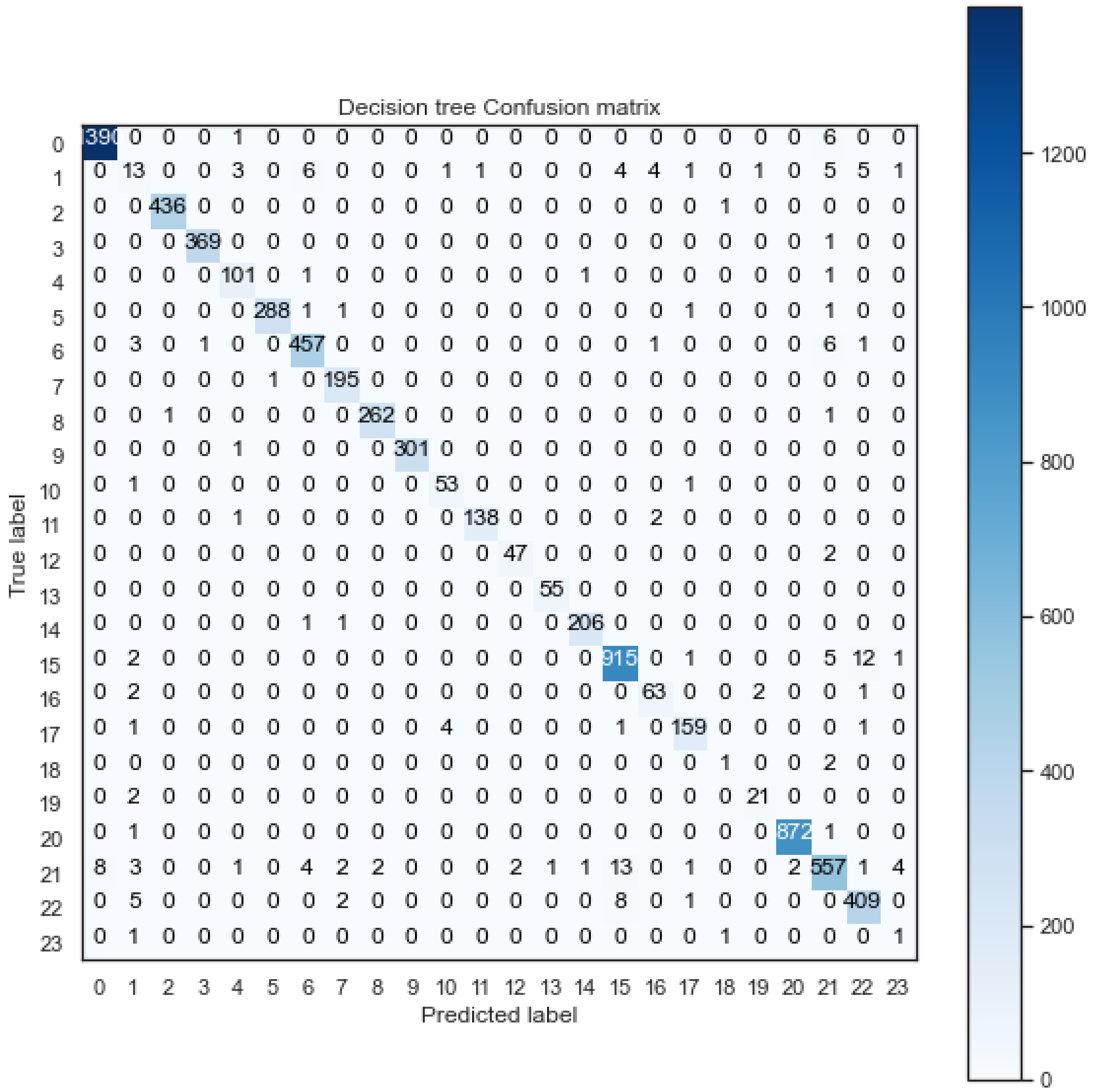

Figure 46 Output: Decision Tree Confusion Matrix

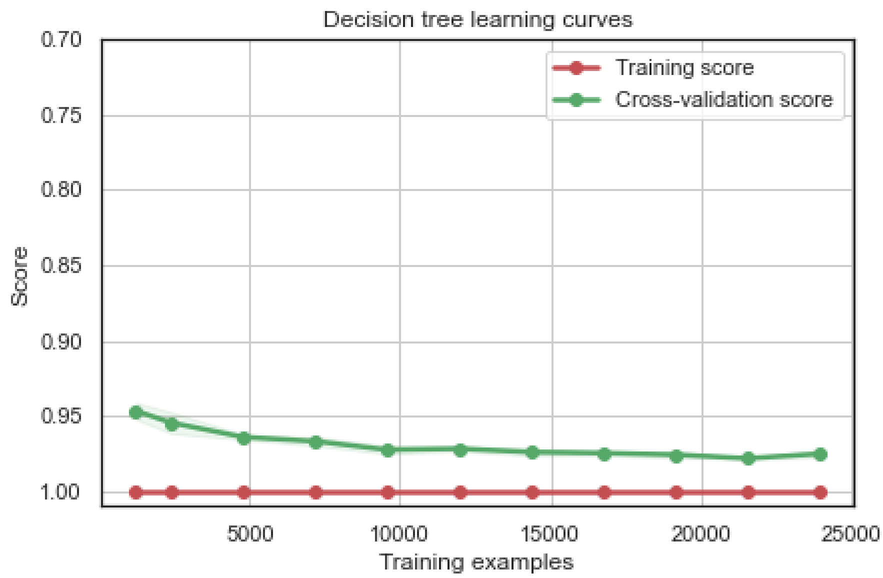

Figure 47 Output: Decision Tree Learning Curves

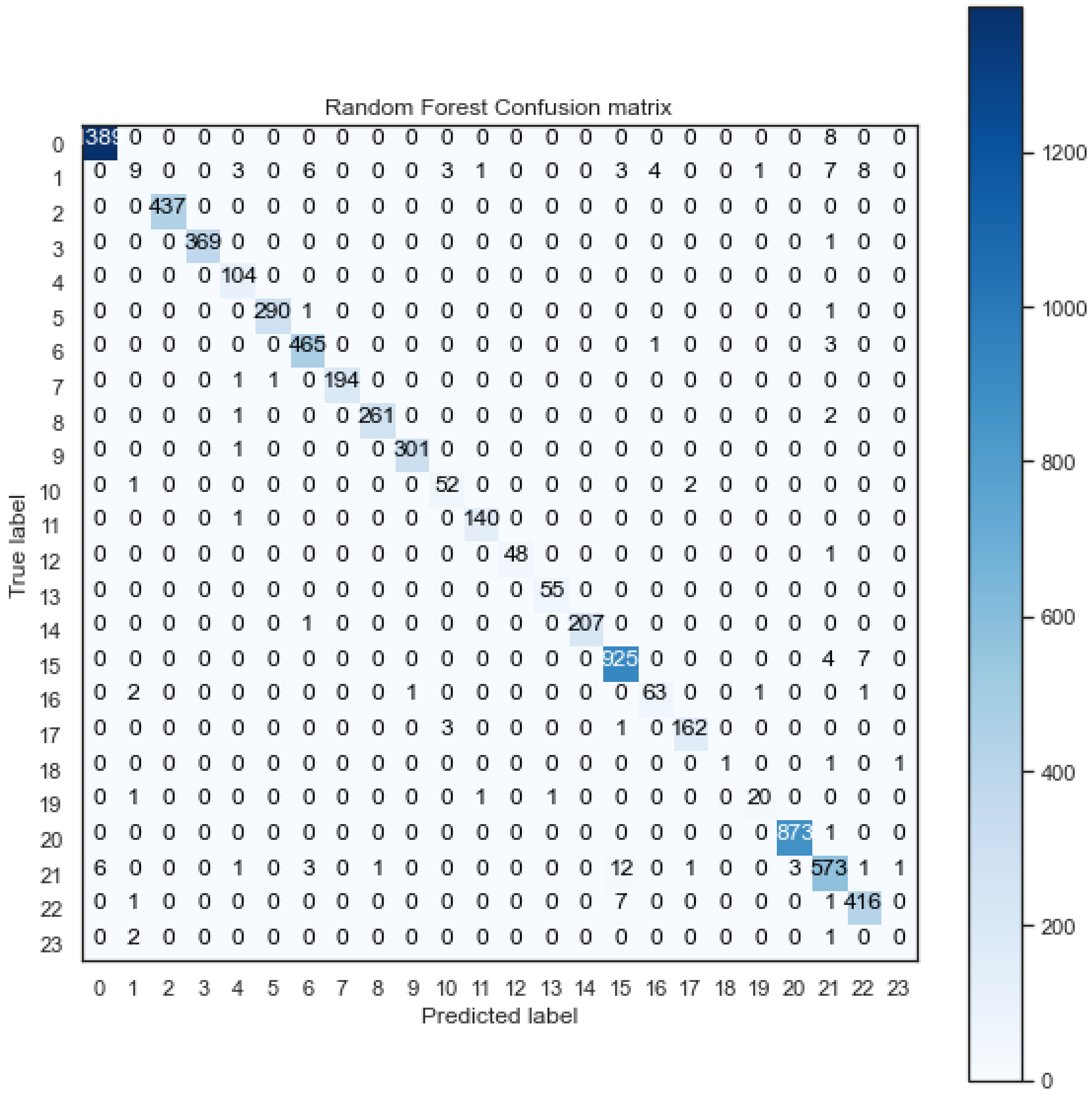

Figure 48 Output: Random Forest Confusion Matrix

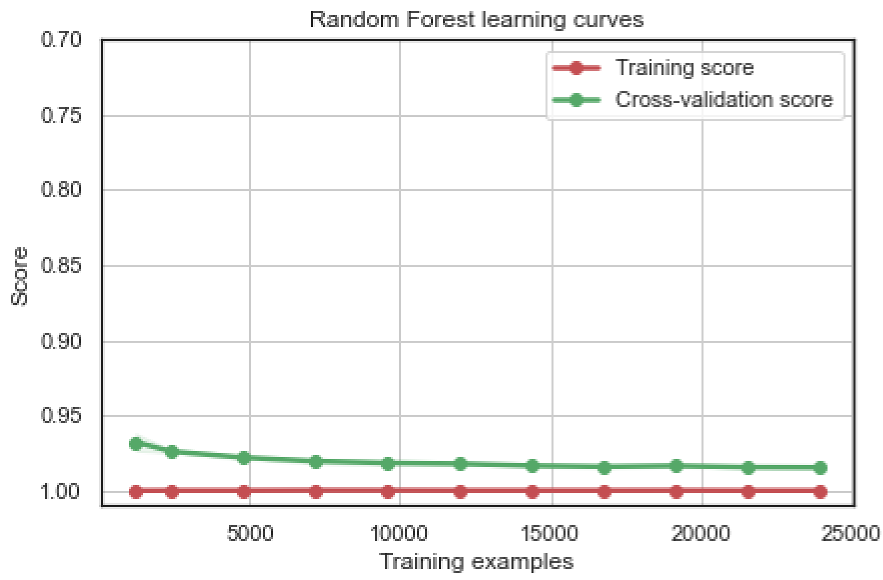

Figure 49 Output: Random Forest Learning Curves

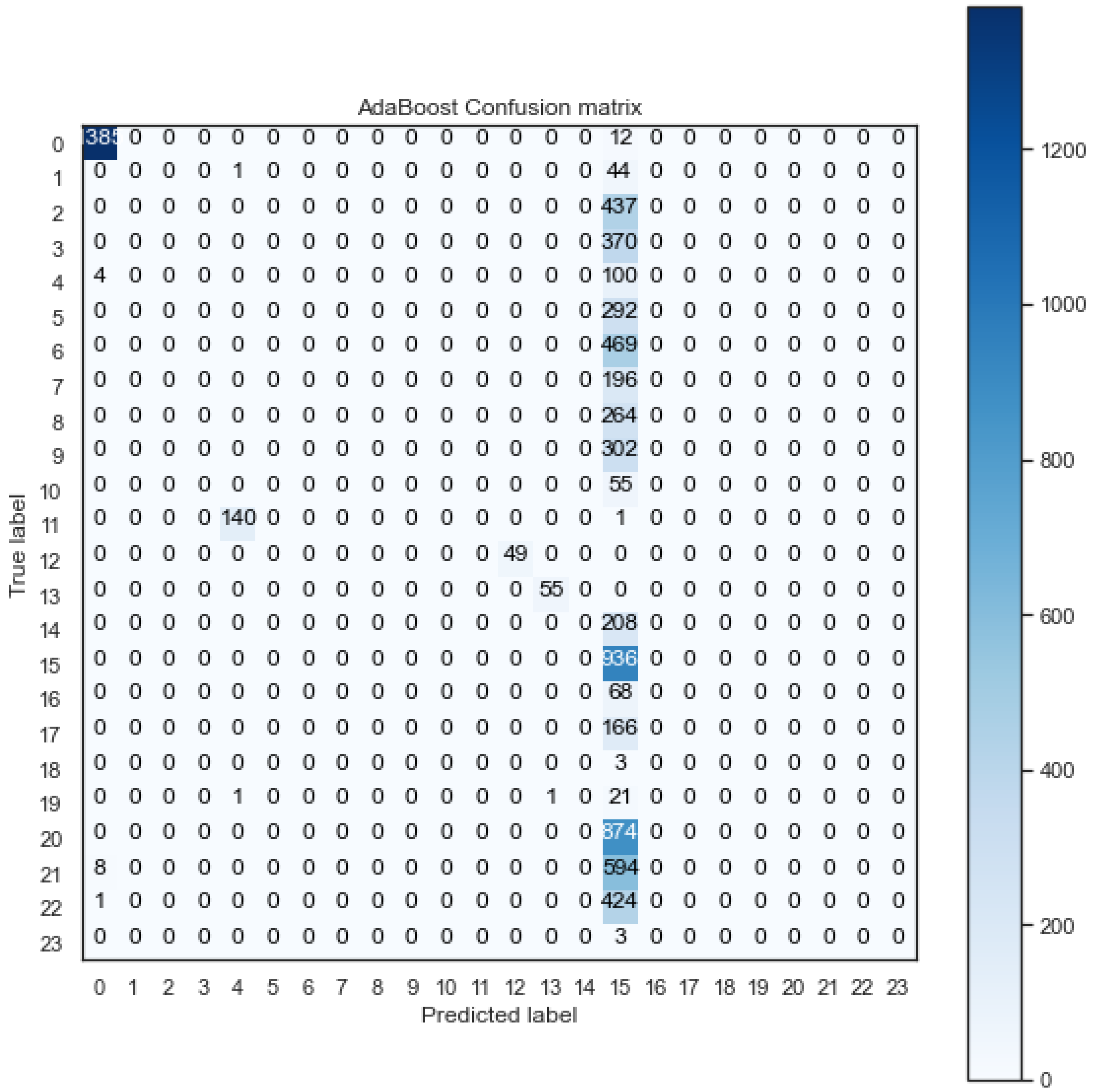

Figure 50 Output: Ada Boost Confusion Matrix

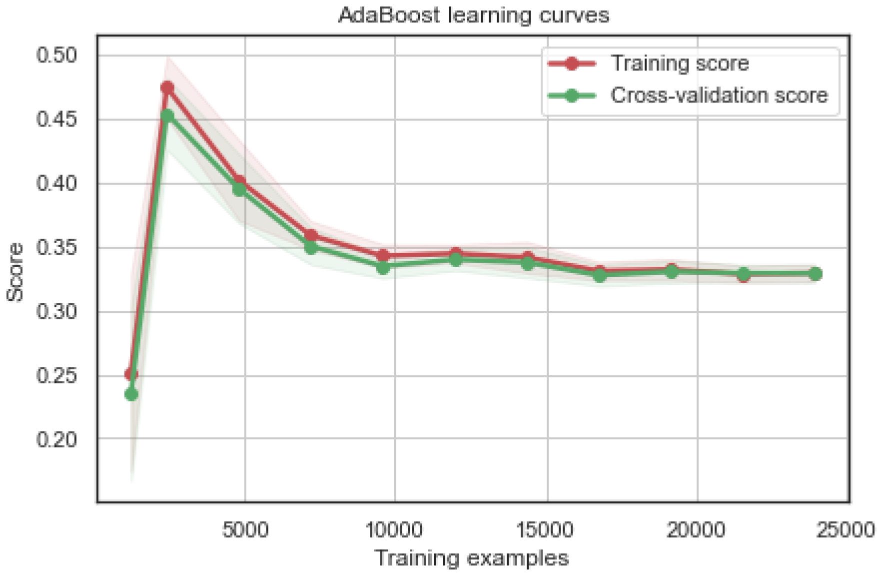

Figure 51 Output: Ada Boost Learning Curve

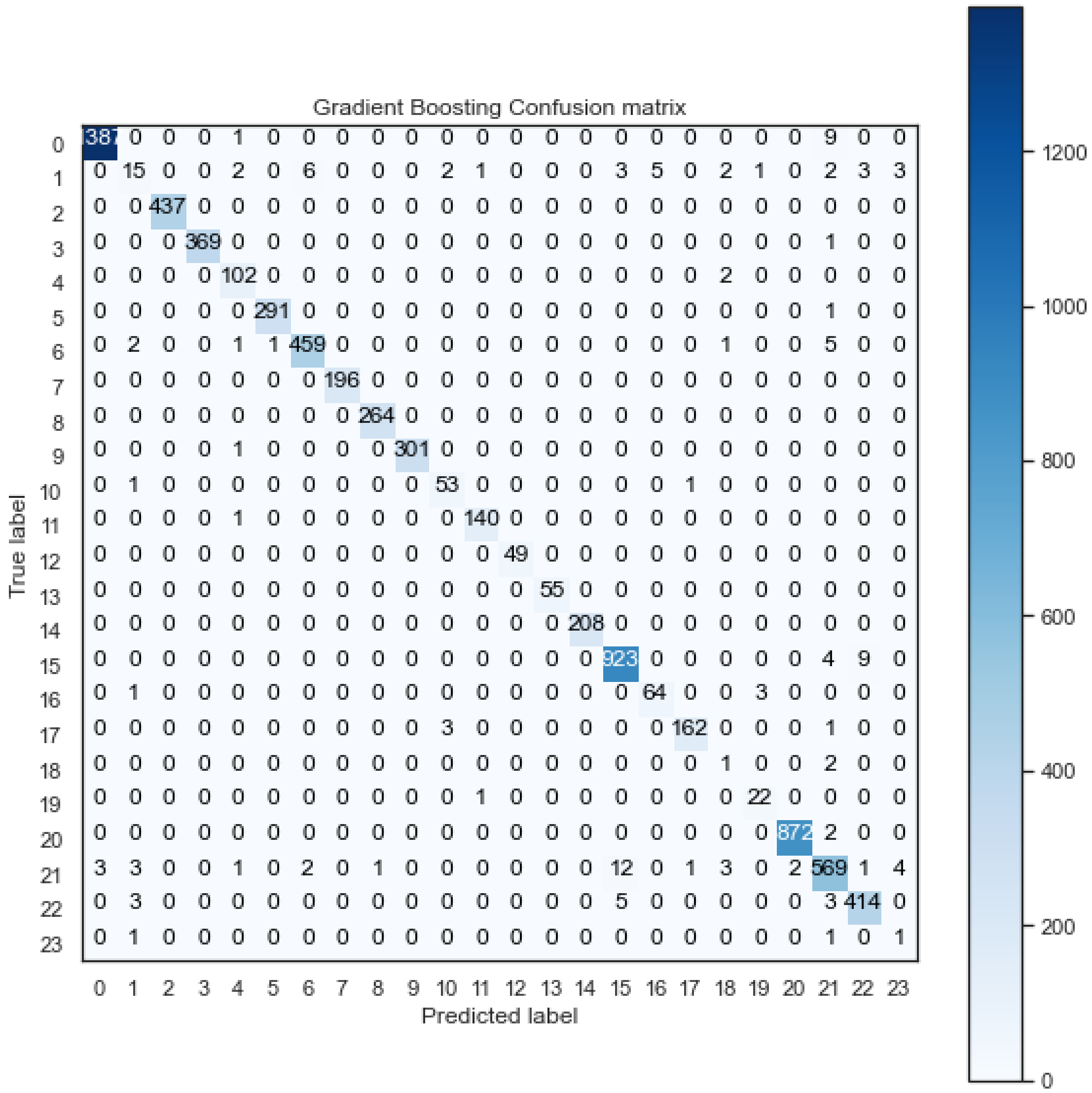

Figure 52 Output: Gradient Boosting Confusion Matrix

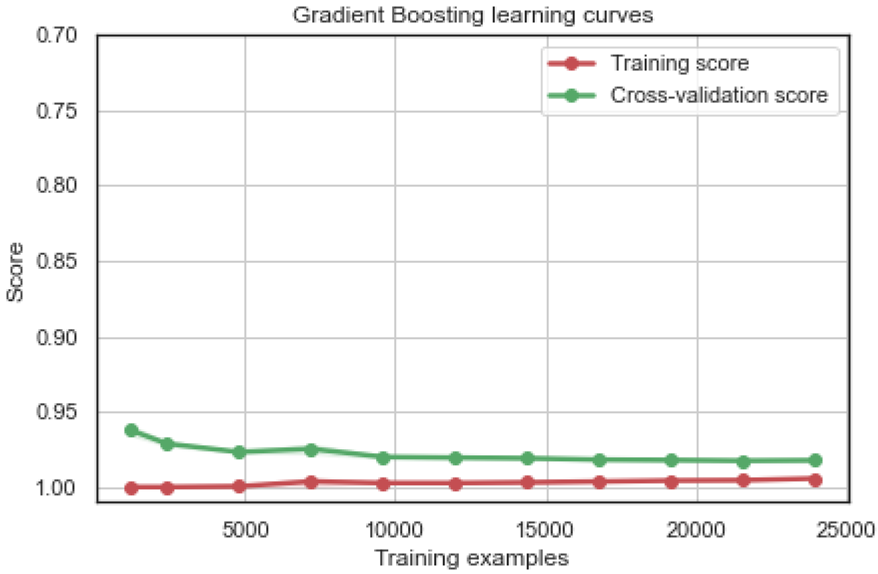

Figure 53 Output: Gradient Boosting Learning Curve

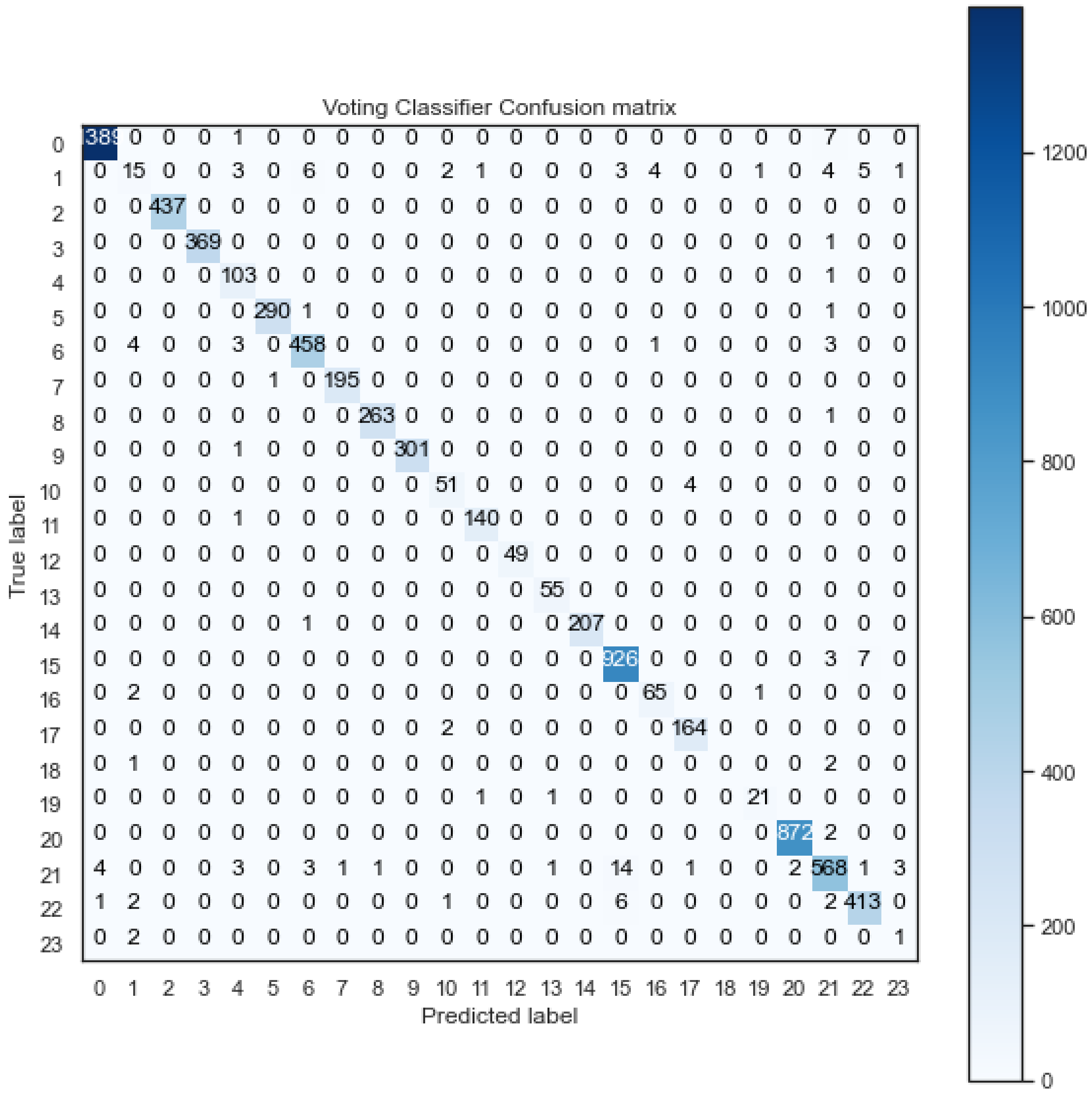

Figure 54 Output: Voting Classifier Confusion Matrix

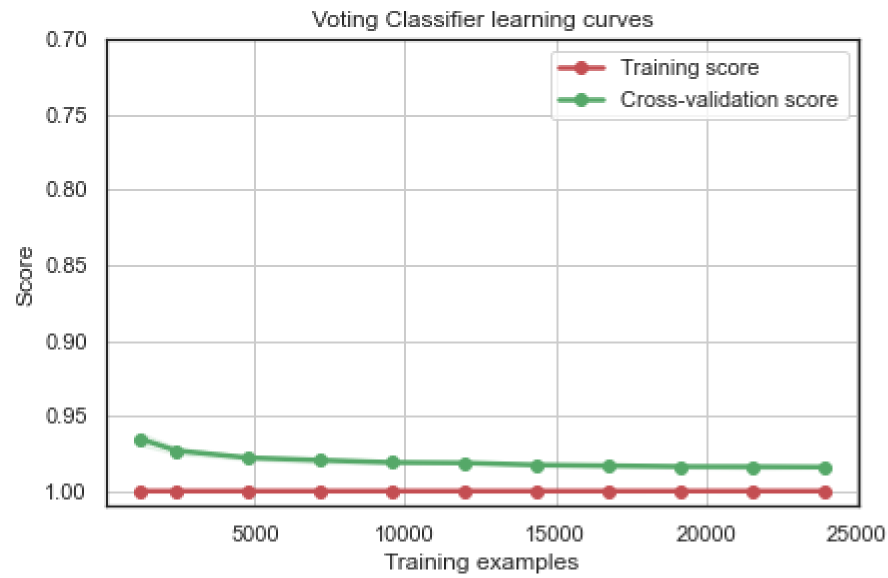

Figure 55 Output: Voting Classifier Learning Curve

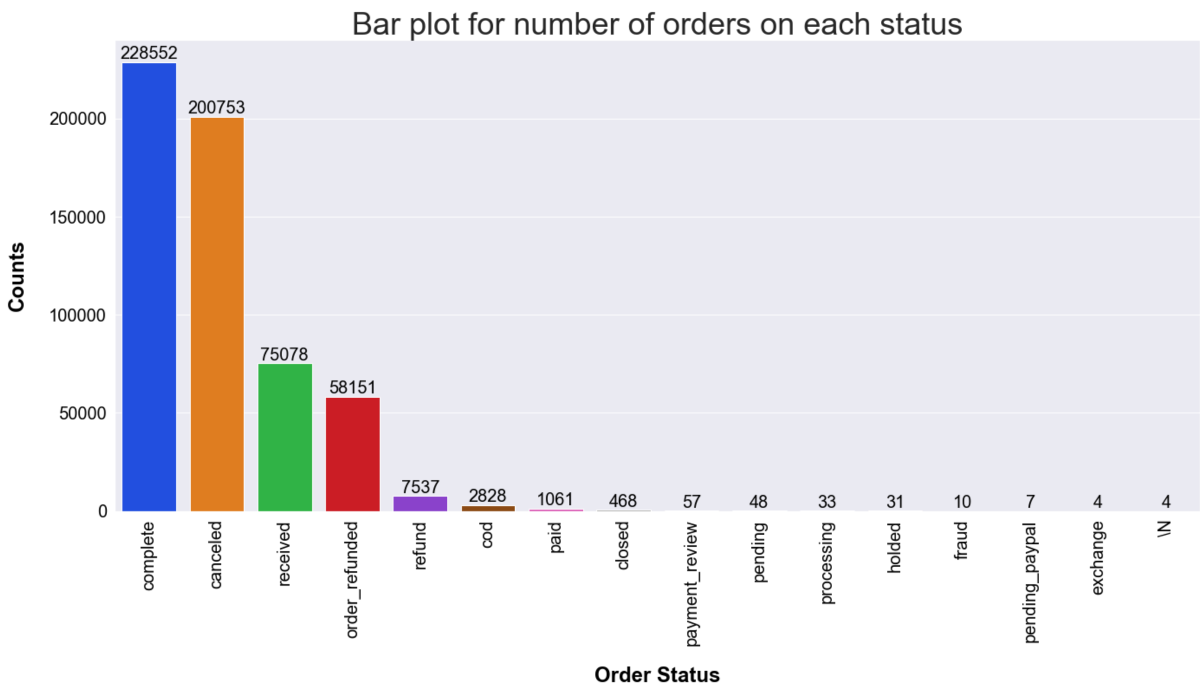

Figure 56 Number of orders in each status

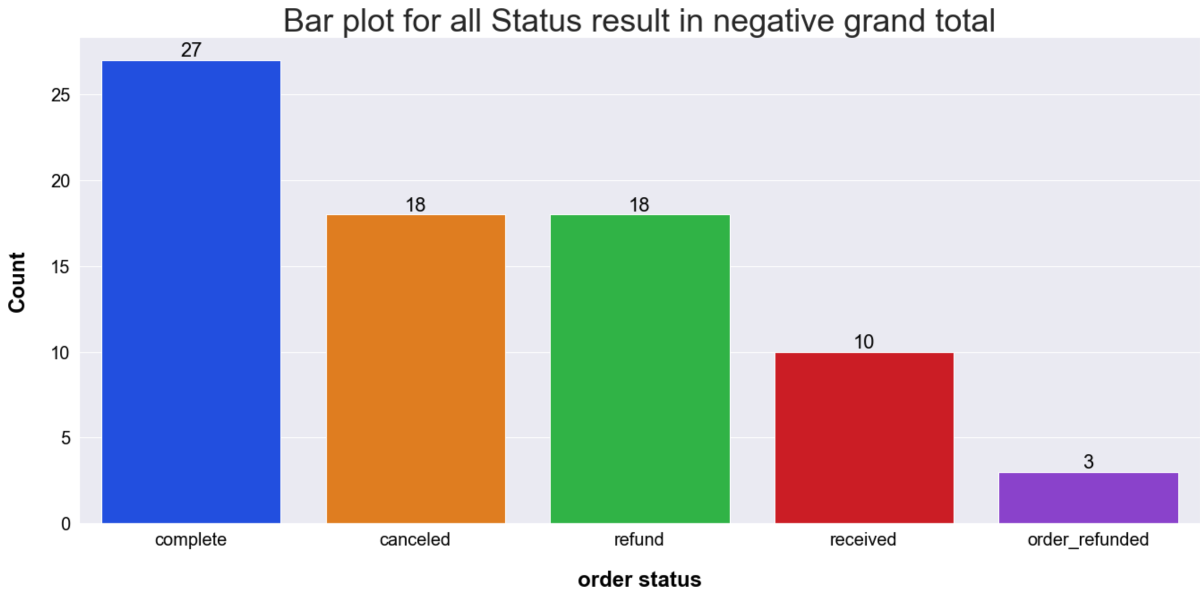

Figure 57 Orders with negative grand total

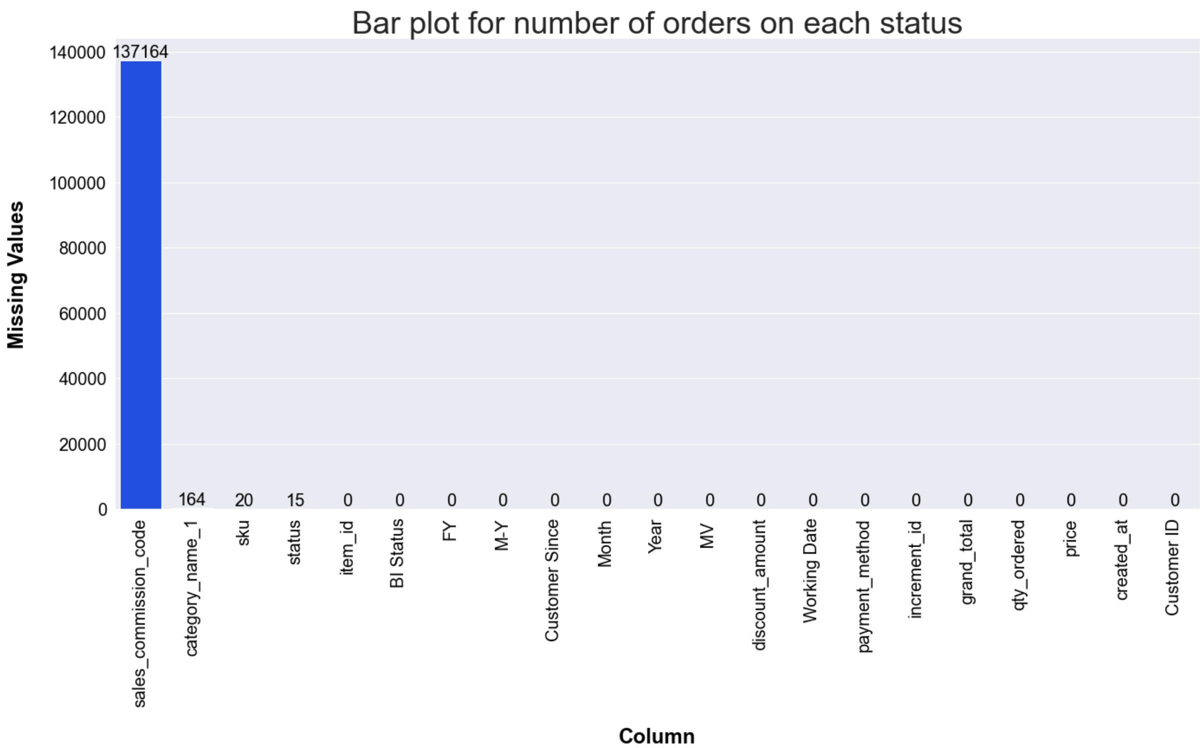

Figure 58 Missing Values in each column

APPENDIX B - Code

```
# 1. Setup

## 1.1.    Importing relevant Libraries and packages

import pandas as pd # data analysis
import numpy as np #mathematical operations
```

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns #interface for informative statistical graphics
import plotly as plotly
import plotly.express as px #Needed this to create pie-chart
import plotly.io as pio #To save graph as png
import datetime, warnings
import matplotlib.cm as cm
import itertools
import sklearn.cluster as cluster #Efficient tools for clustering
import matplotlib.patches as mpatches
import matplotlib
# matplotlib.rc('xtick', labelsize=10)
# matplotlib.rc('ytick', labelsize=10)
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn import preprocessing, model_selection, metrics
from sklearn.model_selection import GridSearchCV, learning_curve
from sklearn.metrics import silhouette_score,silhouette_samples, confusion_matrix, classification_report,
f1_score,fbeta_score,precision_score, recall_score
from sklearn import neighbors, linear_model, svm, tree, ensemble
from sklearn.ensemble import AdaBoostClassifier
from sklearn.decomposition import PCA
from IPython.display import display
from plotly.offline import init_notebook_mode
from scipy import stats
init_notebook_mode(connected=True)
warnings.filterwarnings("ignore")
plt.rcParams["patch.force_edgecolor"] = True
plt.style.use('fivethirtyeight')
mpl.rc('patch', edgecolor = 'dimgray', linewidth=1)
%matplotlib inline
pd.set_option('display.max_columns', 500)

## 1.2.     Importing Dataset as csv

dataset=pd.read_csv(r"C:\Users\bilal\Downloads\Ecommerce Dataset\Dataset.csv", low_memory=False)
#have set low memory to "false", in order to avoid panda being efficient and load whole file together


# 2.     Preprocessing

## 2.1.     Analysing how data look

### 2.1.1.   Analysing Data Shape

print('Dataframe dimensions:', dataset.shape)

### 2.1.2.   Analysing the first few lines of data to build an understanding of the features

# show first lines
display(dataset.head())

## 2.2.     Dropping Unnecessary data
```

```
### 2.2.1.   Dropping dummy columns generated by Panda when reading CSV

dataset = dataset.dropna(axis=1, how='all')
#axis=1 is selected and to drop only those columns which has all NaN values how = "all" is used

#### 2.2.1.1.            Confirming if they are removed by analysing shape again

#Confirming if the extra columns are gone
dataset.shape

### 2.2.2.   Dropping Null Values

#### 2.2.2.1.            Checking the percentage of Null values against data

# gives some information on columns types and numer of null values
tab_info=pd.DataFrame(dataset.dtypes).T.rename(index={0:'column type'})
tab_info=tab_info.append(pd.DataFrame(dataset.isnull().sum()).T.rename(index={0:'null values (nb)'}))
tab_info=tab_info.append(pd.DataFrame(dataset.isnull().sum()/dataset.shape[0]*100).T.
                    rename(index={0:'null values (%)'}))
display(tab_info)


#### 2.2.2.2.            Removing Customer IDs with Null values
```

It **is** observed **from the above output that around 44.25**% of values are **not** even assigned to anything, so they are removed.

```
dataset.dropna(axis = 0, subset = ['Customer ID'], inplace = True)
print('Dataframe dimensions:', dataset.shape)


#### 2.2.2.3.            Checked percentage again to confirm how data looked

tab_info=pd.DataFrame(dataset.dtypes).T.rename(index={0:'column type'})
tab_info=tab_info.append(pd.DataFrame(dataset.isnull().sum()).T.rename(index={0:'null values (nb)'}))
tab_info=tab_info.append(pd.DataFrame(dataset.isnull().sum()/dataset.shape[0]*100).T.
                    rename(index={0:'null values (%)'}))
display(tab_info)

### 2.2.3.   Dropping duplicate values

#### 2.2.3.1.            Checking Duplicate values but did not find any

print('Number of Duplicate values in this dataset = {}'.format(dataset.duplicated().sum()))
dataset.drop_duplicates(inplace = True)

### 2.2.4.   Finding Missing values

#### 2.2.4.1.            Analysed missing values in the dataset and created a graph of that

#Sorting out missing values
missing_values = dataset.isna().sum().sort_values(ascending=False)
```

```
missing_values_df = pd.DataFrame(list(missing_values.items()), columns=['Column', 'Missing_Values'])

plt.figure(figsize=(20,10))

#Creating a Bar Chart
sns.set(font_scale = 3)
ax = sns.barplot(data=missing_values_df, x = 'Column', y = 'Missing_Values', palette='bright')
ax.bar_label(container = ax.containers[0], padding = 0, fontsize = 20, color='black')
ax.set(title='Bar plot for number of orders on each status')


#Setting up axis label size and colours of axis label
plt.xticks( fontsize=20)
plt.yticks( fontsize=20)
ax.tick_params(axis='x', colors='black')
ax.tick_params(axis='y', colors='black')

#setting up axis label name and size
ax.set_xlabel('Column', fontsize=24, labelpad=24, color='black', fontweight='bold')
ax.set_ylabel('Missing Values', fontsize=24, labelpad=24, color='black', fontweight='bold')

#setting up x-axis labels vertically and horizontal alignment to center
plt.xticks(rotation=90, ha='center');

#### 2.2.4.2.         Dropped the "sales_commission_code" and "BI Status" columns because they were not necessary

#In the above graph it is cleared that majorly missing values are in the sales commission column which is understandable. Anyway, we do not need to commission code



dataset = dataset.drop(columns=['sales_commission_code',"BI Status"])
dataset=dataset.dropna()
dataset.shape

### 2.2.5.  Removing date columns which shared the same data but with different names

#### 2.2.5.1.         Checking The Types of Dataset Columns
###### Values are given data types to ensure that they perform properly and without errors. The attribute that tells a computer how to interpret the data is the data type.

dataset.dtypes


#### 2.2.5.2.         Converting Date Columns to Date type

dataset['created_at'] = pd.to_datetime(dataset['created_at'])
dataset['Working Date'] = pd.to_datetime(dataset['created_at'])
dataset['M-Y'] = pd.to_datetime(dataset['M-Y'])
dataset['Customer Since'] = pd.to_datetime(dataset['Customer Since'])


#### 2.2.5.3.         Checking the variable_types of Dataset column again to asure that conversions worked
```

```
dataset.dtypes
```

2.2.5.4. Checking if the column created_at same as working_date

```
dataset.loc[dataset["created_at"] != dataset["Working Date"]].nunique()
```

It shows that both columns have same values

2.2.5.5. Checking if the columns M-Y, has the same information as month and year of created_at column dates

```
dataset.loc[dataset["created_at"].dt.strftime(
  '%m/%Y') != dataset["M-Y"].dt.strftime('%m/%Y')].nunique()
```

2.2.5.6. Dropping the columns of date which represent same data

```
Working Date is same as created_at
M-Y has the same information as the month and year of created_at
Year column has exactly same years as created_at column's
Month column has exactly same month as created_at column's
FY just represent financial year so also not relevant to have separate column


dataset = dataset.drop(columns=['Working Date', "M-Y", "Year", "Month", "FY"])
```

2.2.5.7. Checking the Dataset again to confirm the changes

```
dataset.head()
```

2.2.6. Finding the insight on the statistical analysis

```
dataset.describe() # To find description of numerical data (statistical analysis)
```

2.2.6.1. Checking what status result in negative grand total

```
#To investigate the negative value in grand total minimum
gt = dataset[dataset.grand_total < 0].status.value_counts().to_dict() #checked the less than 0 that are negative values, count them and convert the data set into dictionary
gt_df = pd.DataFrame(list(gt.items()), columns=["Order Status","Count"])

#Defining the figure size
plt.figure(figsize=(20,10))

#Creating a Bar Chart
sns.set(font_scale = 3)
ax = sns.barplot(data=gt_df, x = "Order Status", y="Count", palette = "bright")
ax.set(title='Bar plot for all Status result in negative grand total')
ax.bar_label(container = ax.containers[0],padding = 0, fontsize = 22, color ="black")
```

```
#Setting up axis label size and colours of axis label
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
ax.tick_params(axis="x",colors = "black")
ax.tick_params(axis="y",colors = "black")

#setting up axis label name and size
plt.xlabel("order status",fontsize=24,labelpad=24,color = "black", fontweight='bold')
plt.ylabel("Count",fontsize=24,labelpad=24,color = "black",fontweight='bold')



# it was giving error: AttributeError: 'AxesSubplot' object has no attribute 'bar_label' so updating matplotlib in
1st cell and removed that cell to avoid updating check again and again

#### 2.2.6.2.          Removing values with negative grand total

dataset = dataset[dataset.grand_total > 0] #got rid of negative values

#### 2.2.6.3.          Checking the statistical analysis again to confirm

dataset.describe()  # To find description of numerical data (statistical analysis)

### 2.2.7.   Removing unnecessary statuses

#### 2.2.7.1.          Analysing the statuses against number of orders

plt.figure(figsize=(20,10))

order_status = dataset.status.value_counts()
order_status_df = pd.DataFrame(list(order_status.items()), columns=['Order Status', 'Counts'])

#Creating a Bar Chart
sns.set(font_scale = 3)
ax = sns.barplot(data = order_status_df, x='Order Status', y ="Counts", palette ="bright")
ax.bar_label(container = ax.containers[0], padding = 0, fontsize = 20, color='black')
ax.set(title='Bar plot for number of orders on each status')

#Setting up axis label size and colours of axis label
plt.xticks( fontsize=20)
plt.yticks( fontsize=20)
ax.tick_params(axis='x', colors='black')
ax.tick_params(axis='y', colors='black')

#setting up axis label name and size
plt.xlabel('Order Status', fontsize=24, labelpad=24, color='black',fontweight='bold')
plt.ylabel('Counts', fontsize=24, labelpad=24, color='black',fontweight='bold')

#setting up x-axis labels vertically and horizontal-alignment to center
plt.xticks(rotation=90, ha='center');
```

From above graph it **is** observed that alot of status useless **and** can be generalised:
Received, COD, paid,closed, exchange all means that order **is** completed by the user, **and** so we marked all of them complete

Statuses like order refunded, refund **and** fraud means they weren't completed and so meant to be cancelled

Status like payment_review, processing, holded **and** pending paypal meant they aren't completed, nor they have to be cancelled, so we just put them under pending umbrella

2.2.7.2. Merging the statuses which meant same for us

```
#orders with the status like complete, cod, paid, received, exchanged are completed orders in our point of view in handling this data, so we will group them under completed.
#Orders with cancelled, order refunded, refund and fraud statuses are grouped under cancelled
#Orders with status like payment_review, pending, processing, holded and pending paypal are all grouped under pending

dataset.status = dataset.status.replace({'complete': 'Completed',
                                 'received': 'Completed',
                                 'cod': 'Completed',
                                 'paid': 'Completed',
                                 'closed': 'Completed',
                                 'exchange': 'Completed',
                                 'canceled': 'cancelled',
                                 'order_refunded': 'cancelled',
                                 'refund': 'cancelled',
                                 'fraud': 'cancelled',
                                 'payment_review': 'Pending',
                                 'pending': 'Pending',
                                 'processing': 'Pending',
                                 'holded': 'Pending',
                                 'pending_paypal': 'Pending'})
```

2.2.8. Removing Orders with the cancelled and /N statuses

2.2.8.1. Counting cancelled order against each transaction

```
dataset[dataset["status"]=="cancelled"].count()
```

2.2.8.2. Saved Cancelled orders in a separate dataset

```
#Saving cancelled orders separately
cancelled = dataset[dataset.status == 'cancelled']
```

2.2.8.3. Removed cancelled and /N orders and keeping Completed and Pending

```
dataset = dataset[(dataset.status == 'Completed')
          | (dataset.status == 'Pending')]
status_updated = dataset.status.value_counts()

status_updated_df = pd.DataFrame(list(status_updated.items()), columns=[
                   'Order Status', 'Counts'])

plt.figure(figsize=(20, 10))

#Creating a Bar Chart
sns.set(font_scale=3)
```

```
ax = sns.barplot(data=status_updated_df, x='Order Status',
            y='Counts', palette='bright')
ax.bar_label(container=ax.containers[0], padding=0, fontsize=20, color='black')
ax.set(title='Bar plot for number of orders Completed or on Pending')


#Setting up axis label size and colours of axis label
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
ax.tick_params(axis='x', colors='black')
ax.tick_params(axis='y', colors='black')

#setting up axis label name and size
plt.xlabel('Order Status', fontsize=24, labelpad=24,
        color='black', fontweight='bold')
plt.ylabel('Counts', fontsize=24, labelpad=24,
        color='black', fontweight='bold')

#setting up x-axis labels vertically and horizontal-alignment to center
plt.xticks(rotation=90, ha='center')
```

2.3. Getting Insight on the data

2.3.1. Counting number of Products, Transactions and Customers to get insight how dataset looks

```
pd.DataFrame([{'products': len(dataset['sku'].value_counts()),
            'transactions': len(dataset['increment_id'].value_counts()),
            'customers': len(dataset['Customer ID'].value_counts()),
            }], columns=['products', 'transactions', 'customers'], index=['quantity'])
```

It **is** observed that **79**K(**79309**) customers has bought **62**K(**62531**) products **in** **200**K(**202314**) transactions
Now will find number of products **in** each transaction

2.3.2. Counting number of products in every transaction

```
temp = dataset.groupby(
   by=['Customer ID', 'increment_id', "status"], as_index=False)['created_at'].count()
nb_products_per_basket = temp.rename(columns={'created_at': 'Number of products'})
nb_products_per_basket.head().sort_values('Customer ID')
```

2.3.3. Renaming columns which had confusing names

Created at column **is** same **as** order date **and** category name **1** **is** just the category, increment ID **is** order number

```
dataset = dataset.rename(columns={"created_at": "order_date", "category_name_1":
"category","increment_id":"order_no"})
cancelled = cancelled.rename(columns={
                    "created_at": "order_date", "category_name_1": "category", "increment_id": "order_no"})
```

2.3.4. Finding Orders per customer

```
#Number of Orders vs number of customers

num_of_orders = dataset.groupby("Customer ID")["order_no"].nunique().sort_values(ascending=False)
#Grouping Customers with increment ids and counting unique values
num_of_orders_df = pd.DataFrame(list(num_of_orders.items()),columns = ["Customer ID","Number of
Orders"])

x = num_of_orders_df[num_of_orders_df["Number of Orders"] == 1].value_counts().sum()
y = num_of_orders_df[num_of_orders_df["Number of Orders"] != 1].value_counts().sum()


data = {"order" : ["Customers with Single orders", "Customers with Multiple orders"], "Customer_Counts":
[x,y]}

order_counts = pd.DataFrame.from_dict(data)

#Creating a pie chart
fig = px.pie(order_counts,
        values = order_counts.Customer_Counts,
        names = order_counts.order,
        template = 'plotly_white')

#Setting up the pie chart cosmetics
fig.update_traces(textposition='inside', textinfo='percent+label', textfont_size=18,
            marker = dict(line = dict(color = 'white', width = 2)))
fig.show()
#plotly.offline.iplot(fig, validate=False, filename='worldmap', image='png')
pio.write_image(fig, 'pie1.png')
```

Above chart shows that around 2/3rd of customers have only one order while 1/3rd of customers have placed more than one orders

```
### 2.3.5.  Finding Orders per category

#Number of Orders vs Categories

num_of_prod = dataset.groupby('Customer ID')['category'].nunique().sort_values(ascending=False) #Grouping
Customers with categories and counting unique values

num_of_prod_df = pd.DataFrame(list(num_of_prod.items()), columns=['Customer ID', 'Number of Products'])

a = num_of_prod_df[num_of_prod_df['Number of Products'] == 1].value_counts().sum()
b = num_of_prod_df[num_of_prod_df['Number of Products'] != 1].value_counts().sum()

data = {'Order': ['One Category', 'More than One Category'], 'Customer_Counts': [a, b]}

category_counts = pd.DataFrame.from_dict(data)


#Creating a pie chart
fig = px.pie(category_counts,
        values = category_counts.Customer_Counts,
```

```
        names = category_counts.Order,
        template = 'plotly_white')

#Setting up the pie chart cosmetics
fig.update_traces(textposition='inside', textinfo='percent+label', textfont_size=18,
            marker = dict(line = dict(color = 'white', width = 2)))
fig.show()
pio.write_image(fig, 'pie2.png')
```

Above Pie Chart shows nearly **28**% of customers have ordered **from more than one category but 72**% of customers only bought **from single category**

2.3.6. Analysing Cancelled Orders

2.3.6.1. Finding number of Customers who cancelled

```
# Number of products in every cancelled transaction
temp = cancelled.groupby(
  by=['Customer ID', 'order_no', "status"], as_index=False)['order_date'].count()
cancelled_prod_per_basket = temp.rename(columns={'order_date': 'Number of products'})

#Finding number of cancelled products and dividing it by total number of transactions of cancelled and completed+pending
m_1 = cancelled_prod_per_basket.shape[0]
m_2 = nb_products_per_basket.shape[0]
n2 = m_1+m_2
print('Number of orders cancelled: {}/{} ({:.2f}%) '.format(m_1, n2, m_1/n2*100))


a = dataset['Customer ID'].nunique() + cancelled ['Customer ID'].nunique()
b = cancelled ['Customer ID'].nunique()
print('Number of orders cancelled: {}/{} ({:.2f}%) '.format(b, a, b/a*100))
```

2.3.6.2. Finding number orders which got cancelled

```
a = dataset['Customer ID'].nunique() + cancelled ['Customer ID'].nunique()
b = cancelled ['Customer ID'].nunique()

data = {'Customers': ['Total Customers', 'Those Who cancelled'], 'Customer_Counts': [a, b]}

customer_counts = pd.DataFrame.from_dict(data)


c = dataset['order_no'].nunique() + cancelled['order_no'].nunique()
d = cancelled ['order_no'].nunique()

data = {'Orders': ['Total Orders', 'cancelled Orders'], 'Order_Counts': [c, d]}

order_counts = pd.DataFrame.from_dict(data)

#Setting up background colour of figures, and grid
sns.set(rc={'axes.facecolor':'white', 'figure.facecolor':'white', 'axes.grid' : True,"grid.color": "black"})
```

```
#Creating 2 bar plots one for customer who cancelled and the orders which got cancelled
fig, ax = plt.subplots(1,2, figsize = (20,10))
sns.barplot(ax = ax[0], data = customer_counts, x=customer_counts.Customers,
y=customer_counts.Customer_Counts, palette= 'bright')
sns.barplot(ax = ax[1], data = order_counts, x=order_counts.Orders, y=order_counts.Order_Counts, palette=
'bright')

#Setting 1st barchart cosmetics
ax[0].set_title("Customers Who cancelled", fontsize = 26, pad = 30, color='black', fontweight='bold')
ax[0].set_xlabel("Customers", fontsize = 24, labelpad = 15, color='black', fontweight='bold')
ax[0].set_ylabel("Counts", fontsize = 24, labelpad = 15, color='black', fontweight='bold')
ax[0].tick_params(axis='x', colors='black', labelsize=20)
ax[0].tick_params(axis='y', colors='black', labelsize=20)

#Setting 2nd barchart cosmetics
ax[1].set_title("cancelled Orders", fontsize = 28, pad = 30, color='black', fontweight='bold')
ax[1].set_xlabel("Orders", fontsize = 24, labelpad = 15, color='black', fontweight='bold')
ax[1].set_ylabel("Counts", fontsize = 24, labelpad = 15, color='black', fontweight='bold')
ax[1].tick_params(axis='x', colors='black', labelsize=20)
ax[1].tick_params(axis='y', colors='black', labelsize=20)

plt.tight_layout(pad=2);
```

2.3.6.3. Finding categories whose orders mostly get cancelled

```
#Finding Categories whose orders got cancelled most

cancelled = cancelled.dropna() #Dropping NaN values


fig = px.treemap(cancelled,
        path=['category'], template='seaborn',width=1000, height=1000)
fig.update_traces(textfont_color='black',textfont_size=20, selector=dict(type='treemap'))
fig.show()
```

2.3.7. Understanding product categories

2.3.7.1. Checking all categories

```
#Total categories in a dataset
category=dataset["category"].unique()
print("All categories in the dataset =",category)

n=dataset["category"].nunique()
print("number of categories n =",n)
```

2.3.7.2. Checking category name /N

```
#Exploring category //N
dataset[dataset["category"]=="\\N"]
```

2.3.7.3. Merging category /N with category Others

```
#it seems it is just like others' category, so we will merge both under others
dataset["category"] = dataset["category"].replace('\\N', "Others")

#### 2.3.7.4.           Checking the categories again

#Total categories in a dataset
category=dataset["category"].unique()
print("All categories in the dataset =",category)

#### 2.3.7.5.           Encoding the categories to make them easier to process

##### 2.3.7.5.1.        Using label encoder to encode

#assigning numbers to category name
le = preprocessing.LabelEncoder()
le.fit(dataset["category"])
dataset["category number"] = le.transform(dataset["category"])
dataset.head()


##### 2.3.7.5.2.        Checking what labels are assigned

#checking if the assignment worked
df1=dataset.groupby(["category"])["category"].count()
df2=dataset.groupby(["category number"])["category number"].count()
#print(df2)
#print(df1)
x=(dataset["category"].unique())
y=(dataset["category number"].unique())
tempdf = pd.DataFrame({'Number':y,'Category name':x})
tempdf.sort_values(by="Number").style.hide_index()

### 2.3.8.  Categorising amount of each transaction with each category

plt.figure(figsize=(20,10))

order_status = dataset.status.value_counts()
order_status_df = pd.DataFrame(list(order_status.items()), columns=['Order Status', 'Counts'])

#Creating a Bar Chart
sns.set(font_scale = 3)
ax = sns.barplot(data = order_status_df, x='Order Status', y ="Counts", palette ="bright")
ax.bar_label(container = ax.containers[0], padding = 0, fontsize = 20, color='black')
ax.set(title='Bar plot for number of orders on each status')

#Setting up axis label size and colours of axis label
plt.xticks( fontsize=20)
plt.yticks( fontsize=20)
ax.tick_params(axis='x', colors='black')
ax.tick_params(axis='y', colors='black')

#setting up axis label name and size
plt.xlabel('Order Status', fontsize=24, labelpad=24, color='black',fontweight='bold')
```

```
plt.ylabel('Counts', fontsize=24, labelpad=24, color='black',fontweight='bold')

#setting up x-axis labels vertically and horizontal-alignment to center
plt.xticks(rotation=90, ha='center');
```

2.3.8.1. Creating column names with each category number

For all the categories, columns are made which will have the amount spent in theat particular category

```
for i in range(15):
    col = 'category_{}'.format(i)
    df_temp = dataset[dataset["category number"] == i]
    price_temp = df_temp['price'] * \
        (df_temp['qty_ordered']) - df_temp['discount_amount']
    price_temp = price_temp.apply(lambda x: x if x > 0 else 0)
    dataset.loc[:, col] = price_temp
    dataset[col].fillna(0, inplace=True)

filter_col = [col for col in dataset if col.startswith('category_')]
dataset[["order_no", "grand_total", "category number", 'category_0', 'category_1', 'category_2', 'category_3',
'category_4',
       'category_5', 'category_6', 'category_7', 'category_8', 'category_9', 'category_10', 'category_11',
'category_12', 'category_13', 'category_14']]
```

2.3.8.2. Defining a new variable total price

As the name suggest this new variable will indicate the total price of every purchase

```
dataset['TotalPrice'] = dataset['price'] * (dataset['qty_ordered'])# - dataset['discount_amount']
dataset.sort_values('Customer ID')[:5]
```

2.3.9. Merging all transactions of single order into one

2.3.9.1. Creating a new data frame purchase_price

A new dataframe is created where the information for a particular order is collected and put in as the one single entry as before it was one line per order
It will also have the categories and the total price distributed in those categories columns

```
#Sum of Purchases (user and order)
temp = dataset.groupby(by=['Customer ID', 'order_no'], as_index=False)['TotalPrice'].sum()
purchase_price = temp.rename(columns = {'TotalPrice':'purchase price'})

#product categories
temp = dataset.groupby(by=["Customer ID","order_no"],as_index = False).sum()[["Customer ID","order_no"]
+ [f'category_{i}' for i in range(15)]]
purchase_price = pd.merge(purchase_price, temp, on = ["Customer ID","order_no"])

#Order Date

dataset['order_date_int'] = dataset['order_date'].astype('int64')
temp = dataset.groupby(by=['Customer ID', 'order_no'], as_index=False)['order_date_int'].mean()
```

```
dataset.drop('order_date_int', axis = 1, inplace = True)
purchase_price.loc[:, 'date'] = pd.to_datetime(temp['order_date_int'])

#Selecting positive values of purchase price only and displaying top 5
purchase_price = purchase_price[purchase_price["purchase price"] > 0]
purchase_price.sort_values('Customer ID', ascending = True)[:5]
```

2.3.10. Dividing purchase_price data set into 2 for training and testing

2.3.10.1. Finding the date of first and last order of the whole dataset

```
print(purchase_price['date'].min(), '->',  purchase_price['date'].max())
```

2.3.10.2. Dividing dataset into train and test

The dataset ranges from July'16 to August'18, so dividing the dataset in half to and from July'2017, it has information of 2 years so dividing in one year. It is to develop model to characterise and anticipate the habits of customers from their first visit. In next, we can test

```
test_data = purchase_price[purchase_price['date'] >= pd.to_datetime(datetime.date(2017,7,1))]
train = purchase_price[purchase_price['date'] < pd.to_datetime(datetime.date(2017,7,1))]
purchase_price = train.copy(deep = True)
```

3. Defining Customer Categories

3.1. Analysing Customer Behaviour

3.1.1. Grouping all the transactions of each user and finding the count, minimum, maximum, average and total of all the amount spent by each user

Now as there are multiple entries for similar users, so they all are combined and grouped together under single entry and there purchases are also aggregated. Few additional column's are added which has the maximum, minimum, average and total amount spent by each customer in total.

```
orders_per_customer = purchase_price.groupby(
   by=['Customer ID'])['purchase price'].agg(['count', 'min', 'max', 'mean', 'sum'])
for i in range(15):
   col = 'category_{}'.format(i)
   orders_per_customer.loc[:, col] = purchase_price.groupby(
      by=['Customer ID'])[col].sum() / orders_per_customer['sum']*100

orders_per_customer.reset_index(drop=False, inplace=True)
purchase_price.groupby(by=['Customer ID'])['category_0'].sum()
orders_per_customer.sort_values('Customer ID', ascending=True)[:5]
```

3.1.2. All categories' column has the percentages for each customer, so checking if the values are not exceeding 100 for each category

```
orders_per_customer_sum_categs = orders_per_customer[[
   f'category_{i}' for i in range(15)]].sum(axis=1)
orders_per_customer_sum_categs[orders_per_customer_sum_categs > 100.01]
```

3.1.3. Finding the number of days passed since the first and last purchases

Two additional variables are created: the first purchase **and** the last purchase, which have the dates of the customer's first **and** last purchases.

```
last_date = purchase_price['date'].max().date()

first_registration = pd.DataFrame(purchase_price.groupby(by=['Customer ID'])['date'].min())
last_purchase      = pd.DataFrame(purchase_price.groupby(by=['Customer ID'])['date'].max())

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

orders_per_customer.loc[:, 'last_purchase'] = test2.reset_index(drop = False)['date']
orders_per_customer.loc[:, 'first_purchase'] = test.reset_index(drop = False)['date']

orders_per_customer
```

3.1.4. It is checked how many customers bought one time and how many are the recurring customers

```
n1 = orders_per_customer[orders_per_customer['count'] == 1].shape[0]
n2 = orders_per_customer.shape[0]
print("customers with one-time purchase: {:<2}/{:<5} ({:<2.2f}%)".format(n1,n2,n1/n2*100))
```

There are around **2**/**3** customers who only bought one time

3.1.5. Checking the Anomility in the data

```
median = np.median(orders_per_customer["mean"])
upper_quartile = np.percentile(orders_per_customer["mean"], 75)
lower_quartile = np.percentile(orders_per_customer["mean"], 25)

iqr = upper_quartile - lower_quartile
upper_whisker = orders_per_customer["mean"][orders_per_customer["mean"] <= 
upper_quartile+1.5*iqr].max()
lower_whisker = orders_per_customer["mean"][orders_per_customer["mean"] >= lower_quartile-
1.5*iqr].min()

print("median: ",median,"  upper_quartile  ",upper_quartile,"  lower_quartile  ",lower_quartile,"  .iqr.  ",iqr,"
.upper_whisker.  ",upper_whisker,"  .lower_whisker.  ",lower_whisker)

orders_per_customer[orders_per_customer["mean"]>6549]

mean_mean=orders_per_customer["mean"].mean()
mean_std=orders_per_customer["mean"].std()
mean_value = orders_per_customer["mean"][orders_per_customer["mean"] <= 
mean_mean+4*mean_std].max()
mean_remove = orders_per_customer[orders_per_customer["mean"] > mean_value]["Customer ID"].values
print("customers removed as anomility: ", mean_remove)

dataset.drop(dataset.index[dataset['Customer ID'].isin(mean_remove)], inplace=True)
```

```
orders_per_customer.drop(
  orders_per_customer.index[orders_per_customer['Customer ID'].isin(mean_remove)], inplace=True)
```

3.2. Preprocessing Data for the creation of customer categories

3.2.1. Creating Matrix from required columns of transaction_per_user data frame

orders_per_customer dataframe's each entry correspond to one single client, so now we would be needing a particular information for those customers and create a matrix for it for further processes.

```
list_cols = ['count','min','max','mean'] + [f'category_{i}' for i in range(15)]
selected_customers = orders_per_customer.copy(deep = True)
matrix = selected_customers[list_cols].values
```

3.2.2. Standardising the matrix to avoid the extreme ranges of features

Each of these variable has variety of ranges so to **continue** the analysis **and** stop the processes **from being biassed towards extremely high ranges**, we will standardized the matrix.

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

3.3. Performing K-Means Analysis

3.3.1. Performing the Elbow method to find the optimal number of clusters for K-Mean

As we have to perform k-means analysis, we need to find the optimal number of analysis so we performed elbow method to find the k

```
K=range(1,30)
wss = []
for k in K:
  kmeans=cluster.KMeans(n_clusters=k,init="k-means++")
  kmeans=kmeans.fit(scaled_matrix)
  wss_iter = kmeans.inertia_
  wss.append(wss_iter)

mycenters = pd.DataFrame({'Clusters' : K, 'WSS' : wss})
mycenters

#Plotting Elbow plot
sns.scatterplot(x = 'Clusters', y = 'WSS', data = mycenters, marker="+")
sns.set(font_scale=2)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
x_axis_ticks = plt.xticks(K)
plt.title("Elbow Method to find optimal number of clusters", fontsize = 18)
```

xxx

```
def annot_max(x, y, ax=None):
    xmax = x[np.argmax(y)]
    ymax = max(y)
    text = "x={:.3f}, y={:.3f}".format(xmax, ymax)
    if not ax:
        ax = plt.gca()
    bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
    arrowprops = dict(
        arrowstyle="->", connectionstyle="angle,angleA=0,angleB=60")
    kw = dict(xycoords='data', textcoords="axes fraction",
              arrowprops=arrowprops, bbox=bbox_props, ha="right", va="top")
    ax.annotate(text, xy=(xmax, ymax), xytext=(0.94, 0.96), **kw)

range_n_clusters = np.arange(2,30)
silhouette_avg = []
for num_clusters in range_n_clusters:

 # initialise kmeans
 kmeans = KMeans(n_clusters=num_clusters)
 kmeans.fit(scaled_matrix)
 cluster_labels = kmeans.labels_

 # silhouette score
 silhouette_avg.append(silhouette_score(scaled_matrix, cluster_labels))

plt.plot(range_n_clusters, silhouette_avg)
annot_max(range_n_clusters, silhouette_avg)

plt.xlabel("Values of K",fontsize = 16)
plt.ylabel("Silhouette score", fontsize = 16)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title("Silhouette analysis For Optimal k", fontsize = 18)
plt.show()
```

3.3.2. Verifying the number of the cluster by using the Silhouette score method

The number of clusters which the elbow graph makes an elbow is 18 but to confirm if that is right, so we will perform silhouette analysis to find k

```
Silhouette_Scores = []
for i in range(3, 30):
    labels=cluster.KMeans(n_clusters=i,init="k-means++",random_state=200).fit(scaled_matrix).labels_
    Silhouette_Scores.append((i,
metrics.silhouette_score(scaled_matrix,labels,metric="euclidean",sample_size=1000,random_state=200).round(5)))
    print("Silhouette score for k(clusters) = "+str(i)+" is " + str(Silhouette_Scores[-1][1]))
```

```
plt.bar([t[0] for t in Silhouette_Scores], [t[1] for t in Silhouette_Scores])
plt.ylabel("Number of Clusters", fontsize=16)
plt.xlabel('Silhouette Score', fontsize=16)

plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.title("Silhouette Score for n number of clusters", fontsize=18)


n_clusters = max(Silhouette_Scores,key=lambda item:item[1])[0]
print("we will make "+str(n_clusters)+" of clusters because it has highest Silhouette score of "+
str(max(Silhouette_Scores,key=lambda item:item[1])[1]))
```

3.3.3. Performing K-Means using the number of clusters defined by the elbow method and Silhouette score

It **is** confirmed that optimal number of cluster **is** 24, so we will perfom a k-mean analysis using k **as** 24

```
kmeans = KMeans(init='k-means++', n_clusters=n_clusters, n_init=100)
kmeans.fit(scaled_matrix)
clusters_clients = kmeans.predict(scaled_matrix)
```

3.3.4. Plotting the graph for Silhouette values to look at the quality of separation of clusters again

To find the insight on the cluster quality. Graph **is** taken **from the sklearn documentation**:

```
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
        cmap = cm.get_cmap("Spectral")
        color = cmap(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, 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

# define individual silhouette scores
sample_silhouette_values = silhouette_samples(scaled_matrix, clusters_clients)
#____________________
# and do the graph
graph_component_silhouette(n_clusters, [-1, 1], len(scaled_matrix), sample_silhouette_values,
clusters_clients)

plt.ylabel('Cluster label', fontsize=16)
plt.xlabel('Silhouette Coefficient values', fontsize=16)

plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.title("The Silhouette plot for various clusters", fontsize=18)
```

3.3.5. Creating a data frame for the clusters and the number of customers in each cluster

Finding how many customers are **in** each cluster

```
clust_num = pd.DataFrame(pd.Series(clusters_clients).value_counts(), columns = ['no of
customers']).rename_axis('Cluster No').T

clust_num

sns.barplot(data=clust_num, palette="bright")
plt.ylabel('Number of Customers', fontsize = 16)
plt.xlabel('Cluster Number',fontsize = 16)

plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.title("Silhouette analysis For Optimal k", fontsize = 18)
```

3.4. Performing and displaying PCA

3.4.1. Performing PCA verify the quality of separation between the groups

Because of high disparity **in** the sizes of clusters we have to analyse the seperations between each cluster so we will perform PCA

```
pca = PCA()
pca.fit(scaled_matrix)
pca_samples = pca.transform(scaled_matrix)
```

3.4.2. Representing the amount of variance held by each component

```
fig, ax = plt.subplots(figsize=(17, 8))
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, matrix.shape[1])

ax.set_xticklabels([s if int(s.get_text())%2 == 0 else '' for s in ax.get_xticklabels()])
plt.axhline(y=0.95, color='r', linestyle='-')
plt.text(0.5, 0.85, '95% cut-off threshold', color='red', fontsize=16)

plt.ylabel('Explained variance', fontsize = 16)
plt.xlabel('Principal components', fontsize = 16)
plt.legend(loc='best', fontsize = 13)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.title("PCA for all the components", fontsize = 18)

plt.show()
```

To choose number of components it was planned to select the components **with** 90% variance but they are 15 components which will result **in** that explanation so choosing 6 arbitrary to perform the analysis to keep only a limited number of components since this decomposition **is** only performed to visualize the data

3.4.3. Performed PCA using 6 number of components

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

#print(pca.explained_variance_)
print (pca.explained_variance_ratio_)
#print(pca.explained_variance_ratio_.cumsum())

fig, ax = plt.subplots(figsize=(17, 8))
sns.set(font_scale=1)
plt.step(range(matrix_3D.shape[1]), pca.explained_variance_ratio_.cumsum(), where='mid',
       label='cumulative explained variance')
sns.barplot(np.arange(1, matrix_3D.shape[1]+1), pca.explained_variance_ratio_, alpha=0.5, color='g',
        label='individual explained variance')
plt.xlim(0, matrix_3D.shape[1])

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)
plt.show()


mat
```

3.4.4. Importing and assigning colours for each cluster

```
import matplotlib.colors as colors
color_dic = {}
for i in range(n_clusters):
   for i in range(n_clusters):
      color_dic[i] =[key for key in colors.cnames.keys()][9+i]
color_dic

#### 3.4.5. Plotting the clusters to analyse the data

sns.set_style("white")
sns.set_context("notebook", font_scale=1, rc={"lines.linewidth": 2.5})
label_color = [color_dic[l] for l in mat["cluster"]]

fig = plt.figure(figsize = (12,20))
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

#____________________________________________________
comp_handler = []
for i in range(n_clusters):
   comp_handler.append(mpatches.Patch(color = color_dic[i], label = i))

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

plt.tight_layout(pad=5)

### 3.5.     Adding a variable in the selected_customer data frame to define the clusters each customer belongs to

A new variable "Cluster" is added to the dataframe which will have the values stored in clusters_clients

clust_num


selected_customers.loc[:, 'cluster'] = clusters_clients

selected_customers[selected_customers['cluster'] == 5
```

]

dataset[dataset["Customer ID"] == selected_customers[selected_customers['cluster'] == 5]["Customer ID"].values[0]]

3.6. Another data frame is created, “cust_behaviour” this will have the clusters of the customers, their average purchase price, the number of times they ordered, and the minimum and maximum they spent

Number of customer **in** each group **is** also calculated

dataset[dataset["Customer ID"]==50387]

dataset[dataset["Customer ID"] == selected_customers[selected_customers['cluster'] == 1]["Customer ID"].values[0]]

```
cust_behaviour = 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]
  cust_behaviour = pd.concat([cust_behaviour, test])
#_______________________________________________________
cust_behaviour.drop('Customer ID', axis = 1, inplace = True)
print('number of customers:', cust_behaviour['size'].sum())

cust_behaviour = cust_behaviour.sort_values('sum')
cust_behaviour.sort_index()
```

3.7. Cluster’s Insight (A radar chart is created to represent the insight)

3.7.1. A class is defined to build a radar chart; this is copied from (https://www.kaggle.com/yassineghouzam/don-t-know-why-employees-leave%20-read-this)

```
def _scale_data(data, ranges):
  (x1, x2) = ranges[0]
  d = data[0]
  return [(d - y1) / (y2 - y1) * (x2 - x1) + x1 for d, (y1, y2) in zip(data, ranges)]

class RadarChart():
  def __init__(self, fig, location, sizes,axis_title, variables, ranges,n_ordinate_levels=5):

    angles = np.arange(0, 360, 360./len(variables))

    ix, iy = location[:] ; size_x, size_y = sizes[:]

    axes = [fig.add_axes([ix, iy, size_x, size_y], polar = True, label="axes{}".format(i)) for i in
range(len(variables))]

    _, text = axes[0].set_thetagrids(angles, labels = axis_title)
```

```
        for txt, angle in zip(text, angles):
            if angle > -1 and angle < 181:
                txt.set_rotation(angle - 90)
            else:
                txt.set_rotation(angle - 270)

        for ax in axes[1:]:
            ax.patch.set_visible(False)
            ax.xaxis.set_visible(False)
            ax.grid("off")

        for i, ax in enumerate(axes):
            fig.canvas.draw()
            grid = np.linspace(*ranges[i], num = n_ordinate_levels)
            grid_label = [""]+[""]+["{:.0f}".format(x) for x in grid[2:-1]] + [""]
            ax.set_rgrids(grid, labels = grid_label, angle = angles[i])
            ax.xaxis.set_tick_params(pad=20)
            ax.set_ylim(*ranges[i])

        self.angle = np.deg2rad(np.r_[angles, angles[0]])
        self.ranges = ranges
        self.ax = axes[0]

    def plot(self, data, *args, **kw):
        sdata = _scale_data(data, self.ranges)
        self.ax.plot(self.angle, np.r_[sdata, sdata[0]], *args, **kw)

    def fill(self, data, *args, **kw):
        sdata = _scale_data(data, self.ranges)
        self.ax.fill(self.angle, np.r_[sdata, sdata[0]], *args, **kw)

    def legend(self, *args, **kw):
        self.ax.legend(*args, **kw)

    def title(self, title, *args, **kw):
        self.ax.text(0.9, 1, title, transform = self.ax.transAxes, *args, **kw)
```

3.7.2. Finding ranges for the variables that will be used in Radar Chart

```
attributes = ['count', 'mean', 'sum']+[f'category_{i}' for i in range(15)]
[i for i in zip(cust_behaviour[attributes].min(), cust_behaviour[attributes].max())]


Count_range = [0,650]
Sum_range = [0.063,1800]
Mean_range = [0.013,180]
category_range = [0,100]
```

3.7.3. Parameters are set, and a radar chart is constructed to get insight into the clusters

```
fig = plt.figure(figsize=(25,50))
```

```
axis_title = ['count', 'mean (10^5)', 'sum (10^4)']+[f'category_{i}' for i in range(15)]
attributes = ['count', 'mean', 'sum']+[f'category_{i}' for i in range(15)]
ranges = [Count_range, Sum_range, Mean_range]+[[0,100] for i in range(15)]
index  = np.arange(n_clusters)

n_groups = n_clusters ; i_cols = 3
i_rows = n_groups//i_cols
size_x, size_y = (1/i_cols), (1/i_rows)

for ind in range(n_clusters):
  #Setting figure cosmetics
  ix = ind%3 ; iy = i_rows - ind//3
  pos_x = ix*(size_x + 0.05)
  pos_y = iy*(size_y + 0.05)
  location = [pos_x, pos_y]
  sizes = [size_x, size_y]
  #Filling details of chart
  data = np.array(cust_behaviour.loc[index[ind], attributes])
  radar = RadarChart(fig, location, sizes,axis_title, attributes, ranges)
  radar.plot(data, color = 'b', linewidth=2.0) #linewidth of vlue plot inside
  radar.fill(data, alpha = 0.2, color = 'b') #inside of blue plot
  radar.title(title = 'cluster no {}'.format(index[ind]), color = 'r')
  ind += 1
```

3.8. Cluster Insight using Parallel Coordinates plot

```
df = px.data.iris()
fig = px.parallel_coordinates(cust_behaviour, color="mean",
                              dimensions=['count', 'mean', 'sum']+[f'category_{i}' for i in range(15)],
                              color_continuous_scale=px.colors.diverging.Tealrose,
                              color_continuous_midpoint=2)
fig.show()
pio.write_image(fig, 'parallel_coordinates.png')
```

4. Classification of Customers

We are using multiple machine learning algorithms to classify customers **in** the categories we have already established. To train the algorithms to make predictions **for** the future.
So we will be adjusting classifiers **and** test them

4.1.1. A class is defined to ease up the process functionality

In order to simplify the use, a **class is** created, which will interface several common functionalities **in** the classifiers

```
class Class_Fit():
  # Move all class fields at the top (assign nothing if nothing can be assigned)
  def __init__(self, clf, params=None):
    self.clf = (clf(**params) if params else clf())
    self.grid = None
    self.predictions = None

  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)))
```

4.1.2. Shortlisting only means of the purchase and the categories that order belongs to, assigning that on the X axis and putting clusters on the Y axis

```
columns = ['mean']+[f'category_{i}' for i in range(15)]
X = selected_customers[columns]
Y = selected_customers['cluster']
```

4.1.3. Splitting the dataset into Test and Train using a size of 80%

```
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, train_size = 0.8)
```

4.1.4. Defining class for the learning curve

Code is used from sklearn documentation and a class is made for learning curve

```
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
```

4.1.5. Defining class for the confusion matrix

A class is constructed to make the confusion matrix. The code is taken from sklearn documentation:

```
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')
```

4.2. Performing SVC

4.2.1. Calling Class on the SVC and Setting hyperparameters and number of folds

```
svc = Class_Fit(clf = svm.LinearSVC)
svc.grid_search(parameters=[{'C': np.logspace(-2, 2, 10)}], Kfold=5)
svc.grid_fit(X=X_train, Y=Y_train)
svc.grid_predict(X_test, Y_test)
```

The precision output is good. But as there was an imbalance in the sizes of the clusters, so we will look how the predictions and real values actially look. To do that we will make confusion matrix

4.2.2. Build Confusion Matrix for SVC

```
class_names = [i for i in range(n_clusters)]
cnf_matrix = confusion_matrix(Y_test, svc.predictions)
np.set_printoptions(precision=2)
plt.figure(figsize = (10,10))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize = False, title='SVC Confusion matrix')
```

4.2.3. Classification Report for SVC's confusion matrix

```
x_df=pd.DataFrame(classification_report(Y_test, svc.predictions, output_dict=True)).transpose()
x_df


print("Micro f1 score for SVC: " +
    str(f1_score(Y_test, svc.predictions, average="micro")))
print("Weighted f1 score for SVC: " +
    str(f1_score(Y_test, svc.predictions, average="weighted")))

stats.hmean(x_df.iloc[:,1:4],axis=0)

fbeta_score(Y_test, svc.predictions, average='micro', beta=0.5)


fbeta_score(Y_test, svc.predictions, average='weighted', beta=0.5)

f1 = f1_score(Y_test, svc.predictions, average='weighted')
f0_5 = fbeta_score(Y_test, svc.predictions, beta=0.5, average='weighted')
f2 = fbeta_score(Y_test, svc.predictions, beta=2, average='weighted')
prec = precision_score(Y_test, svc.predictions, average='weighted')
rec = recall_score(Y_test, svc.predictions, average='weighted')

print(f1,f0_5,f2,prec,rec)
```

4.2.5. Plotting Learning Curve

A learning curve **is** used to test quality of a fit. This will detect possible drawbacks **in** the model like over fitting **or** underfitting.

4.2.5.2. Setting parameters for the plot

```
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])
```

4.3. Performing Logistic Regression

4.3.1. Calling class on the logistic regression and setting hyperparameters to find precision

```
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)

class_names = [i for i in range(n_clusters)]
cnf_matrix = confusion_matrix(Y_test, lr.predictions)
np.set_printoptions(precision=2)
plt.figure(figsize = (8,8))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize = False, title='Logistic Regression Confusion
matrix')

pd.DataFrame(classification_report(Y_test, lr.predictions, output_dict=True)).transpose()
```

```
print("Micro f1 score for Logistic Regression: " +
    str(f1_score(Y_test, lr.predictions, average="micro")))
print("Weighted f1 score for Logistic Regression: " +
    str(f1_score(Y_test, lr.predictions, average="weighted")))


#### 4.3.2. Plotting the curve of training and testing scores to observe the quality

g = plot_learning_curve(lr.grid.best_estimator_, "Logistic Regression learning curves", X_train, Y_train,
                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])

### 4.4.    Performing K-Nearest Neighbors

#### 4.4.1. Calling class on the K-Nearest, and setting hyperparameters to find precision

knn = Class_Fit(clf = neighbors.KNeighborsClassifier)
knn.grid_search(parameters = [{'n_neighbors': np.arange(1,100,1)}], Kfold = 5)
knn.grid_fit(X = X_train, Y = Y_train)
knn.grid_predict(X_test, Y_test)

class_names = [i for i in range(n_clusters)]
cnf_matrix = confusion_matrix(Y_test, knn.predictions)
np.set_printoptions(precision=2)
plt.figure(figsize = (8,8))
plot_confusion_matrix(cnf_matrix, classes=class_names,
               normalize=False, title='Nearest Neighbors Confusion matrix')


pd.DataFrame(classification_report(Y_test, knn.predictions, output_dict=True)).transpose()

print("Micro f1 score for Nearest Neighbors: " +
    str(f1_score(Y_test, knn.predictions, average="micro")))
print("Weighted f1 score for Nearest Neighbors: " +
    str(f1_score(Y_test, knn.predictions, average="weighted")))


#### 4.4.2. Plotting the curve of training and testing scores to observe the quality

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])

### 4.5.    Decision Tree

#### 4.5.1. Calling class on the Decision Tree and setting hyperparameters to find precision

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)

class_names = [i for i in range(n_clusters)]
```

```
cnf_matrix = confusion_matrix(Y_test, tr.predictions)
np.set_printoptions(precision=2)
plt.figure(figsize = (8,8))
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      normalize=False, title='Decision tree Confusion matrix')


pd.DataFrame(classification_report(Y_test, tr.predictions, output_dict=True)).transpose()

print("Micro f1 score for decision tree: "+str(f1_score(Y_test, tr.predictions, average="micro")))
print("Weighted f1 score for decision tree: " + str(f1_score(Y_test, tr.predictions, average="weighted")))
```

4.5.2. Plotting the curve of training and testing scores to observe the quality

```
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])
```

4.6. Random Forest

4.6.1. Calling class on the Random Forest and setting hyperparameters to find precision

```
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)

class_names = [i for i in range(n_clusters)]
cnf_matrix = confusion_matrix(Y_test, rf.predictions)
np.set_printoptions(precision=2)
plt.figure(figsize=(8, 8))
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      normalize=False, title='Random Forest Confusion matrix')


pd.DataFrame(classification_report(Y_test, rf.predictions, output_dict=True)).transpose()

print("Micro f1 score for Random Forest: " +
    str(f1_score(Y_test, rf.predictions, average="micro")))
print("Weighted f1 score for Random Forest: " +
    str(f1_score(Y_test, rf.predictions, average="weighted")))
```

4.6.2. Plotting the curve of training and testing scores to observe the quality

```
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])
```

4.7. AdaBoost Classifier

4.7.1. Calling class on the AdaBoost Classifier and setting hyperparameters to find precision

```
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)

class_names = [i for i in range(n_clusters)]
cnf_matrix = confusion_matrix(Y_test, ada.predictions)
np.set_printoptions(precision=2)
plt.figure(figsize=(8, 8))
plot_confusion_matrix(cnf_matrix, classes=class_names,
                normalize=False, title='AdaBoost Confusion matrix')


pd.DataFrame(classification_report(Y_test, ada.predictions, output_dict=True)).transpose()


print("Micro f1 score for AdaBoost: " +
    str(f1_score(Y_test, ada.predictions, average="micro")))
print("Weighted f1 score for AdaBoost: " +
    str(f1_score(Y_test, ada.predictions, average="weighted")))
```

4.7.2. Plotting the curve of training and testing scores to observe the quality

```
g = plot_learning_curve(ada.grid.best_estimator_, "AdaBoost learning curves", X_train, Y_train,
                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])
```

4.8. Gradient Boosting Classifier

4.8.1. Calling class on the Gradient Boosting Classifier and setting hyperparameters to find precision

```
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)

class_names = [i for i in range(n_clusters)]
cnf_matrix = confusion_matrix(Y_test, gb.predictions)
np.set_printoptions(precision=2)
plt.figure(figsize = (8,8))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize = False, title='Gradient Boosting Confusion matrix')


pd.DataFrame(classification_report(Y_test, gb.predictions, output_dict=True)).transpose()

print("Micro f1 score for Gradient Boosting: " +
    str(f1_score(Y_test, gb.predictions, average="micro")))
print("Weighted f1 score for Gradient Boosting: " +
```

```
    str(f1_score(Y_test, gb.predictions, average="weighted")))
```

4.8.2. Plotting the curve of training and testing scores to observe the quality

```
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])
```

4.9. Voting Classifier

4.9.1. Finding the best parameters for each classifier

```
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_)
```

4.9.2. Doing voting from Sklearn to predict the class labels to ensemble well-calibrated classifiers, which will use sums of probabilities

```
votingC = ensemble.VotingClassifier(estimators=[('rf', rf_best),('gb', gb_best),
                                ('dt', tr_best)], voting='soft')
```

4.9.3. Training Dataset on the voting classifier

```
votingC = votingC.fit(X_train, Y_train)
```

4.9.4. Finding precession of the classifier

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

class_names = [i for i in range(n_clusters)]
cnf_matrix = confusion_matrix(Y_test, votingC.predictions)
np.set_printoptions(precision=2)
plt.figure(figsize = (8,8))
plot_confusion_matrix(cnf_matrix, classes=class_names, normalize = False, title='Voting Classifier Confusion matrix')


pd.DataFrame(classification_report(Y_test, votingC.predictions, output_dict=True)).transpose()


print("Micro f1 score for Voting Classifier: " +
    str(f1_score(Y_test, votingC.predictions, average="micro")))
print("Weighted f1 score for Voting Classifier: " +
    str(f1_score(Y_test, votingC.predictions, average="weighted")))


g = plot_learning_curve(votingC, "Voting Classifier 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])


# 5.        Testing the models

#### 5.1.  Using the test data frame, we separated the previous steps

purchase_price = test_data.copy(deep = True)

purchase_price

#### 5.2.  Grouping the dataset as before to create the transaction_per_user dataset

orders_per_customer=purchase_price.groupby(by=['Customer ID'])['purchase
price'].agg(['count','min','max','mean','sum'])
for i in range(15):
   col = 'category_{}'.format(i)
   orders_per_customer.loc[:,col] = purchase_price.groupby(by=['Customer ID'])[col].sum() /\
                                    orders_per_customer['sum']*100

orders_per_customer.reset_index(drop = False, inplace = True)
purchase_price.groupby(by=['Customer ID'])['category_0'].sum()

#_________________________
# Correcting time range
orders_per_customer['count'] = 5 * orders_per_customer['count']
orders_per_customer['sum']   = orders_per_customer['count'] * orders_per_customer['mean']

orders_per_customer.sort_values('Customer ID', ascending = True)[:5]

#### 5.3.  Converting the dataset into the matrix and standardising it

list_cols = ['count','min','max','mean']  + [f'category_{i}' for i in range(15)]
#_________________________________________________________________
matrix_test = orders_per_customer[list_cols].values
scaled_test_matrix = scaler.transform(matrix_test)

#### 5.4.  Performing a K-Means analysis to find the clusters on this dataset

Y = kmeans.predict(scaled_test_matrix)

#### 5.5.  Making predictions from the trained classifier and checking it with the K-Means answer to check
the precision of predictions

columns = ['mean']  + [f'category_{i}' for i in range(15)]
X = orders_per_customer[columns]

#### 5.6.  Improving the quality of classification’s prediction by combining the best outputting classifiers

classifiers = [(svc, 'Support Vector Machine'),
               (lr, 'Logistic Regression'),
               (knn, 'k-Nearest Neighbors'),
               (tr, 'Decision Tree'),
               (rf, 'Random Forest'),
```

```
            (gb, 'Gradient Boosting')]
#______________________________________
for clf, label in classifiers:
   print(30*'_', '\n{}'.format(label))
   clf.grid_predict(X, Y)

# Predicting the **Final Score**

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

6.1 References For Code:

FABIENDANIEL. (2019). Customer Segmentation. *Kaggle*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS One*, *10*(3), e0118432.

YASIR HUSSAIN. (2021). *Customer Segmentation Using RFM Method*. Kaggle.

YASSINE GHOUZAM. (2018). Don’t know why employees leave ? Read this. *Kaggle*.

James Burgess. (2021). *Customer segmentation*.

Nitara Bobal. (2021). *Retail Customer Analysis*.