UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Customer Lifetime Value Prediction using Machine Learning Techniques

Diploma Thesis

Argyrios Adam

Supervisor: Panagiota Tsompanopoulou

June 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Customer Lifetime Value Prediction using Machine Learning Techniques

Diploma Thesis

Argyrios Adam

Supervisor: Panagiota Tsompanopoulou

June 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόβλεψη Αξίας Κύκλου Ζωής Πελατών με Τεχνικές Μηχανικής Μάθησης

Διπλωματική Εργασία

Αργύριος Αδάμ

Επιβλέπουσα: Παναγιώτα Τσομπανοπούλου

Ιούνιος 2022

Approved by the Examination Committee:

Supervisor **Panagiota Tsompanopoulou**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Michael Vassilakopoulos**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Aspassia Daskalopulu**
Assistant Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

Throughout my journey to complete my undergraduate studies, a great number of people supported and encouraged me. Therefore I would like to convey my most heartfelt appreciation to these individuals, without whom I would not have made it.

To begin, I would like to express my profound gratitude to my supervisor, Associate Professor Panagiota Tsompanopoulou, for all of the assistance she has given me and for providing me with the opportunity to work on an very intriguing project that is highly connected to my areas of interest. Additionally, I fell inclined to extend my sincere appreciation to Emeritus Professor Elias N. Houstis, who not only exposed me to the fields of artificial intelligence and data science but also supported me with his expertise and made many resources available to me.

Furthermore, I would like to express my gratitude to my friends and my girlfriend who have provided me with many joyful experiences throughout the years and have been very supportive of me.

Finally, I want to express my heartfelt gratitude to my family, my parents and my siblings, for their unconditional love and support they have shown me during all of these wonderful years. Without you, I know for a fact that I would not be the person that I am today, nor would I have been capable of achieving the things that I have accomplished.

This Diploma Thesis is dedicated to my parents, Konstantinos and Eleni

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Argyrios Adam

Diploma Thesis

Customer Lifetime Value Prediction using Machine Learning Techniques

Argyrios Adam

Abstract

Customer relationship management (CRM) has become more important as businesses strive to preserve a competitive advantage in today's market (CRM). The primary goal of customer relationship management (CRM) is to increase an organization's profitability by creating long-term relationships with profitable customers. However, in order for anything like this to occur, it is necessary to first identify these valuable customers. It is common for companies to define a client's profitability in terms of customer lifetime value (CLV), which is the total amount of money they earn from a customer throughout the period of their relationship with the firm. Ultimately, the purpose of this research is to demonstrate how Machine Learning may be used to forecast CLV at both an individual and a collective level. Two different strategies are employed in order to achieve this goal: in the first approach, we propose a method for CLV prediction on individual consumers; in the second approach, we perform customer segmentation and suggest a method for classifying customers into these groups. Empirical assessments based on a publicly available dataset from the domain of online shopping demonstrate that our suggested models produce outcomes that are competitive in terms of performance.

Keywords:

Artificial Intelligence, Data Science, Machine Learning, Customer Relationship Management, Customer Lifetime Value, RFM Analysis

Διπλωματική Εργασία

Πρόβλεψη Αξίας Κύκλου Ζωής Πελατών με Τεχνικές Μηχανικής Μάθησης

Αργύριος Αδάμ

Περίληψη

Η διαχείριση πελατειακών σχέσεων (CRM) έχει γίνει πιο σημαντική καθώς οι επιχειρήσεις προσπαθούν να διατηρήσουν ένα ανταγωνιστικό πλεονέκτημα στη σημερινή αγορά. Ο πρωταρχικός στόχος της διαχείρισης πελατειακών σχέσεων (CRM) είναι να αυξήσει την κερδοφορία ενός οργανισμού δημιουργώντας μακροπρόθεσμες σχέσεις με κερδοφόρους πελάτες. Ωστόσο, για να συμβεί κάτι τέτοιο, είναι απαραίτητο να εντοπίσουμε πρώτα αυτούς τους πολύτιμους πελάτες. Είναι σύνηθες για τις εταιρείες να ορίζουν την κερδοφορία ενός πελάτη με βάση την αξία ζωής του πελάτη (CLV), που είναι το συνολικό χρηματικό ποσό που κερδίζουν από έναν πελάτη καθ' όλη την περίοδο της σχέσης τους με την επιχείρηση. Ουσιαστικά, ο σκοπός αυτής της έρευνας είναι να καταδείξει πώς η Μηχανική Μάθηση μπορεί να χρησιμοποιηθεί για την πρόβλεψη του CLV τόσο σε ατομικό όσο και σε συλλογικό επίπεδο. Δύο διαφορετικές στρατηγικές χρησιμοποιούνται για την επίτευξη αυτού του στόχου: στην πρώτη προσέγγιση, προτείνουμε μια μέθοδο για την πρόβλεψη CLV σε μεμονωμένους καταναλωτές, ενώ στη δεύτερη προσέγγιση, πραγματοποιούμε τμηματοποίηση πελατών και προτείνουμε μια μέθοδο για την ταξινόμηση των πελατών σε αυτές τις ομάδες. Εμπειρικές αξιολογήσεις που βασίζονται σε ένα δημοσίως διαθέσιμο σύνολο δεδομένων από τον τομέα των διαδικτυακών αγορών καταδεικνύουν ότι τα προτεινόμενα μοντέλα μας παράγουν ανταγωνιστικά αποτελέσματα απόδοσης.

Λέξεις-κλειδιά:

Τεχνητή Νοημοσύνη, Επιστήμη Δεδομένων, Μηχανική Μάθηση, Διαχείριση Πελατειακών Σχέσεων, Αξία Κύκλου Ζωής Πελατών, Ανάλυση RFM.

Table of contents

Appendix

List of figures

List of tables

Abbreviations

CRM	Customer Relationship Management
CLV	Customer Lifetime Value
AI	Artificial Intelligence
OCR	Optical Character Recognition
SVM	Support Vector Machine
MAE	Mean Absolute Error
MSE	Mean Squared Error
MBE	Mean Bias Error
MSLE	Mean Squared Logarithmic Error
KNN	K-Nearest Neighbours
RF	Random Forest
AAC	Allowable Acquisition Cost
IAC	Investment Acquisition Cost
RSS	Residual Sum of Squared Errors
KNN	K-Nearest Neighbours
SSE	Sum of Squared Errors
EDA	Exploratory Data Analysis
RFM	Recency Frequency Monetary

Chapter 1

Introduction

Firms must create innovation activities in order to capture client wants and enhance customer satisfaction and retention in today's business environment, which is becoming increasingly complicated and competitive [1]. When it comes to acquiring and retaining customers, customer relationship management (CRM) is an established method that has been widely used. The primary goal of customer relationship management (CRM) is to establish long-term and profitable relationships with customers [2]. A huge number of databases providing detailed information on demographics and client transactions exist in this area. Different customer relationship management (CRM) systems may be used to examine this data in order to determine client profitability. Customer Lifetime Value (CLV) is a term in client relationship management (CRM) that represents the present value of all future profits generated by a customer [3]. The use of customer lifetime value (CLV) to allocate marketing resources is predicated on the assumption that the future worth of a client can be properly predicted. This assumption is rarely challenged, and there is little actual evidence to support or refute its validity. Customers' future worth may be anticipated with increasing precision along a continuum. One extreme is when customers' prior behavior and the firm's marketing efforts are exactly predicted, this is called a perfect prediction. The other extreme is where the future behavior of consumers is completely independent of their previous behavior and the firm's marketing initiatives. In order to determine whether or not to implement relationship marketing and customer equity tactics, the company must first determine where it lies on the continuum [4]. Spending disproportionate resources on certain clients makes undeniable sense when their future activity can be anticipated correctly; but, when future behavior is unknown, such an investment makes no sense. As a result, the key issue for these organizations

is to figure out how to maximize the likelihood that their investments will be lucrative by developing models that can accurately anticipate future behavior of customers.

1.1 Subject of the thesis

The goal of this thesis is to demonstrate how machine learning techniques can be used to forecast customer lifetime value, a metric that is extremely important for marketing teams of firms who want to determine which customers are more valuable to the organization.

In order to complete this work, we divided it into two phases. First, we handle each customer's situation on an individual basis. Specifically, we attempt to estimate CLV for each customer, in this case the net profit of their transactions, by applying regression techniques to do so. Then we take a different approach to the problem, one that is more collaborative in nature. To be more specific, we seek to segment our customers by generating clusters of consumers, which we then attempt to utilize in order to classify new clients into the predefined categories.

In order to make our models as accurate as possible, we place a strong focus on data-preprocessing, which includes cleaning, normalization, and feature extraction. Finally, we examine and evaluate our models, using different metrics for each methodology, in order to draw conclusions about the efficacy of our models.

1.1.1 Contribution

The overall thesis contributions are summarized as follows:

- To research how CLV may be evaluated and forecasted by using two distinct approaches to the problem: individually and collectively.
- To demonstrate how detailed examination of our data may be utilized to uncover vital information that is concealed behind raw data.
- To examine numerous of machine learning methods for each strategy.
- To assess our proposed models with the goal of determining the one that is best suitable for our dataset.

1.2 Thesis Synopsis

In Chapter 2, we will cover the fundamental theoretical background regarding machine learning, customer lifetime value (CLV), and customer segmentation. In Chapter 3, we discuss the algorithms that will be utilized to estimate CLV for our different approaches. In Chapter 4, we explore and analyze our data. In particular, we do an in-depth exploration before moving on to pre-processing and feature engineering. In Chapters 5 and 6, we will assess the models that have been proposed so that we can pick the ones that perform the best for the different cases. The final chapter, Chapter 7, presents a summary of our work as well as a conclusion and some suggestions for potential future study.

Chapter 2

Theoretical Background

2.1 Introduction to Machine Learning

Machine learning falls under the broad field known as Artificial intelligence (AI). Understanding the structure of data and incorporating that knowledge into models that can be utilized and comprehended by humans is the ultimate purpose of machine learning.

Despite being a sub-field of computer science, machine learning might be considered to be separate from more conventional computational approaches. Traditional algorithms are sets of instructions that have been carefully programmed and are used by computers to do calculations or solve issues. The techniques of machine learning, on the other hand, make it possible for computers to learn from data inputs and then use statistical analysis in order to produce results that are contained within a specific range [5]. As a consequence of this, machine learning enables computers to construct models from sample data in order to automate decision-making mechanisms which are based on the data inputs.

Every single person who uses a piece of technology today has profited from machine learning. Users of social media networks may soon be able to tag and share photographs of their friends with the use of face recognition technology. [6] The technique known as optical character recognition (OCR) can convert photographs of written text into movable type [7]. Recommendation systems driven by machine learning make suggestions to users on which movies or television shows they should watch next based on the user's preferences [8]. Automobiles that drive themselves and rely on machine learning to navigate may soon be available for purchase by consumers [9].

A machine learning process consists, in general, from the following steps:

1. **Data Collection**, that will be used for training and testing the model

2. **Data Pre-Processing**. The data should then be "cleaned", that is, to remove poor quality data, sort it into a desired sequence, and ultimately modify it to have a single format. These first steps are usually time consuming and very important as well since they are the basis on which the models are later built.

3. **Selection of Machine Learning model**. At this stage the experimentation is done in order to choose which models should be used. There are many models of machine learning and there is no clear categorization of which model suits which problem. In this thesis we use various models which will be discussed in more detail later.

4. **Training**. This stage can be considered by many as the essence of machine learning. At this point a portion (approximately 60-80%) of the well-organized data collected in the previous stages passes through the model machine learning in order to "train" and learn to analyze similar data.

5. **Validation**. At this point a smaller chunk of the data (10-20%) is used in order to test whether the now trained model can respond to new inputs and thus make the final calculations and corrections depending on its performance.

6. **Testing**. After that, the last piece of data will be used to measure the final performance of the model.

7. **Model Improvement**. Most of the time after testing, changes need to be done in both the model architecture and the hyperparameters in order to have the desired result. In this case the process returns in the initial steps and the new model is tested again.

The described process can be visualized in Figure 2.1 which shows a standard machine learning workflow diagram.

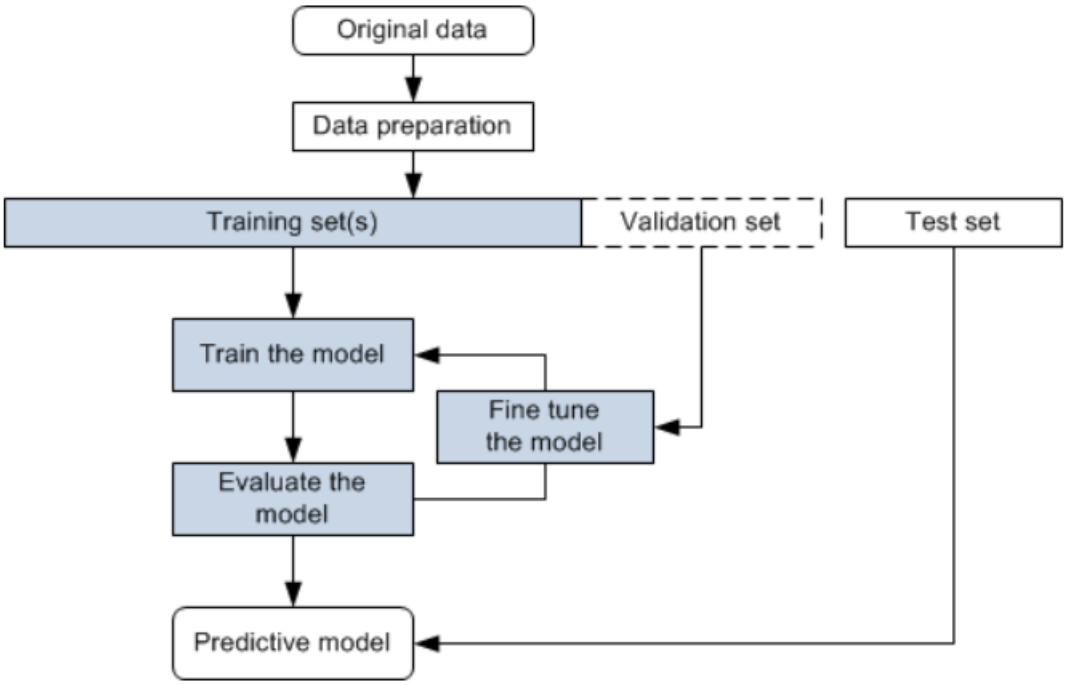


Figure 2.1: Machine Learning Workflow Diagram

Source : http://www.theobjects.com/dragonfly/dfhelp/2020-1/Content/Resources/Images/deep-learning/data-preparation-workflow.png

Machine learning algorithms can be divided into various categories depending on the format of the training data, how it is processed and type of the prediction/forecast. In this research we will focus on supervised and unsupervised learning.

2.2 Supervised Learning

Supervised Learning entails building a mathematical model that maps provided inputs (training set) to known desired outputs, which are frequently referred to as labels, with the ultimate objective of generalizing this model to unknown inputs. For instance, if we want to know if a picture contains a specific item the training data for a supervised learning algorithm will contain photos with and without this object (input data) as well as tags for each image (output data). These labels indicate whether or not each image contains the relevant object. A typical workflow diagram of supervised learning is depicted in Figure 2.2.

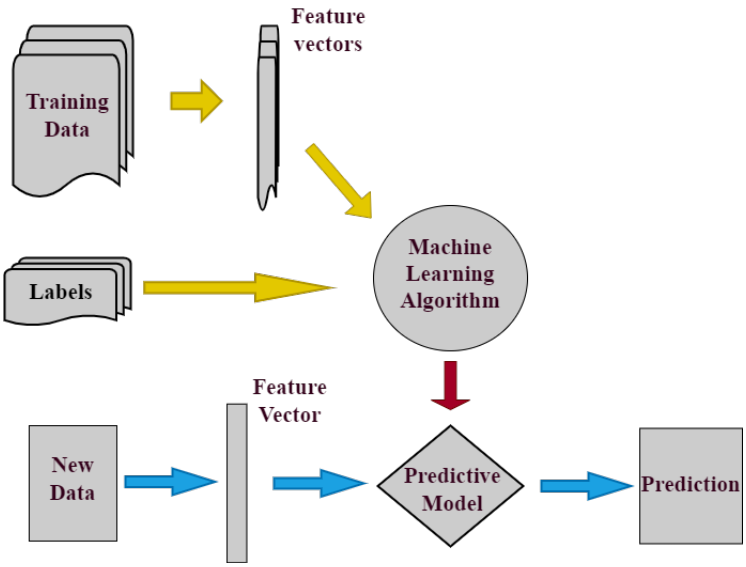


Figure 2.2: Supervised Learning

Let us formalize the supervised machine learning setup. Our training data comes in pairs of inputs $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathcal{R}^d$ is the input instance and y its label. The entire training data is denoted as

$$D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subseteq \mathcal{R}^d \times \mathcal{C} \tag{2.1}$$

where:

- $\mathcal{R}^d$ is the d-dimensional feature space
- $\mathbf{x}_i$ is the input vector of the i^{th} sample
- $\mathbf{y}_i$ is the label of the i^{th} sample
- $\mathcal{C}$ is the label space

The data points $(\mathbf{x}_i, y_i)$ are drawn from some (unknown) distribution $\mathcal{P}(X, Y)$. Ultimately we would like to learn a function h such that for a new pair $(\mathbf{x}, y) \sim \mathcal{P}$, we have $h(\mathbf{x}) = y$ with high probability (or $h(\mathbf{x}) \approx y$) [10].

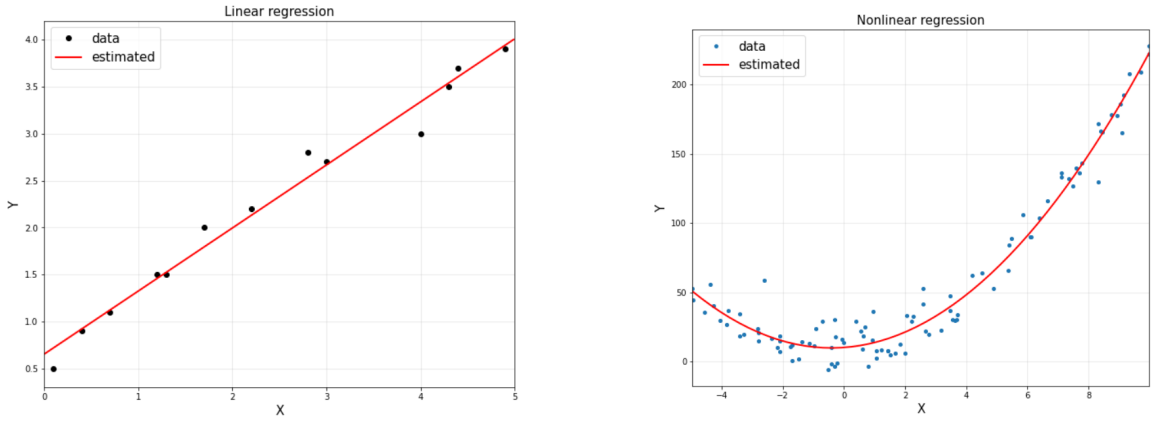
Based on the format of the output / response, he most common supervised learning algorithms are “regression” that fits the data and “classification” that separates the data.

2.2.1 Regression Analysis

A number of different machine learning techniques are included into the process of regression analysis. These techniques enable the value of one or more continuous outcome variables (y) to be predicted based on the value of one or more predictor variables (x). It is currently common practice to apply regression models in a broad range of contexts, including but not limited to the following:

- Financial forecasting (like house price estimates, or stock prices)
- Sales and promotions forecasting
- Testing automobiles
- Weather analysis and prediction
- Drug response modeling
- Time series forecasting

Some of the familiar types of regression algorithms are Linear Regression, Polynomial Regression, Ridge Regression, Lasso Regression, Decision Tree Regression, Support Vector Machine(SVM) Regression and others. A few of them will be extensively discussed and examined for our regression problem in Chapter 3. In Figure 2.3 we can see the difference between linear and nonlinear regression.



(1) Linear Regression (2) Nonlinear Regression

Figure 2.3: Linear vs Nonlinear Regression

2.2.2 Evaluating Regression Analysis

Loss Functions

The sole factor that is taken into consideration while evaluating a statistical model is its performance, which refers to how accurate the model's conclusions are. Because of this, a method is required to quantify the degree to which a certain iteration of the model deviates from the true values. This is where loss functions come into play. Loss functions are used to quantify the degree to which an estimated value deviates from the actual value. The following are some examples of loss functions that might be applied to regression problems:

1. Mean Absolute Error (MAE)
2. Mean Squared Error (MSE)
3. Mean Bias Error (MBE)
4. Mean Squared Logarithmic Error (MSLE)

We will see a couple of them more extensively in Chapter 5, where we evaluate the regression algorithms we used.

Variance

The amount that the estimate of the target function shifts in response to using different sets of training data is referred to as the variance. [11] The relationship between the input (features) and the output variables(values to predict) is determined by the target function, which is usually denoted by $\mathbf{f}$. Since the model should be general enough to adapt to different training data, the target function should be able to maintain its stability with only a small amount of variation. In order to prevent making inaccurate forecasts, we need to make sure that the variance is quite minimal. Because of this, the model has to be expanded so that it can account for previously undiscovered aspects of our data and generate more accurate forecasts.

Bias

Bias may be defined as the tendency of an algorithm to repeatedly learn the incorrect things because it does not take into consideration all the information from the input. It is

essential for our model to have a low level of bias in order for it to be accurate. In the occasion that the dataset has inconsistencies like missing values, outliers or mistakes in the input data, the bias will be large and the value predicted will be inaccurate.

Overfitting vs Underfitting

In order for a model to be considered optimal, it is anticipated that it would have low variance, low bias, and low error. In order to accomplish this, we will need to divide the dataset into a train dataset and a test dataset. After that, the model will learn patterns from the training dataset, and its effectiveness will be tested using the test dataset. In order to minimize the amount of mistake that occurs when the model is learning, we introduce an error function. If the model just memorizes or imitates the training data that is given to it, rather than detecting patterns in the data, then it will make inaccurate predictions on data that has not yet been seen. The accuracy of the model when applied to the test data is poor since the curve that is created from the training model will always pass through all of the data points. This phenomenon is referred to as **overfitting**, and it is caused by high **variance**. On the other hand, **underfitting** occurs when the model performs well on the test data but with poor accuracy on the training data. These fitting cases are depicted in Figure 2.4

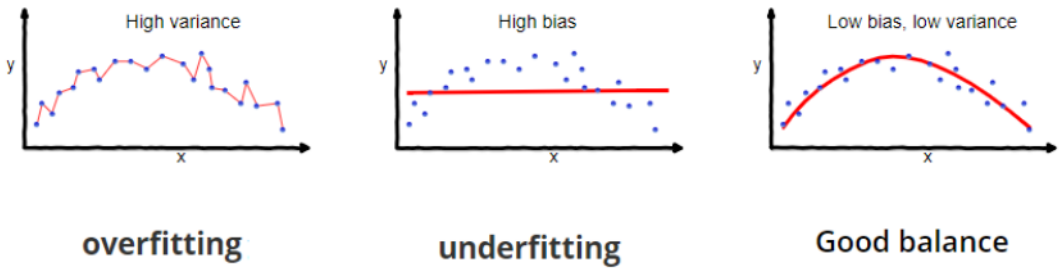


Figure 2.4: Fitting Cases

2.2.3 Classification Analysis

In the field of machine learning, classification is considered a supervised learning approach. It also refers to a challenge in predictive modeling, in which a class label is forecasted for a given example [12]. Mathematically speaking, what it does is map a function (f) from the variables that serve as input (X) to the variables that serve as output (Y) as target, label, or categories. It is possible to carry out the process on either structured or unstructured

data in order to forecast the class of provided data points. For instance, churn prediction for detecting which customers are likely to leave a service or to cancel a subscription, might be an example of a classification challenge. The most important distinction that can be made between classification and regression is that classification can forecast discrete class labels, whereas regression can help forecast a continuous quantity. The differences between classification and regression models are illustrated with a specific example in Figure 2.5. There is frequently located to be some degree of overlap between the two categories of machine learning algorithms.

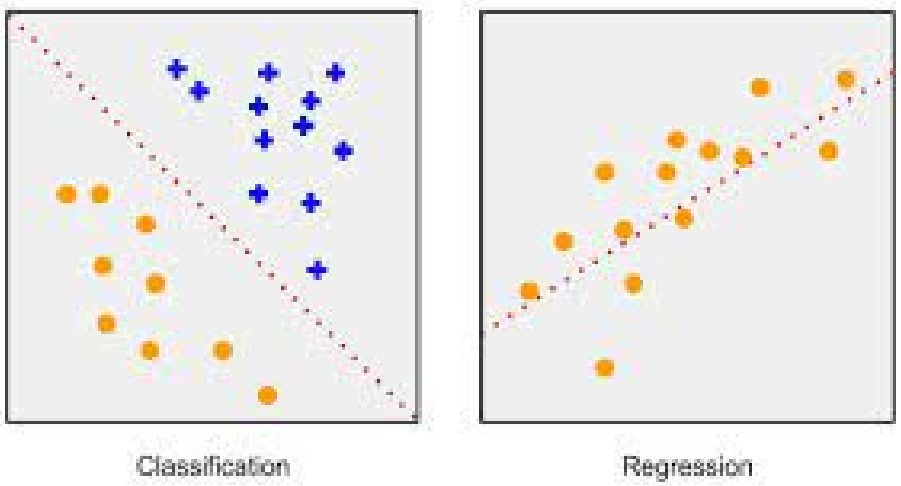


Figure 2.5: Regression vs Classification

Source : https://searchengineland.com/experiment-trying-predict-google-rankings-253621

In what follows, we will provide a synopsis of the major classification problems.

- **Binary Classification:** Binary classification is when there are just two possible class labels, such as “true” and “false” or “yes” and “no” [12]. For instance, in a task requiring binary classification, one class may represent the normal condition, while another class would represent the abnormal state. For example, the phrase “not spam” describes the normal condition of a task that requires filtering emails, but the phrase “spam” describes the problematic state of the work.

- **Multiclass Classification:** Multiclass classification often refers to classification problems that involve more than two different class labels [12]. In contrast to binary classification problems, the multiclass classification does not utilize the idea of normal and abnormal result distributions. Instead, instances are categorized as belonging to one of

many distinct classes from among a range of possible classifications. For instance, it can be a multiclass classification to classify patient hospital re-admission stay, where the possible categories are: 0 days, less than 30 days and more than 30 days [13].

- **Multi-label classification:** When it comes to machine learning, multi-label classification is an essential approach to take into account in situations where a single sample might be linked with many categories or labels. Therefore, it is an extension of multiclass classification, in which the classes involved in the issue are organised hierarchically, and each sample can concurrently belong to more than one class in each hierarchical level, such as in multi-level text classification. For instance, Stack Overflow posts may be presented under different tags such as “python”, “data science”, “machine learning” and so on [14]. In contrast to traditional classification tasks, which utilize class labels that are exclusive to one another, multi-label classification utilizes advanced machine learning algorithms that support the prediction of multiple mutually non-exclusive classes or labels [15].

In Figure 2.6 we see the difference between binary and multiclass classification and in Figure 2.7 the difference between multiclass and multi-label classification.

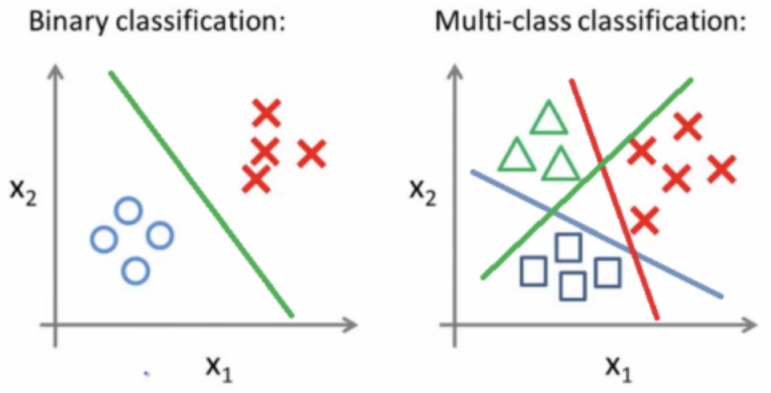


Figure 2.6: Binary vs Multiclass Classification

Source : https://medium.com/analytics-vidhya/ml06-intro-to-multi-class-classification-e61eb7492ffd

In the literature on machine learning and data science, a great deal of emphasis has been placed on the development of classification algorithms. Some of the most popular ones are Logistic Regression, Naïve Bayes, K-Nearest Neighbours(KNN), Decision Tree, Random

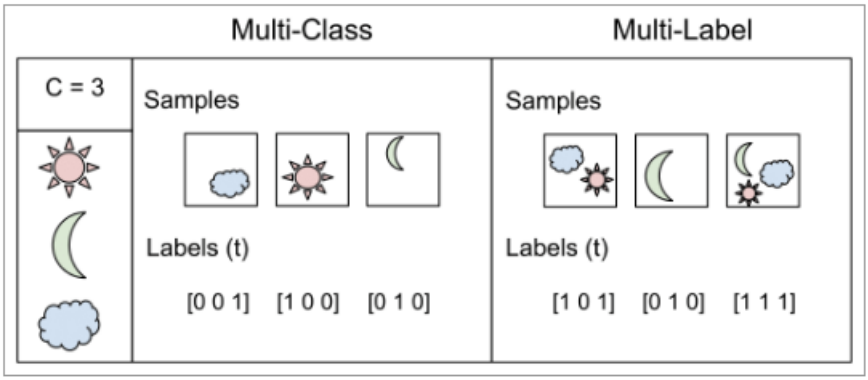


Figure 2.7: Multiclass vs Multi-Label Classification
Source : https://towardsdatascience.com/building-a-multi-label-text-classifier-using-bert-and-tensorflow-f188e0ecdc5d

Forest(RF), Support Vector Machine(SVM) and others. In Chapter 3 we will examine some of them more extensively.

2.3 Unsupervised Learning

Unsupervised learning is a process of machine learning in which a function is approximated to describe hidden structure in unlabeled data (for example, a categorization or a classification that was not included in the original data). Because the samples that are included in the learning model are not labeled in any way, there is no evaluation of the structure that the model ultimately discovers. One of the most popular applications of unsupervised learning is the estimate of the probability density function in distributions. However, unsupervised learning may be used to other issues as well, such as the extraction of concealed features, the identification of relevant patterns and structures and the grouping of data. A typical workflow diagram of unsupervised learning is depicted in Figure 2.8.

Clustering, density estimation, feature learning, dimensionality reduction, finding association rules, anomaly detection, are some of the most prevalent types of unsupervised learning tasks. In this thesis in order to do our customer segmentation task we will focus on clustering methods.

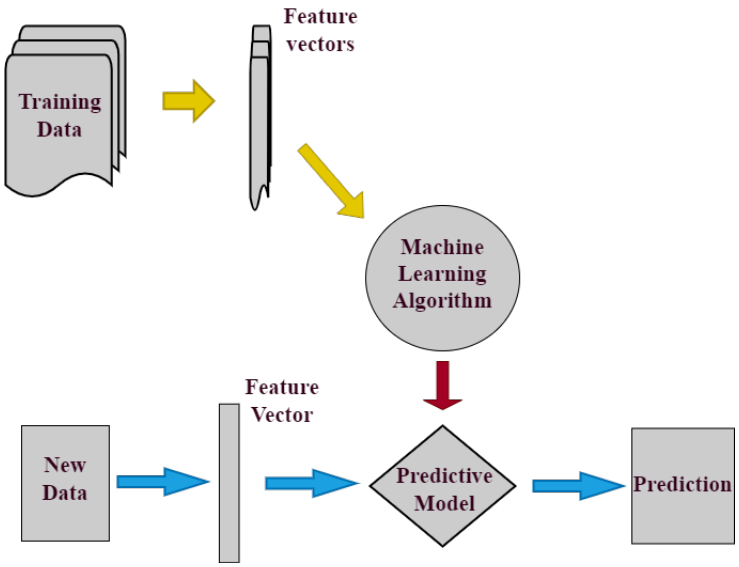


Figure 2.8: Unsupervised Learning

2.3.1 Cluster Analysis

Cluster analysis, which is also known as clustering, is an unsupervised method of machine learning that is used to locate and group similar data points in huge datasets without caring about the precise output. It achieves this by categorizing a set of attributes in such a manner that things in the same category, which is termed a cluster, are in some way more similar to each other than those that are in other clusters [12]. It is a technique for analyzing data that is frequently used to find interesting trends or patterns in the data, such as categories of consumers based on their behavior or transaction history. Clustering has a wide variety of potential applications, including but not limited to market research and customer segmentation, biological data and medical imaging, recommendation engine, pattern recognition, social network analysis, image processing, etc. In what follows, we will explore and outline many sorts of clustering algorithms in a short and concise manner.

- **Partitioning methods:** This clustering strategy organizes the data into a number of different groups or clusters by taking into account the attributes and correlations that exist within the data. In most cases, the number of clusters that should be produced by the techniques of clustering is decided either statically or dynamically by the data scientists or analysts, depending on the kind of applications that are intended to be used. K-means [16], K-Medoids [17], CLARA [18], and many other algorithms are some of the most prevalent clustering approaches that are based on partitioning methods.

- **Density-based methods:** These models utilize the idea that a cluster in the data space is an area of continuous high point density that is separated from other such clusters by regions of continuous low point density. This allows them to recognize separate groups or clusters. The points that do not belong to any cluster are referred to as noise. DBSCAN [19], OPTICS [20], and other similar algorithms are examples of common clustering methods that are based on density. In most cases, the density-based algorithms have difficulty dealing with clusters of data that have a comparable density and high dimensionality.

- **Hierarchical-based methods:** When doing hierarchical clustering, the goal is often to build a hierarchy of clusters, also known as the tree structure. In most cases, there are two distinct categories of hierarchical clustering strategies: **(i) Agglomerative**, which is a "bottom-up" technique in which each observations begin in their own cluster and pairs of clusters are merged as one, progressing up the hierarchy, and **(ii) Divisive**, which is a "top-down" approach in which all observations begin in one cluster and splits are conducted recursively, moving down the hierarchy, as seen in Figure 2.9. Figure 2.9 illustrates both of these approaches.

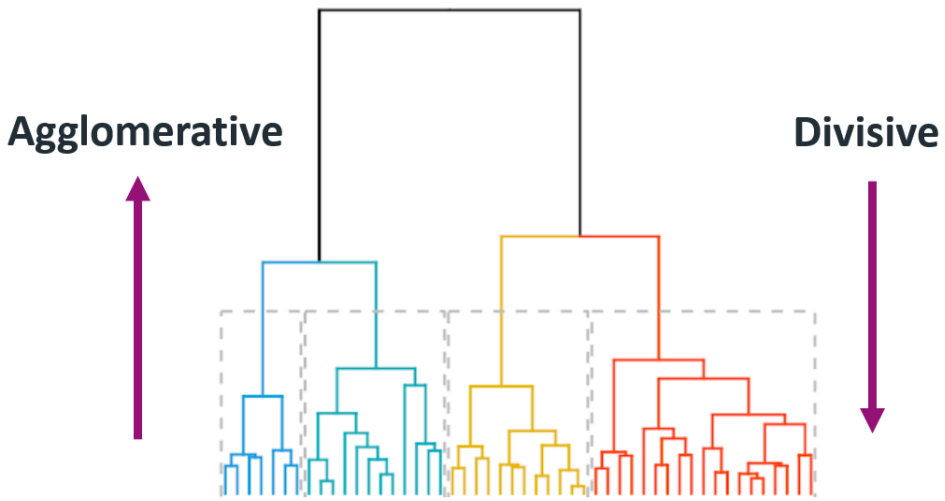


Figure 2.9: Hierarchical Clustering

Source : https://harshsharma1091996.medium.com/hierarchical-clustering-996745fe656b

- **Grid-based methods:** Grid-based clustering is an approach that works particularly well when dealing with enormous datasets. The approach behind obtaining clusters is to first summarize the dataset using a grid representation, and then to merge grid cells

after the grid representation has been completed. STING [21], CLIQUE [22], and other similar algorithms are examples of the conventional grid-based clustering methods.

2.4 Customer Lifetime Value (CLV)

Customer Lifetime Value (CLV) is one of the most essential key performance indicators (KPIs) to track and examine in relation to the customer experience. It is a measurement of how valuable a client is to a company in the long run in comparison to the initial transaction that the customer made with the firm. A fair Cost-per-Acquisition may be determined, among other things, with the use of this statistic, which is helpful to marketers. To be more exact, this measure indicates the total value that a customer brings to an organization during the course of their relationship with that company. There is no question that it is less expensive for businesses to keep a potential client than it is to acquire new customers. Therefore, the goal of most organizations should be to grow by maximizing the value of their current clientele.

For instance, if the CLV of an average client at a store is € 2,000 and it costs that business more than € 2,000 to acquire a new customer (when taking into account the cost of advertising, promotions, and so on), then the store may experience a loss in revenue. Companies are able to build plans to not just bring in new consumers, but also to keep the ones they already have while keeping their profit margins when they have knowledge of the CLV.

2.4.1 Calculation of CLV

Let's say a client has been going to the same supermarket in his area and purchasing cleaning supplies with a total value of € 100 each and every month for the past two years. His CLV for this specific supermarket is worth € 2400 Euros, which is calculated as follows: 100 Euros each purchase, multiplied by 12 Months, and then multiplied by 2 Years. On the other hand, as we can probably guess, determining the CLV for larger organizations is not a straightforward process.

Because of the numerous challenges that might develop, such as volatility in marketing and an inability to support computer systems and technology, a lot of businesses don't even bother to make an attempt to calculate the CLV of their consumers. If, on the other hand, businesses are able to successfully integrate their data silos and precisely quantify the expenses of their numerous operations, then determining the CLV is a simple procedure. CLV can be

calculated with the following steps:

- Determine the touchpoints at which the client contributes value to the company
- Understand and calculate the customer journey, which is the total of the experiences that a customer has with a company or a brand during the course of their relationship with the business
- Measurement of revenue at each touchpoint
- Calculate the sum of all for the lifetime of a customer

2.4.2 Importance of CLV

For giant corporations with hundreds of executives and sophisticated computer systems, the CLV may not be a significant barrier; but, for smaller corporations, the computation of the CLV can be extremely complicated and difficult.

However, as long as you keep in mind the value that a client gives throughout the course of his connection with a firm, there is no need to bother with complex calculations and procedures. With the responses to questions centered on the Customer Experience and the recording of feedback across all touchpoints, we can start to have a grasp on the fundamental CLV principles.

Once we have some information on the CLV of the consumers, we have two choices to choose from in order to determine how much money has to be spent to acquire new customers:

- **Allowable Acquisition Cost (AAC):** This is the maximum amount that we are willing to spend on any individual client provided that the total cost is lower than the profit that we make from the first sale to them. This is a technique that is employed for the short-term and places a greater emphasis on cash flow.
- **Investment Acquisition Cost (IAC):** This is the maximum amount that we are willing to spend on each consumer, knowing that we will make a loss on either the first or subsequent transaction made by that customer. This indicates that we are able to absorb the initial marketing expenditure in this long-term approach and that we have sufficient cash flow and other resources to do so.

Customer lifetime value (CLV) is the single most important factor that will decide how profitable a company is.

2.5 Customer Segmentation

When it comes to marketing, one strategy that may help raise earnings is to connect with consumers in order to learn what those individuals want [23]. Communication is constructed in a manner that takes into account the qualities of the customer. When it comes to communication, taking a personal approach is quite challenging to develop. Customer segmentation refers to the process of dividing consumers into groups based on the traits that they share in common; this process is required because it is essential. Schneider [24] defined market segmentation, as the process of separating prospective consumers into distinct groups. In its e-book, the e-commerce platform Magento[25] describes customer segmentation as an action that involves dividing consumers into groups that all have the same criteria. Customer segmentation offers a number of advantages, including the following: it makes it possible for us to match a customer with an offer of similar products; it modifies the manner in which we communicate with customers based on the information they provide; **it identifies the most profitable customers**; and it enables us to modify our products and services so that they better fulfill customer requirements. Customer segmentation is the process of categorizing or classifying an item or subject as belonging to a group that has been recognized as having certain characteristics in common with one another [23]. To visualize this, we can refer to Figure 2.10. In his study, Baer[23] examines Customer Segmentation Intelligence as a means of enhancing marketing by providing products or services that are tailored to match the requirements of certain customer groups. Customer Segmentation is the process of categorizing or classifying an item into a group that has a similarity in characteristic [26]. Segmentation is also used in CRM (Customer Relationship Management) to classify customers based on some similarities by segmenting the records of customer database.

Figure 2.10: Customer Segmentation

Source : https://www.nutshell.com/blog/market-segmentation

Chapter 3

Overview of the proposed methods

3.1 Linear Regression Models

3.1.1 Linear Regression

A method for modeling the connection that exists between an output variable(or a vector) and one or more input variables(independent variables) is known as **linear regression** [27]. It is utilized in circumstances when it is anticipated that the relationship will be linear. The mathematical formula for the **simple linear regression**, which has one variable serving as an explanatory factor is:

$$\hat{y}_i = \beta_0 + \beta_1 * x_i + \epsilon_i \tag{3.1}$$

where $\hat{y}_i$ is the dependant variable, x_i is the independent variable, β_0 is the intercept, β_1 is the slope and ϵ_i is the error/disturbance term, often called random noise, for instance i. **Multiple linear regression** is the extension of the above, where we consider n observations of one dependent variable and p explanatory variables. The basic multiple linear regression formula is:

$$\hat{y}_i = \beta_0 + \beta_1 * x_{i1} + \beta_2 * x_{i2} + \ldots + \beta_p * x_{ip} + \epsilon_i \tag{3.2}$$

for each observation $i = 1, \ldots, n$

Simplified, the above equation can be written as:

$$\hat{y}_i = \beta_0 + \sum_{j=1}^{p} (\beta_j * x_{ij}) + \epsilon_i \tag{3.3}$$

The model is validated by determining all of the relevant β_j and β_0 coefficients that minimize a cost function. To be more specific, the approach of least squares, which minimizes

the residual sum of squared errors (RSS), is the one that is used the most frequently in fitting data.

$$RSS = \sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}(\beta_j * x_{ij}) - \beta_0)^2 \tag{3.4}$$

where $\hat{y}_i$ are the predictions and y_i are the true values.

3.1.2 Ridge Regression

The Ridge Regression approach applies the same line of reasoning as the standard least squares method, but in addition, it includes a lambda regularization parameter in its analysis. The lambda parameter has a value that is positive, and its purpose is to reduce the size of the regression coefficients β of the linear model. This brings down the variance of the predictions that are produced by the model, as well as the danger of **overfitting** [28]. At this point, the objective is to minimize the loss function:

$$RSS_{ridge} = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}(\beta_j * x_{ij}) - \beta_0)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{3.5}$$

Equation 3.5 becomes the same as Equation 3.4 when the lambda value is set to zero, and at that point, we deal with the problem of normal least squares. When the value of the lambda parameter increases, the size of the coefficients β decrease, and when the coefficients β reach extremely low levels, the model loses its credibility of providing accurate predictions (**Underfitting**) [28]. As a result, the value of the lambda parameter needs to be meticulously chosen and placed at an ideal value. The name of the technique that Ridge Regression employs in order to achieve the desired level of regularization is called **L2 regularization** [29].

3.1.3 Lasso Regression

The Lasso regression method, also known as the Least Absolute Shrinkage and Selection Operator, is another variation of the normal least squares method. This method, like the Ridge model, reduces the size of the regression coefficients b by using a regularization parameter called lambda. However, the Lasso regression method encourages sparsity [30] by providing a regularized feature selection [31], which translates to choosing the most significant ones and ignoring the ones that do not have a major impact on the forecast. The following is the loss equation that has to be minimized:

$$RSS_{lasso} = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}(\beta_j * x_{ij}) - \beta_0)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{3.6}$$

Everything that pertains to the lambda normalization parameter of the Ridge regression is likewise applicable to the Lasso Regression. The primary distinction between the two models is that Lasso model makes use of **L1 regularization**, whereas the Ridge model makes use of the L2 regularization [29]. The size of the coefficients shrink by the regularization L2 process, but they are not driven to zero. In contrast, Lasso regularization L1 accomplishes feature selection by excluding the coefficients for those attributes that have a minor impact on the prediction [31].

3.2 Support Vector Machines (SVM)

Support Vector Machines are a type of supervised machine learning approach that may be used for issues involving regression and classification. This method works by analyzing data in order to find patterns in the data. Support Vector Machines were conceived by Cortes and Vapnik in 1995 [32].

The calculation of a hyperplane of separation is the core concept of support vector machines (SVMs), which aims to maximize the margins between different data classes.It is generally agreed that support vector machines (SVMs) are among the most effective techniques for the categorization and modeling of data. Nevertheless, despite the excellent performance it offers, this method does have a few drawbacks. In particular, its performance on issues involving more than two classes cannot be compared to the performance it has on problems involving only two classes (binary classification). This is due to the fact that it employs an approximation technique to minimize complexity, which ultimately results in the model having less effectiveness.

3.2.1 Linearly Separable Problems

Support Vector Support Engines can be used to solve a binary classification problem by maximizing the distance between the nearest points in each class. The end outcome is the discovery of a single unique divisive hyperplane that maximizes the margin throughout the whole training dataset, hence attaining much improved classification performance.

Let us consider a set of N sample pairs (input-output pairs) of a training set: $\{x_i, y_i\}, i = 1, ..., N$, where $y_i \, \epsilon\{-1, +1\}$ and $x_i \epsilon \mathcal{R}^2$. We want to create a function $f(x)$ that calculates y at x.

$$f(x) = w^\tau x + b = \sum_{i=1}^{p} w_i x_i + b \tag{3.7}$$

where $w^\tau x + b = 0$ is our hyperplane (a line in this case) and w is the vector perpendicular to the hyperplane as we can see in Figure 3.1.

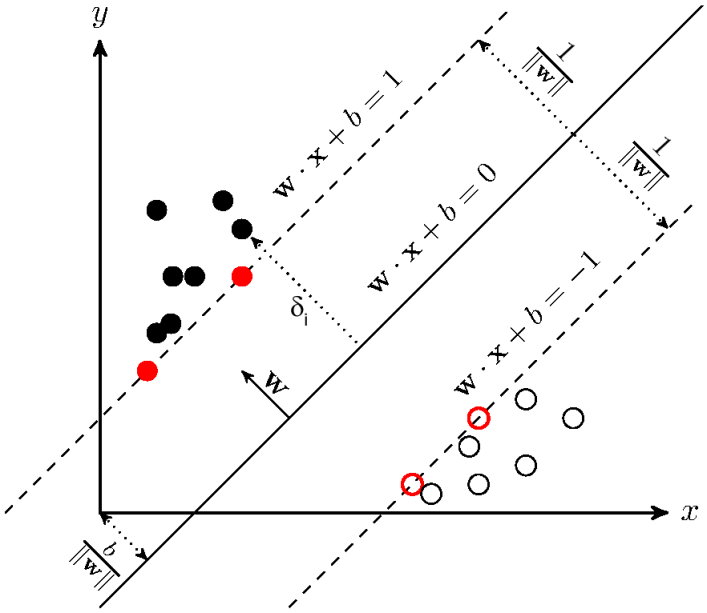


Figure 3.1: Graphic representation of a support vector machine

Source : https://staesthetic.wordpress.com/2014/02/20/update-on-digit-recognizer-using-svm-library/

The support vectors are the points closest to the hyperplane. This method aims at choosing b and w so that the data can be described with the following inequations.

$$w^\tau x_i + b \leq -1, \; with \quad y_i = -1 \tag{3.8}$$

$$w^\tau x_i + b \geq +1, \; with \quad y_i = +1 \tag{3.9}$$

which can be written as

$$y_i(w^\tau x_i + b) \geq 1 \Rightarrow y_i(w^\tau x_i + b) - 1 \geq 0 \tag{3.10}$$

while the support vectors can be described by two hyperplanes (lines)

$$w^\tau x_i + b = -1 \tag{3.11}$$

$$w^\tau x_i + b = +1 \tag{3.12}$$

The distance between these two hyperplanes is equal to $\frac{2}{||w||^2}$. The algorithm's purpose is to maximize this distance

$$\max \frac{2}{||w||^2},\ subject\ to: \quad y_i(w^\tau x_i + b) - 1 \geq 0 \tag{3.13}$$

or

$$\min \frac{||w||^2}{2},\ subject\ to: \quad y_i(w^\tau x_i + b) - 1 \geq 0 \tag{3.14}$$

Hard-Margin

In the case where classes are perfectly separable, the above solution works fine. However, in most real-world problems some points belond to the wrong side of the hyperplane.

Soft-Margin

In order to tackle to aforementioned issue we introduce a set of *"soft"* variables $\xi = \{\xi_1, \xi_2, ..., \xi_n\}$ for the points in the wrong side [33]. This way we accomplish a partial relaxations of our limitations.

Similarly to what we did before the optimization function becomes:

$$\min \frac{||w||^2}{2},\ subject\ to: \begin{cases} y_i(w^\tau x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \\ \sum_{i=1}^{N} \xi_i \leq K \end{cases} \tag{3.15}$$

which can be rewritten as

$$\min \frac{||w||^2}{2} + C\sum_{i=1}^{N} \xi_i,\ subject\ to: \begin{cases} y_i(w^\tau x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \tag{3.16}$$

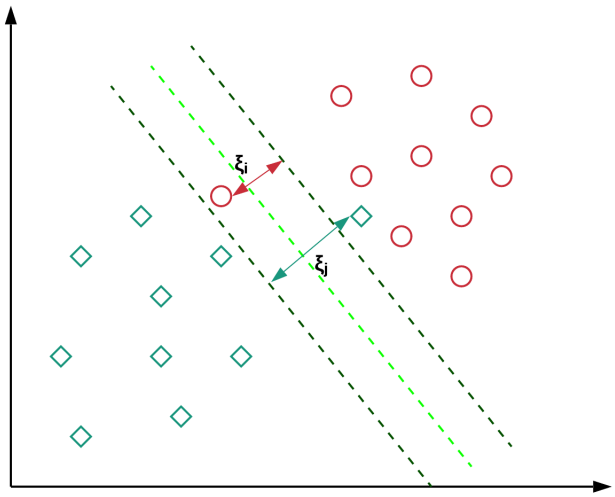


Figure 3.2: Soft-Margin

Source : https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe

where C is the regularization parameter giving more weight in minimizing the error. The soft-margin approach can be seen in Figure 3.2.

3.2.2 Non-Linearly Separable Problems

We went through how SVMs may be used to tackle linearly separable issues in circumstances when the training data can be fully separated using a hyperplane (hard-margin SVM), or at least with some points that have been incorrectly categorized (soft-margin SVM) [33].

In situations in which classes cannot be linearly distinguished in any way, support vector machines can be expanded to handle the task. If this is the case, then we will need to use non-linear functions in order to transfer the points in our data onto a new space that has a higher dimension. The transformation into a space with a greater dimension can be accomplished by the use of a technique known as the kernel trick (Figure 3.3). At this point, we will discuss one of the most well-known kernels, which is called the Gaussian kernel [34], and it is described by the mathematical relationship that is presented below:

$$\phi(x, x') = \exp^{(-\gamma||x-x'||^2)} \tag{3.17}$$

where ϕ is the kernel function that transforms a point x into x' in the new space and $\gamma > 0$ a

parameter controlling the width of the Gaussian distribution. Large γ causes overfitting while small γ causes underfitting.

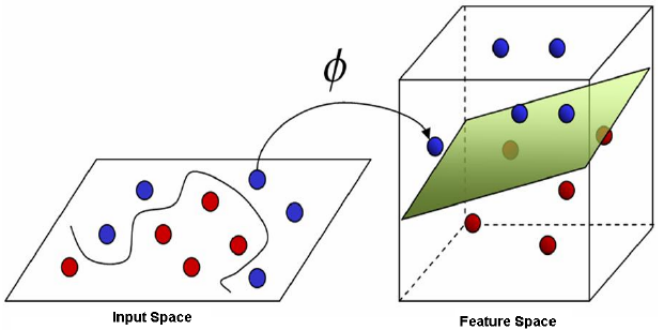


Figure 3.3: Non-linear SVM with the kernel trick

Source : https://www.researchgate.net/figure/Kernel-trick-ph-from-a-Input-Space-to-b-Feature-Space-an-unsupervised-learning-approach_fig2_339097504

3.3 Decision Trees

Decision trees [35] are widely recognized as one of the most effective supervised machine learning methods. It is a technique for building and approximately approximating functions with outputs that can either be discrete or continuous values. This algorithm's output is a tree structure with nodes and branches (Figure 3.4) that describe the data as well as the rules that were used to construct the tree (decision rules). Each internal node of the tree corresponds to a value control condition of an attribute of the instances used in the construction of the tree. Each branch moving away from a parent node (next level) refers to a different potential value or range of values that have a similar class for the feature that was being checked on that node. Moving onto the next level, the data is then segmented according to this attribute, and this process is repeated until the algorithm reaches the outer nodes of the tree (leaves), which correspond to the output values of the data.

The simplicity with which decision trees can provide a natural interpretation of the data that are derived from them is perhaps the most significant benefit that they offer. The computational cost of tree structures has a complexity $O(log_n)$ [36] in the case of binary trees, which are the most frequent. This means that tree structures are not only easy to use and comprehend by the user, but also easy to construct. At each stage of the calculation, the optimal attribute is chosen to be computed (which corresponds to a new node, one level below), and

only the data relating to that node, which is now the algorithm, is involved each time, rather than the complete original dataset.

A principle from the field of information theory is used in the search procedure for the ideal attribute. This is the **Entropy** [37] notion, which that measures the impurity or uncertainty in a group of observations. It determines how a decision tree chooses to split data. Entropy may be calculated using the following formula:

$$E = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{3.18}$$

where p_i the probability of randomly selecting an instance in class i and N the number of classes.

Entropy, of course, does not provide any information on whether or not the entropy of the remaining data will change if the particular feature is chosen. As a result of this, the idea of **Information Gain** [38] is presented. This concept illustrates whether or not the entropy of training data will be decreased in the event that the next feature A_i is chosen from the set of features $A_i, ..., A_n$. The formula for determining Information Gain looks like this:

$$IG(T, A) = E(T) - \sum_{u \epsilon A} \frac{|T_u|}{T} * E(t_u) \tag{3.19}$$

where T is the target column, A is the variable we are testing and u each value in A.

If the feature A_i is selected to act as a separation variable, the information gain will, in essence, result in a decrease in the amount of entropy in the entire data set. A increase in the density of information and, as a result, a more “compact” representation of the data is achieved through the reduction of entropy. In reality, the second term of the above relation 3.19 provides the entropy of the data after it has been separated depending on the specific feature that will be used. Therefore, the information gain for each candidate attribute is computed at each node, and the attribute that consistently yield the maximum overall gain is chosen as the winning candidate.

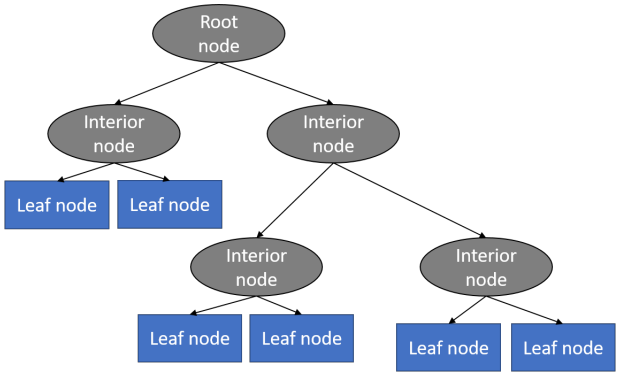


Figure 3.4: Decision Tree example

3.4 Bagging and Boosting Algorithms

3.4.1 Bagging

The **B**ootstrap **Agg**regat**ing** (Bagging) [39] approach is utilized in an effort to improve the stability and precision of machine learning algorithms that are employed in regression and classification. It is a "voting" approach, in which weak learners are diversified by training them in training sets of data that fluctuate very little from one another.

With the Bootstrap method we have L samples with replacement of our original training set. Take into consideration the following example in order to comprehend this idea. Let's say we have a box with N balls of varying colors at our disposal. The next step is to pick an item at random from among them, record its color, place it back in the box and repeat for N times (size of training set). As a consequence of this, certain observations may be selected more than once, while others may not be selected at all. In order to make use of these different variations of data, the base regressor or classifier has to be weak. This ensures that even minor shifts in the data used for training can result in significant shifts in the output. Unless they are weak learners, we will end up with a combination of classifiers or regressors that are nearly identical to one another. The most popular weak learners are decision trees.

Bagging Algorithm :

- Consider a training set with N observations and t samples with replacements.

  1. Take a sample with replacement of size N from the training set
  2. Apply the weak learner algorithm on it
  3. Save the model

- For each of the t models (classification case):

    1. Predict class

    2. Return the class with most predictions

In Figure 3.5 we can see how the workflow for the bagging algorithm in the classification case.

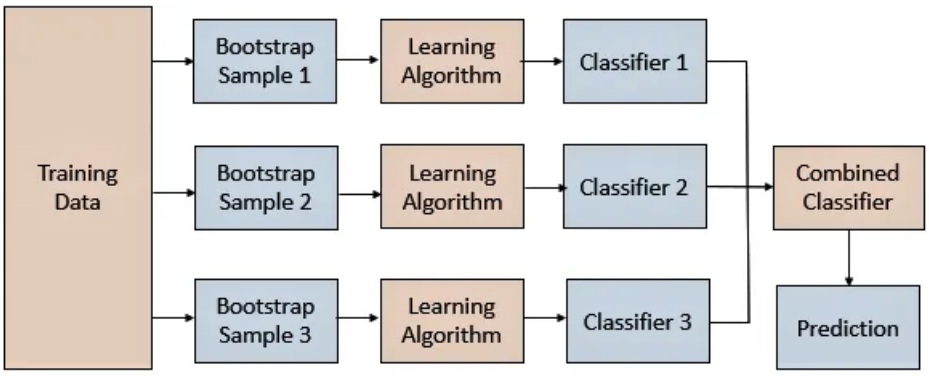


Figure 3.5: Bagging Algorithm for Classification

Source : https://corporatefinanceinstitute.com/resources/knowledge/other/bagging-bootstrap-aggregation/

3.4.2 Random Forest

The Bagging approach [39] discussed previously is what makes the Random Forest algorithm [40] innovative and differentiates it from simple decision trees. The algorithm that is utilized throughout the training process is referred to as the base classifier, and in the case of Random Forest, the Decision Tree is utilized. In the following stage, the findings from this step are compared to the results from the previous step in order to determine which one is the most appropriate. In the case of issues involving classification, the results are determined via the use of the voting technique, but in the case of problems involving regression, the results are often determined by the calculation of an average. In general, one may utilize a variety of statistical approaches in order to compare the findings. The nature of the issue being considered will determine which one of these solutions is applicable.

The ultimate purpose of the random forest technique is to lessen the impact of the bagging variance by lowering the correlation levels between individual trees. We may accomplish this by picking the input variables in a random order. To be more specific, after expanding a tree

in the bootstrapped set and prior to each separation, we randomly choose m of the total p input variables, where $m \leq p$, to serve as candidates for separation.

The implementation algorithm for the Random Forest method, which is depicted in Figure 3.6, is as follows:

for $\mathbf{b} = \mathbf{1}, ..., \mathbf{B}$

1. Create a bootstrap sample C with size N from the training set

2. We generate a random tree, T_b, using our data until the required number of nodes, n_{min}, is attained. This is done by selecting m out of p variables and picking the most appropriate ones.

3. Create the random trees $\left[\mathbf{T_b}\right]_{\mathbf{1}}^{\mathbf{B}}$

4. - **Regression:** We have $\hat{\mathbf{f}}(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{B}} \sum_{\mathbf{b=1}}^{\mathbf{B}} \mathbf{T_b}(\mathbf{x})$
   - **Classification:** We have $\hat{\mathbf{C}}(\mathbf{x}) = \mathbf{majorityvote}\left[\hat{\mathbf{C}}_\mathbf{b}(\mathbf{x})\right]_{\mathbf{1}}^{\mathbf{B}}$

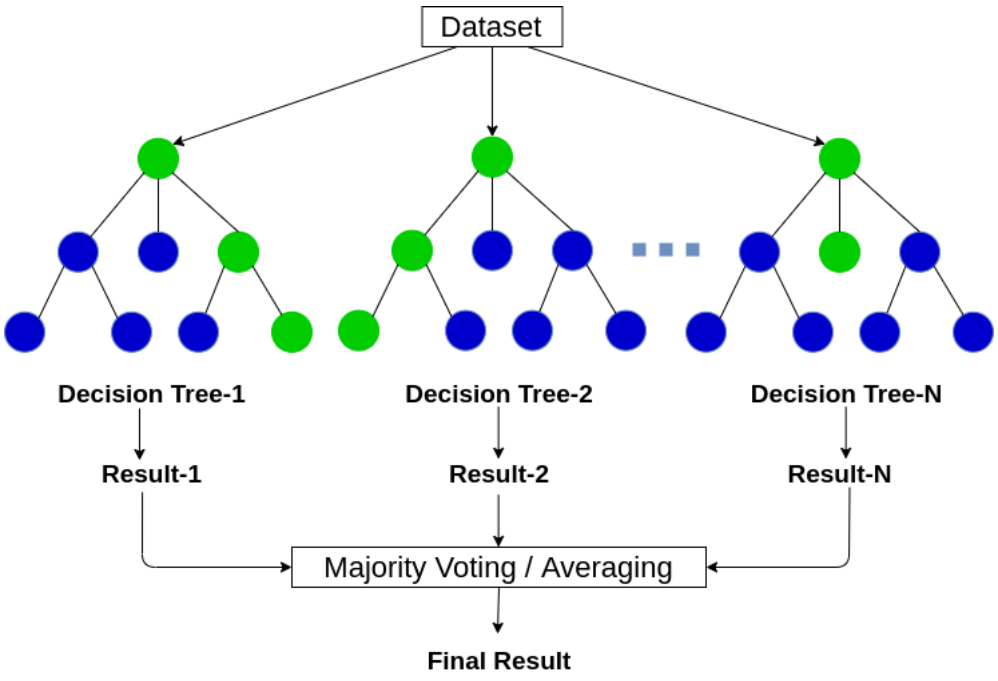


Figure 3.6: Random Forest

Source : https://ai-pool.com/a/s/random-forests-understanding

3.4.3 Boosting

Boosting techniques construct many models from a single piece of data by utilizing other methods of model construction, such as decision trees, which may not always result in an extremely accurate model [41]. The fundamental premise of boosting is to assign a weight to each observation that makes up the data collection. If some of the models wrongly classifies the observation, the weights assigned to them are increased.

The relative weights given to the data tend to move in both positive and negative directions when one model gives way to the next. Because the final model is cumulative and is composed of a series of models, each of the preceding models now has a degree of significance as a result (Figure 3.7). The boosting approach has several benefits, one of which is that it takes very little adjustment, and another is that the only assumption we have to make about base model is that it should be a weak learner. The drawback of boosting is that it is sensitive to random errors and can fail either when there is insufficient data or when the weak models are excessively complicated.

In order to put the method to use, one must first provide satisfactory responses to a pair of fundamental inquiries. Two questions need to be answered: the first is how each weak classifier/regressor should be picked, and the second is how, after multiple of these weak learners have been collected, they may be combined into a single model.

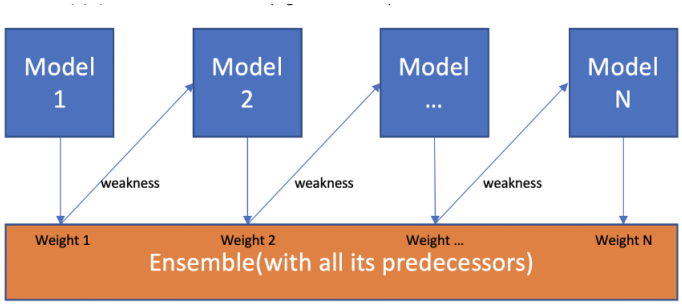


Figure 3.7: Boosting

Source : https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30

3.4.4 Boosting Algorithms

We will briefly refer to the two most popular boosting machine learning algorithms: Adaptive Boosting (AdaBoost) and Extreme gradient boosting (XGBoost) .

- **Adaboost:** Adaptive Boosting, often referred to as AdaBoost is an ensemble learning procedure that ameliorates weak regressors/classifiers by making use of their mistakes during training. In order to achieve that improvement AdaBoost uses an iterative method. Another name for this concept is called meta-learning [42]. The method used by Adaboost is called "Sequential ensembling" [43], while in the previously discussed Random forest technique it is called "Parallel ensembling" [44]. This method is able to produce a powerful learner by merging a large number of weal learners with low performance to achieve high accuracy. Considering the above, AdaBoost is referred to as an adaptive regressor/classifier since it considerably improves the efficiency of the process. The only drawback to that algorithm is its sensitivity to noise and outliers which might cause overfitting. The most ideal and common use of AdaBoost is with decision trees as a base model.

- **XGBoost:** Similarly to the Random Forest algorithm mentioned earlier in our research, Gradient Boosting is an ensemble algorithm that constructs the final model by combining a sequence of weak learners, most frequently decision trees. The gradient is utilized in order to achieve the goal of minimizing the loss function, which is analogous to how gradient descent is used to optimize weights in deep learning (neural networks) [45]. XGBoost is a gradient boosting algorithm that takes into consideration more precise estimates when selecting which model is more accurate. In order to decrease overfitting an enhance the model's generalization and its performance it computes advanced regularizations (L1, L2) [29] mentioned previously in Ridge/Lasso Regression. At the same time in order to minimize the loss it computes second-order gradients [46]. XGBoost is easy to comprehend and can successfully manage datasets of a large scale.

3.5 K-Nearest Neighbours (KNN)

The K-nearest neighbours (KNN) approach is a regression and classification strategy that is quiet widespread and frequently utilized. It is based on the utilization of distance-based metrics [47] [48]. It works on the assumption that the training set contains not only the input data(features), but also the desired target for each individual element. This is because KNN is a supervised learning approach. In order to place each newly given observation into the appropriate class(classification case), it is required to do a distance calculation between the

observation and every other element in the training set. In the end, the item is classified according to the k closest observation or in the regression case based on the average of its k neighbours.

The k-nearest neighbours approach is considered an instance-based learner [49], which means that learning is determined by analogy, rather than a generalized model like it is done in decision trees. Therefore, in the case of KNN, the concept of a "training step" does not exist and no model is formed until a new observation needs to be predicted. Instance-based learners are sometimes referred to as "lazy algorithms" for this reason. This process involves the retention of a large portion of the training set, since in order to estimate a new element previous observations of the training set are used to make the necessary comparisons; in comparison to SVMs, which we previously discussed about, where unnecessary observations that do not provide support can be easily disregarded.

The K-nearest neighbours(KNN) approach has a number of benefits, some of which include the capability of efficiently identifying complicated connections between variables, the ease with which it may be implemented and utilized and the typically high performance in regression and classification problems. The fact that numerous comparisons between observations are necessary, on the other hand, demands the use of correspondingly highly effective indexing techniques. Failing to do so will cause the algorithm to take significantly more time to perform the prediction, particularly in circumstances where the number of neighbours is large. In addition, the outcome is sensitive to the presence of input features with no importance as well as the number of neighbours, which increases the danger of overfitting.

As was noted, the fundamental concept underlying the KNN method is that any element's outcome is predicted according to the elements that are closer to it. To achieve this objective, distance measurements can be applied in order to quantify the degree of similarity that exists between the various observations that make up the dataset. The following are the kinds of distances that are utilized more frequently to measure the similarities of data points.

- **Euclidean Distance:** $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
- **Manhattan Distance:** $\sum_{i=1}^{n} |x_i - y_i|$
- **Minkowski Distance:** $(\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$

We can see an example of the KNN approach in Figure 3.8 , where we try to classify the "star" observation using $k = 5$ neighbours. Three neighbours are "circles" and two are "squares" and so we classify it as a "circle".

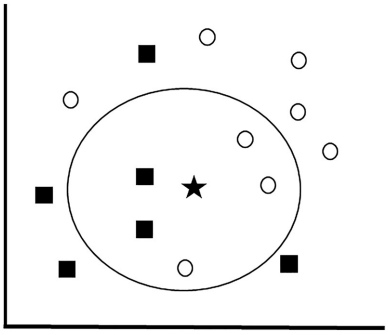


Figure 3.8: K-Nearest Neighbours example with k=5

3.6 K-Means Clustering

Clustering is primarily an unsupervised learning strategy that seeks to group observations based on a similarity index so that those who exhibit the most similarity are in the same group (cluster). Clustering techniques may be broken down into several groups that were mentioned in 2.3.1. The most popular of all clustering algorithms and the one used most frequently is K-means clustering.

K-means clustering is a method of vector quantization that was originated from signal processing and is utilized relatively frequently in the field of data science for the purpose of cluster analysis [50]. The objective is to divide the total number of observations, n, into k clusters, where k is a predefined number. The primary goal here is to establish k centers, one of which should correspond to each of the clusters. Therefore, each observation is though to belong to the cluster that contains the closest center.

Due to the fact that different settings create varying outcomes, these centers have to be positioned in the most optimal manner conceivable. Because of this, the most effective strategy is to put as much distance as possible between them. The Euclidean distance is usually used in the calculation of the distance between observations. The following step is to determine which center is closest to each observation that is part of our dataset. The initial stage of the algorithm, known as the main grouping is considered to have been finished when there are no points left for matching.

After that, the new centers need to be recalculated as gravitational centers of the clusters that were produced in the previous phase [51]. Therefore, in the instance that these k new cluster centers are computed, a fresh match has to be formed between the data points and the new cluster centers that are relatively closest to them. This phase is repeated and the centers of the k clusters are constantly changing until they converge at the optimal points. That is, until they cease making major improvements, at which point we should move on.

The evaluation of optimization is essentially comprised of nothing more than the process of minimizing an objective function, which is also referred to as the sum of squared errors (SSE) which is represented by the following relation:

$$SSE = \sum_{i=1}^{c} \sum_{j=1}^{c_i} (||x_i - y_j||)^2 \tag{3.20}$$

where $||x_i - y_j||$ is the Euclidean distance between x_i which represents the points of the cluster and y_i which represents the center of the cluster, c_i is the number of data points in cluster i and c is the total number of cluster centers.

In Figure 3.9 we can see an example of how data looks before and after k-means clustering (k=3).

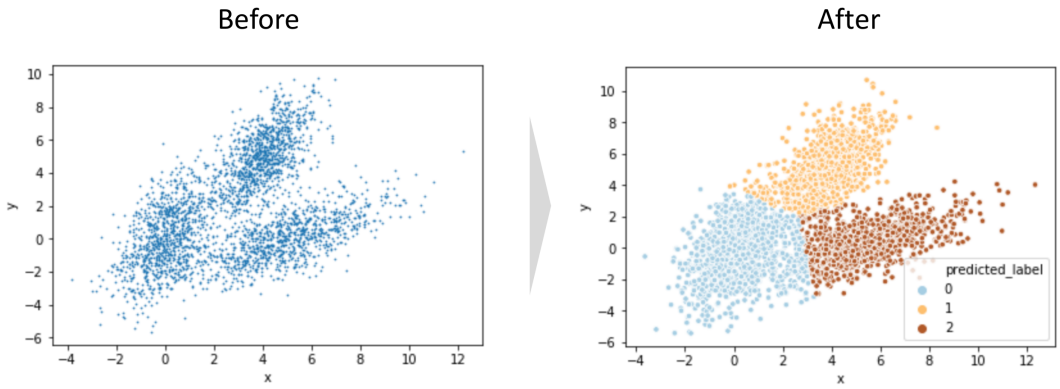


Figure 3.9: K-means clustering

Source : https://h1ros.github.io/posts/k-means-clustering/

Chapter 4

Data

4.1 Dataset

The data are obtained from **UCI Machine Learning Repository**, which is a free collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms[52]. This Online Retail dataset contains all the transactions occurring for a UK-based and registered, non-store online retail between 01/12/2009 and 09/12/2011. Our dataset contains $1.044.848$ rows and 8 columns. In Figure 4.1 we can see a sample of our dataset.

Attribute Information:

- **Invoice**: *Invoice number.* ***Nominal****, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.*
- **StockCode:** *Product (item) code.* ***Nominal****, a 5-digit integral number uniquely assigned to each distinct product.*
- **Description:** *Product (item) name.* ***Nominal****.*
- **Quantity:** *The quantities of each product (item) per transaction.* ***Numeric****.*
- **InvoiceDate:** *Invoice Date and time.* ***Numeric****, the day and time when each transaction was generated.*
- **UnitPrice:** *Unit price.* ***Numeric****, Product price per unit in sterling.*

- **CustomerID:** *Customer number.* ***Nominal****, a 5-digit integral number uniquely assigned to each customer.*

- **Country:** *Country name.* ***Nominal****, the name of the country where each customer resides.*

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
1015814	579518	22139	RETROSPOT TEA SET CERAMIC 11 PC	1	2011-11-29 18:05:00	4.95	12748.0	United Kingdom
910601	571905	22154	ANGEL DECORATION 3 BUTTONS	48	2011-10-19 14:51:00	0.42	12370.0	Cyprus
242134	512797	22089	PAPER BUNTING VINTAGE PAISLEY	12	2010-06-18 10:29:00	2.95	14895.0	United Kingdom
14294	490530	35979	SIX STRING SCANDINAVIAN HEART DECS	12	2009-12-06 16:01:00	2.95	15691.0	United Kingdom
731314	556927	23230	WRAP ALPHABET DESIGN	100	2011-06-15 14:47:00	0.42	14156.0	EIRE

Figure 4.1: Sample of our dataset

4.2 Data Pre-processing

4.2.1 Exploratory Data Analysis (EDA)

During **EDA** we attempt to gain a better understanding of our data by making use of the ***pandas*** [53] library, which is an effective and versatile tool for data analysis and manipulation, and the ***plotly***[54] library, which will allow us to visualize our data through the use of interactive high-quality graphs such as bar-charts, pie-charts, scatter plots, etc.
First and foremost, the first thing that we do is get some fundamental information about our data. To be more precise, we will go over some of our features in order to uncover hidden patterns in data or spot anomalies, both of which can assist us in gaining a deeper understanding of our data.

Invoice

The values in this column, which were described before in Section 4.1, are six-digit integers that represent each transaction. According to the data shown in Figure 4.2 , the majority of the transactions are carried out in the United Kingdom. However, since a single invoice might include a large number of products, so we need to look at the number of unique invoices for each country, which is shown in Figure 4.3.

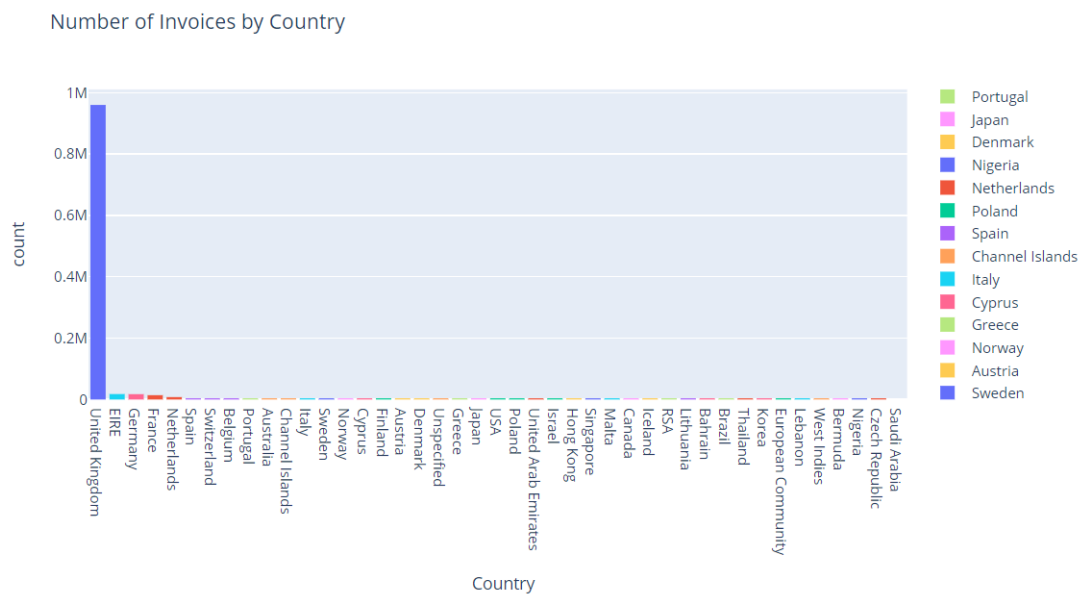


Figure 4.2: Number of Invoices by Country

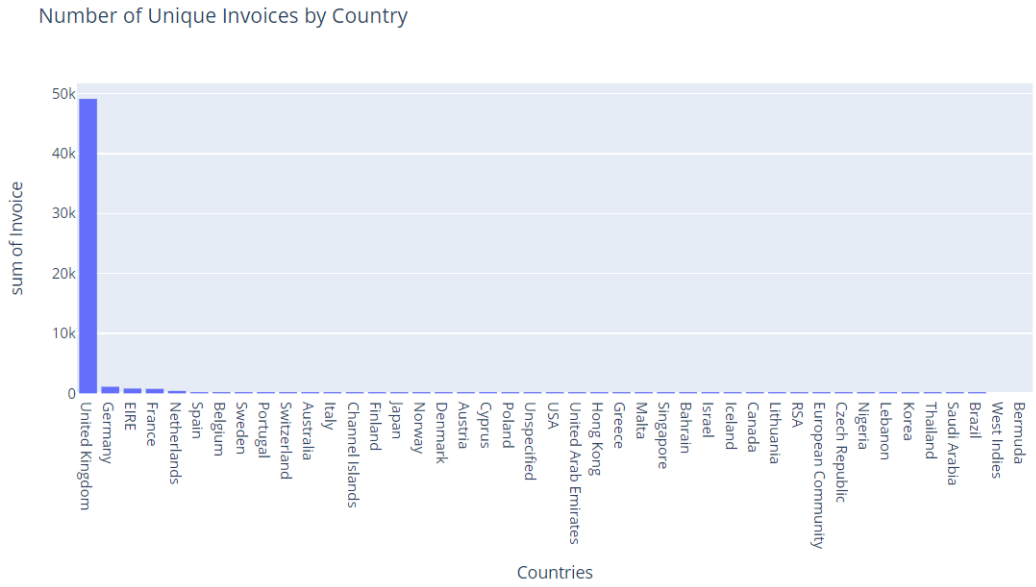


Figure 4.3: Number of Unique Invoices by Country

One thing that stands out is the amount of invoices in the UK. The Tables 4.1 and 4.2 are numerical representations of the graphs that were shown before; they highlight the monopoly that the UK market has in our data.

Table 4.1: Number of Invoices by Country

Country	Number of Invoices	Percentage
UK	959983	**91.88%**
EIRE	17689	1.69%
Germany	17363	1.66%
	Total = 1044848	

Table 4.2: Number of Unique Invoices by Country

Country	Number of Unique Invoices	Percentage
UK	49108	**91.57%**
Germany	1096	2.04%
EIRE	806	1.50%
	Total = 53628	

Additionally, some of our invoices start with the letter **'C'** or the letter **'A'**, which correspond to cancelled or abandoned order respectively. In Figure 4.4 we can see that the percentage of invoices which are either cancelled or abandoned is insignificant compared to normal invoices.

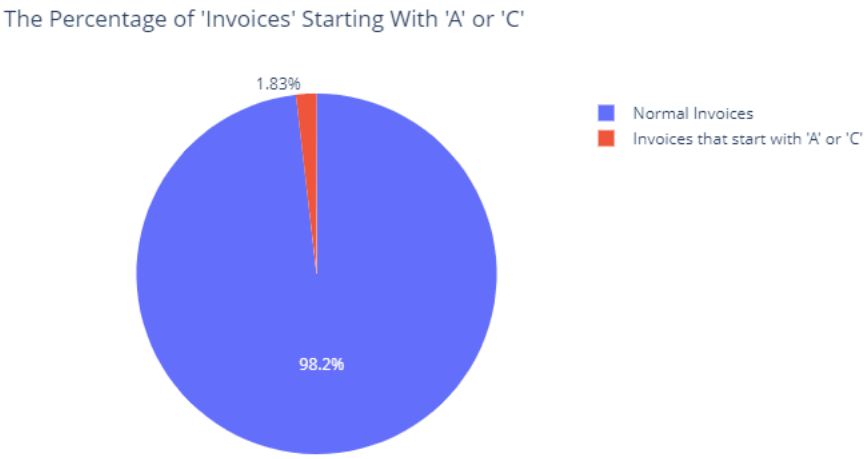


Figure 4.4: Cancelled/Abandoned Invoices vs Normal Invoices

Quantity

The *quantity* attribute indicates the number of times a certain product was bought during a particular transaction. This number should never be lower than 0 from a logical standpoint. However, if we look at Figure 4.5, we can see that the value of *quantity* is negative in some cases. It is clear to us that in each of these instances, either the order was canceled or the *Customer_ID* is not available. The proportion of rows in which the 'quantity' is less than zero is shown in Figure 4.6. As expected, the percentage of these instances is similar to that of cancelled orders.

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer_ID	Country
680158	C552047	22236	CAKE STAND 3 TIER MAGIC GARDEN	-1	2011-05-06 09:55:00	12.75	14796.0	United Kingdom
150065	503707	22173	NaN	-21	2010-04-06 14:04:00	0.00	NaN	United Kingdom
561568	C541257	22111	SCOTTIE DOG HOT WATER BOTTLE	-1	2011-01-16 14:48:00	4.95	18069.0	United Kingdom
279422	C516606	21320	GLASS CHALICE GREEN LARGE	-2	2010-07-21 14:30:00	2.55	14766.0	United Kingdom
200899	508660	35644	NaN	-6	2010-05-17 16:22:00	0.00	NaN	United Kingdom

Figure 4.5: Sample of data with negative values in "Quantity"

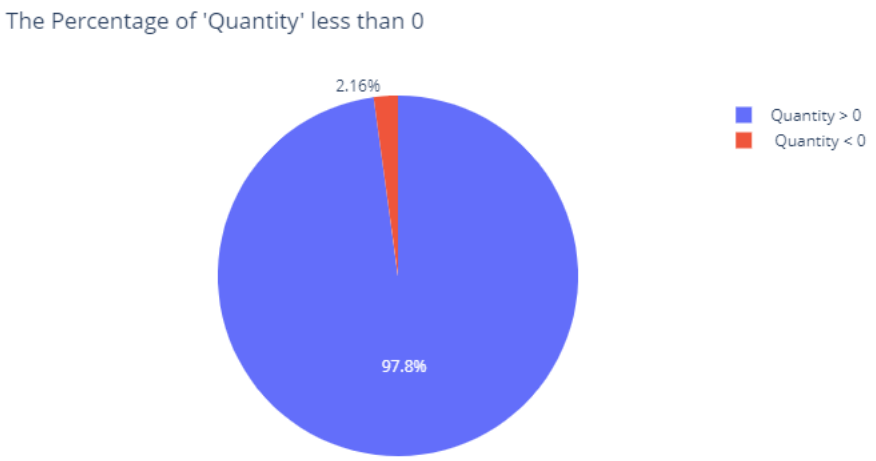


Figure 4.6: Percentage of negative values in "Quantity"

In our analysis of *Quantity* feature we came across rows where we have missing values in *Customer_ID*. In Figure 4.7 we can see the missing values for all of our features. What we observe is that *Customer_ID* has a considerable amount of missing entries. To be more specific we can refer to Table 4.3.

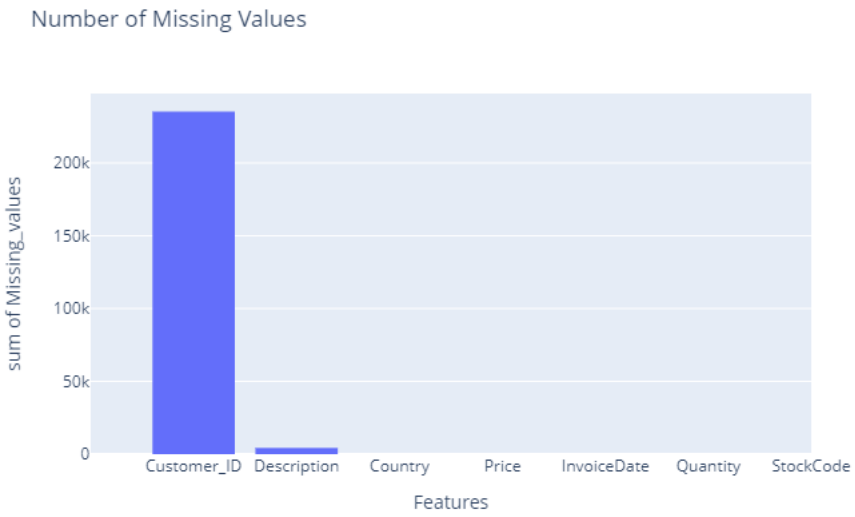


Figure 4.7: Missing values in each attribute

Table 4.3: Number of Missing values

Feature	Number of Missing values	Percentage
Customer_ID	235287	**22.52%**
Description	4275	0.4%

One last significant detail that we should discover about our data is the total number of unique clients that are exist as well as the distribution of these customers throughout the various countries. Similar to what we did for *Invoice* Figure 4.8 and Table 4.4 provide us with the aforementioned information.

Table 4.4: Unique Customers by Country

Country	Number of Unique Customers	Percentage
UK	5410	**91.05%**
Germany	107	1.80%
France	95	1.56%
	Total = 5942	

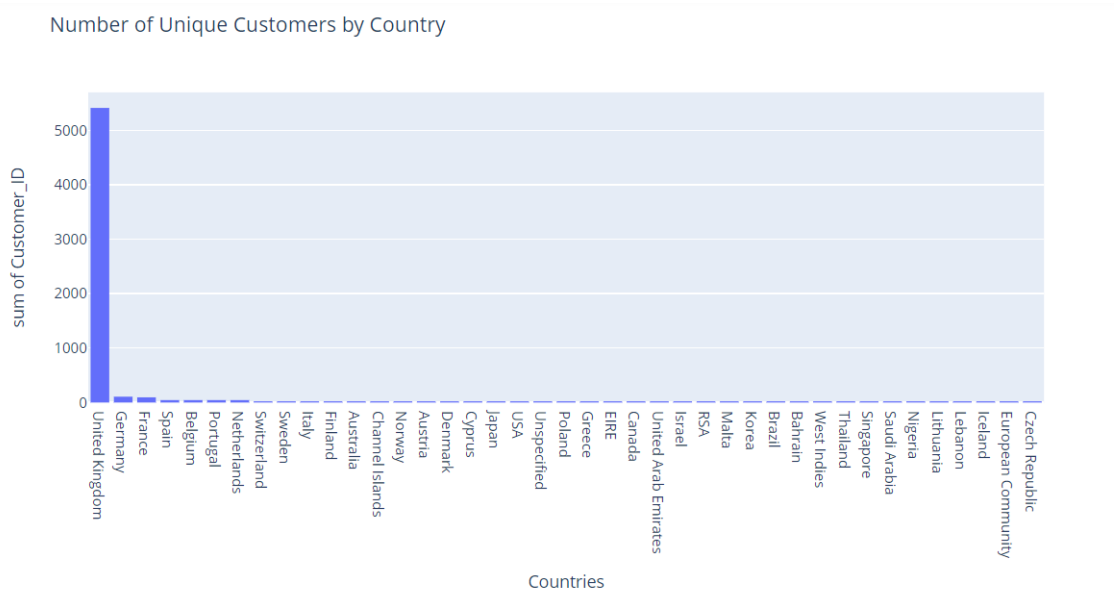


Figure 4.8: Unique Customer by Country

4.2.2 Data Cleaning

The procedure of "data cleaning" refers to the act of preparing our data for further analysis by removing information that is either incorrect or irrelevant [55]. To be more specific, this pertains to information that we feel will have a negative impact on the accuracy of our models. The exploratory study that we carried out in Section 4.2.1 served mostly as preparation for the following work.

The steps for cleaning our data are:

- **Removing rows with *"Invoices"* that begin with with 'A' or 'C'.**

  We saw before that these entries refer to only 1.83% of our data. At the same time we do not believe that abandoned or cancelled orders at such a small percentage add any value to our data. On the contrary they could create noise/anomalies that could affect the efficacy of our models, so we decide to remove them.

- **Removing rows with "Quantity" that are less than 0.**

  For the same reasons, since the percentage of rows that the *"Quantity"* value is less than 0 is very small, to be precise 2.16%, we remove these rows as well.

- **Removing rows where *"Customer_ID"* is not available.**

  Previously, we saw in Table 4.3 that almost one fourth of our data (22.52%) does not have a value for *"Customer_ID"*. We are unable to proceed with the data in its current

state since the issues that we will be addressing are related to Customers. Shifting these transactions to other consumers is one possibility. However, something like that might cause the conclusions we take from the data to be flawed, so we choose to exclude them.

- **Isolation of UK market.**

  In observed in Figures 4.3, 4.8 and Tables 4.1, 4.4 most of our invoices and customers are in the United Kingdom. Since more than $\mathbf{90}\%$ of both clients and transactions are in the UK, we believe that the impact of removing all other Counties is insignificant and focusing on the UK market will make our data more homogeneous and remove probable anomalies that could appear in transactions made in other countries.

In Table 4.5 we can see the results of our ***data cleaning*** procedure.

Table 4.5: Data after cleaning

	Before	**After**
N_rows	1.044.848	**711.928**
Customers	5942	**5353**

4.3 Feature Engineering

Feature engineering is the procedure of modifying raw data into features that better describe the underlying problem to our machine learning models, which ultimately results in enhanced performance [56]. To put it another way, the process of making our data more transparent. However, this is a challenging task because different problems call for a different approaches.

In our situation, feature engineering is of utmost significance. At this point, our dataset has $\mathbf{711.928}$ records, however there are only $\mathbf{5353}$ unique customers. Ultimately , we want to convert the data that we now have into a new dataframe, in which every entry will represent a different customer ($\mathbf{5353\ rows}$). We need to design new features for each customer by utilizing the data that is already accessible in order to get the outcome that we want.

4.3.1 RFM analysis

RFM analysis is a marketing strategy that is used to statistically rank and categorize clients based on the recency, frequency, and monetary total of their transaction records [57]. This is done in order to determine which customers are the most valuable and to carry out tailored marketing campaigns.

In order for us to acquire a better understanding of the RFM analysis and our new features, we need to examine what each of them represents and how to compute them.

- **Recency:** Recency simply refers to the number of days that have gone since the last purchase of a client. To compute Recency we have to subtract the date of the last order from the last day of our data, which we use as a reference date for all customers.

$$\mathbf{Recency} = \mathbf{Latest\_Date} - \mathbf{Last\_Purchase\_Date} \tag{4.1}$$

- **Frequency:** Frequency is the number of orders a customer has completed in their time engaged with a business.

$$\mathbf{Frequency} = \mathbf{Number\_Of\_Orders} \tag{4.2}$$

- **Monetary:** Monetary reflects to the total amount each client has spent for all their transactions.

$$\mathbf{Monetary} = \mathbf{Total\_Money\_Spent} \tag{4.3}$$

After creating these new features and transforming our data so that each row represents each client, our data has the form depicted in Figure 4.9.

	Customer_ID	Recency	Frequency	Monetary
3150	16044.0	435	19	340.84
1868	14731.0	47	45	380.02
4591	17514.0	128	87	542.83
3524	16425.0	302	21	302.70
1054	13900.0	183	95	1610.59

Figure 4.9: Newly formed dataframe

4.3.2 Additional Features

On top of our RFM features and with the help of them, we will create some additional features that will believe will enhance the performance of our models. These new attributes are:

- **Time_engaged:** This feature provides information on the number of days that a client has been involved with the business. In the context of *recency*, we were interested in finding out how much time had passed since the client's most recent order. In the same vein, we count the number of days that have gone since the first order.

$$\mathbf{Time\_engaged} = \mathbf{Latest\_Date} - \mathbf{First\_Purchase\_Date} \tag{4.4}$$

- **Active_period:** Another *"time"* characteristic that will be included in our final dataframe is the amount of time that has passed between the first order placed by each customer until their most recent order.

$$\mathbf{Active\_period} = \mathbf{Last\_Purchase\_Date} - \mathbf{First\_Purchase\_Date} \tag{4.5}$$

- **Time_between:** Here, we develop a new feature that, gives us information on the average amount of time that has elapsed between consecutive orders for each individual client.

$$\mathbf{Time\_between} = \frac{\mathbf{Active\_period}}{\mathbf{Frequency}} \tag{4.6}$$

- **Average_transaction_value:** In addition to the characteristics that were just stated that are focused on time, we are going to include a feature that displays the mean amount of money that each individual client has spent in each transaction.

$$\mathbf{Average\_transaction\_value} = \frac{\mathbf{Monetary}}{\mathbf{Frequency}} \tag{4.7}$$

Chapter 5

Approach A: Experiments and Results

5.1 Individual Approach

First, we will make an effort to estimate CLV for each one of our unique customers by making use of machine learning approaches. Due to the fact that the majority of study focuses on customer segmentation and aggregate CLV, this is a notion that is not commonly employed in the associated work that has been undertaken. Nevertheless, we think that this kind of strategy might be useful, particularly for smaller organizations who have a limited number of clients and are eager to calculate the worth of each one of those clients individually. That way businesses are able to design tailor-made marketing campaigns to raise their CLV in the future.

Ideally, in order to implement such an approach, we would prefer our data to have a target column that contains the CLV generated by the business. On this basis, we would be able to easily carry out experiments using the proposed regression models discussed in Chapter 3. Unfortunately, such a column is not provided in our current data. In order to proceed, we will begin by generating a pseudo-target column that will represent CLV.

Although this target variable is custom-made, we want it to be somewhat realistic but at the same time to contain the element of randomness so that our models will not easily recognize a pattern. The CLV will be calculated with the following equation for each customer:

$$\mathbf{CLV} = \mathbf{Gross\_Profit} - \mathbf{Penalty} * \mathbf{Months\_of\_Inactivity} \tag{5.1}$$

where

$$Gross\_Profit = random\{35\% - 65\%\} * Revenue \tag{5.2}$$

,

$$Penalty = random\{1\% - 5\%\} * Revenue \tag{5.3}$$

and

$$Months\_of\_Inactivity = \frac{Average\_Time\_Between\_Consecutive\_Orders}{30} \tag{5.4}$$

With that said, our dataset before moving to our experiments has the form depicted in Figure 5.1.

	Customer_ID	Recency	Frequency	Monetary	time_engaged	time_between	active_period	avg_basket_value	CLV
3552	16453.0	530	2	251.71	548	274.0	17	125.855	112.598273
2253	15134.0	14	10	1596.29	422	42.2	407	159.629	960.753741
2782	15670.0	296	2	325.20	556	278.0	259	162.600	144.605600
1738	14600.0	305	2	865.17	393	196.5	87	432.585	431.719830
1304	14155.0	266	1	118.75	266	266.0	0	118.750	53.754167

Figure 5.1: Dataframe with CLV target column

5.2 HyperParameter Tuning

5.2.1 Data Separation

Before continuing any further with the analysis, we first need to divide the original data into three distinct subsets, which will be referred to as the training, validation and test sets respectively. In order to do that we utilize the scikit learn library. To illustrate this process we can refer to Figure 5.2.

The test set is usually refereed to as a hold-out set, since we use it at the end in order to see how our model performs on unseen data. The rest of the data is split into training and validation sets. The reason the validation set exists is to evaluate the performance of our model during training before using it on the test set.

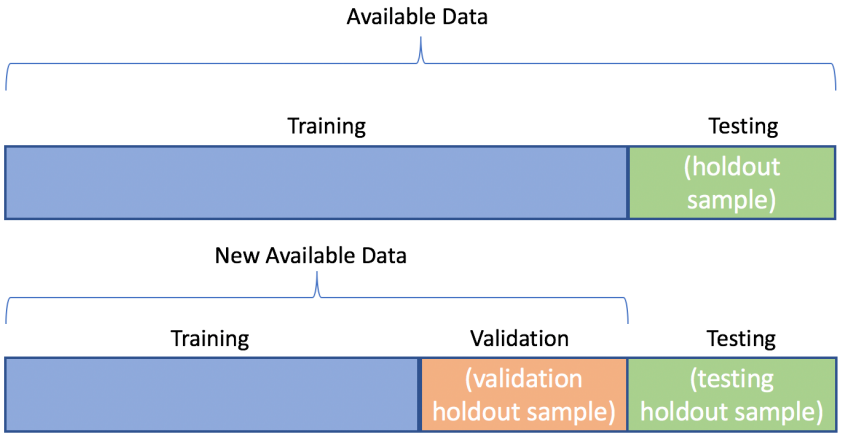


Figure 5.2: Train-Validation-Test split

Source : https://datascience.stackexchange.com/questions/61467/clarification-on-train-test-and-val-and-how-to-use-implement-it

However, the performance metrics of the models are extremely dependant exclusively on these two chunks of data. They are only trained and assessed once, hence their performance is dependant on a single assessment. This procedure frequently results in overfitting, which means that we place and excessive amount of emphasis on training our models so that they perform well on the validation set but perform badly when applied on unseen data (hold-out set). A good solution could be to train and evaluate on several subsets of the same data and look at the average performance.

5.2.2 Cross-Validation

The solution to the aforementioned issue is **Cross-Validation**. The dataset is divided at random into sets for the purposes of cross-validation. One of these sets is designated as the test set an the remaining sets are used to train the model. After doing these steps several times with each of the sets serving as the test set, the final model is constricted by taking the average of all the models.

One of the most frequently methods of cross-validation is **k-fold cross validation**, where k indicates the number of folds in the dataset.

We begin the k-fold cross-validation process by isolation a test/hold-out set from the dataset. This test/hold-out set will be used for the final assessment of our models after they are complete. The data remaining, which includes everything besides the hold-out set, is divided into the specified number of folds(subsets), which is denoted by k. After that, the

cross-validation iterates through the folds, and at each iteration, one of the k-folds is used as the validation set, whereas the remaining are used as the training set. This technique is performed as many times as necessary until each fold has served as a validation set. In Figure 5.3 we can see the procedure carried out for a 5-cross-validation.

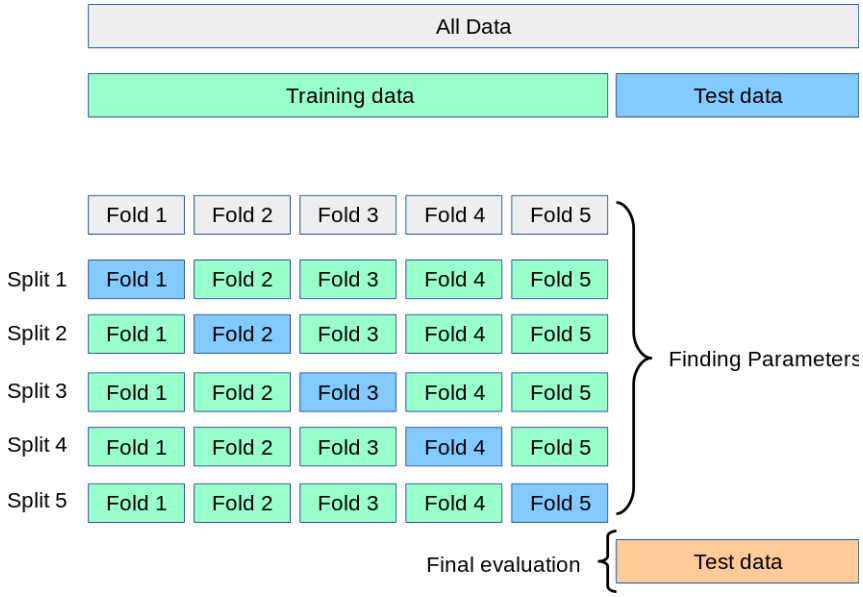


Figure 5.3: 5-fold Cross-validation

Source : https://scikit-learn.org/stable/modules/cross_validation.html

5.2.3 Grid-Search

The grid-search technique is the most straightforward for adjusting hyperparameters. In essence, we create a discrete grid by partitioning the space occupied by combination of these grid's values a chance while simultaneously utilizing cross-validation to compute various performance indicators. The ideal combinations of values for the hyperparameters may be found at the place on the grid that maximizes the average value obtained from the cross-validation process. Grid-search is an extensive technique that covers all of the possible combination points, which enables it to locate the most advantageous location inside the domain. However, the fact that it moves at such a snail's pace is a significant drawback. Checking each and every possible combination of the are involves a significant amount of time, which is not always available. At the same time, since we evaluate our points using k-fold cross-validation, which takes k training steps, the time needed is even more. Grid search, on the other hand, is an excellent concept to consider if the goal is to find the optimal combination of values for the hyperparameters.

In order to perform Grid-Search we utilize the scikit-learn library; in particular the *GridSearchCV()* function [58], that takes as parameters the model, the parameter grid (list of values to explore for the hyperparameters), the score/loss function, but also the cross-validation strategy which we previously define using the *KFold()* function [59] of the scikit-learn library.

5.3 Experiments and Results

Before starting to experiment with our machine learning models analyzed extensively in Chapter 3, we first have to split our data into training and test sets. To do that we utilize the *train_test_split()* function [60] of the scikit-learn library. We split our 5353 customers into 4500 (training) and 853 (test) which is equal to 16% of the total customers. In order to evaluate our models we need a metric that is ideal for our problem. Since we want to predict CLV, which basically corresponds to an amount of money, we will use **MAE**.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{5.5}$$

where y_i the prediction value and x_i the true value. MAE here indicates the amount of money $\pm\, x$ we are off in our prediction.

5.3.1 Linear Regression

We implement Linear Regression utilizing the scikit-learn library and more specifically the *Linear_Regression()* function [61]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. The average MAE of the cross-validation is 69.84. After evaluating on the hold-out set the MAE is 78.85. Table 5.1 shows the results.

Table 5.1: Linear Regression Results

Linear_Regression	**cross-validation**	**hold-out set**
MAE	69.84	78.85

5.3.2 Ridge Regression

We implement Ridge Regression utilizing the scikit-learn library and more specifically the *Ridge()* function [62]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameter *alpha*, we grid-search multiple values from 0 to 1000. The average MAE of the cross-validation is 71.68 and the best *alpha* is 100. After evaluating on the hold-out set the MAE is 72.94. Table 5.2 shows the results.

Table 5.2: Ridge Regression Results

Ridge_Regression	cross-validation	hold-out set
MAE	71.68	72.94

5.3.3 Lasso Regression

We implement Lasso Regression utilizing the scikit-learn library and more specifically the *Lasso()* function [63]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameter *alpha*, we grid-search multiple values from 0 to 1000. The average MAE of the cross-validation is 71.68 and the best *alpha* is 100. After evaluating on the hold-out set the MAE is 72.94. Table 5.3 shows the results.

Table 5.3: Lasso Regression Results

Lasso_Regression	cross-validation	hold-out set
MAE	70.08	67.69

5.3.4 Decision Tree Regression

We implement Decision Tree Regression utilizing the scikit-learn library and more specifically the *DecisionTreeRegressor()* function [64]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameters *max_depth* and *min_samples_leaf*, we grid-search multiple values from 0 to 100 and from 0 to 20 respectively. The average MAE of the cross-validation is 89.39, the best *max_depth* is 30 and the best *min_samples_leaf* is 3. After evaluating on the hold-out set the MAE is 82.78. Table 5.4 shows the results.

Table 5.4: Decision Tree Regression Results

Decision_Tree_Regression	cross-validation	hold-out set
MAE	89.39	82.78

5.3.5 Random Forest Regression

We implement Random Forest Regression utilizing the scikit-learn library and more specifically the *RandomForestRegressor()* function [65]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameter *n_estimators*, we grid-search multiple values from 0 to 100. The average MAE of the cross-validation is 85.14, the best *n_estimators* is 35. After evaluating on the hold-out set the MAE is 47.17. Table 5.5 shows the results.

Table 5.5: Random Forest Regression Results

Random_Forest_Regression	cross-validation	hold-out set
MAE	85.14	47.17

5.3.6 XGBoost Regression

We implement XGBoost Regression utilizing the scikit-learn interface of the XGBoost Python Package and more specifically the *XGBRegressor()* function [66]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameter *n_estimators*, we grid-search multiple values from 0 to 100. The average MAE of the cross-validation is 143.39, the best *n_estimators* is 45. After evaluating on the hold-out set the MAE is 80.81. Table 5.6 shows the results.

Table 5.6: XGBoost Regression Results

XGBoost_Regression	cross-validation	hold-out set
MAE	143.39	80.81

5.3.7 Overall

To summarize the above we can refer to Table 5.7

Table 5.7: Overall Regression Results

	MAE	
Models	**cross-validation**	**hold-out set**
Linear_Regression	69.84	78.85
Ridge_Regression	71.68	72.94
Lasso_Regression	70.08	67.69
Decision_Tree_Regression	89.39	82.78
Random_Forest_Regression	85.14	47.17
XGBoost_Regression	143.39	80.81

What we observe from our experiments is that from the linear models, Lasso_Regression has the lowest MAE with 67.69 on average. From all the models Random_Forest performs the best, predicting the CLV of customers with an average MAE under 50.

Chapter 6

Approach B: Experiments and Results

6.1 Customer Segmentation Approach

We previously discussed the individual approach, where we dealt with a regression problem in an effort to predict the CLV for each one of our customers. We mentioned that this strategy is ideal for small businesses that do not have a large number of customer and want to approach each customer in a unique way. However, most companies with bigger clientele cannot follow this strategy. Therefore, they have to perform Customer Segmentation in order to group these customers based on some features. In order to create these clusters we will use an unsupervised machine learning technique called K-means , which we discussed earlier in Chapter 3. After creating these clusters we attempt to go one step further in our analysis and use these clusters as labels in our data and perform classification experiments using, at this stage, supervised machine learning models in an effort to classify customers in the correct clusters.

6.2 Clustering

The amount of different clusters that our consumers will be placed in will need to be decided upon before we can go on to the k-means clustering method. We employ the **Elbow Method**, which is a method for identifying the ideal number of clusters in k-means clustering, in order to do this. In Figure 6.1, we can see a graph in which the x-axis shows the number of clusters and the y-axis represents the inertia, which is a measurement of how successfully a dataset was clustered using k-means. It is calculated as the sum of the squared distances

between each sample and their centroids, like we mentioned in Equation 3.20. A model that has a low inertia and a small number of clusters is considered to be ideal. On the other hand, this is a trade-off, because as k increases, inertia decreases. We utilize the elbow method to get the best value for k, which implies that we search for the value of k at which the rate of inertia reduction begins to slow down (**elbow point**).

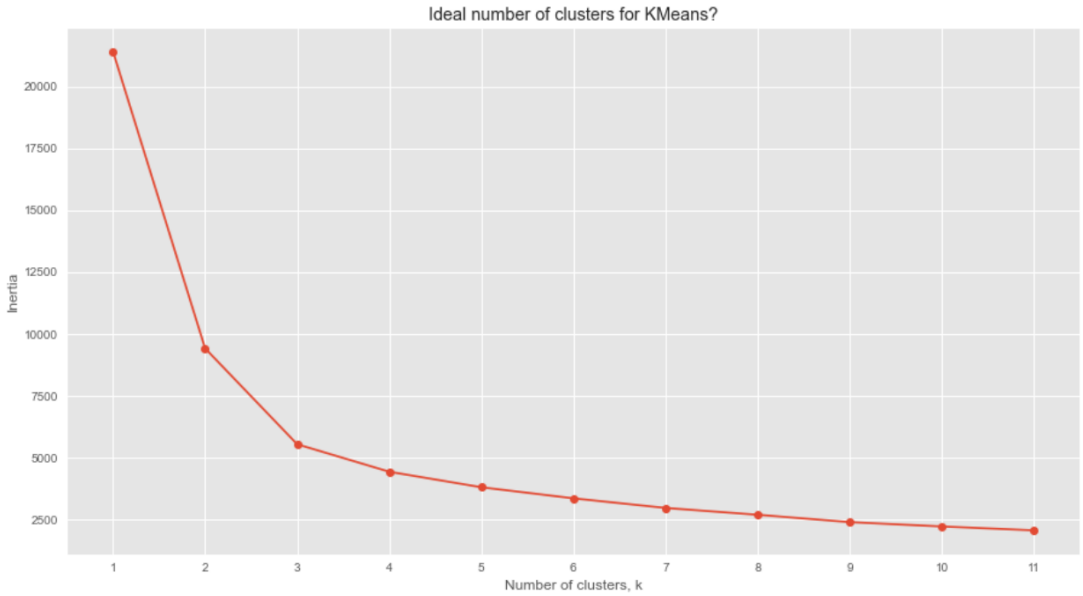


Figure 6.1: Elbow Method

In the above graph we can see that inertia decreases a lot until $\mathbf{k = 3}$ where it begins to slowly drop as we increase the number of clusters. For that reason we choose to continue our analysis with 3 clusters.

We implement K-means clustering utilizing the scikit-learn library and more specifically the *KMeans()* function [67]. After this process we have the following clusters:

- **Cluster 0:** $\frac{1420}{5353} = \mathbf{26.53}\%$ of total customers.
- **Cluster 1:** $\frac{1875}{5353} = \mathbf{35.03}\%$ of total customers.
- **Cluster 2:** $\frac{2058}{5353} = \mathbf{38.44}\%$ of total customers.

The aforementioned results can also be visualized in Figure 6.2.

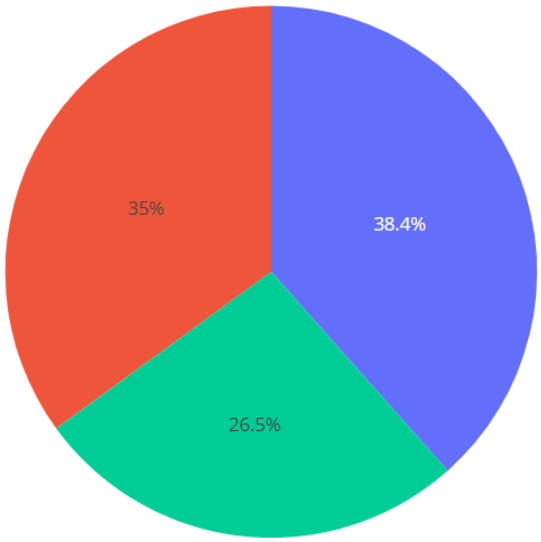


Figure 6.2: Clusters

We create a new feature in our data to store the cluster to each customer and we name it *Cluster_ID*. We can see the new dataframe in Figure 6.3.

	Recency	Frequency	Monetary	time_engaged	time_between	active_period	avg_basket_value	ClusterID
Customer_ID								
12346.0	325	12	77556.46	725	60.416667	400	6463.038333	0
12608.0	404	1	415.79	404	404.000000	0	415.790000	1
12745.0	486	2	723.85	574	287.000000	87	361.925000	1
12746.0	540	1	254.55	540	540.000000	0	254.550000	1
12747.0	2	26	8917.98	732	28.153846	730	342.999231	0
...	...	...	...	...	...	...	...	...
18283.0	3	22	2736.65	658	29.909091	654	124.393182	0
18284.0	431	1	461.68	431	431.000000	0	461.680000	1
18285.0	660	1	427.00	660	660.000000	0	427.000000	1
18286.0	476	2	1296.43	723	361.500000	247	648.215000	1
18287.0	42	7	4182.99	571	81.571429	528	597.570000	0

Figure 6.3: Dataframe with Clusters

The next step is to understand what these clusters mean. In particular, we would like to see which cluster represents which kind of customer. To do that we look at the RFM metrics of the Clusters in Figure 6.4.

What we observe is the following:

- **Cluster 0:** Low Recency, High Frequency, High Monetary
- **Cluster 1:** High Recency, Low Frequency, Low Monetary

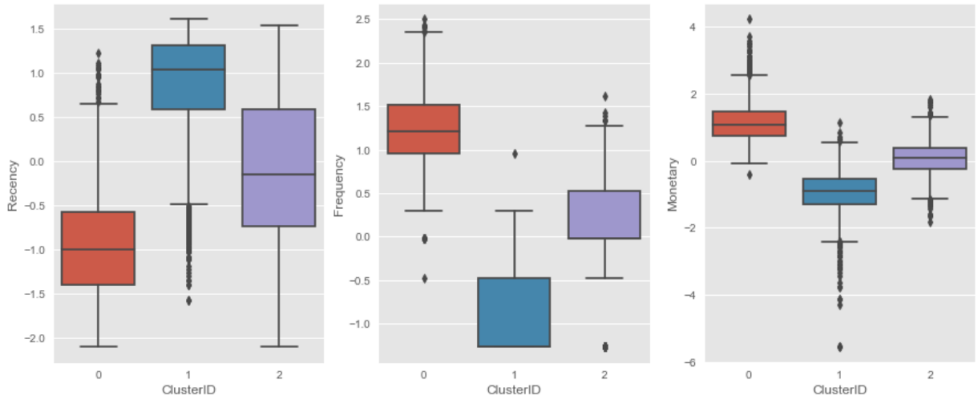


Figure 6.4: RFM of Clusters

- **Cluster 2:** Average Recency, Average Frequency, Average Monetary.

which means that **Cluster 0** represents the **Best** Customers, **Cluster 1** represents the **Bad** Customers and **Cluster 2** represents the **Average** Customers.

6.3 Classification Performance Metrics

6.3.1 Confusion Matrix

Usually, the performance of classification models is evaluated using a Confusion Matrix. In the simple case of two classes (Positive, Negative) the confusion matrix is like in Table 6.1.

Table 6.1: Confusion Matrix

		Predicted	
		Positive	Negative
Actual	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

where

- **True Positive (TP):** the observations predicted as positive and were indeed positive.
- **True Negative (TN):** the observations predicted as negative and were indeed negative.

- **False Positive (FP):** the observations predicted as positive and were actually negative.
- **False Negative (FN):** the observations predicted as negative and were actually positive.

The table can be normalized such that it contains percentages in addition to the number of observations, as seen above. In light of this, TP and TN will be stated as the percentages of the correct categories, and FP and FN will be expressed as the percentages of the wrong classifications. The following metrics of evaluation are able to be derived from the data included in the Confusion Matrix:

- **Accuracy:** It is the most common metric when evaluating a machine learning model and it describes the percentage of correct predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

- **Precision:** Describes the probability of the correct classification of a positive observation. Specifically, it is the ratio of observations correctly classified in label "positive" to all observations classified in that label.

$$Precision = \frac{TP}{TP + FP} \tag{6.2}$$

- **Recall:** It describes the percentage of accurate predictions for our positive observations. In particular, it is the ratio of observations correctly classified in label "positive" to all observations that belong in that label.

$$Recall = \frac{TP}{TP + FN} \tag{6.3}$$

- **F1-Score:** It is the harmonic mean between recall and precision.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.4}$$

6.3.2 Multi-Class Confusion Matrix

In the event that the data may be classified into more than two classes, the value in each cell (i, j) if the Confusion Matrix (Figure 6.5) represents the number of observations from class i that were classified in class j. The number of successfully classified observations

		Predicted Class					
		C_1	C_2	...	C_i	...	C_n
Actual Class	C_1	P_{11}	P_{12}	...	P_{1i}	...	P_{1n}
	C_2	P_{21}	P_{22}	...	P_{2i}	...	P_{2n}
	...	...	...	...	...	...	...
	C_i	P_{i1}	P_{i2}	...	P_{ii}	...	P_{in}
	...	...	...	...	...	...	...
	C_n	P_{n1}	P_{n2}	...	...	...	P_{nn}

Figure 6.5: Multi-class Confusion Matrix

is represented in the table by the diagonal elements, where $(i = j)$, whereas the number of observations that were not correctly classified is represented by the reminder elements, where $(i \neq j)$.

Calculation the evaluation metrics for a multi-class classification problem can be done for each class individually, but also as an average that reflects the performance of the model, using the micro and macro formulas [68]. When computing macro formulas all the classes are considered equal, whereas in micro formulas more weight is given to classes with the most elements. Therefore, in cases where the samples are imbalanced we prefer micro formulas. The following are the types of evaluation metrics we mentioned above.

- **Average Accuracy:**

$$Average \quad Accuracy = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i} \tag{6.5}$$

- **Precision:**

$$Macro\_Precision = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i} \tag{6.6}$$

$$Micro\_Precision = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FP_i)} \tag{6.7}$$

- **Recall:**

$$Macro\_Recall = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i} \tag{6.8}$$

$$Micro\_Recall = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N}(TP_i + FN_i)} \tag{6.9}$$

- **F1-score:**

$$F1 - score(macro\ or\ micro) = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.10}$$

6.4 Classification Experiments and Results

In this section we will evaluate various classification models, similar to what we did in Chapter 5. Similarly, we split our 5353 customers into 4500 (training) and 853 (test). Then we apply a 5-fold cross-validation with Folds of 900 values in our models and grid-search to find the best hyperparameters that minimize the accuracy of our classification models. For each of our models we will measure the accuracy based on the confusion matrix, but at the same time we will explore in detail the misclassifications. The reason for that is to see if our models are good in general and in the cases they fail if they **overestimate** or **underestimate** the customer.

1. **Overestimation:**
    - Average Customers classified as Best
    - Bad Customers classified as Average or Best
2. **Underestimation:**
    - Average Customers classified as Bad
    - Best Customers classified as Average or Bad

6.4.1 Decision Tree Classification

We implement Decision Tree Classification utilizing the scikit-learn library and more specifically the *DecisionTreeClassifier()* function [69]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameters *max_depth* and *min_samples_leaf*, we grid-search multiple values from 0 to 100 and from 0 to 20 respectively. The average Accuracy of the cross-validation is 95.71%, the best *max_depth* is 40 and the best *min_samples_leaf* is 3. After evaluating on the hold-out set the Accuracy is $95.67\% = \frac{816}{853}$ correct classifications. Figure 6.6 and Tables 6.2, 6.3 show the results.

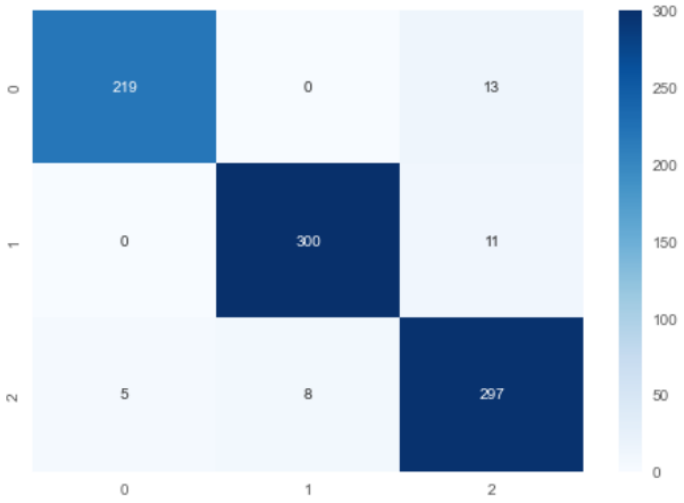


Figure 6.6: Decision Tree Classification

Table 6.2: Decision Tree Classification Results

Decision_Tree_Classification	cross-validation	hold-out set
Accuracy	95.71%	95.67%

Table 6.3: Decision Tree Misclassified Customers

Decision_Tree_Classification	misclassifications	overestimations	underestimations
Number of Customers	37	16	21

- Out of the 16 overestimations 11 bad customers were classified as average customers and 5 average customers were classified as best customers.

- Out of the 21 underestimations 13 best customers were classified as average customers and 8 average customers were classified as bad customers.

6.4.2 Random Forest Classification

We implement Random Forest Classification utilizing the scikit-learn library and more specifically the *RandomForestClassifier()* function [70]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameter *n_estimators*, we grid-search multiple values from 0 to 100. The average Accuracy of the cross-validation is 97.2%, the best *n_estimators* is 45. After evaluating on the hold-out set the Accuracy is $97.3\% = \frac{830}{853}$ correct classifications. Figure 6.7 and Tables 6.4, 6.5 show the results.

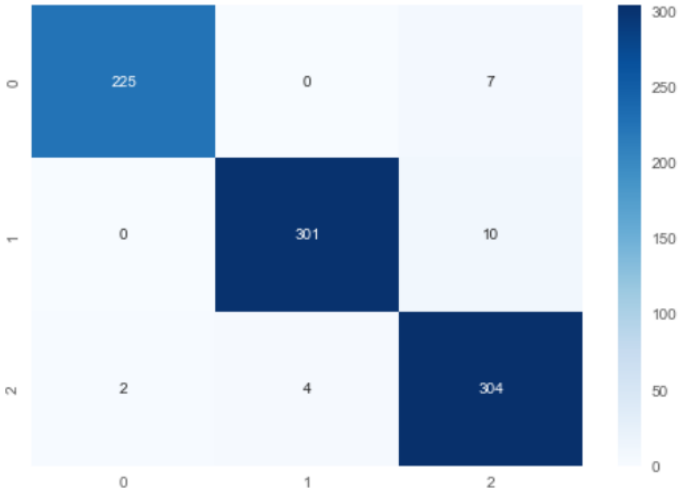


Figure 6.7: Random Forest Classification

Table 6.4: Random Forest Classification Results

Random_Forest_Classification	cross-validation	hold-out set
Accuracy	97.2%	97.3%

Table 6.5: Random Forest Misclassified Customers

Random_Forest_Classification	misclassifications	overestimations	underestimations
Number of Customers	23	12	11

- Out of the 12 overestimations 10 bad customers were classified as average customers and 2 average customers were classified as best customers.

- Out of the 11 underestimations 7 best customers were classified as average customers and 4 average customers were classified as bad customers.

6.4.3 K-Nearest Neighbours Classification

We implement K-Nearest Neighbours Classification utilizing the scikit-learn library and more specifically the *KNeighborsClassifier()* function [71]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameter *n_neighbors*, we grid-search multiple values from 1 to 45. The average Accuracy of the cross-validation is 95%, the best *n_neighbors* is 5. After evaluating on the hold-out set the Accuracy is $96\% = \frac{819}{853}$ correct classifications. Figure 6.8 and Tables 6.6, 6.7 show the results.

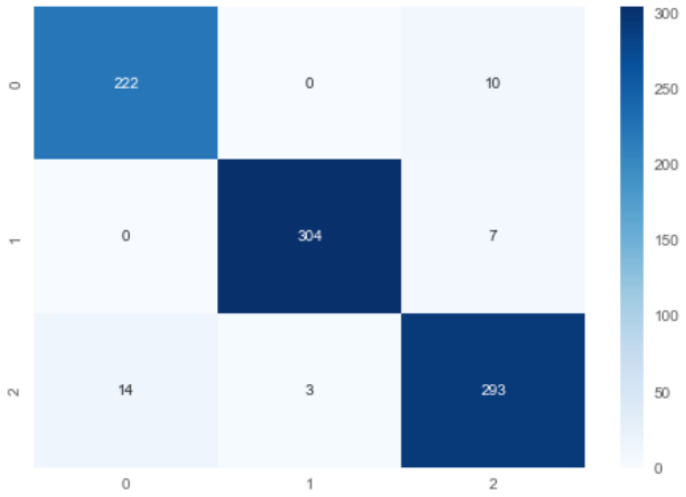


Figure 6.8: K-nearest Neighbours Classification

Table 6.6: K-Nearest Neighbours Classification Results

KNN_Classification	cross-validation	hold-out set
Accuracy	95%	96%

Table 6.7: K-Nearest Neighbours Misclassified Customers

KNN_Classification	misclassifications	overestimations	underestimations
Number of Customers	34	21	13

- Out of the 21 overestimations 7 bad customers were classified as average customers and 14 average customers were classified as best customers.

- Out of the 13 underestimations 10 best customers were classified as average customers and 3 average customers were classified as bad customers.

6.4.4 XGBoost Classification

We implement XGBoost Regression utilizing the scikit-learn interface of the XGBoost Python Package and more specifically the *XGBClassifier()* function [66]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameter *n_estimators*, we grid-search multiple values from 0 to 100. The average Accuracy of the cross-validation is 97.64%, the best *n_estimators* is 45. After evaluating on the hold-out set the Accuracy is $97.54\% = \frac{832}{853}$ correct classifications. Figure 6.9 and Tables 6.8, 6.9 show the results.

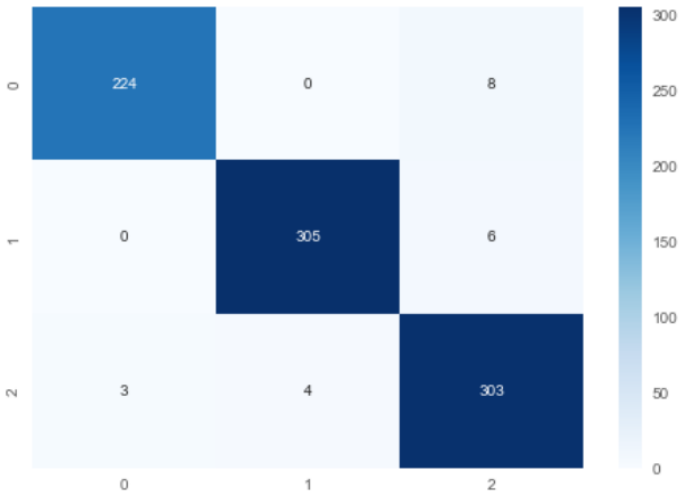


Figure 6.9: XGBoost Classification

Table 6.8: XGBoost Classification Results

XGBoost_Classification	cross-validation	hold-out set
Accuracy	97.64%	97.54%

Table 6.9: XGBoost Misclassified Customers

XGBoost_Classification	misclassifications	overestimations	underestimations
Number of Customers	21	9	12

- Out of the 9 overestimations 6 bad customers were classified as average customers and 3 average customers were classified as best customers.
- Out of the 12 underestimations 8 best customers were classified as average customers and 4 average customers were classified as bad customers.

6.4.5 AdaBoost Classification

We implement AdaBoost Regression utilizing the scikit-learn interface of the XGBoost Python Package and more specifically the *AdaBoostClassifier()* function [72]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameter *n_estimators*, we grid-search multiple values from 0 to 100. The average Accuracy of the cross-validation is 87.96%, the best *n_estimators* is 45. After evaluating on the hold-out set the Accuracy is $92.61\% = \frac{790}{853}$ correct classifications. Figure 6.10 and Tables 6.10, 6.11 show the results.

Table 6.10: AdaBoost Classification Results

AdaBoost_Classification	cross-validation	hold-out set
Accuracy	87.96%	92.61%

Table 6.11: AdaBoost Misclassified Customers

AdaBoost_Classification	misclassifications	overestimations	underestimations
Number of Customers	63	19	44

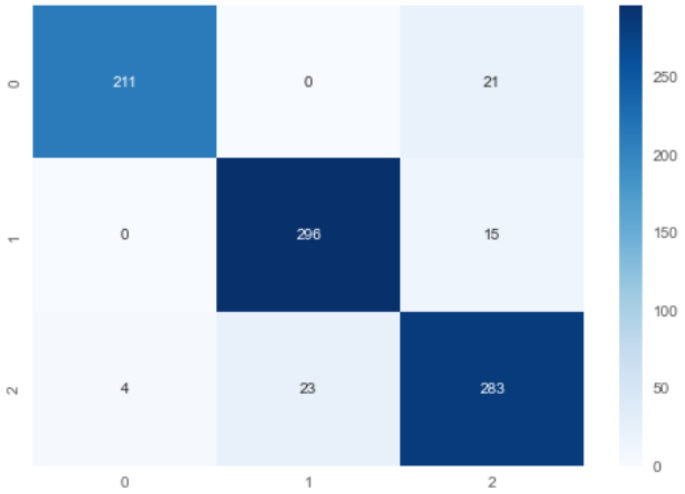


Figure 6.10: AdaBoost Classification

- Out of the 19 overestimations 15 bad customers were classified as average customers and 4 average customers were classified as best customers.

- Out of the 44 underestimations 21 best customers were classified as average customers and 23 average customers were classified as bad customers.

6.4.6 SVM Classification

We implement AdaBoost Regression utilizing the scikit-learn interface of the XGBoost Python Package and more specifically the *SVC()* function [73]. Furthermore, we apply a 5-fold cross-validation with Folds of 900 values. To find the optimal hyperparameters C and *kernel*, we grid-search multiple values from 1 to 10000 for C and the following kernels: {'linear', 'poly', 'rbf', 'sigmoid'}. The average Accuracy of the cross-validation is 87.43%, the best C is 10 and the best *kernel* is rbf. After evaluating on the hold-out set the Accuracy is $92.85\% = \frac{792}{853}$ correct classifications. Figure 6.11 and Tables 6.12, 6.13 show the results.

Table 6.12: SVM Classification Results

SVM_Classification	cross-validation	hold-out set
Accuracy	87.43%	92.85%

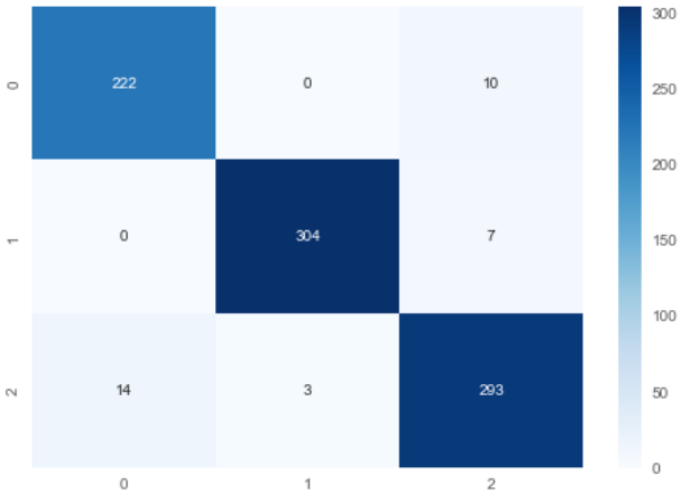


Figure 6.11: SVM Classification

Table 6.13: SVM Misclassified Customers

SVM_Classification	misclassifications	overestimations	underestimations
Number of Customers	61	11	50

- Out of the 11 overestimations 5 bad customers were classified as average customers and 6 average customers were classified as best customers.

- Out of the 50 underestimations 24 best customers were classified as average customers and 26 average customers were classified as bad customers.

6.4.7 Overall

To summarize the above we can refer to Tables 6.14 and 6.15

Table 6.14: Overall Classification Results

	Accuracy	
Models	**cross-validation**	**hold-out set**
Decision_Tree_Classification	95.71%	95.67%
Random_Forest_Classification	97.2%	97.3%
KNN_Classification	95%	96%
XGBoost_Classification	97.64%	97.54%
AdaBoost_Classification	87.96%	92.61%
SVM_Classification	87.43%	92.85%

Table 6.15: Overall Misclassification Results

Models	**misclassifications**	**overestimations**	**underestimations**
Decision_Tree_Classification	37	16	21
Random_Forest_Classification	23	12	11
KNN_Classification	34	21	13
XGBoost_Classification	21	9	12
AdaBoost_Classification	63	19	44
SVM_Classification	61	11	50

From the above Tables we observe the following:

- All of our models have high accuracy, which indicates that the clustering conducted in the previous step was quite successful and the classes are easily distinguished.
- The best performing models are **XGBoost Classifier** and **Random Forest Classifier** with the former having slightly better accuracy with **2** less misclassifications. **XGBoost Classifier** has the least overestimations, which means that it is the best model when it comes to **not** giving more value than needed to customers. On the other hand, **Random Forest Classifier** has the least underestimations, which means that it is the best model when it comes to **not** undervaluing the customers.
- The worst performing models are **AdaBoost Classifier** and **SVM Classifier** with the former having slightly worse accuracy with **2** more misclassifications. **SVM Classifier**

has the most underestimations, which means that it is the worst model when it comes to **not** undervaluing the customers. **KNN Classifier**, although it has an average overall performance shows a weakness when it comes to overestimations, making it the worst model when it comes to **not** giving more value than needed to customers.

Chapter 7

Conclusion

In this section, we will provide a summary of our work by providing the milestones, the findings, and the contribution to the thesis. In conclusion, we discuss the ways in which we anticipate expanding upon this work in the future.

7.1 Summary and Conclusions

In this thesis, we begin with an introduction to the theoretical background of machine learning techniques as well as fundamental concepts regarding Customer Relationship Management (CRM), such as Customer Lifetime Value (CLV) and Customer Segmentation.

In Chapter 3, we continue by digging into all of the methods used in our research. These machine learning methods include regression methods used for our individual approach and clustering/classification methods used for our collective approach. We later focus on analyzing our data, which we obtained from a public dataset and they represent transactions of an online retail store. This analysis is all about preparing our data for our experiments in a way that maximizes the performance of our models. This is accomplished through two key procedures: data pre-processing and feature engineering.

The experiments for our individual and collective methods are presented in Chapters 5 and 6, respectively. In our individual approach, we start by framing the problem and creating a target variable (CLV) for our models to predict. The accuracy of our models is measured using the Mean Absolute Error as a performance metric. The results show that from the Linear regression models we tested, Lasso was the the most efficient, whilst the Random Forest Regressor appears to be the most efficient overall. In the other approach, we start by utilizing the

Kmeans clustering methods, which resulted in the formation of 3 clusters that correspond to 3 types of customers: bad, average and best. We then utilize these clusters as labels and experiment with our classification methods in an attempt to evaluate their performance. We observe a high accuracy from all of our models which is an indicator that the clustering conducted in the previous step was quite successful and the classes are easily distinguished. What stands out is the outstanding performance of the XGBoost and Random Forest Classifiers with only a few misclassifications.

To contribute to this study further in the future, we may experiment with new datasets that are relevant to our problem scenarios, as well as enhance our experiment tools by leveraging deep learning and neural networks.

Bibliography

[1] Huan-Ming Chuang and Chia-Cheng Shen. A study on the applications of data mining techniques to enhance customer lifetime value — based on the department store industry. *2008 International Conference on Machine Learning and Cybernetics*, 1:168–173, 2008.

[2] V. Ravi. *Advances in Banking Technology and Management: Impacts of ICT and CRM*. Premier Reference Source Series. Information Science Reference, 2008.

[3] Sunil Gupta and Donald R. Lehmann. Customers as assets. *Journal of Interactive Marketing*, 17(1):9–24, 2003.

[4] Francis J. Mulhern. Customer profitability analysis: Measurement, concentration, and research directions. *Journal of Interactive Marketing*, 13(1):25–40, 1999.

[5] J. Brownlee. Basic concepts in machine learning. *Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/basicconcepts-in-machine-learning/*, December 2015.

[6] Jiachen Chen and W. Kenneth Jenkins. Facial recognition with pca and machine learning methods. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 973–976, 2017.

[7] Shriansh Srivastava, J. Priyadarshini, Sachin Gopal, Sanchay Gupta, and Har Shobhit Dayal. Optical character recognition on bank cheques using 2d convolution neural network. In Hasmat Malik, Smriti Srivastava, Yog Raj Sood, and Aamir Ahmad, editors, *Applications of Artificial Intelligence Techniques in Engineering*, pages 589–596, Singapore, 2019. Springer Singapore.

[8] Claudio Biancalana, Fabio Gasparetti, Alessandro Micarelli, Alfonso Miola, and Giuseppe Sansonetti. Context-aware movie recommendation based on signal processing and machine learning. 10 2011.

[9] Abhishek Soni, Dharamvir Dharmacharya, Amrindra Pal, Vivek Kumar Srivastava, Rabindra Nath Shaw, and Ankush Ghosh. *Design of a Machine Learning-Based Self-driving Car*, pages 139–151. Springer Singapore, Singapore, 2021.

[10] `https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote01_MLsetup.html`.

[11] `https://builtin.com/data-science/regression-machine-learning`.

[12] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques (3rd ed.)*. Elsevier/Morgan Kaufmann, 2012.

[13] Yuanzheng Hu and Marina Sokolova. Explainable multi-class classification of medical data. 12 2020.

[14] `https://analyticsindiamag.com/what-is-extreme-multilabel-text-classification/`.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Xin Jin and Jiawei Han. *K-Means Clustering*, pages 563–564. Springer US, Boston, MA, 2010.

[17] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2, Part 2):3336–3341, 2009.

[18] Tanvi Gupta and Supriya P Panda. Clustering validation of clara and k-means using silhouette amp; dunn measures on iris dataset. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pages 10–13, 2019.

[19] Anant Ram, Jalal Sunita, Anand Jalal, and Kumar Manoj. A density based algorithm for discovering density varied clusters in large spatial databases. *International Journal of Computer Applications*, 3, 06 2010.

[20] Mihael Ankerst, Markus Breunig, Hans-Peter Kriegel, and Joerg Sander. Optics: Ordering points to identify the clustering structure. volume 28, pages 49–60, 06 1999.

[21] Veselina Bureva, Evdokia Sotirova, Stanislav Popov, Deyan Mavrov, and Velichka Traneva. Generalized net of cluster analysis process using sting: A statistical information grid approach to spatial data mining. pages 239–248, 05 2017.

[22] Marek Chrobak, Christoph Dürr, Aleksander Fabijan, and Bengt Nilsson. Online clique clustering. *Algorithmica*, 82:1–28, 04 2020.

[23] D. Baer. Csi: Customer segmentation intelligence for increasing profits.

[24] Gary P. Schneider. page 643. Course Technology Cengage Learning, 2011.

[25] `https://blog.magestore.com/magento-ebook-an-introduction-to-customer-segmentation/`.

[26] Randall S. Collica. *Customer segmentation and clustering using SAS enterprise miner*, page 1–14. SAS, 2011.

[27] Mark Schmidt. Least squares optimization with l1-norm regularization, 2005.

[28] Husam H. Alkinani, Abo Taleb T. Al-Hameedi, Shari Dunn-Norman, Munir Aldin, Deepak Gokaraju, Andreina Guedez, and Atheer M. Alattar. Regularized ridge regression models to estimate static elastic moduli from wireline measurements: Case study from southern iraq - journal of petroleum exploration and production technology, Dec 2021.

[29] Anuja Nagpal. L1 and l2 regularization methods, Oct 2017. `https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c`.

[30] Ryan J. Tibshirani and Larry A. Wasserman. Sparsity , the lasso , and friends statistical machine learning , spring 2017. 2017.

[31] R Muthukrishnan and R Rohini. Lasso: A feature selection technique in predictive modeling for machine learning. In *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*, pages 18–20, 2016.

[32] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[33] Di-Rong Chen, Qiang Wu, Yiming Ying, and Ding-Xuan Zhou. Support vector machine soft margin classifiers: Error analysis. *Journal of Machine Learning Research*, 5:1143–1175, 09 2004.

[34] Wenjian Wang, Zongben Xu, Wei-Zhen Lu, and Xiaoyun Zhang. Determination of the spread parameter in the gaussian kernel for classification and regression. *Neurocomputing*, 55:643–663, 10 2003.

[35] Lior Rokach and Oded Maimon. *Decision Trees*, volume 6, pages 165–192. 01 2005.

[36] Mikhail Moshkov. Time complexity of decision trees. *Transactions on Rough Sets*, 3:244–459, 01 2005.

[37] Joaquim Sá, João Gama, Raquel Sebastião, and Luís Alexandre. Decision trees using the minimum entropy-of-error principle. pages 799–807, 09 2009.

[38] T. Suryakanthi. 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, 01 2020.

[39] Bradley Efron and Gail Gong. A leisurely look at the bootstrap, the jackknife, and. 1983.

[40] L Breiman. Random forests. *Machine Learning*, 45:5–32, 10 2001.

[41] Vladimir Koltchinskii and B. Yu. Three papers on boosting: An introduction. *Ann Stat*, 32, 02 2004.

[42] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm, 1996.

[43] Shebuti Rayana, Wen Zhong, and Leman Akoglu. Sequential ensemble learning for outlier detection: A bias-variance perspective. pages 1167–1172, 12 2016.

[44] Carlos Valle, Francisco Saravia, Héctor Allende, Raul Monge, and César Fernández. Parallel approach for ensemble learning with locally coupled neural networks. *Neural Processing Letters*, 32:277–291, 12 2010.

[45] Jinsol Lee and Ghassan Alregib. Gradients as a measure of uncertainty in neural networks. pages 2416–2420, 10 2020.

[46] Yuzheng Hu, Licong Lin, and Shange Tang. Second-order information in first-order optimization methods, 12 2019.

[47] B.W. Silverman and M. C. Jones. E. fix and j.l. hodges(1951): an important contribution to nonparametric discriminant analysis and density estimation. *International Statistical Review*, 57(3):233–247, 1989.

[48] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[49] W. Aha, Dennis Kibler, and Marc Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 01 1991.

[50] Archana Singh, Avantika Yadav, and Ajay Rana. K-means with three different distance metrics. *International Journal of Computer Applications*, 67:13–17, 04 2013.

[51] Mohd Ansari, Anand Prakash, and Mainuddin Siddique. *Gravitational K-Means Algorithm*, pages 420–429. 05 2020.

[52] Daqing Chen. Online Retail II. UCI Machine Learning Repository, 2019.

[53] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[54] Plotly Technologies Inc. Collaborative data science. 2015.

[55] Ratnadeep Deshmukh and Vaishali Wangikar. Data cleaning: Current approaches and issues. 01 2011.

[56] Tara Rawat and Vineeta Khemchandani. Feature engineering (fe) tools and techniques for better classification performance. 05 2019.

[57] Jo-Ting Wei, Shih-Yen Lin, and Hsin-Hung Wu. A review of the application of rfm model. *African Journal of Business Management December Special Review*, 4:4199–4206, 01 2010.

[58] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[59] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html.

[60] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html.

[61] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

[62] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html.

[63] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html?highlight=lasso#sklearn.linear_model.Lasso.

[64] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html?highlight=decision#sklearn.tree.DecisionTreeRegressor.

[65] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=random#sklearn.ensemble.RandomForestRegressor.

[66] https://xgboost.readthedocs.io/en/stable/python/python_intro.html#scikit-learn-interface.

[67] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[68] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing Management*, 45:427–437, 07 2009.

[69] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[70] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[71] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[72] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

[73] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

APPENDICES

Appendix

Code and System Specifications

1 Code

The code written in this thesis, is given in the jupyter notebook that can be accessed via Github:

`https://github.com/argiris98/Diploma-Thesis/blob/main/Diploma_Thesis.ipynb`

We will mention below some key parts of our code.

1.1 Feature Engineering

```
## df containts the original data after pre-processing
df['Revenue'] = df['Quantity'] * df['Price'] # create revenue value
Latest_Date = dt.datetime(2011,12,10) # Latest Date in our data

## RFM Features

final_df = df.groupby('Customer_ID').agg({'InvoiceDate': lambda x: (
    Latest_Date - x.max()).days,
     'Invoice' : lambda x: x.nunique(),
     'Revenue' : lambda x:  sum(x)}
     )
final_df.rename(columns={'InvoiceDate' :'Recency',
                         'Invoice' : 'Frequency',
                         'Revenue': 'Monetary'}, inplace = True)

```

```
## Additional Features
final_df['time_engaged'] = Latest_Date - df.groupby('Customer_ID')['
    InvoiceDate'].min()
final_df['time_engaged'] = final_df['time_engaged'].dt.days
final_df['time_between'] = final_df['time_engaged']/final_df['Frequency']
final_df['active_period'] = df.groupby('Customer_ID')['InvoiceDate'].max
    () - df.groupby('Customer_ID')['InvoiceDate'].min()
final_df['active_period'] = final_df['active_period'].dt.days
final_df['avg_basket_value'] = final_df['Monetary'] / final_df['Frequency
    ']
final_df.reset_index().sample(5)
```

1.2 Regression Example

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold

# define model
model = Ridge()
# define evaluation
cv = KFold(n_splits=5)
# define search space
space = dict()
space['alpha'] = [1, 10, 20,50,100]
# define search
search = GridSearchCV(model, space, scoring='neg_mean_absolute_error',
    n_jobs=-1, cv=cv)
# execute search
result = search.fit(X_train, y_train)
# summarize result
print('Best Score: %s' % result.best_score_)
print('Best Hyperparameters: %s' % result.best_params_)
```

```
clf = Ridge(alpha=100) #model with best Hyperparameters
clf.fit(X_train,y_train)
y_hat = clf.predict(X_test) # predictions
difference_array = abs(np.subtract(y_hat, np.array(y_test)))
mae = difference_array.mean() # mean absolute error
```

```
print(mae)
```

1.3 Clustering

```
ks = range(1, 12) ## Elbow Method to find optimal k
inertias=[]
for k in ks :
    # Create KMeans clusters
    kc = KMeans(n_clusters=k, random_state=42)
    kc.fit(cluster_df_scaled)
    inertias.append(kc.inertia_)

# Plot ks vs inertias
f, ax = plt.subplots(figsize=(15, 8))
plt.plot(ks, inertias, '-o')
plt.xlabel('Number of clusters, k')
plt.ylabel('Inertia')
plt.xticks(ks)
plt.style.use('ggplot')
plt.title('Ideal number of clusters for KMeans?')
plt.show()
```

```
 # Kmeans Clustering with k =3
kmeans = KMeans(n_clusters = 3).fit(cluster_df_scaled)
kmeans.fit_predict(cluster_df_scaled)
labels = kmeans.labels_
cluster_df['ClusterID']=labels # assign the clusters to customers under '
    ClusterID' column
```

1.4 Classification Example

```
from sklearn.ensemble import RandomForestClassifier
# define model
model = RandomForestClassifier()
# define evaluation
cv = KFold(n_splits=5)
# define search space
space = dict()
```

```
space['n_estimators'] =
    [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,20,25,30,35,40,45]
# define search
search = GridSearchCV(model, space, scoring='accuracy', n_jobs=-1, cv=cv)
# execute search
result = search.fit(X_train, y_train)
# summarize result
print('Best Score: %s' % result.best_score_)
print('Best Hyperparameters: %s' % result.best_params_)
```

```
# RF classifier with best hyperparameters
clf = RandomForestClassifier(n_estimators=45)
clf.fit(X_train,y_train)
y_hat = clf.predict(X_test) #predictions

from sklearn.metrics import confusion_matrix
cf = confusion_matrix(y_test, y_hat)
print(sns.heatmap(cf, annot=True, cmap='Blues',fmt='')) #confusion matrix
print('\nClassification Report\n') #report (Accuracy,Recall,Precision,F1-
    score
print(classification_report(y_test, y_hat))
```

2 System Specifications

The thesis was run on a personal computer with the following specifications:

- **Processor (CPU):** Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz 2.50 GHz
- **RAM:** 16 GB
- **System type:** 64-bit operating system, x64-based processor
- **Hard Drive:** 512GB SSD
- **Graphics Card:** NVIDIA GeForce GTX 860M