MATHEMATICAL MODELS FOR MARKETING

A REPORT

submitted in partial fulfillment of the requirements

for the award of the dual degree of

Bachelor of Science - Master of Science

in

MATHEMATICS

by

PRAJWAL OMPRAKASH DUPARE

(19110)

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH BHOPAL
BHOPAL - 462066

April 2025

भारतीय विज्ञान शिक्षा एवं अनुसंधान संस्थान भोपाल
Indian Institute of Science Education and Research
Bhopal
(Estb. By MHRD, Govt. of India)

CERTIFICATE

This is to certify that **PRAJWAL OMPRAKASH DUPARE**, BS-MS (MATHEMATICS), has worked on the project entitled **'Mathematical Models for Marketing'** under my supervision and guidance. The content of this report is original and has not been submitted elsewhere for the award of any academic or professional degree.

April 2025
IISER Bhopal

DR. KARTICK ADHIKARI
(IISER Bhopal)

Committee Member	**Signature**	**Date**
Dr. Kartick Adhikari	_____	_____
Dr. Dheeraj Kulkari	_____	_____
Dr. Rahul Garg	_____	_____

ACADEMIC INTEGRITY AND COPYRIGHT DISCLAIMER

I hereby declare that this project report is my own work and due acknowledgement has been made wherever the work described is based on the findings of other investigators. This report has not been accepted for the award of any other degree or diploma at IISER Bhopal or any other educational institution. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

I certify that all copyrighted material incorporated into this document is in compliance with the Indian Copyright (Amendment) Act (2012) and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and safeguard IISER Bhopal from any claims that may arise from any copyright violation.

April 2025 **PRAJWAL OMPRAKASH DUPARE**

IISER Bhopal

ACKNOWLEDGEMENT

I would like to express heartfelt gratitude to my supervisor **Dr. Kartick Adhikari**, for his constant guidance, support and insightful feedback that have shaped the direction of this work. I am equally thankful to my committee members, **Dr. Dheeraj Kulkarni** and **Dr. Rahul Garg** for their valuable suggestions and support throughout this journey. I would also like to thank the faculty members and staff of the **Mathematics Department** for their support, timely assistance, and for providing a nurturing environment.

This thesis would not have been possible without the love and support of my family. To my father **Mr. Omprakash Dupare** and mother **Mrs. Seema Dupare**, thank you for being my greatest source of strength and for always believing in me. I am also grateful to my brothers **Rajratna** and **Prashik** and to **Rupali** Vahini, for their encouragement and quiet support. A special mention to my nephew **Krishiv**, whose innocent presence brought joy even during the most stressful days. I fondly remember my late aunt **Kalpana**, Her memories and blessings have always stayed with me.

I extend my gratitude to my fellow batchmates, especially **Lokender, Mizan, Shamil, Ashish, Aman** for the shared learning, discussions, and occasional escapes from routine that made this journey more meaningful.

Lastly, I am deeply grateful to my friends. This journey came with its fair share of stress, setbacks and moments of doubt but somehow, it never felt truly heavy, because I had all of you beside me. To **PD-Prathmesh, Nafdi-Yasmin, VG-Prajwal, Bodhya, Anushka-Chilut, Dallu-Sneh, Gujju, Pranya, Shrujan, Priya Didia, Anuvinda Chechida, Nishta Di, Saniya, Deepak, Vihar, Lakshita, and many more**, thank you for being the constants through every high and low, and for making even the hardest days easier.

PRAJWAL OMPRAKASH DUPARE

ABSTRACT

Marketing has evolved from product-centric strategies to customer-centric frameworks, requiring firms to quantitatively model consumer behavior across various stages of the customer lifecycle. This thesis develops and applies mathematical models to address core marketing challenges through the lens of Customer Relationship Management (CRM). Beginning with foundational tools in Statistical modeling, Machine learning and Deep learning, we construct a suite of models to forecast and optimize customer acquisition, retention, churn, and win-back strategies. Using frameworks such as Binary classification models, Linear regression models, Survival analysis, Neural networks, etc. we address both cross-sectional and temporal aspects of customer behavior.

The thesis models key outcomes such as the probability of acquisition, retention likelihood, churn risk, win-back potential and expected duration of customer-firm relationships. It further estimates Customer Lifetime Value (CLV) and the other customer level metrics using probabilistic, econometric, and learning-based techniques to support strategic resource allocation. By integrating these models across the entire CRM lifecycle, we provide a robust, data-driven framework for actionable marketing decisions. Each model is empirically tested using realistic marketing datasets, ensuring theoretical rigor and practical relevance. Overall, this work unifies mathematical modeling with marketing strategy to help firms maximize long-term customer profitability through informed and targeted interventions.

LIST OF SYMBOLS OR ABBREVIATIONS

Ω	Sample space (state space)
X	Random variable (mapping $\Omega \to \mathbb{R}$)
$\mathbb{P}(X = x)$	Probability mass function for discrete X
$f(x)$	Probability density function
$F(x)$	Cumulative distribution function
$\sigma(x)$	Sigmoid activation function: $\frac{1}{1+e^{-x}}$
η	Learning rate in gradient descent
θ	Parameter vector
$\nabla \ell(\theta)$	Gradient (score function)
$\nabla^2 \ell(\theta)$	Hessian matrix of log-likelihood
$I(\theta)$	Fisher Information Matrix
$\phi(x)$	Standard normal density
$\Phi(x)$	Standard normal CDF
λ	Rate parameter (Exponential, Weibull)
β	Shape parameter (Weibull) or coefficient vector
α	Parameter in Beta distribution
$B(\alpha, \beta)$	Beta function
$\Gamma(\cdot)$	Gamma function
z_i^*	Latent utility (Probit model)
y_i	Observed binary outcome
x_i	Covariate vector
μ_i	Random intercept for unit i

ϵ_i	Error term
$P(y_i = 1)$	Probability of success
$S(t)$	Survival function: $\mathbb{P}(T > t)$
$h(t)$	Hazard function: $\frac{f(t)}{S(t)}$
$H(t)$	Cumulative hazard: $\int_0^t h(u)\,du$
δ_i	Event indicator (1 = event, 0 = censored)
Y_t	k-dimensional vector in VAR model
A_i	Coefficient matrix in VAR
ϵ_t	White noise error at time t
$d(x, x_i)$	Euclidean distance: $\|x - x_i\|_2$
$I(\cdot)$	Indicator function
x_k	Hermite quadrature node
w_k	Hermite quadrature weight
CLV	Customer Lifetime Value
LTV	Lifetime Value
CE	Customer Equity
CRM	Customer Relationship Management
SOW	Share of Wallet
KNN	K-Nearest Neighbors
MLP	Multilayer Perceptron
MSE	Mean Squared Error
MCMC	Markov Chain Monte Carlo
VAR	Vector AutoRegressive Model
OLS	Ordinary Least Squares
2SLS	Two-Stage Least Squares
3SLS	Three-Stage Least Squares

SGD	Stochastic Gradient Descent
NN	Neural Network
ANN	Artificial Neural Network
ReLU	Rectified Linear Unit
AUC	Area Under the Curve
CDF	Cumulative Distribution Function
PDF	Probability Density Function

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

Marketing is a vital component of every organization's strategic framework as it governs how firms engage with their customers, create value, and sustain profitability. Over the decades, the discipline of marketing has evolved from being product-oriented to customer-oriented, acknowledging that long-term success is achieved not merely by selling products but by building relationships and delivering superior value to customers [2].

In the early stages, marketing was predominantly product-centric, focusing on mass production, product innovation, and aggressive selling techniques [3]. However, with the proliferation of choices, increased market competition, and the digital revolution, customers have become more empowered, informed, and selective. This has led to the emergence of a customer-centric marketing paradigm, where the emphasis is on understanding consumer behavior, managing customer relationships, and enhancing customer lifetime value [1].

The primary objective of this thesis is to bridge the gap between marketing theory and quantitative modeling by providing a structured mathematical approach to key marketing problems, particularly in the domain of Customer Relationship Management (CRM). This study leverages fundamental concepts from Probability theory, Statistics, and Machine learning to model and analyze marketing strategies aimed at customer acquisition, retention, and revenue maximization.

The flow of the thesis is structured as follows

- **Chapter 2 Marketing:** This chapter introduces the fundamental concepts of marketing and highlights the motivation for the study. It outlines the shift from product-centric to customer-centric marketing and

discusses the increasing need for data-driven decision-making. The chapter also presents the objectives and significance of the research along with the overall structure of the thesis.

- **Chapter 3 Mathematical Tools:** This chapter provides the essential mathematical background required for modeling marketing problems. It covers the fundamentals of probability theory, random variables, optimization techniques such as gradient descent, and the Newton-Raphson method which are used in model estimation.

- **Chapter 4 Mathematical Models:** In this chapter, various statistical and machine learning models relevant to marketing analytics are developed and discussed. It includes binary classification models like the Logit and Probit models, linear regression models, neural networks and survival Analysis. The mathematical formulation and estimation techniques for these models are elaborated in detail.

- **Chapter 5 Applications in CRM:** This chapter demonstrates the practical application of the developed models in the field of CRM. It focuses on customer acquisition, retention, churn prediction and win-back along with the estimation of Customer Lifetime Value (CLV). The chapter emphasizes how mathematical models can assist in making informed marketing decisions.

Through this structured approach, the thesis seeks to provide a quantitative framework for addressing key marketing challenges. By integrating statistical methods and machine learning with core marketing principles, the study enhances the analytical rigor of marketing decision-making and bridges the gap between theory and practice.

2. MARKETING

2.1 Introduction

Marketing is essential for brands because it shapes how they identify customer needs, create value, and build lasting relationships. A brand that effectively applies marketing principles is better positioned to differentiate itself in a competitive marketplace. As Michael J. Baker explains in The Marketing Book, marketing is not just a business function but a strategic orientation that integrates customer insight into every aspect of brand decision-making. It allows brands to move beyond product-centric approaches and become truly customer-driven, ensuring long-term relevance and competitiveness [3].

From a modern standpoint, marketing also serves as the core driver of brand performance and growth. The Principles of Marketing by OpenStax [4] emphasizes that marketing creates value not only for the brand but also for customers and society, through well-coordinated activities such as product development, pricing, distribution, and promotion.

In the competitive landscape of consumer goods, companies often encounter significant challenges that require innovative marketing strategies to overcome. A notable example is PepsiCo's approach during the 1970s Cola Wars. In the mid-1970s, PepsiCo faced the formidable challenge of competing against Coca-Cola's dominant position in the soft drink market. To address this, PepsiCo launched the "Pepsi Challenge" in 1975, a bold marketing campaign that directly confronted consumer perceptions and Coca-Cola's supremacy. The Pepsi Challenge involved conducting blind taste tests in public venues such as shopping malls, where participants sampled unmarked cups of both Pepsi and Coca-Cola. After tasting, individuals were asked to

choose their preferred beverage, and the results often showed a preference for Pepsi. PepsiCo capitalized on these outcomes by featuring them prominently in their advertising, suggesting that, based solely on taste, consumers favored Pepsi over Coca-Cola. This campaign not only garnered significant media attention but also challenged the existing consumer loyalty towards Coca-Cola. By emphasizing the taste preference revealed in these tests, PepsiCo effectively positioned itself as a formidable competitor, leading to increased market share and altering the dynamics of the cola market during that period. This case exemplifies how innovative marketing strategies, such as engaging consumers directly and challenging established competitors, can effectively alter market dynamics and improve a company's position.

Given the profound role marketing plays in shaping brand success and consumer perception, it is crucial to clearly define and understand what marketing entails. The following section will delve into the fundamental concepts of marketing, clarifying its meaning, scope, and key functions in contemporary business practice.

2.2 What is Marketing

Marketing is defined as the *"activity and processes for creating, communicating, delivering and exchanging offerings that have value for customers, clients, partners, and society at large."* This definition emphasizes the multifaceted role marketing plays, from understanding customer needs to delivering value in a socially responsible way. Marketing involves identifying consumer needs, designing products or services to meet those needs, and then using strategic communication and distribution to ensure that the right product reaches the right person at the right time. Marketing also extends beyond the business context, it contributes to societal well-being by creating products and services that improve lives and sustain economic growth. At its most basic level, marketing is made up of every process involved in moving a product or service from the organization to the consumer [4].

2.2.1 Objectives of Traditional Marketing

Marketing, as a strategic and operational function, is driven by a set of interconnected objectives that align customer needs with business performance. These objectives are not limited to promotional activities but encompass the entire process of value creation, delivery, and relationship management.

1. **Customer Acquisition:** A foundational objective of marketing is to acquire new customers by identifying unmet needs, creating awareness, and delivering value propositions that resonate with specific market segments. This process is facilitated through segmentation, targeting, and positioning (STP), When Spotify launched in new markets, it offered a free trial period with personalized playlists—this campaign effectively targeted tech-savvy, cost-conscious users and led to rapid user acquisition. As Kotler and Keller emphasize that successful acquisition must built on understanding consumer behavior and deploying the right marketing mix to initiate long-term relationships [2].

2. **Revenue Generation:** Marketing acts as a direct contributor to the firm's revenue stream by generating and sustaining demand for its offerings. Through pricing strategies, promotional campaigns, channel selection, and product innovations, marketing ensures that offerings are not only visible but also accessible and appealing to customers. OpenStax highlights that the role of marketing in the value chain is to move products from producers to consumers efficiently while maximizing both customer satisfaction and profitability [4].

3. **Market Share Growth:** Another key objective is expanding the firm's share in the total market. This involves increasing customer base, improving brand recognition, and outperforming competitors through superior value delivery. Kotler outlines that companies can pursue offensive or defensive strategies—such as product innovation, market penetration, or repositioning—to increase their share within target markets. Apple expanded its market share in the smartphone industry by continuously launching upgraded iPhone models with ecosystem integration

and strong brand equity. Metrics like relative market share and brand recall serve as benchmarks for this objective [2].

4. **Customer Retention and Loyalty:** Acquiring customers is only the beginning. Long-term profitability depends heavily on retaining existing customers and increasing their engagement. This is achieved by customer satisfaction, delivering consistent value and building trust. Effective Marketing involves identifying profitable customers and tailoring strategies to increase their lifetime duration and value [1]. Amazon Prime retains users through fast delivery, exclusive content, and personalized recommendations, leading to higher purchase frequency and loyalty.

5. **Profit Maximization via CLV:** Marketing must ensure that all customer related efforts result in financial returns. This involves calculating Customer Lifetime Value (CLV) and allocating resources toward customers who yield the highest long-term profits. CLV helps firms in allocating marketing investments optimally across acquisition, retention, and win-back channels [1].

Definition 2.2.1. (Customer Lifetime Value (CLV)) Customer Lifetime Value (CLV) is the present value of all future profits generated from a customer over the duration of their relationship with the firm. It is a key metric used to evaluate the long-term financial contribution of a customer.

2.2.2 The Traditional Marketing Mix: The 4Ps Framework

The traditional marketing mix, commonly known as the 4Ps framework, represents the foundational model used by marketers to develop and deliver value in a product-centric market environment. Proposed by E. Jerome McCarthy in the 1960s, this model includes four key tactical elements—Product, Price, Place, and Promotion—which organizations strategically combine to meet customer needs and achieve business objectives. Each component plays

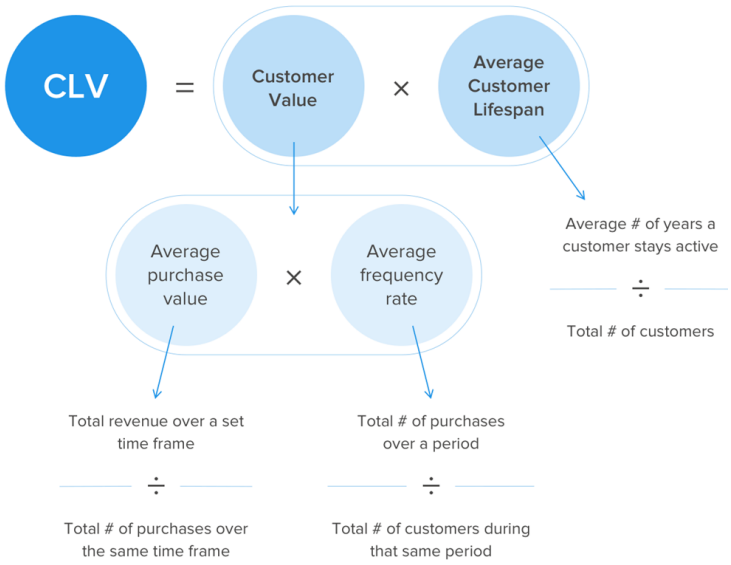


Fig. 2.1: Customer Lifetime Value Illustration

a specific role in influencing demand, positioning offerings, and sustaining competitive advantage. The elements of the 4Ps as discussed in Figure 2.2 are as follows:

Fig. 2.2: The 4Ps Framework

1. **Product:** The product is the core offering of a business that satisfies a specific customer need or want. It encompasses not only the tangible physical goods but also intangible components such as services, experiences, features, quality, and branding. In marketing, the product is more than just an item; it includes everything from design and packaging to after-sale support and warranties. anything that can be offered to a market to satisfy a want or need, which includes goods, services, experiences, events, persons, places, properties, organizations, information, and ideas [2]. Starbucks doesn't just sell coffee—it offers a product experience consisting of quality beverages, ambient store design, personalized service, and brand engagement. All these aspects are intentionally managed under the product strategy.

2. **Price:** Price refers to the amount of money a customer pays to obtain a product or service. It plays a crucial role in influencing consumer demand, brand positioning, and profitability. Unlike other elements of the marketing mix, price directly affects a company's revenue and must be carefully aligned with the perceived value of the offering. Pricing decisions should consider customer expectations, cost structures, competitive conditions, and the company's strategic objectives [2]. Netflix uses tiered pricing to appeal to different market segments, offering various plans based on resolution and screen access. This flexible strategy maximizes reach while retaining profitability. Pricing decisions are often guided by the "Five Cs"—company objectives, customers, costs, competition, and channel partners—which together help firms determine an optimal price point that balances demand and value [4].

3. **Place:** Place refers to the methods and channels through which a product or service is made available to the target customers. It ensures that the offering reaches the right people, at the right time, and in the right condition. Kotler and Keller [2] describe place as encompassing the distribution channels, logistics, and intermediaries that facilitate product movement from the producer to the consumer. Amazon leverages a highly efficient global distribution network to deliver products

quickly and reliably to customers across geographies. This strategic use of warehousing, transportation, and digital platforms exemplifies how place decisions can offer competitive advantage. As noted in Principles of Marketing, effective distribution planning ensures customer convenience, minimizes delivery time, and enhances the overall customer experience [4].

4. **Promotion:** Promotion refers to the communication strategies used to inform, persuade, and remind customers about a product or service. It includes advertising, personal selling, public relations, sales promotions, and digital marketing. Promotion is essential for shaping customer perceptions and creating awareness about the brand's value proposition [2]. A classic example is Coca-Cola's use of integrated marketing communications (IMC)—combining TV ads, social media campaigns, sponsorships, and seasonal promotions to maintain global brand visibility and emotional engagement. OpenStax highlights that promotion should be consistent across all platforms to reinforce the brand message and drive customer action [4].

2.2.3 Evolution from Product-Centric to Customer-Centric Marketing

Marketing strategies have undergone a significant transformation over the past few decades, evolving from a product-centric to a customer-centric orientation. In a product-centric approach, firms primarily focused on mass production, product features, and operational efficiency, assuming that superior products would automatically drive demand. The objective was to sell what the company could produce, often overlooking individual customer preferences and long-term relationships [3].

However, with increasing market saturation, growing competition, and empowered consumers, this model proved inadequate. The contemporary marketplace demands a customer-centric approach, where organizations prioritize understanding customer needs, preferences, and experiences, and tai-

lor their offerings accordingly. This shift not only redefined marketing strategies but also led to the conceptual expansion of the traditional 4Ps marketing mix to the 7Ps framework. The additional elements—People, Process, and Physical Evidence—were introduced to address the dynamic requirements of service-oriented and relationship-driven markets, thereby reinforcing the customer-centric philosophy [1].

Fig. 2.3: The 7Ps Framework

1. **People:** The "People" element in the marketing mix refers to everyone directly or indirectly involved in the delivery of a product or service. In service-based industries, where the offering is intangible and often consumed during interaction, the attitudes, behavior, and competence of staff are vital in shaping customer experience. According to The Marketing Book, the role of employees in relationship marketing is central, as services are produced and consumed simultaneously, making the service personnel part of the value proposition [3]. Singapore Airlines is globally recognized for its customer service, which is embedded into its brand identity. Staff training, empowerment, and consistency play a key role in ensuring customer satisfaction and loyalty. As Kumar and Petersen emphasize, in CRM environments, customer-facing employees are instrumental in influencing outcomes such as retention, churn, and

Customer Lifetime Value (CLV), making them critical to long-term profitability [1].

2. **Process:** Process refers to the systems, workflows, and procedures through which a service or product is created, delivered, and consumed. In services marketing, where intangibility and simultaneity dominate, standardized and efficient processes ensure consistent delivery and customer satisfaction. The Marketing Book notes that well-managed service processes help reduce variability, minimize delays, and enhance perceived reliability, especially when customer involvement is high during the service encounter [3]. A strong example is McDonald's, which maintains globally consistent service quality through standardized cooking, ordering, and delivery processes. This allows them to serve millions with uniform quality. Process design should align with customer expectations while balancing cost efficiency and service personalization to support long-term competitiveness [4].

3. **Physical Evidence:** Physical evidence refers to the tangible cues and environment that support the delivery of a service and help customers evaluate it before and during consumption. Since services are intangible and cannot be assessed in advance like physical goods, customers rely on physical surroundings, branding, design, and ambiance to form perceptions of quality. The Marketing Book highlights that physical evidence acts as a proxy for trust and credibility, particularly in high-contact service settings [3]. Nike's immersive retail environments, with digital displays, athlete imagery, and branded interiors, create a strong emotional and physical connection to performance and innovation. tangible aspects like staff uniforms, service brochures, facilities, and website interfaces are all part of the physical evidence that shapes the customer's confidence in the service [4].

The shift from product-centric to customer-centric marketing was operationalized primarily through the strategic framework of Segmentation, Targeting, and Positioning (STP). This approach emphasizes dividing a hetero-

geneous market into smaller, homogeneous segments based on shared characteristics, selecting the most appropriate segments to serve, and crafting tailored value propositions for each. The STP process enables firms to move away from the "one-size-fits-all" mindset of mass marketing and instead deliver differentiated offerings that align with the preferences and needs of specific customer groups [1].For instance, in the airline industry, firms segment customers based on factors such as income, travel purpose, and frequency of travel. Targeted marketing strategies are then developed for economy, business, and premium class travelers. Effective segmentation and targeting lead to more efficient resource allocation, higher customer satisfaction, and stronger competitive positioning [4].

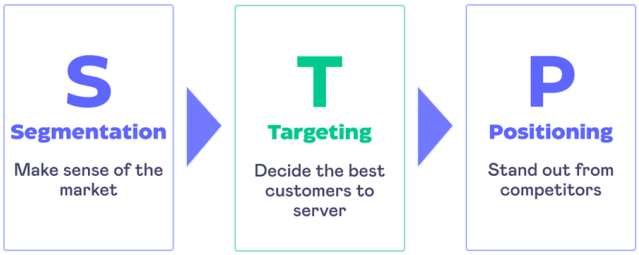


Fig. 2.4: STP Framework

An interesting illustration of effective marketing influencing consumer behavior can be observed in the energy drink market. Consider the scenario of retail margins: retailers earn approximately 4 Rs to 5 Rs on selling a 20 Rs bottle like Sting. As prices increase, the retailer's profit margin rises slightly to around 10 Rs on 60 Rs can of HELL. With Red Bull, priced at 150 Rs, the retailer margin further increases to 35 Rs per can. Monster, priced similarly at 125 Rs, offers an even higher margin of 55 Rs per can. Despite Monster offering a notably higher profit margin, retailers predominantly favor selling Red Bull. The primary reason behind this preference lies in Red Bull's robust branding and targeted marketing campaigns, which significantly influence consumer choices. Effective branding ensures that consumers often prefer Red Bull over competitors, irrespective of price advantages or retailer

margins. Consequently, although Monster provides higher incentives to retailers, its weaker brand positioning and lower consumer demand make it less attractive to retailers.

This example underscores the vital role effective marketing play in shaping consumer preferences.

As marketing evolved towards customer-centricity, the focus shifted from short-term transactions to building long-term, mutually beneficial relationships with customers. This transition gave rise to the concept of Relationship Marketing, which emphasizes retaining existing customers, fostering loyalty, and enhancing customer lifetime value rather than solely acquiring new customers. Baker (2003) points out that relationship marketing aims to create ongoing engagement with customers by delivering consistent value, thereby reducing churn and increasing profitability [3].

The evolution from product-centric to customer-centric marketing has brought about a significant shift in how firms design and deliver value. Rather than focusing solely on transactions and product performance, firms now emphasize building and managing long-term relationships with customers. This shift has led to the strategic adoption of Customer Relationship Management (CRM), where data and analytics play a central role.

At the core of Customer Relationship Management (CRM) are four fundamental objectives that collectively enhance firm profitability. The first is *customer acquisition*, which involves identifying and attracting potential customers. Next is *customer retention*, aimed at sustaining relationships with high-value customers to maximize their lifetime value. *Customer churn management* focuses on predicting and reducing the likelihood of customer attrition, while *customer win-back* refers to re-engaging lapsed customers through targeted strategies.

In modern marketing, achieving core objectives demands not only strategic vision but also quantitative rigor. Accordingly, the upcoming chapters develop the mathematical tools necessary for modeling key CRM elements. Chapter 3 introduces foundational techniques, while Chapter 4 builds CRM-specific models. These tools are applied in Chapter 5, where each of the four objectives is examined through real-world analytical frameworks.

3. MATHEMATICAL TOOLS

3.1 Introduction

This chapter establishes the mathematical foundation required to develop, train, and optimize learning-based models for customer behavior prediction. It systematically introduces essential tools spanning probability theory, optimization algorithms, and advanced numerical methods, all critical for building robust predictive systems—particularly neural networks and probabilistic models.

The chapter begins with a formal treatment of random variables, distinguishing between discrete and continuous types. Key distributions such as Bernoulli, Binomial, Poisson, Normal, and Exponential are explored, alongside their applications in modeling real-world phenomena like customer interactions or system failures. These concepts underpin the probabilistic frameworks used in later chapters.

Next, the focus shifts to optimization techniques, the engine of model training. Gradient descent variants—batch, stochastic, and mini-batch—are derived in the context of minimizing loss functions like Mean Squared Error (MSE). The discussion extends to second-order methods such as Newton-Raphson and Fisher Scoring, which leverage curvature information for faster convergence.

The latter sections address advanced computational tools: Gauss-Hermite quadrature for efficient numerical integration in latent variable models, and Markov Chain Monte Carlo (MCMC) methods for Bayesian inference. These techniques enable scalable parameter estimation and uncertainty quantification in complex, high-dimensional settings.

By unifying probability, optimization, and numerical computation, this chapter equips readers with the analytical core needed to implement and refine the customer-level models introduced in subsequent chapters. Mastery of these tools ensures not only a deeper understanding of model mechanics but also the ability to adapt them to diverse real-world challenges.

3.2 Random Variables

When an experiment is performed we are sometimes primarily concerned about the value of some numerical quantity determined by the result. These quantities of interest that are determined by the results of the experiment are known as random variables [5].

Definition 3.2.1 (Random Variable)**.** A random variable is a function that assigns a real number to each outcome in a sample space. Formally, a random variable is defined as a function $X : \Omega \to \mathbb{R}$, where Ω is the sample space of a probability experiment and $X(\omega)$ maps each outcome $\omega \in \Omega$ to a real number [6].

Example 3.2.2 (Tossing a fair coin)**.** Consider a random variable X that represents the number of heads in a single toss of a fair coin. The sample space is $\Omega = \{\text{Heads, Tails}\}$

We define X as

$$X(\text{Heads}) = 1, \quad X(\text{Tails}) = 0.$$

The probability mass function (PMF) of X is:

$$P(X = 1) = \frac{1}{2}, \quad P(X = 0) = \frac{1}{2}.$$

We primarily classify random variables into two fundamental types as following,

- **Discrete Random Variable:** These take values in a countable set (finite or countably infinite).
- **Continuous Random Variable:** These take uncountably infinite values within a real interval.

In the following subsections, we will explore *discrete* and *continuous* random variables in detail, including their properties, distributions, and applications.

3.2.1 Discrete Random Variable

Definition 3.2.3 (Discrete Random Variable)**.** A random variable that can take either a finite or at most a countable number of possible values is said to be discrete.

For a discrete random variable X, we define its probability mass function $p(x)$ by

$$p(x) = P\{X = x\} \tag{3.1}$$

If X is a discrete random variable that takes on one of the possible values $x_1, x_2, \ldots$, then, since X must take on one of these values, we have

$$\sum_{i=1}^{\infty} p(x_i) = 1 \tag{3.2}$$

Example 3.2.4 (Fair 6-Sided Die)**.** Let X represent the outcome of rolling a fair 6-sided die. The possible values that X can take are $\{1, 2, 3, 4, 5, 6\}$. The probability mass function (PMF) of X is given by $p(x) = \frac{1}{6}$ for each $x \in \{1, 2, \ldots, 6\}$.

Discrete random variables are often classified according to their probability mass functions. We now consider some of these random variables.

1. **The Bernoulli Random Variable:** Suppose that a trial, or an experiment, whose outcome can be classified as either a "success" or as a "failure" is performed. If we let X equal 1 if the outcome is a success

and 0 if it is a failure, then the probability mass function of X is given by

$$p(0) = P(X = 0) = 1 - p \tag{3.3}$$

$$p(1) = P(X = 1) = p \tag{3.4}$$

where p, $0 \leq p \leq 1$, is the probability that the trial is a "success."

A random variable X is said to be a Bernoulli random variable if its probability mass function is given by the above equations for some $p \in (0, 1)$.

Example 3.2.5. A fair coin is flipped once. What is the probability of getting a head? Letting X equal 1 if the outcome is a head ("success") and 0 if it is a tail ("failure"), then X is a Bernoulli random variable with parameter $p = \frac{1}{2}$. Hence,

$$P\{X = 1\} = p = \frac{1}{2}.$$

2. **The Binomial Random Variable:** Suppose that n independent trials, each of which results in a "success" with probability p and in a "failure" with probability $1 - p$, are to be performed. If X represents the number of successes that occur in the n trials, then X is said to be a binomial random variable with parameters (n, p).

   The probability mass function of a binomial random variable having parameters (n, p) is given by

$$p(i) = \binom{n}{i} p^i (1-p)^{n-i}, \qquad i = 0, 1, \ldots, n \tag{3.5}$$

   where

$$\binom{n}{i} = \frac{n!}{(n-i)!\, i!} \tag{3.6}$$

   equals the number of different groups of i objects that can be chosen from a set of n objects.

Example 3.2.6. Four fair coins are flipped. If the outcomes are assumed independent, what is the probability that two heads and two tails are obtained? Letting X equal the number of heads ("successes") that appear, then X is a binomial random variable with parameters $(n = 4, p = \frac{1}{2})$. Hence,

$$P\{X = 2\} = \binom{4}{2}\left(\frac{1}{2}\right)^2\left(\frac{1}{2}\right)^2 = \frac{3}{8}$$

3. **The Geometric Random Variable:** Suppose that independent trials, each having probability p of being a success, are performed until a success occurs. If we let X be the number of trials required until the first success, then X is said to be a geometric random variable with parameter p. Its probability mass function is given by

$$p(n) = P\{X = n\} = (1-p)^{n-1}p, \qquad n = 1, 2, \ldots \tag{3.7}$$

Equation (3.7) follows since in order for X to equal n it is necessary and sufficient that the first $n-1$ trials be failures and the n-th trial a success. Equation (3.7) follows since the outcomes of the successive trials are assumed to be independent.

To check that $p(n)$ is a probability mass function, we note that

$$\sum_{n=1}^{\infty} p(n) = p\sum_{n=1}^{\infty}(1-p)^{n-1} = 1$$

4. **The Poisson Random Variable:** A random variable X is called a Poisson random variable with parameter $\lambda > 0$ if it models the number of events occurring in a fixed interval of time or space, where events happen independently at a constant mean rate λ. Its probability mass function is given by

$$P(X = i) = \frac{e^{-\lambda}\lambda^i}{i!}, \qquad i = 0, 1, 2, \ldots \tag{3.8}$$

Equation (3.8) arises because the Poisson distribution approximates the number of rare events in a large number of trials. To verify that $P(X = i)$ is a valid PMF, observe that:

$$\sum_{i=0}^{\infty} P(X = i) = e^{-\lambda} \sum_{i=0}^{\infty} \frac{\lambda^i}{i!} = e^{-\lambda} \cdot e^{\lambda} = 1$$

(using the Taylor series expansion of e^{λ}).

Example 3.2.7 (Calls at a Call Center)**.** Suppose a call center receives an average of $\lambda = 5$ calls per hour. What is the probability of receiving exactly 3 calls in a given hour? Let X represent the number of calls. X is a Poisson random variable with $\lambda = 5$. Using Equation (2.8):

$$P(X = 3) = \frac{e^{-5}5^3}{3!} = \frac{125 \cdot e^{-5}}{6} \approx 0.1404$$

5. **The Negative Binomial Random Variable:** Suppose independent trials, each with probability p of success, are performed until the r-th success occurs. Let X denote the number of trials required. Then X is a negative binomial random variable with parameters (r, p). Its probability mass function is given by

$$P(X = n) = \binom{n-1}{r-1} p^r (1-p)^{n-r}, \qquad n = r, r+1, r+2, \ldots \tag{3.9}$$

Equation (3.9) follows because to achieve the r-th success on the n-th trial, there must be exactly $r - 1$ successes in the first $n - 1$ trials, followed by a success on the n-th trial.

To verify normalization:

$$\sum_{n=r}^{\infty} \binom{n-1}{r-1} p^r (1-p)^{n-r} = p^r \sum_{k=0}^{\infty} \binom{r+k-1}{r-1} (1-p)^k = p^r \cdot \frac{1}{(1-(1-p))^r} = 1$$

(using the identity $\sum_{k=0}^{\infty} \binom{r+k-1}{k} a^k = \frac{1}{(1-a)^r}$, valid for $|a| < 1$).

Example 3.2.8 (Quality Control Testing)**.** A factory tests components until 3 defective ones are found. The probability of a component being defective is $p = 0.1$. What is the probability that exactly 10 components are tested? Let X represent the number of trials. X is a negative binomial random variable with $r = 3$, $p = 0.1$. Using Equation (2.6):

$$P(X = 10) = \binom{10-1}{3-1}(0.1)^3(0.9)^{10-3} = \binom{9}{2}(0.1)^3(0.9)^7 \approx 0.033$$

3.2.2 Continuous Random Variable

In this section, we shall concern ourselves with random variables whose set of possible values is uncountable. Let X be such a random variable. We say that X is a *continuous* random variable if there exists a nonnegative function $f(x)$, defined for all real $x \in (-\infty, \infty)$, having the property that for any set B of real numbers

$$P(X \in B) = \int_B f(x)\,dx \tag{3.10}$$

The function $f(x)$ is called the probability density function of the random variable X.

In words, Equation (3.10) states that the probability that X will be in B may be obtained by integrating the probability density function over the set B. Since X must assume some value, $f(x)$ must satisfy

$$P\{X \in (-\infty, \infty)\} = \int_{-\infty}^{\infty} f(x)\,dx = 1 \tag{3.11}$$

The relationship between the cumulative distribution $F(\cdot)$ and the probability density $f(\cdot)$ is expressed by

$$F(a) = P(X \in (-\infty, a]) = \int_{-\infty}^{a} f(x)\,dx \tag{3.12}$$

Differentiating both sides of the preceding yields

$$\frac{d}{da}F(a) = f(a)$$

That is, the probability density function is the derivative of the cumula-

tive distribution function.

Continuous random variables are often classified according to their probability density functions. We now consider some of these random variables.

1. **Uniform Random Variable:** A random variable is said to be uniformly distributed over the interval $(0, 1)$ if its probability density function is given by

$$f(x) = \begin{cases} 1, & 0 < x < 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

Note that the preceding is a density function since $f(x) \geq 0$ and

$$\int_{-\infty}^{\infty} f(x)\, dx = \int_{0}^{1} dx = 1 \tag{3.14}$$

Since $f(x) > 0$ only when $x \in (0, 1)$, it follows that X must assume a value in $(0, 1)$. Also, since $f(x)$ is constant for $x \in (0, 1)$, X is just as likely to be "near" any point in that interval.

Example 3.2.9 (Uniform Distribution over $(0, 10)$)**.** If X is uniformly distributed over $(0, 10)$, calculate the probability that:

(a) $X < 3$

(b) $1 < X < 6$

Solution:

(a)

$$P\{X < 3\} = \frac{\int_0^3 dx}{10} = \frac{3}{10}$$

(b)

$$P\{1 < X < 6\} = \frac{\int_1^6 dx}{10} = \frac{5}{10} = \frac{1}{2}$$

2. **Normal Random Variable:** We say that X is a *normal random variable* (or simply that X is normally distributed) with parameters μ and σ^2 if the density of X is given by

$$f(x) = \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-(x-\mu)^2/2\sigma^2}, \quad -\infty < x < \infty \tag{3.15}$$

This density function is a bell-shaped curve that is symmetric around μ.

The cumulative distribution function (CDF) of a normal random variable X with parameters μ and σ^2 is given by:

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-(t-\mu)^2/(2\sigma^2)}\, dt, \quad -\infty < x < \infty \tag{3.16}$$

the variable t ranges over the interval $(-\infty, x]$.

For the standard normal distribution ($\mu = 0$, $\sigma = 1$), the CDF is denoted by $\Phi(x)$,

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}}\, e^{-t^2/2}\, dt \tag{3.17}$$

For a general normal variable $X \sim \mathcal{N}(\mu, \sigma^2)$, the CDF can be expressed as

$$F(x) = \Phi\left(\frac{x-\mu}{\sigma}\right) \tag{3.18}$$

Example 3.2.10 (Student Heights)**.** Assume the heights of college students are normally distributed with a mean of 170 cm and a standard deviation of 10 cm. What is the probability that a randomly selected student has a height between 160 cm and 180 cm? Let X represent the height. X is a normal random variable with parameters $\mu = 170$ and $\sigma = 10$. The probability density function (PDF) of X is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \frac{1}{10\sqrt{2\pi}} e^{-\frac{(x-170)^2}{200}}$$

To compute $P(160 \leq X \leq 180)$, we standardize the values using Z-

scores:

$$Z = \frac{X - \mu}{\sigma}$$

For $X = 160$, $Z = \frac{160-170}{10} = -1$

For $X = 180$, $Z = \frac{180-170}{10} = 1$

Thus,

$$P(160 \leq X \leq 180) = P(-1 \leq Z \leq 1)$$

Using standard normal distribution tables or a calculator:

$$P(-1 \leq Z \leq 1) = \Phi(1) - \Phi(-1) = 0.8413 - 0.1587 = 0.6826$$

So, the probability is approximately 0.6826.

3. **Gamma Random Variable:** A continuous random variable whose density is given by

$$f(x) = \begin{cases} \frac{\lambda e^{-\lambda x}(\lambda x)^{\alpha-1}}{\Gamma(\alpha)}, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.19}$$

for some $\lambda > 0$, $\alpha > 0$, is said to be a *gamma random variable* with parameters α, λ. The quantity $\Gamma(\alpha)$ is called the gamma function and is defined by

$$\Gamma(\alpha) = \int_0^\infty e^{-x} x^{\alpha-1}\, dx \tag{3.20}$$

It is easy to show by induction that for integral α, say, $\alpha = n$,

$$\Gamma(n) = (n-1)!$$

4. **Log-Normal Random Variable:** A random variable X is said to be *log-normally distributed* with parameters μ and σ^2 if its natural logarithm $\ln X$ follows a normal distribution:

$$\ln X \sim \mathcal{N}(\mu, \sigma^2)$$

The probability density function (PDF) of a log-normal random variable is given by:

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right), \quad x > 0 \tag{3.21}$$

The cumulative distribution function (CDF) is:

$$F(x) = \mathbb{P}(X \leq x) = \Phi\left(\frac{\ln x - \mu}{\sigma}\right), \quad x > 0 \tag{3.22}$$

where $\Phi(\cdot)$ denotes the standard normal CDF.

Log-normal distributions are used to model positively skewed data, particularly in finance, insurance, and reliability engineering.

Example 3.2.11 (Household Income)**.** Assume household income in a region follows a log-normal distribution, where the natural logarithm of income is normally distributed with mean $\mu = 3$ and standard deviation $\sigma = 0.5$. What is the probability that a randomly selected household has an income between \$50,000 and \$150,000? Let X represent household income. Since X is log-normally distributed, $Y = \ln(X)$ follows a normal distribution with parameters $\mu = 3$ and $\sigma = 0.5$. The probability density function (PDF) of X is:

$$f_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}, \quad x > 0$$

To compute $P(50{,}000 \leq X \leq 150{,}000)$, transform to the normal variable Y:

$$P(\ln(50{,}000) \leq Y \leq \ln(150{,}000)) =$$
$$P\left(\frac{\ln(50{,}000) - 3}{0.5} \leq Z \leq \frac{\ln(150{,}000) - 3}{0.5}\right)$$

where Z is the standard normal variable. Calculate the bounds:

$$\ln(50{,}000) \approx 10.82, \quad \ln(150{,}000) \approx 11.92$$

$$Z_1 = \frac{10.82 - 3}{0.5} = 15.64, \quad Z_2 = \frac{11.92 - 3}{0.5} = 17.84$$

Using standard normal tables or software:

$$P(15.64 \leq Z \leq 17.84) \approx \Phi(17.84) - \Phi(15.64) \approx 1 - 1 = 0$$

(Note: This result reflects the extreme rarity of values beyond $Z = 3$. For practical purposes, incomes in this range are nearly impossible under the given parameters.)

5. **Logistic Random Variable:** A continuous random variable is said to be a *logistic random variable* if its cumulative distribution function (CDF) is given by

$$F(x) = \frac{1}{1 + e^{-(x-\mu)/s}}, \quad -\infty < x < \infty \tag{3.23}$$

where $\mu \in \mathbb{R}$ is the location parameter and $s > 0$ is the scale parameter. The corresponding probability density function (PDF) is:

$$f(x) = \frac{e^{-(x-\mu)/s}}{s(1 + e^{-(x-\mu)/s})^2}, \quad -\infty < x < \infty \tag{3.24}$$

Example 3.2.12 (Customer Response to a Discount Offer)**.** Suppose a firm models the probability $P(Y = 1|x)$ that a customer responds to a discount offer x, using a logistic function with parameters $\mu = 0$, $s = 1$. What is the probability of response when the discount is 2 units?

Let $X \sim \text{Logistic}(0, 1)$. Then:

$$P(X \leq 2) = F(2) = \frac{1}{1 + e^{-2}} \approx \frac{1}{1 + 0.1353} \approx 0.8808$$

So, there's an 88.08% chance the customer responds to a discount of 2 units.

6. **Gumbel Random Variable:** A continuous random variable is said to be a *Gumbel random variable* (Type I Extreme Value Distribution) if its cumulative distribution function (CDF) is given by

$$F(x) = \exp\left(-e^{-(x-\mu)/\beta}\right), \quad -\infty < x < \infty \tag{3.25}$$

where $\mu \in \mathbb{R}$ is the location parameter and $\beta > 0$ is the scale parameter. The corresponding probability density function (PDF) is:

$$f(x) = \frac{1}{\beta} \exp\left(-\frac{x-\mu}{\beta}\right) \exp\left(-e^{-(x-\mu)/\beta}\right), \quad -\infty < x < \infty \tag{3.26}$$

Example 3.2.13 (Maximum Daily Temperature Exceedance)**.** Suppose the maximum daily temperature X (in °C) in a city during summer follows a Gumbel distribution with $\mu = 35$ and $\beta = 2$. What is the probability that the temperature exceeds 40°C?

We compute:

$$P(X > 40) = 1 - F(40) = 1 - \exp\left(-e^{-(40-35)/2}\right)$$

$$P(X > 40) = 1 - \exp\left(-e^{-2.5}\right) \approx 1 - \exp(-0.0821) \approx 1 - 0.9212 = 0.0788$$

So, there's a 7.88% chance the temperature exceeds 40°C on a summer day.

7. **Exponential Random Variable:** A continuous random variable whose probability density function is given, for some $\lambda > 0$, by

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{3.27}$$

is said to be an *exponential random variable* with parameter λ. The cumulative distribution function F is

$$F(a) = \int_0^a \lambda e^{-\lambda x}\, dx = 1 - e^{-\lambda a}, \quad a \geq 0 \tag{3.28}$$

Note that $F(\infty) = \int_0^\infty \lambda e^{-\lambda x}\, dx = 1$.

Example 3.2.14 (Time Between Customer Arrivals)**.** A service center receives customers at an average rate of $\lambda = 0.5$ customers per minute. What is the probability that the time between two consecutive customers exceeds 3 minutes? Let X represent the time between arrivals. X follows an exponential distribution with $\lambda = 0.5$. Using the CDF:

$$P(X > 3) = 1 - F(3) = e^{-\lambda \cdot 3} = e^{-0.5 \cdot 3} = e^{-1.5} \approx 0.2231.$$

The probability that the time between two consecutive customer arrivals exceeds 3 minutes is approximately 0.2231. This is derived using the exponential distribution with rate parameter $\lambda = 0.5$.

8. **Weibull Random Variable:** A continuous random variable whose density is given by

$$f(x) = \begin{cases} \beta\lambda^\beta x^{\beta-1} e^{-(\lambda x)^\beta}, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.29}$$

for some $\lambda > 0$, $\beta > 0$, is said to be a *Weibull random variable* with parameters λ, β.

The cumulative distribution function (CDF) of the Weibull distribution is given by:

$$F(x) = \begin{cases} 1 - e^{-(\lambda x)^\beta}, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.30}$$

This distribution is commonly used in reliability analysis and survival studies. For $\beta = 1$, it reduces to the exponential distribution with rate λ.

9. **Beta Random Variable**

A continuous random variable whose density is given by

$$f(x) = \begin{cases} \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}, & 0 < x < 1 \\ 0, & \text{otherwise} \end{cases}$$

for some $\alpha > 0$, $\beta > 0$, is said to be a *beta random variable* with parameters α, β.

The quantity $B(\alpha, \beta)$ is the beta function, defined as

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\, dt \tag{3.31}$$

The beta function is related to the gamma function(Equation 3.20) by

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{3.32}$$

The beta distribution is defined on the interval $(0, 1)$ and is useful in modeling proportions and probabilities.

Example 3.2.15 (Modeling Proportion of Time)**.** Suppose the proportion of time a machine is in use during a day follows a Beta distribution with parameters $\alpha = 2$, $\beta = 3$. What is the probability that the machine is used less than half the time?

Let X be the proportion of usage time. Then $X \sim \text{Beta}(\alpha = 2, \beta = 3)$. The required probability is:

$$P(X < 0.5) = \int_0^{0.5} \frac{x^{2-1}(1-x)^{3-1}}{B(2,3)} dx = \int_0^{0.5} \frac{x(1-x)^2}{B(2,3)} dx$$

where $B(2,3) = \frac{\Gamma(2)\Gamma(3)}{\Gamma(5)} = \frac{1\cdot 2}{24} = \frac{1}{12}$. Using numerical integration or a Beta CDF:

$$P(X < 0.5) \approx 0.6875$$

10. **Log-Logistic Random Variable** A continuous random variable whose density is given by

$$f(x) = \begin{cases} \frac{(\beta/\alpha)(x/\alpha)^{\beta-1}}{[1+(x/\alpha)^{\beta}]^2}, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{3.33}$$

for some $\alpha > 0$, $\beta > 0$, is said to be a *log-logistic random variable* with parameters α, β.

The cumulative distribution function (CDF) is given by:

$$F(x) = \begin{cases} \frac{1}{1+(x/\alpha)^{-\beta}}, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{3.34}$$

This distribution is used in survival analysis and reliability applications, especially when the hazard function is non-monotonic.

Example 3.2.16 (Failure Time Modeling)**.** Suppose the failure time (in hours) of a certain device follows a log-logistic distribution with shape parameter $\beta = 2$ and scale parameter $\alpha = 3$. What is the probability that the device fails before 4 hours?

Let $X \sim$ Log-Logistic($\alpha = 3, \beta = 2$). The cumulative distribution function (CDF) is:

$$F(x) = \frac{1}{1 + \left(\frac{\alpha}{x}\right)^{\beta}}, \quad x > 0$$

Plugging in $x = 4$:

$$P(X < 4) = F(4) = \frac{1}{1 + \left(\frac{3}{4}\right)^2} = \frac{1}{1 + 0.5625} = \frac{1}{1.5625} \approx 0.64$$

11. **Exponential Power (Expo-Power) Random Variable:** A continuous random variable whose density is given by

$$f(x) = \begin{cases} \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp\left(-\left(\frac{|x-\mu|}{\alpha}\right)^{\beta}\right), & x \in \mathbb{R} \\ 0, & \text{otherwise} \end{cases} \tag{3.35}$$

for parameters $\mu \in \mathbb{R}$, $\alpha > 0$, and $\beta > 0$, is said to be an *exponential power random variable* (also called generalized normal or generalized error distribution).

Here μ is the location parameter, α is the scale parameter, β is the shape parameter.

When $\beta = 2$, the distribution becomes the normal distribution. $\beta = 1$, it becomes the Laplace (double exponential) distribution.

The distribution is symmetric around μ and is used for robust modeling in statistics and signal processing.

Example 3.2.17 (Deviation Modeling)**.** Suppose a variable X follows an Exponential Power (also known as Generalized Normal) distribution with parameters $\mu = 0$, $\alpha = 1$, and shape $\beta = 2$. What is the probability that X lies between -1 and 1?

The PDF of the Expo-Power distribution is:

$$f(x) = \frac{\beta}{2\alpha\Gamma(1/\beta)} \exp\left(-\left|\frac{x-\mu}{\alpha}\right|^{\beta}\right)$$

For $\mu = 0$, $\alpha = 1$, $\beta = 2$, this reduces to the standard normal-like form:

$$f(x) = \frac{1}{\sqrt{\pi}} \exp(-x^2), \quad \text{since } \Gamma(1/2) = \sqrt{\pi}$$

Then,

$$P(-1 < X < 1) = \int_{-1}^{1} \frac{1}{\sqrt{\pi}} e^{-x^2} dx \approx 0.6826$$

3.3 Gradient Descent

Gradient descent is a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ w.r.t. to the parameters. The learning rate η determines the size of the steps we take to reach a (local) minimum. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a valley.

There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update.

3.3.1 Batch gradient descent

Vanilla gradient descent, aka batch gradient descent, computes the gradient of the cost function w.r.t. to the parameters θ for the entire training dataset:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \cdot \nabla_\theta J(\theta_{\text{old}}) \tag{3.36}$$

As we need to calculate the gradients for the whole dataset to perform just *one* update, batch gradient descent can be very slow and is intractable for datasets that do not fit in memory [7].

We then update our parameters in the direction of the gradients with the learning rate determining how big of an update we perform. Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces.

3.3.2 Stochastic gradient descent

Stochastic gradient descent (SGD) in contrast performs a parameter update for *each* training example $x^{(i)}$ and label $y^{(i)}$:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \cdot \nabla_\theta J(\theta_{\text{old}}; x^{(i)}; y^{(i)}) \tag{3.37}$$

Batch gradient descent perform redundant computations for large dataset as it recomputes gradients for similar examples before each parameter update. SGD does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn online. SGD performs frequent updates with a high variance that cause the objective function to fluctuate heavily as in Figure 3.1 [7].

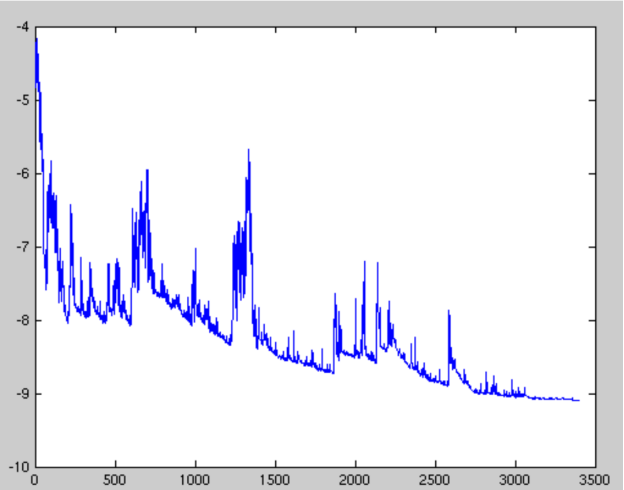


Fig. 3.1: SGD fluctuation

While batch gradient descent converges to the minimum of the basin the parameters are placed in, SGD's fluctuation, on the one hand, enables it to jump to new and potentially better local minima. On the other hand, this ultimately complicates convergence to the exact minimum, as SGD will keep overshooting. However, it has been shown that when we slowly decrease the learning rate, SGD shows the same convergence behavior as batch gradient descent, almost certainly converging to a local or the global minimum for non-convex and convex optimization respectively.

3.3.3 Mini-batch gradient descent

Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \cdot \nabla_\theta J(\theta_{\text{old}}; x^{(i:i+n)}, y^{(i:i+n)}) \tag{3.38}$$

This way, it a) reduces the variance of the parameter updates, which can lead to more stable convergence; and b) can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient. Common mini-batch sizes range between 50 and 256, but can vary for different applications. Mini-batch gradient descent is typically the algorithm of choice when training a neural network and the term SGD usually is employed also when mini-batches are used [7].

3.4 Newton-Raphson Method

Newton's method (also acknowledged as the Newton–Raphson method), named after Isaac Newton and Joseph Raphson, is a technique for judgment sequentially superior approximations to the extraction (or zeroes) of a real-valued function.

$$x : f(x) = 0.$$

The Newton-Raphson method is derived using the Taylor series expansion of a function $f(x)$ around an initial guess x_0.

Expanding $f(x)$ around x_0 using the first-order Taylor series:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \mathcal{O}((x - x_0)^2)$$

Ignoring higher-order terms $\mathcal{O}((x - x_0)^2)$, we approximate:

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0)$$

To find the root of $f(x)$, we set $f(x) = 0$:

$$0 = f(x_0) + (x - x_0)f'(x_0)$$

Rearranging for x:

$$x - x_0 = -\frac{f(x_0)}{f'(x_0)}$$

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Since this process is iterative, we generalize it to:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

This is the Newton-Raphson iteration formula, which is used to iteratively refine the approximation of a root.

It can also be used to find a minimum or maximum of such a function, by finding a zero in the function's first derivative, see Newton's method as an optimization algorithm.

In optimization, Newton's method is applied to find stationary points by setting the gradient to zero, i.e., solving $\nabla J(\theta) = 0$. The iterative update rule is given by:

$$\theta_{n+1} = \theta_n - \frac{\nabla J(\theta_n)}{H_J(\theta_n)}$$

where $\nabla J(\theta)$ is the gradient vector, which consists of the first-order partial derivatives of $J(\theta)$ with respect to θ, and $H_J(\theta)$ is the Hessian matrix, which consists of the second-order partial derivatives of $J(\theta)$, capturing the curvature of the function.

3.5 Fisher Scoring

Fisher Scoring is an iterative optimization technique used to obtain the Maximum Likelihood Estimates (MLEs) of parameters in statistical models. It

is a variant of the Newton-Raphson method where the observed information matrix is replaced by the expected information matrix. This replacement often leads to more stable convergence, particularly in generalized linear models and mixed effects models [8].

Suppose we wish to maximize the log-likelihood function $\ell(\theta)$ with respect to a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^p$. The Newton-Raphson update for estimating the MLE $\hat{\theta}$ is given by:

$$\theta^{(t+1)} = \theta^{(t)} - \left[\nabla^2 \ell(\theta^{(t)})\right]^{-1} \nabla \ell(\theta^{(t)}), \tag{3.39}$$

where $\nabla \ell(\theta)$ is the gradient (score function) and $\nabla^2 \ell(\theta)$ is the Hessian (observed information matrix).

However, in Fisher Scoring, we replace the observed information matrix with the expected information matrix:

$$\mathcal{I}(\theta) = \mathbb{E}\left[-\nabla^2 \ell(\theta)\right], \tag{3.40}$$

resulting in the following iterative scheme:

$$\theta^{(t+1)} = \theta^{(t)} + \mathcal{I}^{-1}(\theta^{(t)}) \nabla \ell(\theta^{(t)}). \tag{3.41}$$

This method ensures that the curvature of the likelihood is approximated using its average behavior under the model, which often enhances numerical stability and convergence [9].

3.6 Gauss-Hermite Quadrature

In many statistical and econometric applications, it is often necessary to evaluate integrals of the form:

$$I = \int_{-\infty}^{\infty} f(x) e^{-x^2} dx, \tag{3.42}$$

especially when the function $f(x)$ is multiplied by the Gaussian kernel e^{-x^2}. A common example arises in marginal likelihood estimation in models with

Gaussian random effects or latent variables, such as mixed-effects models, hierarchical Bayes models, and generalized linear mixed models.

3.6.1 Quadrature Approximation

Gauss-Hermite quadrature provides an efficient numerical method to approximate integrals of the form in Equation (3.42). It does so by evaluating the function $f(x)$ at specific nodes and weighting them appropriately:

$$I \approx \sum_{k=1}^{K} w_k f(x_k), \tag{3.43}$$

where x_k are the roots of the K-th order Hermite polynomial $H_K(x)$, and w_k are the corresponding weights. This approximation is exact when $f(x)$ is a polynomial of degree up to $2K-1$, and it performs well for many smooth, bounded integrands [10].

3.6.2 Transformations for General Gaussian Integration

Gauss-Hermite quadrature assumes the weight function e^{-x^2}, whereas many practical problems involve the standard normal density that is

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}. \tag{3.44}$$

To adapt Gauss-Hermite quadrature to this context, we apply a change of variables. If $x = \frac{z}{\sqrt{2}}$, then:

$$\int_{-\infty}^{\infty} f(z)\phi(z)\,dz = \int_{-\infty}^{\infty} f(\sqrt{2}x) \cdot \frac{1}{\sqrt{\pi}} e^{-x^2}\,dx \approx \frac{1}{\sqrt{\pi}} \sum_{k=1}^{K} w_k f(\sqrt{2}x_k). \tag{3.45}$$

This transformation allows Gauss-Hermite quadrature to be used for computing expectations over normal distributions, which is a frequent requirement in latent variable models [11].

3.7 Markov Chain Monte Carlo (MCMC)

3.7.1 Introduction

In modern computational statistics, particularly in Bayesian inference, the need to evaluate high-dimensional integrals or sample from complex probability distributions is ubiquitous. Traditional numerical integration methods often fail in these scenarios due to the curse of dimensionality. This challenge has led to the development of Markov Chain Monte Carlo (MCMC) methods—an elegant class of algorithms that allow for approximate inference by simulating dependent samples from a desired target distribution.

MCMC methods are built upon two foundational ideas: Markov chains, which provide a framework for constructing sequences of dependent random variables with a specified stationary distribution, and Monte Carlo integration, which allows us to estimate expectations with respect to a distribution using random samples. When combined, these ideas enable practical and scalable estimation in settings where analytical solutions are unavailable.

To fully appreciate the mechanism and utility of MCMC, it is important to first understand its two components separately. We therefore begin by briefly reviewing the theory of Markov chains and Monte Carlo methods before discussing how they are fused in MCMC algorithms.

3.7.2 Markov Chains

A *Markov Chain* is a stochastic process that satisfies the Markov property, meaning that the future state of the process depends only on the present state and not on the sequence of states that preceded it. For a discrete-time stochastic process $\{X_t\}_{t=0}^{\infty}$, this property is formally written as

$$\begin{aligned} P(X_{t+1} = x_{t+1} \mid X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0) \\ = P(X_{t+1} = x_{t+1} \mid X_t = x_t) \end{aligned} \tag{3.46}$$

This memoryless property makes Markov chains particularly tractable

and widely used in statistical computing and simulation-based inference. Each step in a Markov chain is governed by a transition probability, which may be encoded in a transition probability matrix $P = [P_{ij}]$ for discrete state spaces. Here, P_{ij} denotes the probability of transitioning from state i to state j.

A Markov chain is said to be *irreducible* if it is possible to reach any state from any other state (possibly in multiple steps), and *aperiodic* if the chain does not cycle in a deterministic manner. Under these conditions, the chain possesses a unique stationary distribution π, satisfying:

$$\pi = \pi P \tag{3.47}$$

This stationary distribution represents the long-run behavior of the Markov chain, such that, regardless of the initial state, the distribution of states after many transitions converges to π [12]. A particularly important subclass of Markov chains are those that satisfy the detailed balance condition,

$$\pi(i)P(i \to j) = \pi(j)P(j \to i), \quad \forall i, j \tag{3.48}$$

A chain satisfying this condition is said to be *reversible* with respect to π. Reversibility ensures that the transitions of the Markov chain mirror one another in a probabilistic sense, and it simplifies the verification that π is indeed the stationary distribution [13].

Example 3.7.1. Consider a simple weather model where the states are Sunny, Rainy, and Cloudy. Suppose today's weather only depends on yesterday's weather, and not on any earlier conditions. Then, the system can be modeled using a Markov chain with a transition matrix like

$$P = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.4 & 0.4 & 0.2 \\ 0.3 & 0.3 & 0.4 \end{bmatrix}$$

Each row sums to 1, and repeated application of this transition matrix eventually leads the system to a stationary distribution, representing the long-term weather probabilities [14].

Relevance to MCMC: Markov chains form the foundational backbone of Markov Chain Monte Carlo (MCMC) methods. In MCMC, one constructs a Markov chain in such a way that its stationary distribution is equal to a complex target distribution (often a Bayesian posterior). By simulating the chain for a sufficiently long time, samples from this complex distribution can be obtained, thus facilitating approximate inference [12, 13].

3.7.3 Monte Carlo Methods

Monte Carlo (MC) methods are a class of computational algorithms that rely on repeated random sampling to compute numerical estimates of mathematical quantities. These methods are especially powerful when applied to high-dimensional problems where analytical or deterministic numerical techniques are impractical or infeasible.

Suppose $h(x)$ is a function of interest and $p(x)$ is a probability density function defined over domain $\mathcal{X}$. The expectation of $h(x)$ under $p(x)$ is defined as:

$$\mathbb{E}_p[h(x)] = \int_{\mathcal{X}} h(x)p(x)\,dx \tag{3.49}$$

When direct integration is infeasible, this expectation can be approximated using Monte Carlo sampling. Drawing independent and identically distributed (i.i.d.) samples $x^{(1)}, \ldots, x^{(N)} \sim p(x)$, the Monte Carlo estimator is:

$$\mathbb{E}_p[h(x)] \approx \frac{1}{N} \sum_{i=1}^{N} h(x^{(i)}) \tag{3.50}$$

As $N \to \infty$, the estimator converges almost surely to the true expectation by the Strong Law of Large Numbers [15].

When it is difficult to sample from $p(x)$ directly, an alternative approach is to use a proposal distribution $q(x)$ from which sampling is easier. The expectation can then be rewritten as:

$$\mathbb{E}_p[h(x)] = \int h(x)\frac{p(x)}{q(x)}q(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N} h(x^{(i)}) \cdot \frac{p(x^{(i)})}{q(x^{(i)})} \tag{3.51}$$

where $x^{(i)} \sim q(x)$ and the term $\frac{p(x)}{q(x)}$ serves as an importance weight [16].

3.7.4 Mathematical Foundation

Let $\pi(x)$ be a target probability distribution defined on a state space $\mathcal{X} \subset \mathbb{R}^d$, known only up to a normalizing constant:

$$\pi(x) = \frac{p(x)}{Z}, \quad \text{where} \quad Z = \int_{\mathcal{X}} p(x)\,dx \text{ is intractable.} \tag{3.52}$$

Our goal is to estimate expectations of the form:

$$\mathbb{E}_\pi[h(x)] = \int_{\mathcal{X}} h(x)\pi(x)\,dx. \tag{3.53}$$

If we can generate samples $x^{(1)}, \ldots, x^{(T)}$ from a Markov chain whose stationary distribution is $\pi(x)$, then the Monte Carlo estimate is:

$$\hat{\mu}_T = \frac{1}{T}\sum_{t=1}^{T} h(x^{(t)}). \tag{3.54}$$

By the Ergodic Theorem, if the Markov chain is irreducible, aperiodic, and positive Harris recurrent, then $\hat{\mu}_T \xrightarrow{a.s.} \mathbb{E}_\pi[h(x)]$ as $T \to \infty$ [17].

3.7.5 The Metropolis-Hastings Algorithm

The most fundamental MCMC algorithm is the Metropolis-Hastings (MH) algorithm. Starting from an initial state $x^{(0)}$, the algorithm iterates:

1. Given current state $x^{(t)}$, sample a candidate x^* from a proposal distribution $q(x^* \mid x^{(t)})$.

2. Compute the acceptance probability:

$$\alpha(x^{(t)}, x^*) = \min\left(1, \frac{p(x^*)q(x^{(t)} \mid x^*)}{p(x^{(t)})q(x^* \mid x^{(t)})}\right) \tag{3.55}$$

3. Set:

$$x^{(t+1)} = \begin{cases} x^*, & \text{with probability } \alpha(x^{(t)}, x^*) \\ x^{(t)}, & \text{otherwise.} \end{cases}$$

This algorithm ensures that the resulting Markov chain satisfies the detailed balance condition

$$\pi(x)q(x' \mid x)\alpha(x, x') = \pi(x')q(x \mid x')\alpha(x', x), \tag{3.56}$$

which implies that $\pi(x)$ is the stationary distribution of the chain [18, 19].

3.7.6 Special Cases

If the proposal distribution is symmetric, i.e., $q(x' \mid x) = q(x \mid x')$, the acceptance probability simplifies to:

$$\alpha(x, x') = \min\left(1, \frac{p(x')}{p(x)}\right), \tag{3.57}$$

which defines the original Metropolis algorithm.

3.7.7 Gibbs Sampling

Gibbs Sampling is a special case of the Metropolis-Hastings algorithm, particularly well-suited for high-dimensional distributions when full conditional distributions are available in closed form.

Assume we aim to sample from a joint distribution $\pi(x_1, x_2, \ldots, x_d)$ where we can derive each full conditional as

$$\pi(x_i \mid x_{-i}), \quad \text{for } i = 1, 2, \ldots, d, \tag{3.58}$$

where x_{-i} denotes the vector of all components except x_i. Gibbs sampling proceeds by sequentially sampling as

$$\begin{aligned} x_1^{(t+1)} &\sim \pi(x_1 \mid x_2^{(t)}, x_3^{(t)}, \ldots, x_d^{(t)}) \\ x_2^{(t+1)} &\sim \pi(x_2 \mid x_1^{(t+1)}, x_3^{(t)}, \ldots, x_d^{(t)}) \\ &\vdots \\ x_d^{(t+1)} &\sim \pi(x_d \mid x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_{d-1}^{(t+1)}) \end{aligned} \tag{3.59}$$

Each update leaves the target distribution invariant. Gibbs sampling always accepts proposed values and requires no tuning parameters, making it efficient when applicable [20, 21].

Both MH and Gibbs sampling produce Markov chains. After a suitable burn-in period, samples are assumed to approximate the target distribution.

This chapter has equipped readers with essential mathematical tools to design, train, and optimize learning-based models for customer behavior prediction. From foundational probability theory (discrete/continuous distributions) to optimization methods (gradient descent, Newton-Raphson) and advanced computational techniques (MCMC, quadrature), these concepts form the backbone of modern predictive analytics. By bridging theoretical principles with practical implementation, the chapter lays the groundwork for applying these tools to real-world models in subsequent sections, ensuring robust, scalable, and interpretable solutions for customer-level analysis.

4. MATHEMATICAL MODELS

4.1 Introduction

This chapter focuses on the development of mathematical models to quantify customer behavior and optimize firm-level decision-making in Customer Relationship Management (CRM). These models provide structured frameworks to estimate key outcomes such as acquisition likelihood, retention probability, churn timing, and customer lifetime value. By leveraging statistical and probabilistic foundations, we translate real-world customer interactions into tractable equations that can be estimated, interpreted, and acted upon. Whether the outcome is binary (e.g., response vs. no response), continuous (e.g., duration of engagement) or categorical (e.g., store choice), each model is tailored to reflect the data structure and business objective.

We cover a wide range of models that extend beyond classical statistical techniques to include machine learning and deep learning approaches, each grounded in a rigorous mathematical foundation. The chapter explores models such as logistic regression, Tobit models, hazard-based survival analysis, and share-of-wallet estimation using attraction models and K-nearest neighbors. In addition, we delve into neural network-based classifiers and duration models capable of capturing non-linearity and handling high-dimensional data, which are increasingly relevant in modern CRM applications. Each framework is examined through its mathematical formulation, estimation methodology, and empirical relevance. Collectively, these models form a comprehensive toolkit for predicting customer behavior, identifying high-value segments, and guiding data-driven strategies to optimize long-term business outcomes.

4.2 Binary Classification Models

Binary classification models are used to predict outcomes that take one of two possible values, typically represented as $Y \in \{0, 1\}$. These models estimate the probability that a given observation belongs to a specific category based on a set of predictor variables.

Binary models often assume an underlying latent continuous variable Y^*, which determines the observed binary outcome Y. The general form of the latent variable model is

$$Y^* = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k + \epsilon, \quad \epsilon \sim F_\epsilon(0, \sigma^2) \tag{4.1}$$

where Y^* is an unobserved continuous latent variable, $X = (X_1, X_2, \ldots, X_k)$ represents the independent variables, $\beta = (\beta_0, \beta_1, \ldots, \beta_k)$ are the regression coefficients, and ϵ is an error term capturing unobservable factors and follows a distribution $F_\epsilon(0, \sigma^2)$.

The observed binary outcome Y is defined as

$$Y = \begin{cases} 1, & \text{if } Y^* > 0 \\ 0, & \text{if } Y^* \leq 0 \end{cases} \tag{4.2}$$

Thus, the probability of observing $Y = 1$ is

$$P(Y = 1|X) = P(Y^* > 0|X) = P(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k + \epsilon > 0) \tag{4.3}$$

$$\Rightarrow \; P(Y = 1|X) = P(\epsilon > -(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k)) \tag{4.4}$$

where ϵ follows a specific probability distribution $F_\epsilon(\cdot)$. The choice of the error term ϵ distribution leads to different binary classification models. Two commonly used models are as

1. **Logit Model**: Assumes the error term follows a logistic distribution.
2. **Probit Model**: Assumes the error term follows a standard normal distribution.

Each model applies a different cumulative distribution function (CDF) to transform the linear predictor βX into a probability.

4.2.1 Logit Model

The logit model is derived by assuming that the error term ϵ follows a logistic distribution (see Section 3.1.2.5). The cumulative distribution function (CDF) of a logistic distribution is given by:

$$F_\epsilon(t) = \frac{1}{1 + e^{-t}} \tag{4.5}$$

Using this assumption, the probability of observing $Y = 1$ is:

$$P(Y = 1|X) = P(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k + \epsilon > 0) \tag{4.6}$$

which can be rewritten as:

$$P(Y = 1|X) = P(\epsilon > -(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k)) \tag{4.7}$$

Applying the logistic CDF:

$$P(Y = 1|X) = 1 - F_\epsilon(-(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k)) \tag{4.8}$$

Since the logistic CDF satisfies the symmetry property $F_\epsilon(-t) = 1 - F_\epsilon(t)$, we obtain:

$$P(Y = 1|X) = F_\epsilon(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k)}} \tag{4.9}$$

This function, known as the sigmoid function, ensures that probability values remain within the range $(0, 1)$, see Figure - 4.1. To make classification decisions, we assign $Y = 1$ if the predicted probability is greater than or equal to 0.5, and $Y = 0$ otherwise.

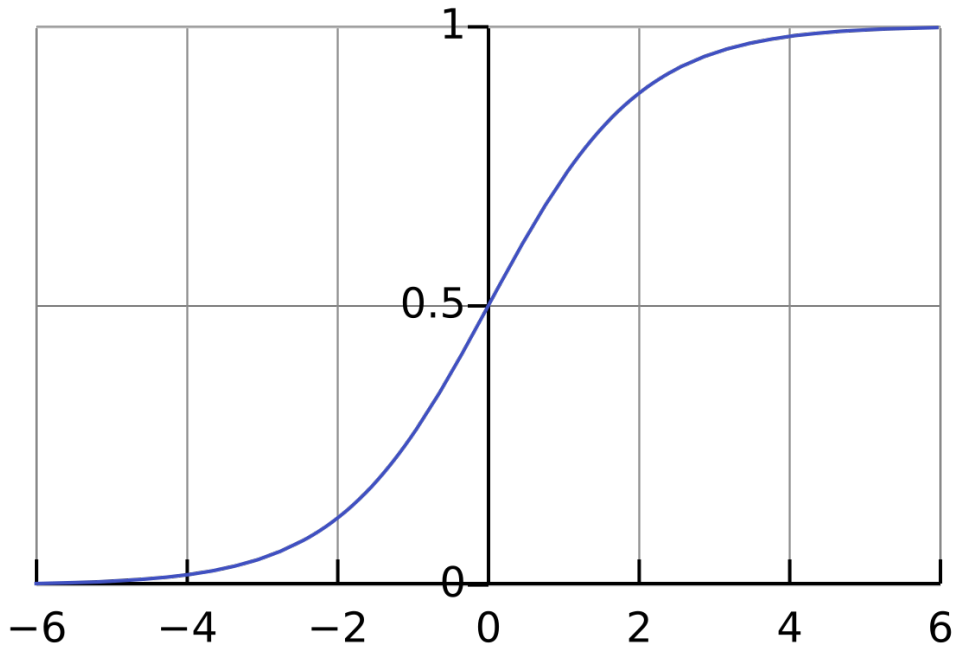


Fig. 4.1: Sigmoid function

4.2.2 Probit Model

In the probit model, we assume that the error term ϵ follows a standard normal distribution (see Section 3.1.2.2),

$$\epsilon \sim \mathcal{N}(0, 1) \tag{4.10}$$

Thus, the probability of observing $Y = 1$ is:

$$P(Y = 1|X) = P(\beta X + \epsilon > 0) \tag{4.11}$$

$$= P(\epsilon > -\beta X) \tag{4.12}$$

Using the cumulative distribution function (CDF) of the standard normal distribution $\Phi(t)$, we express this probability as:

$$P(Y = 1|X) = \Phi(\beta X) \tag{4.13}$$

where $\Phi(t)$ is the CDF of the standard normal distribution:

$$\Phi(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{-z^2/2} dz \tag{4.14}$$

Unlike the logistic function, which has an analytical form, the probit function requires numerical integration.

The probit model predicts $P(Y = 1)$ using the normal CDF:

$$P(Y = 1|X) = \Phi(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k) \tag{4.15}$$

where β_j represents the impact of X_j on the probability.

To make classification decisions, we assign $Y = 1$ if the predicted probability is greater than or equal to 0.5, and $Y = 0$ otherwise.

4.2.3 Random Intercept Model

The Random Intercept Model is a special case of a mixed effects model used to analyze clustered or hierarchical data. It accounts for unobserved heterogeneity across groups by allowing the intercept to vary randomly.

Model Specification

Consider a binary response variable $Y_{ij} \in \{0, 1\}$, where $i = 1, \ldots, N$ indexes the groups (e.g., subjects, firms), and $j = 1, \ldots, T_i$ indexes observations within each group. The latent variable formulation is given by:

$$Y_{ij}^* = \mu_i + \boldsymbol{\beta}^\top \mathbf{x}_{ij} + \varepsilon_{ij} \tag{4.16}$$

where $\mathbf{x}_{ij} \in \mathbb{R}^p$ is a vector of covariates, $\boldsymbol{\beta} \in \mathbb{R}^p$ is a vector of fixed-effect coefficients, $\mu_i \sim \mathcal{N}(0, \sigma^2)$ is the random intercept for group i, and $\varepsilon_{ij} \sim F$, with F typically logistic, standard normal, or Gumbel.

The observed binary outcome is determined by:

$$Y_{ij} = \begin{cases} 1 & \text{if } Y_{ij}^* > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.17}$$

Probability Model

The conditional probability of success given the random intercept μ_i is:

$$\mathbb{P}(Y_{ij} = 1 \mid \mu_i) = G(\mu_i + \boldsymbol{\beta}^\top \mathbf{x}_{ij}) \tag{4.18}$$

where G is the cumulative distribution function (CDF) of ε_{ij}.

Marginal Likelihood

Since μ_i is unobserved, the likelihood contribution for group i is obtained by integrating over the distribution of the random intercept:

$$L_i(\boldsymbol{\beta}, \sigma^2) = \int_{-\infty}^{\infty} \prod_{j=1}^{T_i} G(\mu + \boldsymbol{\beta}^\top \mathbf{x}_{ij})^{Y_{ij}} [1 - G(\mu + \boldsymbol{\beta}^\top \mathbf{x}_{ij})]^{1-Y_{ij}} \cdot \phi(\mu \mid 0, \sigma^2)\, d\mu \tag{4.19}$$

The total log-likelihood over all groups is:

$$\log L(\boldsymbol{\beta}, \sigma^2) = \sum_{i=1}^{N} \log L_i(\boldsymbol{\beta}, \sigma^2) \tag{4.20}$$

Estimation

Estimation is typically performed via maximum likelihood estimation (see Section 4.2.4) using numerical integration methods such as adaptive Gauss-Hermite quadrature (see Section 3.6) or Monte Carlo integration (see Section 3.7).

The Random Intercept Model accounts for correlation within groups and allows flexible modeling of hierarchical or panel data structures. It assumes that unobserved heterogeneity across groups can be captured by a group-specific random effect. This structure enhances model accuracy by separating within-group variability from between-group variability.

4.2.4 Maximum Likelihood Estimation (MLE)

Since logistic regression and probit model do not yield closed-form solutions for parameter estimation, unlike Ordinary Least Squares (OLS) (see Section 4.3.1), their parameters are estimated using Maximum Likelihood Estimation (MLE).

The likelihood function for a dataset of size N is given by

$$L(\beta) = \prod_{i=1}^{N} P(y_i)^{y_i} (1 - P(y_i))^{(1-y_i)} \tag{4.21}$$

where $P(y_i)$ is the probability that $Y_i = 1$, $1 - P(y_i)$ is the probability that $Y_i = 0$ and the likelihood function is the product of probabilities across all observations.

Taking the log-likelihood function as

$$\ell(\beta) = \sum_{i=1}^{N} [y_i \log P(y_i) + (1 - y_i) \log(1 - P(y_i))] \tag{4.22}$$

The optimal parameter β is obtained by maximizing $\ell(\beta)$, which does not have a closed-form solution and must be solved using numerical optimization techniques.

To maximize the log-likelihood function, we compute its first and second derivatives.

The gradient (first derivative) is given by

$$\nabla \ell(\beta) = \sum_{i=1}^{N} X_i \left(y_i - \sigma(X_i^\top \beta) \right) \tag{4.23}$$

where X_i represents the feature vector for the i-th observation.

The Hessian (second derivative) is given by:

$$\nabla^2 \ell(\beta) = \frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^\top} = -\sum_{i=1}^{N} X_i X_i^\top \, \sigma(X_i^\top \beta) \left(1 - \sigma(X_i^\top \beta) \right) \tag{4.24}$$

The Hessian matrix is used in second-order optimization methods such as the Newton-Raphson algorithm to update the parameters efficiently.

Optimization Techniques

1. **Gradient Descent**

   Gradient descent (see Section - 3.3) is an iterative method used to update the parameters β in the direction of the gradient of the log-likelihood function:

   $$\beta^{(t+1)} = \beta^{(t)} + \alpha \nabla \ell(\beta^{(t)}) \tag{4.25}$$

   where α is the learning rate, and $\nabla \ell(\beta^{(t)})$ is the gradient of the log-likelihood function evaluated at iteration t.

2. **Newton-Raphson Method**

   The Newton-Raphson (see Section - 3.4) method is a second-order optimization technique that uses both the gradient and hessian matrix for faster convergence:

   $$\beta^{(t+1)} = \beta^{(t)} - \left[\nabla^2 \ell(\beta^{(t)})\right]^{-1} \nabla \ell(\beta^{(t)}) \tag{4.26}$$

   where $\nabla \ell(\beta^{(t)})$ is the gradient of the log-likelihood function evaluated at $\beta^{(t)}$ and $\nabla^2 \ell(\beta^{(t)})$ is the Hessian matrix (matrix of second derivatives) evaluated at $\beta^{(t)}$.

   Newton-Raphson converges faster than gradient descent but requires computing the second derivative, making it computationally expensive for large datasets.

   Gradient Descent uses the first-order derivative (Gradient) and has slower convergence. Newton-Raphson uses the second-order derivative (Hessian) and converges faster.

3. **Fisher Scoring**

   Fisher Scoring (see Section 3.5) is an iterative optimization method used to estimate parameters θ by maximizing the log-likelihood $\ell(\theta)$. It updates the parameters using,

$$\theta^{(t+1)} = \theta^{(t)} + \mathcal{I}^{-1}(\theta^{(t)})\nabla\ell(\theta^{(t)}) \tag{4.27}$$

   where, $\nabla\ell(\theta)$ is the score function (gradient of the log-likelihood) and $\mathcal{I}(\theta) = \mathbb{E}[-\nabla^2\ell(\theta)]$ is the Fisher information matrix.

   It is a variant of Newton-Raphson where the expected information replaces the observed Hessian.

4.3 The Linear Regression Model

Linear regression models the relationship between a dependent variable and one or more independent variables, assuming a linear relationship between them. This means the dependent variable is expressed as a weighted sum of the independent variables.

The model is given by

$$Y = X\beta + \epsilon \tag{4.28}$$

where Y is an $n \times 1$ vector of the dependent variable, X is an $n \times k$ matrix of independent variables (with a column of ones for the intercept), β is a $k \times 1$ vector of coefficients, and ϵ is an $n \times 1$ vector of error terms.

For an individual observation i, the model is written as:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_{k-1} X_{i(k-1)} + \epsilon_i \tag{4.29}$$

where β_0 is the intercept, and β_j represents the effect of each independent variable X_{ij} on Y_i.

The estimation of parameters in linear regression is done by mainly OLS (4.3.1), 2SLS (4.3.2) AND 3SLS (4.3.3).

4.3.1 Ordinary Least Squares (OLS)

Ordinary Least Squares (OLS) is a fundamental method for estimating the parameters of a linear regression model. It aims to find the best-fitting line by minimizing the sum of squared differences between the observed values and the predicted values.

Given the linear regression model:

$$Y = X\beta + \epsilon \tag{4.30}$$

Assumptions

For OLS to provide unbiased and efficient estimates, the following Assumptions must hold:

- **Independence:** Observations are independently and identically distributed (i.i.d.).
- **Homoscedasticity:** The error terms have constant variance.
- **Normality of Errors:** The error terms ϵ follow a normal distribution.

OLS estimates β such that the residuals $\epsilon = Y - X\beta$ are minimized by minimizing the sum of squared residuals (SSR):

$$SSR(\beta) = \|Y - X\beta\|^2 = (Y - X\beta)'(Y - X\beta) \tag{4.31}$$

Expanding the quadratic form:

$$SSR(\beta) = Y'Y - 2\beta'X'Y + \beta'X'X\beta \tag{4.32}$$

To find the minimizer $\hat{\beta}$, we compute the first-order condition:

$$\frac{\partial}{\partial\beta}SSR(\beta) = -2X'Y + 2X'X\beta \tag{4.33}$$

Setting the gradient to zero:

$$-2X'Y + 2X'X\beta = 0 \tag{4.34}$$

$$\implies X'X\beta = X'Y \tag{4.35}$$

If $X'X$ is invertible, the unique solution for β is:

$$\hat{\beta} = (X'X)^{-1}X'Y \tag{4.36}$$

The invertibility of $X'X$ is ensured if X has full column rank ($\text{rank}(X) = k$), which guarantees a unique solution.

Limitations

OLS has several limitations that can affect its validity as follows

- **Multicollinearity:** High correlation among independent variables inflates standard errors.
- **Endogeneity:** Correlation between independent variables and errors causes bias.
- **System of Equations:** OLS is unsuitable for simultaneous equation models.
- **Collinear Errors Across Equations:** OLS fails when error terms are correlated across equations.

4.3.2 Two-Stage Least Squares (2SLS)

The Two-Stage Least Squares (2SLS) method is used to estimate the parameters of a structural equation where one or more explanatory variables are endogenous. Endogeneity arises when an explanatory variable is correlated with the error term, leading to biased and inconsistent estimates under Ordinary Least Squares (OLS).

To address this, 2SLS replaces the endogenous variable with an exogenous component obtained through an instrumental variable (IV). The instrumental variable must be correlated with the endogenous regressor and uncorrelated with the error term to satisfy relevance and exogeneity conditions. The procedure consists of two stages as follows.

Stage 1: Instrumental Projection (First-Stage Regression)

Given the structural equation:

$$Y = X\beta + \epsilon, \tag{4.37}$$

where X is endogenous, we introduce an instrumental variable Z and regress the endogenous variable X on the instrument Z as follows:

$$\hat{X} = Z\gamma + \nu \tag{4.38}$$

where

- γ is the vector of coefficients,
- ν is the error term.

Stage 2: Structural Estimation (Second-Stage Regression)

In the second stage, the original structural equation is re-estimated using the fitted values $\hat{X}$ in place of X:

$$Y = \hat{X}\beta + \epsilon \tag{4.39}$$

Since $\hat{X}$ is constructed only from exogenous components, estimation using OLS in this step yields a consistent estimator for β.

i.e.

$$\hat{\beta} = (\hat{X}'\hat{X})^{-1}\hat{X}'Y \tag{4.40}$$

4.3.3 Three-Stage Least Squares (3SLS)

Three-Stage Least Squares (3SLS) extends Two-Stage Least Squares (2SLS) by estimating multiple equations simultaneously, correcting for endogeneity and cross-equation error correlation. It improves efficiency by applying Generalized Least Squares (GLS).

System of Equations

Consider the following system of M simultaneous equations:

$$\begin{aligned} Y_1 &= X_1\beta_1 + \epsilon_1, \\ Y_2 &= X_2\beta_2 + \epsilon_2, \\ &\vdots \\ Y_M &= X_M\beta_M + \epsilon_M. \end{aligned} \tag{4.41}$$

Each equation consists of a dependent variable Y_m, a matrix of explanatory variables X_m (including endogenous variables), a coefficient vector β_m, and an error term ϵ_m. The error terms are correlated across equations:

$$E[\epsilon_i \epsilon_j'] = \Sigma, \quad i, j = 1, 2, \ldots, M. \tag{4.42}$$

If Σ is not diagonal, OLS and 2SLS yield inefficient estimates.

Stage 1: Instrumental Variable Regression

Each endogenous variable X_m is regressed on a set of instrumental variables Z_m as

$$X_m = Z_m\gamma_m + v_m. \tag{4.43}$$

The fitted values are obtained as:

$$\hat{X}_m = P_{Z_m} X_m, \quad P_{Z_m} = Z_m(Z_m'Z_m)^{-1}Z_m'. \tag{4.44}$$

Here, $P_{Z_m} = Z_m(Z_m'Z_m)^{-1}Z_m'$ is the projection matrix for instruments Z_m.

Stage 2: Two-Stage Least Squares Estimation

Using $\hat{X}_m$ instead of X_m, each equation is estimated via 2SLS:

$$\hat{\beta}_m = (X_m'P_{Z_m}X_m)^{-1}X_m'P_{Z_m}Y_m. \tag{4.45}$$

Stage 3: Generalized Least Squares (GLS) Estimation

The system is rewritten in matrix form:

$$Y = X\beta + \epsilon, \tag{4.46}$$

where:

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_M \end{bmatrix}, \quad X = \begin{bmatrix} X_1 & 0 & \cdots & 0 \\ 0 & X_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & X_M \end{bmatrix}, \quad \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_M \end{bmatrix}. \tag{4.47}$$

Since $E[\epsilon\epsilon'] = \Sigma \otimes I_n$, the Generalized Least Squares (GLS) estimator for 3SLS is:

$$\hat{\beta}_{3SLS} = (X'(\Sigma^{-1} \otimes I_n)X)^{-1}X'(\Sigma^{-1} \otimes I_n)Y. \tag{4.48}$$

Conditions for 3SLS Validity

For 3SLS to provide efficient estimates, the following conditions must hold: - Z_m must be strongly correlated with the endogenous variables X_m (instrument relevance). - Z_m must be uncorrelated with the error terms (instrument exogeneity). - The error terms must be correlated across equations to justify GLS.

4.3.4 Tobit Model

In many statistical applications, the dependent variable may be subject to censoring, where observations are only partially observed beyond certain thresholds. For example, the exact value of a variable may be known only if it exceeds a given limit, while values below that threshold are censored at a fixed point. In such scenarios, classical linear regression techniques like Ordinary Least Squares (OLS) (see Section 4.3.1) can yield biased and inconsistent estimates due to the violation of normality and homoscedasticity

assumptions. The Tobit model, proposed by James Tobin (1958) [22], addresses this issue by assuming an underlying latent variable governed by a linear relationship, while incorporating the censoring mechanism into the observed outcome.

Model Specification

The Tobit model assumes the existence of a latent (unobserved) variable y_i^* that depends linearly on a vector of covariates $\mathbf{x}_i$ and a normally distributed error term:

$$y_i^* = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2) \tag{4.49}$$

However, we do not observe y_i^* directly. Instead, we observe the censored outcome y_i defined as:

$$y_i = \begin{cases} y_i^*, & \text{if } y_i^* > 0 \\ 0, & \text{if } y_i^* \leq 0 \end{cases} \tag{4.50}$$

This censoring can be generalized to other thresholds (left-censored, right-censored, or interval-censored), but the standard Tobit model assumes left-censoring at zero.

Estimation of Parameters

The parameters $\boldsymbol{\beta}$ and σ in the Tobit model are estimated using the method of Maximum Likelihood. The model assumes that the observed outcome y_i is censored from the latent linear model $y_i^* = \mathbf{x}_i^\top \boldsymbol{\beta} + \varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$.

The full likelihood function $\mathcal{L}(\boldsymbol{\beta}, \sigma)$ accounts for both censored and uncensored observations. Let $\phi(\cdot)$ and $\Phi(\cdot)$ denote the standard normal density and cumulative distribution function, respectively. Then the likelihood function is:

$$\mathcal{L}(\boldsymbol{\beta}, \sigma) = \prod_{i \in \mathcal{U}} \left[\frac{1}{\sigma} \phi\left(\frac{y_i - \mathbf{x}_i^\top \boldsymbol{\beta}}{\sigma} \right) \right] \cdot \prod_{i \in \mathcal{C}} \left[\Phi\left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta}}{\sigma} \right) \right] \tag{4.51}$$

Taking the logarithm of the likelihood yields the log-likelihood function $\ell(\boldsymbol{\beta}, \sigma)$:

$$\ell(\boldsymbol{\beta}, \sigma) = \sum_{i \in \mathcal{U}} \log \left(\frac{1}{\sigma} \phi \left(\frac{y_i - \mathbf{x}_i^\top \boldsymbol{\beta}}{\sigma} \right) \right) + \sum_{i \in \mathcal{C}} \log \left(\Phi \left(\frac{\mathbf{x}_i^\top \boldsymbol{\beta}}{\sigma} \right) \right) \tag{4.52}$$

where $\mathcal{U}$ is the set of uncensored observations, i.e., $y_i > 0$, and $\mathcal{C}$ is the set of censored observations, i.e., $y_i = 0$.

Maximizing this log-likelihood function yields the maximum likelihood estimators of $\boldsymbol{\beta}$ and σ. Due to the presence of the nonlinear cumulative distribution term $\Phi(\cdot)$, closed-form solutions are not available. Therefore, numerical optimization methods such as the Newton-Raphson algorithm or Fisher Scoring are employed to obtain consistent and efficient parameter estimates.

4.4 Vector Autoregressive (VAR) Model

The Vector Autoregressive (VAR) model is a multivariate generalization of the univariate autoregressive (AR) model. It captures the linear interdependencies among multiple time series variables.

Let $\{\mathbf{Y}_t\}_{t=1}^T$ be a k-dimensional vector-valued time series, i.e.,

$$\mathbf{Y}_t = \begin{bmatrix} Y_{1t} \\ Y_{2t} \\ \vdots \\ Y_{kt} \end{bmatrix}, \quad t = 1, 2, \ldots, T$$

A VAR(p) model of order p is defined as:

$$\mathbf{Y}_t = A_1 \mathbf{Y}_{t-1} + A_2 \mathbf{Y}_{t-2} + \cdots + A_p \mathbf{Y}_{t-p} + \boldsymbol{\epsilon}_t \tag{4.53}$$

where $A_i \in \mathbb{R}^{k \times k}$ are coefficient matrices for $i = 1, \ldots, p$, $\boldsymbol{\epsilon}_t \in \mathbb{R}^k$ is a white noise process: $\mathbb{E}[\boldsymbol{\epsilon}_t] = 0$, $\mathbb{E}[\boldsymbol{\epsilon}_t \boldsymbol{\epsilon}_s^\top] = \Sigma \delta_{ts}$, Σ is a positive definite

covariance matrix, and δ_{ts} is the Kronecker delta.

In matrix lag notation, the model can be compactly expressed as:

$$\mathbf{Y}_t = \sum_{i=1}^{p} A_i \mathbf{Y}_{t-i} + \boldsymbol{\epsilon}_t \tag{4.54}$$

The VAR model assumes that all components of $\mathbf{Y}_t$ are endogenous and influenced by their own and each other's past values.

4.5 K-Nearest Neighbors: Classification

Let the training dataset be

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}, \quad \mathbf{x}_i \in \mathbb{R}^p, \quad y_i \in \mathcal{Y} = \{1, 2, \ldots, C\},$$

where each $\mathbf{x}_i$ is a feature vector and y_i is a class label.

Given a new query point $\mathbf{x} \in \mathbb{R}^p$, the K-Nearest Neighbors (KNN) classifier identifies the set $\mathcal{N}_K(\mathbf{x}) \subseteq \{1, \ldots, n\}$ of indices corresponding to the K training points closest to $\mathbf{x}$ under a chosen distance metric, typically the Euclidean distance:

$$d(\mathbf{x}, \mathbf{x}_i) = \|\mathbf{x} - \mathbf{x}_i\|_2 = \sqrt{\sum_{j=1}^{p} (x_j - x_{ij})^2}.$$

Let $n_y^{(K)}(\mathbf{x})$ denote the number of neighbors among $\mathcal{N}_K(\mathbf{x})$ that belong to class y. The KNN classifier assigns to $\mathbf{x}$ the class with the highest local frequency:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} \frac{1}{K} \sum_{i \in \mathcal{N}_K(\mathbf{x})} \mathbb{I}(y_i = y),$$

where $\mathbb{I}(\cdot)$ is the indicator function.

This method estimates the class-posterior probability as:

$$\widehat{\mathbb{P}}(Y = y \mid \mathbf{x}) = \frac{1}{K} \sum_{i \in \mathcal{N}_K(\mathbf{x})} \mathbb{I}(y_i = y).$$

KNN is a non-parametric method. Its decision boundary is determined entirely by the geometry of the training data and the distance function used. For $K = 1$, the decision boundary corresponds to the Voronoi tessellation of the training data.

The choice of K controls the bias-variance trade-off. Smaller values of K lead to lower bias and higher variance, while larger K values increase bias and reduce variance.

4.6 Artificial Neural Network

4.6.1 Introduction

In recent decades, the complexity and dimensionality of real-world data have increased substantially, leading to challenges in accurately modeling and predicting such data using traditional statistical techniques. Conventional parametric models like linear regression, logistic regression, and decision trees often fail to capture intricate non-linear relationships, particularly when the underlying data distribution is unknown or highly complex [23]. To address these limitations, the concept of Artificial Neural Networks (ANNs) has emerged as a powerful and flexible class of models capable of approximating complex, non-linear functions.

The conceptual basis of neural networks originates from the functioning of biological neural systems, specifically the human brain, as first introduced by McCulloch and Pitts (1943) [24]. A biological neuron receives multiple electrical signals through dendrites, processes them within the cell body, and generates an output signal transmitted via the axon. This process of signal aggregation, thresholding, and transmission has been mathematically abstracted into the structure of artificial neurons. An artificial neuron takes weighted inputs, sums them, adds a bias, and applies a non-linear activation function to produce an output [25]. As shown in Figure 4.2, the structure of a biological neuron can be conceptually mapped to an artificial neuron in neural networks.

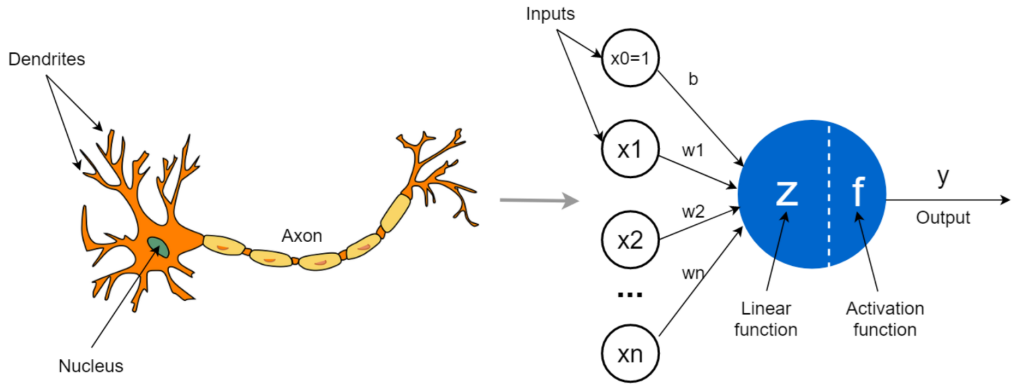


Fig. 4.2: Biological vs. Artificial Neuron: A Comparison

4.6.2 Structure of Neural Network

A neural network is composed of an interconnected set of computational units, commonly referred to as artificial neurons or nodes, systematically arranged in multiple layers. These layers are typically organized into three categories: an input layer, one or more hidden layers, and an output layer [23].

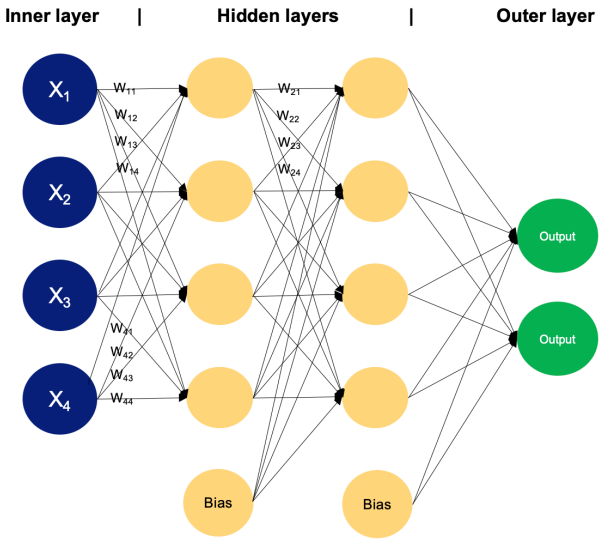


Fig. 4.3: Multi-Layer Perceptron (MLP)

Figure 4.3 illustrates a fully connected feedforward neural network, where

each neuron in one layer is connected to every neuron in the subsequent layer, including bias nodes. The input layer serves the purpose of receiving the feature vector corresponding to an observation. This feature vector contains the numerical representation of the input data and is transmitted to the hidden layers for further processing. The hidden layers consist of a collection of neurons that perform non-linear transformations on the received inputs, thereby enabling the network to capture complex and non-linear relationships inherent in the data. The final layer, referred to as the output layer, produces the predicted output based on the transformations applied by the preceding layers.

Each neuron within the network executes a two-step computational process [25]:

1. **Aggregation:** The neuron computes a weighted summation of its input signals. Mathematically, this involves multiplying each input by an associated weight and adding a bias term.

2. **Activation:** The aggregated sum is passed through a non-linear activation function(e.g., ReLU, Sigmoid, Tanh) (see Figure 4.4). This step introduces the capacity to model complex, non-linear patterns that cannot be captured by linear models.

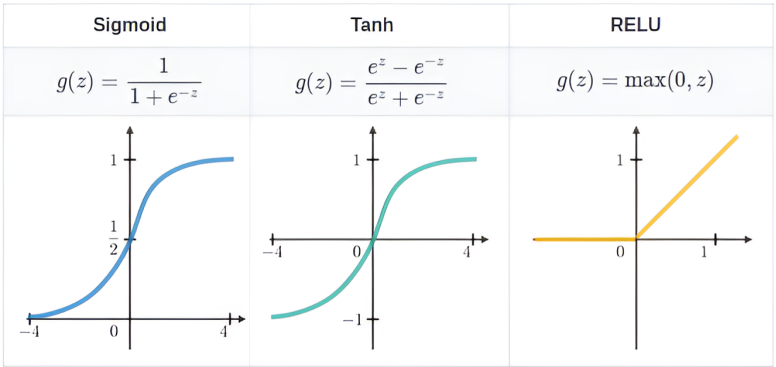


Fig. 4.4: Plot of commonly used activation functions

By systematically combining these operations across multiple layers, neural networks are capable of learning intricate mappings between input and

output spaces. The hierarchical structure of transformations allows the network to learn low-level to high-level feature representations in a data-driven manner, eliminating the need for manual feature engineering.

4.6.3 Mathematics of Neural Network

Mathematically, an Artificial Neural Network (ANN) is a parametric, non-linear model designed to approximate an unknown function

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

such that, for a given input vector $\mathbf{x} \in \mathbb{R}^n$, the network produces an output $\hat{\mathbf{y}} \in \mathbb{R}^m$ that closely estimates the true output $\mathbf{y} \in \mathbb{R}^m$.

The network consists of multiple layers of neurons, where each layer performs an affine transformation followed by a non-linear activation. Specifically, the neural network defines a composite function of the form:

$$\hat{\mathbf{y}} = F(\mathbf{x}; \boldsymbol{\theta}) = f^{[L]} \circ f^{[L-1]} \circ \cdots \circ f^{[1]}(\mathbf{x}) \tag{4.55}$$

where L is the total number of layers (excluding the input layer), $f^{[l]}$ represents the transformation at layer l, and $\boldsymbol{\theta} = \{\mathbf{W}^{[l]}, \mathbf{b}^{[l]}\}_{l=1}^{L}$ is the complete set of **trainable parameters**, consisting of weight matrices $\mathbf{W}^{[l]}$ and bias vectors $\mathbf{b}^{[l]}$.

Each transformation in layer l is defined as:

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}, \quad \mathbf{a}^{[l]} = \phi^{[l]}(\mathbf{z}^{[l]})$$

where $\mathbf{a}^{[0]} = \mathbf{x}$ is the input vector, $\mathbf{a}^{[l]}$ is the output of layer l (also known as activation), and $\phi^{[l]}(\cdot)$ is a non-linear activation function (e.g., ReLU, Sigmoid, Tanh).

This mathematical structure enables the network to approximate non-linear decision boundaries efficiently [26].

Theorem 4.6.1 (Universal Approximation Theorem)**.** *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a non-constant, bounded, and continuous activation function. Then, for any*

continuous function $f : \mathbb{R}^n \to \mathbb{R}$ defined on a compact subset $K \subset \mathbb{R}^n$, and for any $\varepsilon > 0$, there exists a feedforward neural network with a single hidden layer, a finite number of neurons, and activation function σ, such that the network $\hat{f}$ satisfies

$$\sup_{x \in K} \left| f(x) - \hat{f}(x) \right| < \varepsilon.$$

That is, $\hat{f}$ uniformly approximates f on K to arbitrary precision.

$$\forall\, \varepsilon > 0,\ \exists\, \hat{f}(x) \text{ such that } |f(x) - \hat{f}(x)| < \varepsilon \quad \forall x \in \Omega$$

where $\hat{f}$ is represented by a neural network and $\Omega \subset \mathbb{R}^n$ is compact.

This result highlights that neural networks are universal function approximators, justifying their effectiveness in modeling complex, non-linear systems [27].

The learning objective of the neural network is to find an optimal set of parameters $\boldsymbol{\theta}^*$ such that the expected loss over the dataset is minimized:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \ \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(F(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)}\right) \tag{4.56}$$

where $\mathcal{L}(\cdot, \cdot)$ is a loss function that quantifies the error between predicted and actual outputs (e.g., Mean Squared Error or Cross-Entropy Loss), and N is the total number of training samples.

Thus, a neural network can be formally characterized as a high-capacity, parameterized function that learns complex mappings from input space to output space by optimizing its parameters to minimize the empirical risk over the observed data.

4.6.4 Training Neural Networks: Backpropagation and Gradient Descent

To train a neural network, the goal is to find parameters $\boldsymbol{\theta} = \{\mathbf{W}^{[l]}, \mathbf{b}^{[l]}\}$ that minimize a chosen loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$, which measures the difference between predicted output $\hat{\mathbf{y}}$ and true label $\mathbf{y}$.

This optimization is performed using Gradient Descent (see Section 3.3, where each parameter is updated as:

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial \mathcal{L}}{\partial \theta}$$

Here, η is the learning rate, and $\frac{\partial \mathcal{L}}{\partial \theta}$ is the gradient of the loss with respect to the parameter θ.

Backpropagation Algorithm

Backpropagation is the core algorithm used to train artificial neural networks by efficiently computing gradients of the loss function with respect to the network's parameters. Introduced by Rumelhart, Hinton, and Williams (1986) [28], it applies the chain rule of calculus in a recursive manner to propagate the error backward through the layers of the network.

Given a loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$, where $\hat{\mathbf{y}}$ is the predicted output and $\mathbf{y}$ is the ground truth, the goal is to update parameters $\theta \in \{\mathbf{W}^{[l]}, \mathbf{b}^{[l]}\}$ such that:

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial \mathcal{L}}{\partial \theta}$$

Here, η is the learning rate. The backpropagation algorithm computes $\frac{\partial \mathcal{L}}{\partial \theta}$ for all weights and biases using the following steps:

1. **Forward Pass:** Compute activations $\mathbf{a}^{[l]}$ for each layer using:

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}, \quad \mathbf{a}^{[l]} = \phi(\mathbf{z}^{[l]})$$

2. **Compute Output Error:** For the output layer L, compute:

$$\delta^{[L]} = \nabla_{\hat{\mathbf{y}}} \mathcal{L} \odot \phi'(\mathbf{z}^{[L]})$$

3. **Backpropagate Error:** For all hidden layers $l = L-1, \ldots, 1$:

$$\delta^{[l]} = \left(\mathbf{W}^{[l+1]^\top} \delta^{[l+1]}\right) \odot \phi'(\mathbf{z}^{[l]})$$

4. **Compute Gradients:**

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} = \delta^{[l]} \mathbf{a}^{[l-1]\top}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} = \delta^{[l]}$$

This process allows for efficient training of deep networks by reusing intermediate computations. Backpropagation is typically used with stochastic or mini-batch gradient descent to iteratively update network parameters and minimize the loss over the dataset [23].

4.7 Survival Analysis and Hazard Models

Survival analysis is a statistical approach used to analyze time-to-event data, where the primary focus is on estimating the time until an event of interest occurs. It is widely applied in fields such as medicine, customer churn, engineering, and social sciences to study outcomes like patient survival time, time when customer is likely to churn, machine failure rates, and employment durations. A distinctive feature of survival analysis is its ability to handle censored data, where the exact time of the event is unknown for some subjects. Key components include the survival function, which estimates the probability of surviving beyond a given time, and the hazard function, which measures the instantaneous risk of the event occurring. Advanced methods like the Kaplan-Meier estimator and the Cox proportional hazards model allow researchers to analyze survival data effectively while incorporating explanatory variables. By addressing challenges like censoring and time-dependent risks, survival analysis provides valuable insights into time-to-event processes.

4.7.1 Key Components of Survival Analysis

Survival analysis relies on fundamental mathematical functions that describe the distribution of time-to-event data while handling censored observations. The three primary components are the Survival Function $S(t)$, the Hazard

Function $h(t)$, and the treatment of Censoring. These functions allow for accurate estimation of survival probabilities, hazard rates, and risk assessment.

1. **Survival Function ($S(t)$)**
   The survival function represents the probability that an event has not yet occurred by time t, and is formally defined as:

$$S(t) = P(T > t), \tag{4.57}$$

   where T is a non-negative random variable denoting the time-to-event. The survival function is related to the cumulative distribution function (CDF) $F(t)$ of T:

$$S(t) = 1 - F(t) = 1 - P(T \leq t) = \int_t^\infty f(u)\, du, \tag{4.58}$$

   where $f(t)$ is the probability density function (PDF) of T. The survival function satisfies:

$$S(0) = 1, \quad \lim_{t\to\infty} S(t) = 0, \quad \text{and} \quad S(t) \text{ is non-increasing.} \tag{4.59}$$

2. **Hazard Function ($h(t)$)**
   The hazard function quantifies the instantaneous risk of an event occurring at time t, given survival up to t. It is defined as:

$$h(t) = \lim_{\Delta t\to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t}. \tag{4.60}$$

   Rewriting in terms of the PDF and survival function:

$$h(t) = \frac{f(t)}{S(t)} = \frac{f(t)}{1 - F(t)}. \tag{4.61}$$

   The cumulative hazard function $H(t)$ is:

$$H(t) = \int_0^t h(u)\, du, \tag{4.62}$$

which gives the fundamental relation:

$$S(t) = e^{-H(t)}. \tag{4.63}$$

This relationship highlights that survival probability decreases as cumulative hazard increases.

3. **Censoring**
   Censoring occurs when the exact event time is unknown for some subjects. The three main types are:

   - **Right Censoring**: The event has not occurred by the study's end ($T > C$).
   - **Left Censoring**: The event occurred before the subject entered the study ($T < C$).
   - **Interval Censoring**: The event occurred within a known time range (T_{i1}, T_{i2}).

4.8 Hazard Models in Survival Analysis

In this section, we discuss the mathematical formulation of hazard models used in survival analysis. We distinguish models based on the nature of the time variable: continuous and discrete. The classification of models into parametric, semi-parametric, or non-parametric categories is mentioned within each model description, but the primary structure is based on time measurement.

4.8.1 Continuous-Time Hazard Models

In continuous-time survival analysis, the time-to-event variable T is treated as a continuous random variable, and the analysis focuses on estimating the instantaneous risk of event occurrence at each point in time. This framework is particularly useful in applications such as modeling customer churn, equip-

ment failure, or patient survival, where time is recorded with fine granularity (e.g., days, months, years).

Hazard Function and Basic Definitions

The fundamental quantity in continuous-time survival analysis is the hazard function, which captures the instantaneous rate of occurrence of the event at time t, conditional on survival up to that time:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t \mid T \geq t)}{\Delta t}. \tag{4.64}$$

This function can be interpreted as the conditional failure rate at time t, and is linked to the probability density function $f(t)$ and survival function $S(t)$ through the relationship:

$$h(t) = \frac{f(t)}{S(t)}. \tag{4.65}$$

The cumulative hazard function is defined as:

$$H(t) = \int_0^t h(u)\, du, \tag{4.66}$$

and satisfies the key identity:

$$S(t) = \exp(-H(t)), \tag{4.67}$$

which illustrates how survival probability decays exponentially as cumulative hazard increases. These expressions form the backbone of all continuous-time hazard modeling approaches [29, 30].

Parametric Models

Parametric survival models assume a specific functional form for the hazard or survival function, leading to tractable likelihoods and interpretable parameters. Below, we describe two commonly used parametric models: the Exponential and Weibull models.

Exponential Hazard Model The Exponential model assumes a constant hazard rate over time:

$$h(t \mid \mathbf{x}) = \lambda, \tag{4.68}$$

where $\lambda > 0$ is the constant hazard. The corresponding survival and density functions are:

$$S(t \mid \mathbf{x}) = \exp(-\lambda t), \quad f(t \mid \mathbf{x}) = \lambda \exp(-\lambda t). \tag{4.69}$$

This model implies a memoryless process and is often used as a baseline in comparison with more flexible models.

Weibull Hazard Model The Weibull model generalizes the exponential by introducing a shape parameter p that controls how the hazard evolves over time:

$$h(t \mid \mathbf{x}) = \lambda p t^{p-1}, \quad \lambda > 0,\ p > 0. \tag{4.70}$$

The survival and density functions are given by:

$$S(t \mid \mathbf{x}) = \exp(-\lambda t^p), \quad f(t \mid \mathbf{x}) = \lambda p t^{p-1} \exp(-\lambda t^p). \tag{4.71}$$

Interpretation of shape parameter p, For $p > 1$, the hazard increases with time (aging or deterioration); for $p = 1$, the hazard is constant (reduces to exponential); and for $p < 1$, the hazard decreases with time (infant mortality or early attrition).

This flexibility makes the Weibull model suitable for a wide range of empirical scenarios in business and healthcare [31, 32].

Semi-Parametric Model: Cox Proportional Hazards

The Cox Proportional Hazards (PH) model [33] is a semi-parametric model that specifies the hazard as a product of a baseline hazard and a covariate effect

$$h(t \mid \mathbf{x}) = h_0(t) \exp(\boldsymbol{\beta}^\top \mathbf{x}), \tag{4.72}$$

Where $h_0(t)$ is the unspecified baseline hazard function, $\mathbf{x}$ is a vector of

covariates, and $\boldsymbol{\beta}$ represents their associated log hazard ratios.

The survival function is:

$$S(t \mid \mathbf{x}) = [S_0(t)]^{\exp(\boldsymbol{\beta}^\top \mathbf{x})}, \tag{4.73}$$

where $S_0(t) = \exp\left(-\int_0^t h_0(u)\,du\right)$ is the baseline survival.

Rather than estimating the full likelihood, the Cox model relies on a partial likelihood:

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{i:\delta_i=1} \frac{\exp(\boldsymbol{\beta}^\top \mathbf{x})}{\sum_{j \in R(t_i)} \exp(\boldsymbol{\beta}^\top \mathbf{x})}, \tag{4.74}$$

where $R(t_i)$ is the risk set at time t_i, and δ_i indicates if the event was observed. This approach makes minimal assumptions on $h_0(t)$, allowing greater model robustness and interpretability.

4.8.2 Discrete-Time Hazard Models

In discrete-time survival analysis, the time-to-event variable T is measured in discrete intervals (e.g., weeks, months, quarters). This setting is especially suitable when event times are recorded in grouped formats or when decisions (such as purchases, exits, or renewals) are made periodically.

Hazard Function and Survival in Discrete Time

Let $T \in \{1, 2, 3, \dots\}$ denote the time of event occurrence, and define the discrete-time hazard function as the conditional probability of event occurrence in interval t, given survival up to t:

$$h(t) = P(T = t \mid T \geq t). \tag{4.75}$$

Given the hazard function $h(t)$, the survival function $S(t)$ is:

$$S(t) = P(T \geq t) = \prod_{j=1}^{t-1} [1 - h(j)]. \tag{4.76}$$

The probability mass function (PMF) is:

$$P(T = t) = h(t) \prod_{j=1}^{t-1} [1 - h(j)]. \tag{4.77}$$

Modeling with Link Functions

In discrete-time models, the hazard function is typically modeled using a binary response formulation:

$$Y_{it} \in \{0, 1\}, \quad \text{where } Y_{it} = 1 \text{ if the event occurs at time } t \text{ for unit } i.$$

The probability of event occurrence at time t is linked to covariates $\mathbf{x}_{it}$ via a suitable link function:

$$h_{it} = P(Y_{it} = 1 \mid Y_{i,t-1} = 0, \mathbf{x}_{it}) = g^{-1}(\alpha_t + \mathbf{x}_{it}^\top \boldsymbol{\beta}), \tag{4.78}$$

where α_t is the baseline hazard for time t, $g^{-1}(\cdot)$ is an inverse link function.

Common link functions:

- **Logit link:**
$$\log\left(\frac{h_{it}}{1 - h_{it}}\right) = \alpha_t + \mathbf{x}_{it}^\top \boldsymbol{\beta}$$
- **Complementary log-log (cloglog) link:**
$$\log[-\log(1 - h_{it})] = \alpha_t + \mathbf{x}_{it}^\top \boldsymbol{\beta}$$

The cloglog link arises naturally from discretizing a continuous-time proportional hazards model with constant hazard within intervals [34].

Likelihood and Estimation

Let $y_{it} \in \{0, 1\}$ be the indicator of event occurrence in interval t. The conditional likelihood for individual i across T_i intervals is:

$$\mathcal{L}_i = \prod_{t=1}^{T_i} h_{it}^{y_{it}} (1 - h_{it})^{1-y_{it}}, \tag{4.79}$$

and the log-likelihood over n individuals is:

$$\log \mathcal{L} = \sum_{i=1}^{n} \sum_{t=1}^{T_i} \left[y_{it} \log(h_{it}) + (1 - y_{it}) \log(1 - h_{it}) \right]. \tag{4.80}$$

The parameters $\boldsymbol{\beta}$ and α_t can be estimated using standard binary response model techniques such as maximum likelihood estimation (MLE) (see Section 4.2.4).

Applications and Interpretation

Discrete-time hazard models are widely used in situations where:

- The event timing is only known up to an interval (e.g., monthly observations),
- The decision process is inherently periodic (e.g., renewal, review),
- Time-varying covariates are to be included flexibly,
- The proportional hazards assumption of continuous-time models is unsuitable.

Each coefficient β_j corresponds to the effect of covariate x_j on the log-odds or log-hazard of experiencing the event in a given period. This allows for interpretability in terms of odds ratios (logit) or hazard ratios (cloglog), depending on the link function used [35].

5. APPLICATIONS IN CRM

5.1 Introduction

In the previous chapters, we established the strategic shift from product-centric to customer-centric marketing and introduced the mathematical foundations necessary for quantitative modeling. With this groundwork in place, we now move toward the practical implementation of CRM strategies using statistical and machine learning models.

This chapter focuses on the four central objectives of Customer Relationship Management (CRM) that are Customer Acquisition, Customer Retention, Customer Churn Prediction and Customer Win-Back.

Each of these objectives is operationalized using the models and techniques discussed in Chapters 3 and 4. We apply these models to interpret customer behavior, make predictions, and support data-driven marketing decisions. Where applicable, we also highlight managerial insights and business implications, demonstrating how mathematical rigor translates into actionable CRM strategies.

5.2 What is Customer Relationship Management (CRM)?

Customer Relationship Management (CRM) refers to the strategic and analytical process by which firms acquire, retain, and develop relationships with individual customers to improve profitability and customer satisfaction. According to Kumar and Petersen [1], CRM involves the systematic collection and analysis of customer-level data and the use of that information to drive

marketing decisions. Unlike traditional marketing approaches that operate at the segment or aggregate level, CRM focuses on the micro-level—treating each customer as a distinct unit of analysis.

The essence of CRM lies in its ability to manage relationships across the entire customer lifecycle, which includes four key processes:

1. **Customer Acquisition**: Identifying and attracting valuable prospects.
2. **Customer Retention**: Encouraging continued engagement through satisfaction and loyalty.
3. **Customer Churn Management**: Predicting and mitigating the risk of customer attrition.
4. **Customer Win-Back**: Re-engaging customers who have previously churned.

CRM systems utilize customer data to answer critical questions such as: Who are the most profitable customers? Which customers are likely to churn? What interventions can prevent defection or improve engagement? These decisions are supported by analytical models that estimate probabilities of customer behavior, predict lifetime value, and quantify marketing ROI.

Kumar and Petersen emphasize that effective CRM strategies require going beyond intuition and managerial heuristics. Instead, they must rely on customer-level metrics like Customer Lifetime Value (CLV), retention probability, and share of wallet to guide decision-making [1]. This model-driven approach ensures that marketing resources are allocated efficiently, and customers are managed as assets, with the objective of maximizing long-term firm value.

In summary, CRM is not simply a technological solution but a firm-wide strategic orientation that integrates data, analytics, and marketing science to optimize the value of customer relationships over time.

5.3 Goals of CRM and Requirements for Implementation

The primary goal of Customer Relationship Management (CRM) is to optimize the long-term value generated from customer relationships. CRM enables firms to build a structured approach to managing customers across their lifecycle — from acquisition to retention and eventual re-engagement. According to Buttle and Maklan [36], the essence of CRM lies in treating customers not as isolated transactions, but as long-term assets that require strategic investment and management.

The core goals of CRM include:

- **Customer Acquisition:** Identify and attract high-potential customers using data-driven targeting and personalized offers.
- **Customer Retention:** Enhance customer satisfaction, loyalty, and repeat purchase by delivering consistent value and proactive service.
- **Churn Management:** Predict which customers are at risk of defection and intervene through tailored offers or engagement strategies.
- **Customer Win-Back:** Reacquire previously lost customers who still hold high future value.

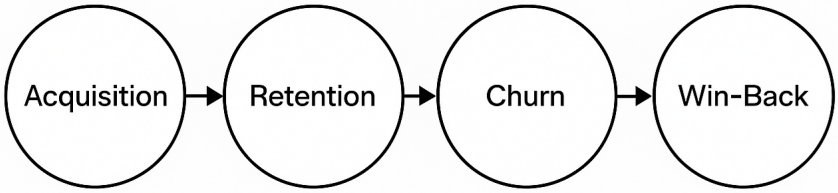


Fig. 5.1: The four key stages in Customer Relationship Management

Rather than maximizing any single metric (e.g., retention rate), CRM aims to maximize *customer equity* — the total combined value of a firm's customer base. Harvard Business Review notes that "not all customers are created equal," and CRM provides the framework to allocate resources based on predicted profitability [37].

5.3.1 What is Needed to Implement CRM Strategies?

The implementation of CRM strategies depends on three interconnected pillars: data, technology, and metrics.

- **Customer-Level Data:** High-quality, granular data is the foundation of CRM. This includes transactional history, behavioral signals, service interactions, and demographic attributes. OpenStax [4] emphasizes that collecting consistent and relevant customer data enables marketers to understand preferences, anticipate needs, and deliver personalized experiences.
- **Technology Infrastructure:** CRM systems such as Salesforce, HubSpot, or Microsoft Dynamics provide the platforms to store, access, and analyze customer data. These tools often include automation features for sales, service, and marketing, as well as integration with analytics engines that support real-time decision-making [38].
- **Customer-Centric Metrics:** Effective CRM decisions are guided by key performance indicators like Customer Lifetime Value (CLV), Net Promoter Score (NPS), Retention Rate, and Share of Wallet. These metrics enable firms to quantify the value of customers, prioritize investment, and monitor CRM outcomes over time.

As Reinartz, Krafft, and Hoyer [39] argue, CRM success is not about collecting massive volumes of data but rather about extracting actionable insights through analytical modeling. A firm must be capable of translating raw customer data into predictive models that inform acquisition, retention, and loyalty-building strategies. When supported by the right tools, metrics, and organizational alignment, CRM enables firms to move from reactive engagement to predictive, value-driven marketing.

With the strategic CRM framework in place, we now focus on the application of mathematical models to each key objective. The following sections present a model-driven approach to customer acquisition, retention, churn

prediction, and win-back. Each objective is addressed using appropriate statistical or machine learning techniques that enable precise, data-informed marketing decisions.

5.4 Customer Acquisition

Customer acquisition is the foundational step in the CRM lifecycle. It refers to the process of attracting and converting prospects into paying customers. While acquisition is essential for business growth, its high cost demands a strategic, data-driven approach that ensures profitability over the long term. As highlighted by Villanueva et al. [40], acquiring customers with higher potential lifetime value (LTV) is more beneficial than maximizing acquisition volume alone. Hansotia and Wang [41] emphasize that acquisition modeling should consider not just the probability of conversion but also expected revenue and cost components.

5.4.1 Importance of Customer Acquisition

Customer acquisition plays a pivotal role in shaping a firm's growth trajectory. It is not merely a marketing function but a strategic imperative that determines a firm's ability to expand its customer base, diversify revenue sources, and increase market share. Effective acquisition strategies help firms offset natural attrition and churn, thereby sustaining a healthy customer pipeline.

From a financial standpoint, acquiring profitable customers early in their lifecycle can significantly enhance long-term Customer Lifetime Value (CLV). High-quality acquisition also enables the formation of robust CRM strategies by ensuring that retained customers are both valuable and strategically aligned with the firm's offerings. Moreover, customer acquisition is essential during product launches, market expansions, and when targeting new segments—serving as a lever for top-line growth. Thus, customer acquisition is not just a tactical requirement but a critical element in achieving sustainable competitive advantage.

5.4.2 What We Do in Customer Acquisition

The process of customer acquisition encompasses a sequence of structured activities aimed at identifying, engaging, and converting potential customers. Although modeling-based estimations are addressed in subsequent sections, here we outline the strategic and operational activities typically undertaken:

- **Segmentation and Targeting:** Firms begin by identifying potential customer groups based on demographics, psychographics, behaviors, and needs. Data-driven segmentation allows for precise targeting.
- **Value Proposition and Positioning:** Tailored messaging is developed to align with each segment's specific expectations. This includes articulating the product's functional and emotional benefits.
- **Channel Strategy:** Organizations choose appropriate acquisition channels—digital (search, social media), offline (retail, events), or hybrid—based on reach, cost, and alignment with customer preferences.
- **Conversion and Incentivization:** Tactics such as free trials, discounts, referral rewards, and bundling are employed to lower the barrier to first purchase and maximize initial conversion.
- **Measurement and Optimization:** Key performance indicators (KPIs) such as Cost Per Acquisition (CPA), early-stage CLV, and conversion rate are continuously monitored. Insights from A/B testing, clickstream data, and predictive modeling inform real-time optimizations.

These actions collectively form the acquisition funnel—from awareness generation to purchase—and their integration within a firm's marketing architecture ensures that the process is both scalable and economically viable. As illustrated in Figure 5.2, we employ a range of quantitative models to analyze and support these activities. In following section we will learn each of the model in detail with their mathematical background along with that we will also see their applications in the retention modeling and the different methodologies we used to estimated the parameters of the models.

Model Used	Research Interest	Estimation Technique
Logistic/ Probit Regression	Estimating probability of acquisition (binary response)	Maximum Likelihood Estimation (MLE)
Linear Regression	Estimating number of new customers acquired in campaigns	Ordinary Least Squares (OLS)
Linear Regression	Joint modeling of acquisition & order size with endogeneity	Three-Stage Least Squares (3SLS)
Vector Autoregressive (VAR) Model	Dynamic interdependencies in acquisition and performance over time	Lag estimation + Impulse Response Analysis
Tobit Model	Estimating long-term customer value under right-censoring	MLE for censored regression

Fig. 5.2: Customer acquisition models, research focus, and estimation

5.4.3 Probability of Acquisition

To estimate the probability of a customer responding to a campaign, we model acquisition as a binary outcome using logit model, also called as logistic regression (see Section 4.2.1). Let y_i denote whether customer i responds ($y_i = 1$) or not ($y_i = 0$), and x_i be the vector of explanatory variables for customer i. Following the latent utility framework:

$$y_i^* = \beta^\top x_i + \epsilon_i \tag{5.1}$$

The observed binary outcome is defined as:

$$y_i = \begin{cases} 1 & \text{if } y_i^* > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

Assuming $\epsilon_i \sim \text{Logistic}(0, \pi^2/3)$, the probability of acquisition is

$$P(y_i = 1 \mid x_i) = \frac{1}{1 + e^{-\beta^\top x_i}} \tag{5.3}$$

The coefficients β are estimated by Maximum Likelihood Estimation (MLE) (see Section 4.2.4). The likelihood function for a dataset of size

N is given by

$$L(\beta) = \prod_{i=1}^{N} P(y_i)^{y_i}(1 - P(y_i))^{(1-y_i)} \tag{5.4}$$

where $P(y_i)$ represents the probability that $Y_i = 1$, and $1 - P(y_i)$ represents the probability that $Y_i = 0$.

Taking the log-likelihood function we get

$$\ell(\beta) = \sum_{i=1}^{N} [y_i \log P(y_i) + (1 - y_i) \log(1 - P(y_i))] \tag{5.5}$$

Optimization can be performed using methods such as gradient ascent or Newton-Raphson (see Section 3.4. Since gradient descent has been discussed previously (see Section 3.3), we note here that gradient ascent is conceptually similar but moves in the direction that increases the log-likelihood rather than minimizing a loss.

If the error term ϵ_i is assumed to follow a standard normal distribution instead of a logistic one, the appropriate specification is the Probit model (see Section 4.2.2)

$$P(y_i = 1 \mid x_i) = \Phi(\beta^\top x_i) \tag{5.6}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function. The coefficients are estimated using Maximum Likelihood Estimation.

5.4.4 Number of Newly Acquired Customers

In CRM-driven campaign analysis, it is often necessary to estimate the number of customers acquired in response to specific marketing actions. According to Kumar and Petersen [1], this quantity can be modeled using a linear regression (see Section 4.3) framework where the dependent variable is the number of newly acquired customers (CAcq), and the independent variables are campaign-specific factors such as price promotions, referral incentives, shipping fees, and marketing exposure.

The general form of the regression model is

$$\text{CAcq}_i = \alpha + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \epsilon_i \tag{5.7}$$

Where CAcq_i = Number of new customers acquired in campaign i, $x_{1i}, x_{2i}, \ldots, x_{ki}$ = Campaign-level explanatory variables (e.g., base size, shipping fees, promotional offer, etc.), $\beta_1, \ldots, \beta_k$ = Coefficients measuring the marginal effect of each variable, and ϵ_i = Error term capturing unobserved influences.

This model helps quantify how each marketing lever influences acquisition volume, enabling marketers to simulate the outcome of changes in campaign design. When acquisition is modeled jointly with other outcomes (such as average order size), a system of equations and methods such as Three-Stage Least Squares (3SLS) (see Section 4.3.3) may be used to address endogeneity and improve estimation efficiency.

5.4.5 Joint Modeling

When customer acquisition and initial order size are simultaneously affected by shared variables (e.g., discounts, shipping incentives), a system of equations may be necessary but due to potential correlation between the error terms, Ordinary Least Squares (OLS) (see Section 4.3.1) applied independently to each equation may yield biased estimates. To correct for this, we employ the Three-Stage Least Squares (3SLS) method.

The simultaneous system is

$$\text{CAcq}_i = \alpha + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \epsilon_i \tag{5.8}$$

$$\text{Amount}_{\text{new},i} = \gamma_0 + \gamma_1 x_{1i} + \gamma_2 x_{2i} + \cdots + \gamma_k x_{ki} + \nu_i \tag{5.9}$$

The 3SLS approach involves

1. Estimating each equation using 2SLS (see Section 4.3.2) to handle endogeneity,
2. Estimating the covariance structure of the residuals,
3. Applying Generalized Least Squares (GLS) to the system using this structure.

This methodology, as discussed by Kumar and Petersen [1], allows for more accurate estimation of interdependent marketing outcomes, improving both predictive accuracy and decision quality.

Villanueva et al. [42] proposed a dynamic framework to analyze how different customer acquisition channels influence both the volume of new customers and firm performance over time. The model incorporates three interdependent time series: the number of customers acquired through the firm's direct marketing actions (x_t), those acquired through word-of-mouth (y_t), and a performance metric such as customer equity or revenue (z_t). Here, word-of-mouth refers to organic customer-to-customer communication that influences purchase behavior, often driven by satisfaction, loyalty, or referral incentives [1].

The relationship among these variables is modeled using a vector autoregression (VAR) (see Section 4.4) system, as follows

$$\begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix} = \begin{pmatrix} a_{10} \\ a_{20} \\ a_{30} \end{pmatrix} + \sum_{l=1}^{p} \begin{pmatrix} d_{11}^l & d_{12}^l & d_{13}^l \\ d_{21}^l & d_{22}^l & d_{23}^l \\ d_{31}^l & d_{32}^l & d_{33}^l \end{pmatrix} \begin{pmatrix} x_{t-l} \\ y_{t-l} \\ z_{t-l} \end{pmatrix} + \begin{pmatrix} \epsilon_{1t} \\ \epsilon_{2t} \\ \epsilon_{3t} \end{pmatrix}, \tag{5.10}$$

where p is the lag order of the model, and the error terms $(\epsilon_{1t}, \epsilon_{2t}, \epsilon_{3t})'$ are assumed to be white-noise disturbances distributed as $N(0, \Sigma)$. The coefficient matrix captures the directional influence across time: for instance, the impact of past marketing acquisitions (x_{t-l}) and word-of-mouth (y_{t-l}) on present firm performance (z_t), and vice versa.

Importantly, this model captures both direct and feedback effects. For example, it includes how past firm performance can influence future customer acquisition (feedback), and how acquisition channels interact with each other (cross-effects). The diagonal terms account for self-reinforcing behavior within each variable.

Impulse response functions (IRFs) are then used to estimate the dynamic response of one variable to a shock in another, allowing assessment of both short-run and long-run effects. This structure enables quantification of indirect and temporal spillovers between acquisition strategies and outcomes.

5.4.6 Firm's Performance: LTV, CLV, and CE

Customer acquisition decisions should not be based solely on the probability of response or order quantity. Firms must also evaluate the long-term profitability of acquired customers, quantified using metrics such as Lifetime Value (LTV), Customer Lifetime Value (CLV), and Customer Equity (CE). These measures estimate the net profit expected from customers over their relationship with the firm.

One of the primary challenges in estimating LTV or CLV is the presence of censored data — some customers may not have completed their purchasing cycle within the observation window. To address this, Hansotia and Wang (1997) [43] proposed using a censored regression framework, specifically the Tobit model. Let the latent performance variable y_i^* for customer i be given by:

$$y_i^* = \boldsymbol{\beta}^\top \mathbf{x}_i + u_i, \quad u_i \sim \mathcal{N}(0, \sigma^2) \tag{5.11}$$

Due to censoring at a threshold y_L, the observed outcome y_i is defined as:

$$y_i = \begin{cases} y_i^*, & \text{if } y_i^* \leq y_L \\ y_L, & \text{if } y_i^* > y_L \end{cases} \tag{5.12}$$

The log-likelihood function for M customers, where $s_i = 1$ if the observation is uncensored and 0 otherwise, is:

$$L = \prod_{i=1}^{M} \left[\Pr(y_i = \text{observed})\right]^{s_i} \left[\Pr(y_i^* > y_L)\right]^{1-s_i} \tag{5.13}$$

The Tobit model accommodates censored observations and provides consistent estimates of customer value, even when some outcomes are not fully observed.

Reinartz and Kumar (2003) [44] further extended this framework using a right-censored Tobit model to estimate both duration and profitability of B2B customers. They demonstrated that spending on acquisition and

catalog marketing significantly influences customer profitability. Similarly, vector autoregressive (VAR) (see Section 4.4) models can be employed when acquisition variables such as customer counts, order frequency, and profits exhibit temporal interdependencies. A first-order VAR model is:

$$Y_t = A\mathbf{Y}_{t-1} + \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \Sigma) \tag{5.14}$$

where Y_t represents a vector of acquisition metrics and A captures inter-variable dynamics.

For modeling the duration until customer churn, the Accelerated Failure Time (AFT) model is an effective parametric survival model:

$$\log(T_i) = \boldsymbol{\beta}^\top \mathbf{x}_i + \sigma\epsilon_i \tag{5.15}$$

where ϵ_i follows a known distribution such as exponential, Weibull, or log-normal. This model captures how covariates accelerate or decelerate the time to churn.

In this chapter, we modeled customer acquisition using both classification and duration-based frameworks. However, acquiring a customer is only the first step. Next, we turn to the challenge of retaining them, which often demands entirely different models and strategies.

5.5 Customer Retention

5.5.1 Introduction

Customer retention refers to the ability of a firm to retain its existing customers over a period of time. It is a central pillar of Customer Relationship Management (CRM) and plays a critical role in both contractual settings (e.g., telecom, banking) and non-contractual settings (e.g., retail, consumer goods) [1]. Retention is not merely about extending a customer's tenure but about fostering repeat interactions that generate long-term profitability.

While the traditional focus in marketing has been customer acquisition, increasing competition, saturation, and rising costs have shifted strategic em-

phasis toward retention. Modern firms recognize that retaining a profitable customer can yield significantly greater lifetime value than acquiring a new one.

5.5.2 Importance of Customer Retention

Retaining customers is often more cost-effective than acquiring new ones. According to empirical studies, a 5% increase in retention rate can lead to an increase in profits of 25% to 85%, depending on the industry [1]. This is due to lower acquisition costs, higher purchase frequency, and greater cross-buying potential of loyal customers.

Moreover, loyal customers are more likely to refer new customers via word-of-mouth, which refers to the process of customers sharing their experiences and recommendations with others, either directly or through social media. They also exhibit a higher willingness to pay and are more forgiving of service failures.

Conversely, customer churn—the loss of existing customers—reduces revenue predictability and inflates acquisition costs.

Figure 5.3 Flowchart illustrating key drivers and consequences of customer retention within a CRM framework.

Example 5.5.1 (Sprint's Strategic Customer Drop)**.** A notable example is Sprint's decision to discontinue services to over 1,000 unprofitable customers who made excessive customer service calls. While unconventional, this decision emphasized the importance of strategic retention, not all customers are worth retaining. Firms must focus on retaining customers with high Customer Lifetime Value (CLV), rather than attempting to retain every customer indiscriminately.

5.5.3 What Do We Do in Retention Modeling?

Customer retention modeling is a systematic process that involves identifying customers who are at risk of discontinuing their relationship with the firm and executing actions to proactively retain them. The process starts with

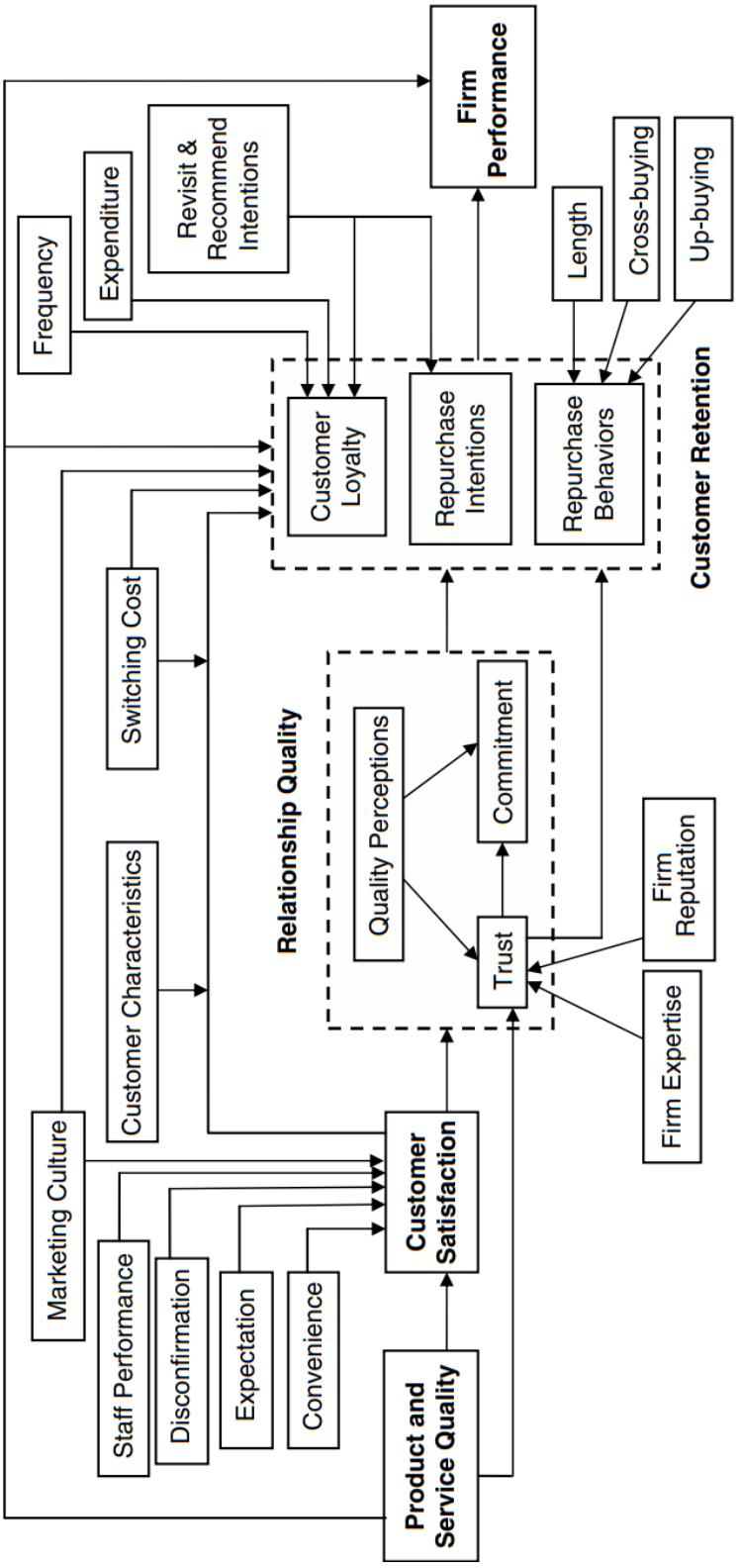


Fig. 5.3: Integrated relationships addressed in customer retention [1].

transforming raw transactional and behavioral data into structured variables such as purchase frequency, recency, monetary value (RFM), time since last interaction, and historical response to campaigns [44].

Retention modeling typically addresses the following objectives:

- **Detection:** Identify customers at high risk of churn based on predictive indicators.
- **Prioritization:** Rank customers not just by churn risk, but by their expected future value (e.g., CLV).
- **Action:** Design targeted retention campaigns such as personalized offers, loyalty programs, or engagement emails.

A well-known real-world application is Amazon Prime's retention strategy. Customers exhibiting declining purchase frequency may receive early renewal discounts or extended free trials to re-engage them. Netflix, on the other hand, uses machine learning to monitor user activity and proactively recommends personalized content to reduce inactivity and subsequent churn.

Figure 5.4 Conceptual model of customer retention linking satisfaction, relationship quality, and firm performance.

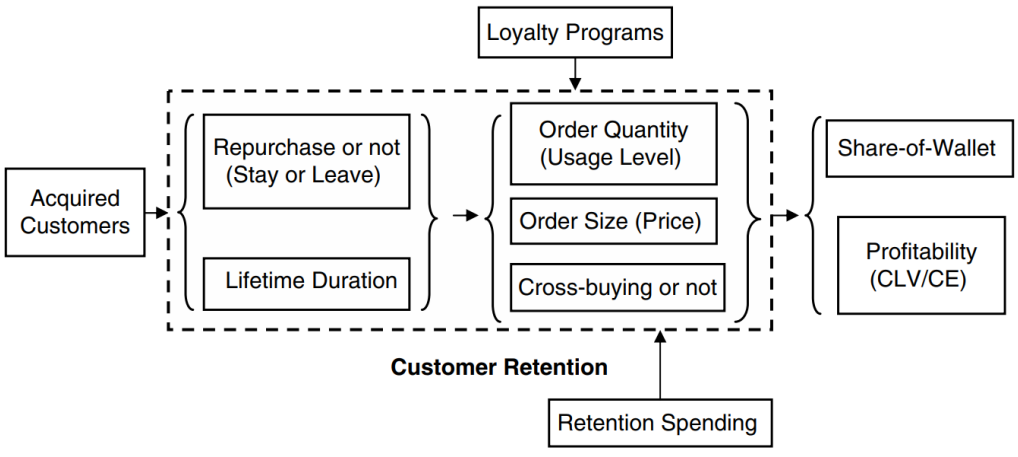


Fig. 5.4: Integrated relationships addressed in customer retention [1].

Retention modeling also enables selective retention, wherein only customers with positive long-term value are targeted for retention interventions. For instance, Sprint's controversial but analytically supported decision to drop 1,000 unprofitable customers demonstrates that not all churn is bad [1].

Finally, retention models feed into a continuous optimization loop, where response to retention actions is tracked, and models are recalibrated using experimental methods such as A/B testing. This ensures that resources are allocated efficiently and adaptively across different customer segments and life stages [45].

Retention efforts, when executed using a model-driven framework, improve not just repurchase rates but also overall firm profitability by increasing the tenure and value of high-potential customers.

The effectiveness of retention efforts hinges on the ability to model and predict customer behavior with precision. This necessitates the use of robust quantitative techniques that can account for variability in repurchase decisions, capture time-dependent churn dynamics, and incorporate customer-level heterogeneity. In the following sections, we present the formal modeling frameworks employed in this study to estimate retention probabilities and duration using both classical statistical methods and modern machine learning approaches.

Table 5.5 summarizes the specific models used in our analysis, along with their associated research objectives and estimation techniques. This classification enables a clear understanding of the methodological contributions and the rationale for selecting each model and in following section we will learn each of the model in detail with their mathematical background along with that we will also see their applications in the retention modeling and the different methodologies we used to estimated the parameters of the models.

Model Used	Research Interest	Estimation Technique
Random Intercept Binary Choice Model	Predicting binary repurchase (Yes/No)	Gaussian-Hermite Quadrature + Fisher Scoring
Discrete-Time Hazard Model	Modeling when repurchase happens (timing)	Maximum Likelihood Estimation
Dropout/Defection Model	Modeling probability of customer defection	Hierarchical Bayesian MCMC
K - Nearest Neighbours	Purchase probability via classification	

Fig. 5.5: Models Used with Research Focus and Estimation Techniques

5.5.4 Probability of Repurchase

To model binary repurchase decisions while accounting for unobserved firm-level heterogeneity, we use a random intercept specification. Let $Y_{ic} \in \{0, 1\}$ denote the binary outcome for customer c in firm i, where $Y_{ic} = 1$ indicates repurchase. The latent utility model is given by:

$$U_{ic} = \mu_i + \boldsymbol{\beta}^\top \mathbf{x}_{ic} + \varepsilon_{ic} \tag{5.16}$$

where $\mathbf{x}_{ic}$ is a vector of observed covariates, $\boldsymbol{\beta}$ is a parameter vector, μ_i is a firm-specific intercept, and $\varepsilon_{ic} \sim \text{Gumbel}(0, 1)$.

The decision rule is:

$$Y_{ic} = \begin{cases} 1 & \text{if } U_{ic} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.17}$$

Using the cumulative distribution function of the Gumbel distribution (see Section 3.1.2.6), the repurchase probability becomes:

$$\mathbb{P}(Y_{ic} = 1) = 1 - \exp\left(-\exp(\mu_i + \boldsymbol{\beta}^\top \mathbf{x}_{ic})\right) \tag{5.18}$$

The random intercept μ_i captures firm-level unobserved heterogeneity

and is modeled as:

$$\mu_i \sim \mathcal{N}(\mu, \sigma^2) \tag{5.19}$$

The conditional likelihood for all customers in firm i, given μ_i, is:

$$\mathcal{L}_i(\mu_i) = \prod_{c=1}^{T_i} \left[1 - \exp\left(-\exp(\mu_i + \boldsymbol{\beta}^\top \mathbf{x}_{ic})\right)\right]^{y_{ic}} \cdot \left[\exp\left(-\exp(\mu_i + \boldsymbol{\beta}^\top \mathbf{x}_{ic})\right)\right]^{1-y_{ic}} \tag{5.20}$$

To eliminate the unobserved μ_i, we compute the marginal likelihood by integrating over its normal distribution:

$$L_i(\theta) = \int_{-\infty}^{\infty} \mathcal{L}_i(\mu_i) \cdot \phi(\mu_i \mid \mu, \sigma^2)\, d\mu_i \tag{5.21}$$

where $\phi(\mu_i \mid \mu, \sigma^2)$ is the Gaussian density function.

Since the integral in the marginal likelihood does not have a closed-form solution, we approximate it using Gaussian-Hermite quadrature (see Section 3.6). Let z_j and w_j denote the abscissas and weights for the Hermite polynomial of order Q, then

$$L_i(\theta) \approx \sum_{j=1}^{Q} w_j \cdot \mathcal{L}_i\left(\sqrt{2}\sigma z_j + \mu\right) \tag{5.22}$$

The overall log-likelihood is then given by:

$$\ell(\theta) = \sum_{i=1}^{N} \log L_i(\theta) \tag{5.23}$$

We estimate parameters $\theta = (\boldsymbol{\beta}, \mu, \sigma^2)$ by maximizing $\ell(\theta)$. The optimization is carried out using the Fisher scoring algorithm (see Section 3.5), which updates parameters as:

$$\theta^{(t+1)} = \theta^{(t)} + \mathcal{I}^{-1}(\theta^{(t)}) \cdot \nabla \ell(\theta^{(t)}) \tag{5.24}$$

where $\mathcal{I}(\theta)$ is the expected Fisher information matrix and $\nabla \ell(\theta)$ is the score function.

5.5.5 When Will a Customer No Longer Repurchase?

Understanding when a customer will cease repurchasing is central to Customer Relationship Management (CRM). This question can be effectively addressed using survival analysis, a statistical methodology designed to analyze the time until the occurrence of an event—here, customer defection or churn.

Let T be a non-negative continuous random variable representing the time until a customer lapses (i.e., stops repurchasing). The key function in survival analysis is the *hazard function*, defined as:

$$h(t) = \lim_{\Delta t \to 0} \frac{\Pr(t \leq T < t + \Delta t \mid T \geq t)}{\Delta t} \tag{5.25}$$

This function captures the instantaneous rate at which an event occurs, given that it has not occurred up to time t.

Bhattacharya (2008) [46] applied this approach in the context of a membership-based service, modeling the customer's hazard of lapsing using a proportional hazards framework. The model assumes:

$$h_u(t) = h_0(t) \cdot \exp(\boldsymbol{\beta}^\top \mathbf{x}_u(t-1)) \tag{5.26}$$

where, the hazard rate for customer u at time t is $h_u(t)$, the baseline hazard function is $h_0(t)$, the vector of lagged covariates representing the customer's behavior or characteristics up to time $t-1$ is $\mathbf{x}_u(t-1)$, and the vector of regression coefficients is $\boldsymbol{\beta}$.

The model estimates the effect of prior behavior on the likelihood of lapsing in future periods. Parameters are typically estimated using partial likelihood maximization in the Cox proportional hazards model, though parametric approaches (e.g., Weibull, exponential) can also be employed if distributional assumptions are tenable.

Such modeling provides firms a dynamic view of customer retention, aiding in proactive targeting of customers at higher risk of churn. For implementation details and empirical illustrations, see Bhattacharya (2008) [46] and Kumar and Petersen (2012) [1].

We apply the discrete-time proportional hazards framework (see Section 4.8.2) developed by Kivetz, Ran and Urminsky (2006) [47] and extended by Seetharaman (2003) [48] to model customer repurchase behavior. This model operationalizes the discrete-time survival theory introduced in Section 4.8.2 by parameterizing the hazard probability using covariates and estimating it via maximum likelihood.

Let $S(t, \mathbf{X}_t)$ denote the survival function, i.e., the probability that a customer has not made a purchase by time t. It is defined as:

$$S(t, \mathbf{X}_t) = \exp\left(- \sum_{u=1}^{t} \exp(\mathbf{X}_u^\top \boldsymbol{\beta}) \int_{u-1}^{u} h_0(w)\, dw \right), \tag{5.27}$$

where $\mathbf{X}_u$ is the vector of time-varying covariates, $\boldsymbol{\beta}$ is the vector of coefficients, and $h_0(w)$ is the baseline hazard function defined over continuous time.

The probability of purchase at time t, conditional on no purchase prior to t, becomes

$$\Pr(t, \mathbf{X}_t) = 1 - \frac{S(t, \mathbf{X}_t)}{S(t-1, \mathbf{X}_{t-1})} = 1 - \exp\left(- \exp(\mathbf{X}_t^\top \boldsymbol{\beta}) \int_{t-1}^{t} h_0(u)\, du \right) \tag{5.28}$$

This probability captures the effect of covariates and baseline hazard in determining the likelihood of purchase at a specific time point.

The full individual-level likelihood over T periods is specified as

$$L = \prod_{t=1}^{T} \left[\Pr(t, \mathbf{X}_t)^{\delta_t} \cdot (1 - \Pr(t, \mathbf{X}_t))^{1-\delta_t} \right], \tag{5.29}$$

where $\delta_t = 1$ if the customer makes a purchase at time t, and 0 otherwise.

This structure accommodates covariate dynamics and temporal hazard shifts, making it well-suited for customer repurchase modeling within CRM analytics.

While Section 4.8.2 introduced the general discrete-time hazard framework, which models the conditional probability of an event occurring in interval t given survival up to t, Borle et al. (2008) [49] extend this structure

specifically to estimate customer defection behavior. Their formulation is particularly suited for modeling the likelihood that a customer discontinues purchasing after a given transaction history.

In this approach, the discrete hazard represents the probability that a customer h will leave the company without making the i^{th} purchase, conditional on having completed the $(i-1)^{\text{th}}$ purchase. This is defined using a logistic transformation of a latent defection score δ_{hi}:

$$h(\text{LIFE}_{hi}) = [1 + \exp(-\delta_{hi})]^{-1}, \tag{5.30}$$

where $h(\text{LIFE}_{hi})$ denotes the hazard of lifetime defection for customer h at purchase occasion i. This hazard captures the risk that a customer drops out before completing the i^{th} purchase.

The defection score δ_{hi} is parameterized using both transaction-level variables and a third-order polynomial in i, capturing non-stationarity in the dropout process

$$\delta_{hi} = \delta_0 + \delta_1 i + \delta_2 i^2 + \delta_3 i^3 + \delta_4 \log(\text{TIME}_{hi}) + \delta_5 \log(\text{AMNT}_{hi}) + \delta_6 \text{GENDER}_{hi}, \tag{5.31}$$

where: TIME_{hi} denotes the time since the previous purchase, AMNT_{hi} represents the monetary value of the prior purchase, and GENDER_{hi} is a binary indicator variable capturing the customer's gender.

The inclusion of polynomial terms in i (i, i^2, i^3) accounts for temporal dynamics, recognizing that a customer's propensity to leave may evolve nonlinearly with repeated purchases.

To model individual-level variation in purchasing behavior, the coefficients for lagged behavioral predictors are treated as random effects drawn from normal distributions:

$$\delta_{4h} \sim \mathcal{N}(\bar{\delta}_4, \tau_4^2), \tag{5.32}$$

$$\delta_{5h} \sim \mathcal{N}(\bar{\delta}_5, \tau_5^2). \tag{5.33}$$

This hierarchical structure captures heterogeneity across customers in

how interpurchase time and past spending influence defection probability [1].

The full model is estimated jointly with interpurchase times using Bayesian techniques, specifically via a Markov Chain Monte Carlo (MCMC) (see Section 3.7) algorithm. This enables robust inference even in the presence of complex data dependencies and individual variability.

This defection model complements the repurchase hazard model discussed earlier in this chapter by shifting the focus to dropout behavior. Together, they offer a comprehensive view of customer lifecycle dynamics in discrete-time, consistent with the survival analysis foundations introduced in Section 4.8.2.

5.5.6 Share of Wallet Estimation

In customer retention analytics, Share of Wallet (SOW) serves as a proxy for engagement and commitment. A higher SOW indicates that a customer allocates a greater portion of their category spending to the focal firm, often implying stronger loyalty and lower churn propensity. Monitoring changes in SOW can help firms identify customers at risk and implement timely retention strategies.

Let S be the number of competing firms in the category. For a given customer i, let A_{is} denote the latent attraction or preference toward firm s. The share of wallet is defined as:

$$\text{SOW}_{is} = \frac{A_{is}}{\sum_{j=1}^{S} A_{ij}}.$$

In practice, A_{is} is unobserved. Instead, we estimate the probability that customer i will make their next purchase from firm s using a data-driven method. Specifically, we employ a K-Nearest Neighbors (KNN) (see Section 4.5) classifier trained on historical purchase data. Let $\mathbf{x}_i$ denote the feature vector for customer i (e.g., transaction frequency, recency, product

preferences). The KNN model estimates:

$$\widehat{\mathbb{P}}(Y_i = s \mid \mathbf{x}_i),$$

where Y_i is the observed firm from which the purchase was made. This probability is then interpreted as the estimated share of wallet:

$$\widehat{\text{SOW}}_{is} = \frac{1}{K} \sum_{j \in \mathcal{N}_K(\mathbf{x}_i)} \mathbb{I}(y_j = s),$$

where $\mathcal{N}_K(\mathbf{x}_i)$ denotes the indices of the K nearest neighbors of $\mathbf{x}_i$ in the training set, and y_j is the store label for neighbor j.

Customers with consistently high $\widehat{\text{SOW}}_{is}$ are deemed loyal and stable. Declining SOW over time, especially when shifted toward competitors, signals disengagement. Hence, estimated SOW becomes a critical input to retention models that seek to predict future value and prioritize proactive interventions [50, 51].

In this chapter, we explored various models for understanding and enhancing customer retention, including repurchase likelihood, lifetime duration, cross-buying, and share-of-wallet. Retaining a customer is crucial for long-term profitability, but retention is not always guaranteed. The next challenge we address is customer churn, identifying when and why a customer might leave, and how firms can act in advance to reduce attrition.

5.6 Customer Churn

5.6.1 Introduction

Customer churn, also known as customer attrition, is the loss of customers or clients over time. It is the opposite of customer retention, where the goal is to keep customers loyal. Churn is a critical metric for businesses, as it directly affects revenue and profitability. High churn rates can indicate poor customer satisfaction, inefficiencies in customer service, or better offerings from competitors. Understanding and managing churn is essential for companies

aiming to grow sustainably and improve customer lifetime value (CLV).

In most industries, preventing churn is more cost-effective than acquiring new customers. As a result, churn management has become a crucial part of Customer Relationship Management (CRM) strategies. Businesses rely on churn prediction models to identify at-risk customers and take proactive steps to prevent attrition.

5.6.2 Importance of Customer Churn

Customer churn has a profound impact on a company's financial performance. As we discussed in retention section (see Section 5.5.2) have shown that reducing churn by as little as 5% can increase profits by 25%-85% depending on the industry. This is due to the fact that retaining customers is significantly less expensive than acquiring new ones. In fact, acquiring a new customer can cost anywhere from five to twenty-five times more than retaining an existing one [52].

Churn also affects customer lifetime value (CLV), which is the total revenue a company can expect from a customer over their relationship. As churn increases, CLV decreases, and the firm loses out on long-term revenue from loyal customers. Additionally, high churn rates can damage a company's brand reputation and create instability in cash flow, making financial forecasting more difficult.

A well-known example of churn in practice is the telecom industry. Companies like *Sprint* and *Verizon* often face high customer turnover due to competition, pricing, and service quality. For example, Sprint reported that despite offering aggressive pricing and promotional deals, they were still unable to retain a substantial number of customers due to poor customer service and network issues [53]. This led to a direct loss in revenue and market share.

On the other hand, companies such as *T-Mobile* have managed to reduce churn by investing in improving customer service and offering value-added services. T-Mobile's "Un-carrier" strategy, which eliminated long-term contracts and offered more flexible plans, helped them reduce churn and retain customers by focusing on their pain points and improving experience [54].

5.6.3 What Do We Do in Churn Management?

Customer churn modeling focuses on identifying customers who are likely to terminate their relationship with the firm and understanding the drivers behind this behavior. The primary goal is to predict the likelihood of churn before it occurs, allowing the firm to take timely action to retain valuable customers.

The process begins with defining what constitutes churn, which varies by context:

- In **contractual settings** (e.g., telecom, SaaS), churn is observed as explicit service cancellation.
- In **non-contractual settings** (e.g., retail, e-commerce), churn is inferred from prolonged inactivity or reduced engagement.

Once churn definitions are set, firms extract features from historical data such as recency, frequency, service usage, customer complaints, payment behavior, and digital engagement. These variables serve as input to predictive churn models.

Key tasks in churn modeling include:

- **Risk Scoring:** Estimating the probability that a customer will churn in a given future window.
- **Segment-Based Targeting:** Identifying customer segments with high churn probability and sufficient economic value.
- **Root Cause Analysis:** Interpreting drivers of churn to uncover actionable insights (e.g., dissatisfaction, price sensitivity).

For example, telecom companies like Vodafone use churn scores updated in real time to trigger automatic retention offers. Similarly, Spotify monitors skips, listening frequency, and app uninstall patterns to identify disengaged users.

Critically, not all churn is equally harmful. High-risk customers with low future value may not merit intervention. Hence, modern churn management

frameworks integrate churn risk with profitability metrics like CLV to enable selective win-back or disengagement [1, 55].

Lastly, churn modeling supports continuous experimentation. Firms implement retention interventions (e.g., discounts, personalized messages) and use A/B testing to assess effectiveness, thereby refining both prediction and treatment strategies [56].

Churn modeling thus combines data engineering, predictive analytics, and strategic filtering to reduce revenue loss and sustain long-term customer equity.

To effectively manage churn, firms must blend predictive analytics with strategic decision-making. This involves estimating churn risk, identifying influential drivers, and prioritizing high-value customers for intervention. As summarized in Table 5.6, the tools used range from interpretable models for estimating churn probabilities to more flexible approaches that capture nonlinear behavior and individual-level heterogeneity. In the subsequent sections, we outline these approaches in greater detail, highlighting how they support proactive and profitable churn management.

Model Used	Research Interest	Estimation Technique
Logistic Regression	Estimating probability of churn	Maximum Likelihood Estimation
Artificial Neural Network (ANN)	Learning complex nonlinear churn patterns from high-dimensional data	Stochastic Gradient Descent (SGD)
Weibull Survival Model	Modeling time to churn with parametric assumptions	Maximum Likelihood Estimation
Discrete-Time Hazard Model	Estimating time-specific churn likelihood with time-varying covariates	Maximum Likelihood Estimation
Polynomial Logistic Hazard	Capturing churn risk over purchase cycles with random effects	Markov Chain Monte Carlo

Fig. 5.6: Modeling approaches for churn management

5.6.4 Probability of Churn

Customer churn refers to the decision by a customer to terminate their relationship with a firm. Modeling churn is essential for proactive retention strategies and customer portfolio optimization. Understanding who is likely to churn allows for efficient resource allocation in marketing and customer relationship management (CRM) [1].

Let the churn decision be represented by a binary outcome $y_n \in \{0, 1\}$, where $y_n = 1$ indicates that customer n has churned. This decision is driven by a latent utility framework:

$$U_{jn} = V_{jn} + \varepsilon_{jn}, \quad j \in \{\text{churn}, \text{stay}\} \tag{5.34}$$

where V_{jn} is the deterministic component (e.g., based on satisfaction, price sensitivity, or service quality) and ε_{jn} is a random error term. The customer chooses the action that yields higher utility.

Assuming the errors ε_{jn} follow a logistic distribution (see Section 3.1.2.5), the probability that customer n will churn is given by the logit model (see Section 4.2.1):

$$P_n = \Pr(y_n = 1) = \frac{1}{1 + \exp(-\mathbf{x}_n^\top \boldsymbol{\beta})} \tag{5.35}$$

Here, $\mathbf{x}_n$ is a vector of customer-specific covariates, and $\boldsymbol{\beta}$ is the vector of coefficients to be estimated.

This probabilistic framework provides a powerful tool to predict which customers are most likely to churn, enabling firms to intervene selectively. Predictive models like this are often estimated using maximum likelihood methods (see Section 4.2.4).

While logistic regression provides a parametric and interpretable model for estimating churn probabilities, its linearity limits its ability to capture complex, nonlinear relationships present in customer data. To address this, feedforward neural networks offer a flexible and powerful alternative.

Artificial Neural Network (see Section 4.6) can also be used to approximates the probability of churn by learning hierarchical representations through multiple hidden layers. Given input features $\mathbf{x}_n \in \mathbb{R}^d$, the predicted proba-

bility of churn is computed as

$$\begin{aligned} \mathbf{z}^{(1)} &= \sigma(\mathbf{W}^{(1)}\mathbf{x}_n + \mathbf{b}^{(1)}) \\ \mathbf{z}^{(2)} &= \sigma(\mathbf{W}^{(2)}\mathbf{z}^{(1)} + \mathbf{b}^{(2)}) \\ &\vdots \\ \hat{y}_n &= \sigma(\mathbf{w}^{(L)} \cdot \mathbf{z}^{(L-1)} + b^{(L)}) \end{aligned} \tag{5.36}$$

where $\sigma(\cdot)$ is a nonlinear activation function such as ReLU or sigmoid, and $\hat{y}_n \in (0, 1)$ represents the predicted churn probability.

Model parameters $\Theta = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}$ are optimized via stochastic gradient descent by minimizing the Mean Squared Error (MSE) loss:

$$\mathcal{L}(\Theta) = \frac{1}{N} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 \tag{5.37}$$

Neural networks are particularly advantageous in high-dimensional settings where interactions among features are complex and unknown. Although they may trade off interpretability, their predictive accuracy often surpasses traditional methods, especially when trained on large labeled datasets with diverse behavioral indicators.

5.6.5 When Will a Customer Churn

Several modeling frameworks used to estimate the timing of customer churn also appear in the context of customer retention, particularly when the goal is to analyze the duration until the next repurchase or the end of customer engagement. As a result, some mathematical foundations for estimating time-to-churn overlap with those already discussed in Section 5.5.5, where continuous-time and discrete-time hazard models were developed for retention modeling. However, in churn management, these methods are specifically tailored to identify the likely point at which a customer exits the relationship permanently.

Let T denote a non-negative continuous random variable representing the time to churn. The key construct in survival analysis is the hazard function,

defined as

$$h(t) = \lim_{\Delta t \to 0} \frac{\Pr(t \leq T < t + \Delta t \mid T \geq t)}{\Delta t} = \frac{f(t)}{S(t)},$$

where $f(t)$ is the probability density function and $S(t) = \Pr(T > t)$ is the survival function. These are related through the cumulative hazard function $H(t) = \int_0^t h(u)\, du$, giving the identity $S(t) = \exp(-H(t))$.

In the parametric setting, we use the Weibull model (see Section 4.8.1), where the hazard function is specified as

$$h(t \mid \mathbf{x}) = \lambda p t^{p-1}, \quad \lambda = \exp(-\mathbf{x}^\top \beta).$$

This model flexibly accommodates increasing, decreasing, or constant churn risk over time, and is particularly useful in modeling customer duration with right-censored data (see Section 5.4.4 and Section 5.6.4).

To relax distributional assumptions, we use the Cox proportional hazards model, specified as

$$h(t \mid \mathbf{x}) = h_0(t) \exp(\mathbf{x}^\top \beta),$$

where $h_0(t)$ is an unspecified baseline hazard function. This model was introduced earlier in the retention framework (Section 5.5.5) but applies equally to churn prediction, where the event of interest is customer defection.

In scenarios where transaction data are recorded at discrete intervals, we adopt a discrete-time hazard modeling framework, as developed in Section 4.8.2 and applied in Section 5.5.5. The discrete-time hazard is given by

$$\Pr(T = t \mid T \geq t, \mathbf{x}_t) = 1 - \exp\left(-\exp(\mathbf{x}_t^\top \beta) \int_{t-1}^{t} h_0(u)\, du\right),$$

allowing churn likelihood to be estimated per time interval based on dynamic covariates.

In addition, to capture nonlinear hazard trajectories and customer heterogeneity, we employ a polynomial logistic hazard specification of the form

$$h(\text{LIFE}_{hi}) = [1 + \exp(-\delta_{hi})]^{-1},$$

where,

$$\delta_{hi} = \delta_0 + \delta_1 i + \delta_2 i^2 + \delta_3 i^3 + \delta_4 \log(\text{TIME}_{hi}) + \delta_5 \log(\text{AMNT}_{hi}) + \delta_6 \text{GENDER}_{hi}.$$

This model was introduced in Section 5.5.5 and extended here to estimate dropout hazard over repeated purchase cycles, integrating both transaction history and demographic signals. Coefficients such as δ_4 and δ_5 are modeled as random effects, and estimation is performed via Markov Chain Monte Carlo (MCMC) (see Section 3.7).

Thus, by leveraging both parametric and semi-parametric survival analysis methods discussed in Chapter 4 and applied in Chapter 5, we provide a rigorous framework for estimating when a customer is likely to churn, allowing for timely intervention and effective churn management.

This chapter focused on modeling customer churn using techniques such as logistic regression, hazard models, and classification frameworks. By identifying potential churners and estimating their likelihood of leaving, firms can intervene strategically. However, not all customer loss is preventable. In the next chapter, we turn to customer win-back — reactivating lost customers and understanding the conditions under which they are most likely to return.

5.7 Customer Win-Back

5.7.1 Introduction

Customer win-back is a critical yet often underutilized strategy in Customer Relationship Management (CRM), involving the reactivation and re-engagement of previously lost customers. Unlike customer retention strategies which aim to prevent churn, win-back initiatives are post-churn interventions that seek to re-establish a relationship with defected customers. According to Kumar and Petersen [1], customer win-back lies at the intersection of strategic marketing and predictive analytics, leveraging historical

behavioral data to identify defectors with a high likelihood of return.

The process typically begins with segmentation of past customers based on factors such as their former profitability, recency of departure, and reasons for churn. By focusing resources on high-value defectors—customers who had previously exhibited significant lifetime value (CLV) and remain reengagement candidates—firms can achieve a favorable return on investment. The strategic logic underlying win-back is that these customers, already familiar with the brand and its value proposition, require lower persuasion effort and are more responsive to reactivation stimuli compared to completely new prospects.

The operationalization of a win-back strategy may include tailored incentives, direct communication (e.g., personalized emails or phone outreach), and modifications to the core offering based on feedback or churn diagnostics. These approaches are often embedded within statistical modeling frameworks that estimate both the probability of return and the projected duration of the renewed relationship.

5.7.2 Importance of Customer Win-Back

The strategic value of customer win-back arises from three fundamental advantages: cost efficiency, brand familiarity, and profit potential. First, reacquiring former customers is frequently more economical than acquiring new ones. The cost of acquisition for a new customer can be five to seven times higher than the cost associated with winning back a previous customer [1]. This economic differential underscores the tactical relevance of win-back in marketing budget allocations.

Second, former customers possess pre-existing familiarity with the firm's brand, products, and services. This accumulated brand knowledge reduces the informational burden on the firm and increases the likelihood of rapid reengagement. It also enables a more nuanced and personalized communication strategy during the win-back campaign.

Third, evidence from empirical studies shows that successfully reacquired customers can become even more profitable than before. These customers of-

ten return with revised expectations and, having previously experienced loss aversion, may engage with greater loyalty and higher purchase frequency. This has direct implications for CLV modeling and long-term revenue forecasting.

Furthermore, win-back strategies provide diagnostic intelligence by illuminating the reasons for churn. This feedback loop allows firms not only to enhance their offerings but also to proactively adjust their retention mechanisms, thereby creating a virtuous cycle of customer-centric improvement. From a portfolio optimization perspective, selective win-back targeting aligns with the broader objective of maximizing customer equity rather than indiscriminately pursuing volume-based retention.

Example 5.7.1. Verizon Wireless implemented a successful win-back program aimed at former customers who had churned due to perceived pricing issues. By offering limited-time promotional plans with enhanced data packages and personalized outreach from retention teams, Verizon was able to achieve a reacquisition rate exceeding 25% among targeted segments. Additionally, the average revenue per user (ARPU) for reacquired customers increased over a 12-month horizon, surpassing that of newly acquired users [57].

5.7.3 What We Do in Win-Back Strategies

Customer win-back is not merely a reactive outreach mechanism—it is a structured and deliberate process that combines customer insight, segmentation, targeted communication, and timely follow-up. The aim is to convert previously lost customers into profitable, long-term relationships. The following outlines the general flow of activities involved in designing and executing a customer win-back strategy.

1. Identification of Target Customers

The win-back process begins with identifying which former customers are worth pursuing. Not all defected customers are viable or profitable candidates for return. Typically, firms apply business rules or scoring metrics

based on historical value, recency of churn, and reason for departure to shortlist customers most likely to respond favorably. This step often draws upon customer lifetime value (CLV), churn classifications, and qualitative feedback obtained at the point of exit.

2. Root Cause Analysis of Churn

Understanding why a customer left is essential for crafting an effective comeback strategy. This may involve mining feedback from exit surveys, complaint records, or service logs. Root causes can vary widely—from dissatisfaction with pricing or service quality to competitive switching or life events. Diagnosing these drivers helps tailor the messaging and value proposition accordingly.

3. Strategic Design of Win-Back Campaigns

Once the target group and churn reasons are understood, firms design interventions that directly address those issues. Common elements include:

- Personalized communication (e.g., referencing past purchase behavior).
- Limited-time offers, price discounts, or upgrades.
- Highlighting improved features or changes in service.
- Reinforcing the brand's unique value or emotional appeal.

These campaigns can be delivered through multiple channels—email, SMS, phone calls, or direct sales—depending on the segment's responsiveness and past interaction history.

4. Implementation and Timing

Effective win-back is not only about content but also about timing. Campaigns often prioritize recently churned customers, as brand familiarity and

emotional engagement diminish over time. Additionally, some firms implement tiered outreach, initiating light-touch reminders before progressing to more aggressive incentives.

5. Measurement and Feedback Loop

The final step involves evaluating the campaign's success through well-defined KPIs, such as:

- Reacquisition rate.
- Incremental revenue from reacquired customers.
- Post-return CLV and retention duration.
- Campaign-level ROI.

The insights from these metrics not only inform future win-back initiatives but also feed back into customer retention strategies more broadly.

6. Role of Modeling in Enhancing Effectiveness

While the above steps can be implemented using rule-based systems, modeling plays an increasingly important role in scaling and optimizing win-back strategies. Predictive models not only help identify customers with the highest likelihood of returning and estimate the expected duration of their renewed relationship, but they also allow for simulating various offer structures to maximize return on investment (ROI). Despite their potential, there has been relatively limited academic attention to quantitative modeling in the win-back context, especially compared to acquisition and retention modeling. In this study, we aim to address this gap by formulating a structured modeling framework for customer win-back. The details of the estimation strategy, variables considered, and empirical implications will be discussed in the subsequent sections.

Table 5.7 summarizes the key analytical methods used to support customer win-back efforts. These approaches focus on estimating the likelihood

of a previously lost customer returning, predicting the expected duration of the renewed relationship, and quantifying the financial value generated post reacquisition. Each method aids in prioritizing outreach and tailoring incentives for maximum impact.

Model Used	Research Interest	Estimation Technique
Probit Model	Estimating the probability of reacquisition	Maximum Likelihood Estimation (MLE)
Tobit Model	Estimating length of second customer relationship	Censored Regression via MLE
Tobit Model	Estimating second customer lifetime value (SCLV)	Censored Regression via MLE

Fig. 5.7: Analytical approaches supporting customer win-back

5.7.4 Probability of Reacquisition

To estimate whether a previously churned customer is likely to return, we model the reacquisition decision as a binary outcome using a probit model (see Section 4.2.2)
The latent utility model is given by

$$z_i^* = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0,1) \tag{5.38}$$

where z_i^* denotes the unobserved propensity of customer i to return, $\boldsymbol{x}_i$ is a vector of observed features (e.g., prior relationship duration, offer type, time since churn), and $\boldsymbol{\beta}$ is a vector of parameters to be estimated.
The observed reacquisition decision is:

$$z_i = \begin{cases} 1 & \text{if } z_i^* > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.39}$$

The probability of reacquisition is therefore modeled as:

$$P(z_i = 1) = \Phi(\boldsymbol{x}_i^\top \boldsymbol{\beta}) \tag{5.40}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution.
Parameters $\boldsymbol{\beta}$ are estimated via Maximum Likelihood Estimation (MLE) as described in Section 4.2.4.

5.7.5 Duration of the Second Relationship

Once a customer is reacquired, it is important to estimate how long they are likely to stay in their second lifecycle. This duration is modeled using a censored regression (Tobit) (see Section 4.3.4) model to account for right-censoring.
The latent duration is modeled as:

$$y_i^* = \boldsymbol{x}_i^\top \boldsymbol{\gamma} + \eta_i, \quad \eta_i \sim \mathcal{N}(0, \sigma^2) \tag{5.41}$$

where y_i^* is the unobserved duration of the second relationship for customer i, $\boldsymbol{x}_i$ is a vector of covariates (e.g., characteristics at reacquisition), and $\boldsymbol{\gamma}$ and σ^2 are parameters to be estimated.
The observed duration y_i is:

$$y_i = \begin{cases} y_i^* & \text{if } y_i^* \leq c \\ c & \text{if } y_i^* > c \end{cases} \tag{5.42}$$

where c denotes the censoring threshold (e.g., the end of observation window). The likelihood function for censored observations is composed of a mixture of normal density for uncensored and cumulative normal for censored observations.
Estimation is performed using Maximum Likelihood methods, discussed in Section 4.2.4.

5.7.6 Second Customer Lifetime Value (SCLV)

After estimating the probability of reacquisition and the duration of the second relationship, we compute the second Customer Lifetime Value (SCLV) to evaluate the financial worth of reacquired customers.
The latent SCLV for customer i is modeled as:

$$\text{SCLV}_i^* = \boldsymbol{x}_i^\top \boldsymbol{\delta} + \zeta_i, \quad \zeta_i \sim \mathcal{N}(0, \sigma^2) \tag{5.43}$$

where $\boldsymbol{x}_i$ includes variables such as reacquisition channel, previous CLV, and duration predictors. The parameters $\boldsymbol{\delta}$ and σ^2 are estimated using censored regression techniques also called as Tobit (see Section 4.3.4).
The observed SCLV is given by:

$$\text{SCLV}_i = \begin{cases} \text{SCLV}_i^* & \text{if } \text{SCLV}_i^* \leq c \\ c & \text{if } \text{SCLV}_i^* > c \end{cases} \tag{5.44}$$

where c is the right-censoring threshold for lifetime value due to limited observation windows.
The estimation is performed via Maximum Likelihood Estimation (see Section 4.2.4).

Here, we modeled the process of reacquiring lost customers using frameworks based on customer value, offer perception, and reactivation probability. While win-back completes the core CRM cycle, the real strength lies in integrating these models into a unified framework. The following chapters focus on implementation, showing how firms deploy these CRM models in practice.

5.8 Some Practical Implementations

While the core focus of this thesis lies in developing rigorous mathematical frameworks for modeling key stages of the customer lifecycle, we supplement our theoretical contributions with illustrative implementation examples. These examples are not meant to serve as optimized predictive pipelines, but rather to offer practical interpretations of how such models can be instan-

tiated and interpreted using customer-level marketing data. The objective here is explanatory rather than computational performance.

The Data used in this section is majorly of B2B firm and the source of the data is Kumar and Peterson (2012) [1].

Illustrative Code Implementations

1. **Customer Acquisition via Logistic Regression**
   A logistic regression model is implemented to predict the likelihood of customer acquisition based on firm-level and marketing expenditure variables. This helps in understanding how acquisition spending translates to conversion probability.
   Refer below to LOGISTIC REGRESSION: Customer Acquisition

2. **First Purchase Amount using Linear Regression**
   A linear regression model is used to estimate the first purchase value among acquired customers. This serves as an important input to downstream CLV calculations.
   Refer below to Two-Step Model: Customer Acquisition + First Purchase

3. **Customer Lifetime Value using Tobit Model**
   Due to censoring in CLV data (many customers are still active), a Tobit regression is applied to model truncated lifetime value while accounting for latent contribution.
   Refer below to Tobit Model for CLV

4. **Repurchase Likelihood using Logit**
   This model estimates the probability that a customer will repurchase, factoring in lagged purchase behavior and random intercepts for firm heterogeneity.
   Refer below to Logistic Regression: Customer Repurchase Prediction

5. **Store Preference Classification using KNN**
   A K-Nearest Neighbors classifier is employed to predict a customer's

likely store preference based on observed transactional patterns.
Refer below to KNN for SOW - Store Prediction

6. **Churn Prediction via Artificial Neural Network**
   A simple feedforward ANN is trained to classify whether a customer is likely to churn, capturing complex interactions across behavioral variables.
   Refer below to Artificial Neural Network for Churn Prediction

7. **Churn Timing via Discrete-Time Hazard Model**
   A hazard-based survival model is used to estimate the expected churn timing conditional on historical data and covariates.
   Refer below to Hazard Model for Churn Duration

8. **SCLV Modeling with Neural Networks**
   A neural network is trained to estimate the Share of Customer Lifetime Value (SCLV) distribution across segments or individuals, offering a flexible non-linear approach to long-term value estimation.
   Refer below to Artificial Neural Network for SCLV

LOGISTIC REGRESSION: Customer Acquisition

```
#Mount Google drive

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#Import necessary libraries

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    roc_auc_score
)
```

```
# Load the dataset from Excel file

df = pd.read_excel('/content/drive/MyDrive/Chapter 3 - Customer Acquisition Data.xls')
df.head()
```

	Customer	Acquisition	First_Purchase	CLV	Duration	Censor	Acq_Expense	Acq_Expense_SQ	Industry	Revenue	Employees	Ret_Ex
0	1	1	433.64	0.0000	384	0	760.36	578147.33	1	30.16	1240	23
1	2	0	0.00	0.0000	0	0	147.70	21815.29	1	39.80	166	
2	3	0	0.00	0.0000	0	0	252.56	63786.55	1	54.93	1016	
3	4	1	225.84	5.7316	730	1	609.73	371770.67	1	45.83	122	2
4	5	1	363.04	0.0000	579	0	672.36	452067.97	1	69.03	313	

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#Extract relevant features for modeling
# Creating a subset of useful variables for modeling

df1 = df[['Acquisition', 'Acq_Expense', 'Acq_Expense_SQ',   'Industry', 'Revenue',  'Employees', ]]
print(df1.head())
```

```
   Acquisition  Acq_Expense  Acq_Expense_SQ  Industry  Revenue  Employees
0            1       760.36       578147.33         1    30.16       1240
1            0       147.70        21815.29         1    39.80        166
2            0       252.56        63786.55         1    54.93       1016
3            1       609.73       371770.67         1    45.83        122
4            1       672.36       452067.97         1    69.03        313
```

```
#Define feature matrix `X` and target vector `y`

x = df1.drop('Acquisition', axis=1)
y = df1['Acquisition']

#Check class distribution
y.value_counts()
```

Acquisition	count
1	292
0	208

dtype: int64

```
#Split data into training and testing sets (80/20 split)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
y_test
```

	Acquisition
361	0
73	1
374	0
155	0
104	1
...	...
347	1
86	0
75	1
438	0
15	1

100 rows × 1 columns

dtype: int64

```
#Fit a Logistic Regression model

model = LogisticRegression()
model.fit(x_train, y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

LogisticRegression

LogisticRegression()

```
#Predict and evaluate model on test set

y_pred = model.predict(x_test)


# Confusion Matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[37  6]
 [ 8 49]]
```

```
# Classification Report: Precision, Recall, F1-score, Accuracy
print("\n Classification Report:")
print(classification_report(y_test, y_pred))
```

```
 Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.86      0.84        43
           1       0.89      0.86      0.88        57

    accuracy                           0.86       100
   macro avg       0.86      0.86      0.86       100
weighted avg       0.86      0.86      0.86       100
```

```
# ROC AUC Score
y_pred_proba = model.predict_proba(x_test)[:, 1]
roc_score = roc_auc_score(y_test, y_pred_proba)
print("\n ROC AUC Score:", roc_score)
```

```
 ROC AUC Score: 0.9188086495308038
```

```
y_pred_proba = model.predict_proba(x_test)[:,1]
print(roc_auc_score(y_test, y_pred_proba))
```

```
0.9188086495308038
```

```
#Extract Model Coefficients

feature_names = x.columns
coefficients = model.coef_[0]  # Coefficient values
intercept = model.intercept_[0]

# Create a DataFrame for coefficients

coeff_df = pd.DataFrame({
    'Variable': feature_names,
    'Estimate': coefficients
}).sort_values(by='Variable', ascending=False)

# Add Intercept at the top

intercept_df = pd.DataFrame({'Variable': ['Intercept'], 'Estimate': [intercept]})
coeff_df = pd.concat([intercept_df, coeff_df], ignore_index=True)

print("\n Logistic Regression Coefficients:")
print(coeff_df)
```

```
 Logistic Regression Coefficients:
         Variable  Estimate
0       Intercept -0.009194
1         Revenue -0.020020
2        Industry -0.004824
3       Employees  0.001976
4  Acq_Expense_SQ  0.000042
5     Acq_Expense -0.021405
```

Number of Customers Aquired

```
#Count of customers predicted as acquired (Acquisition = 1)
predicted_acquired = np.sum(y_pred == 1)
print(f"Number of Customers Predicted as Acquired in Test Set: {predicted_acquired}")
```

```
Number of Customers Predicted as Acquired in Test Set: 55
```

```
#Count of correctly predicted acquired customers (True Positives)
true_positives = np.sum((y_test == 1) & (y_pred == 1))
print(f"Number of Correctly Predicted Acquired Customers (True Positives): {true_positives}")
```

```
Number of Correctly Predicted Acquired Customers (True Positives): 49
```

Two-Step Model: Customer Acquisition + First Purchase

```
#Import Libraries

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.metrics import (
    accuracy_score, confusion_matrix, classification_report,
    roc_auc_score, mean_squared_error, r2_score
)
import matplotlib.pyplot as plt
import seaborn as sns


#Step 2: Load Data
df = pd.read_excel('/content/drive/MyDrive/Chapter 3 - Customer Acquisition Data.xls')

#Prepare for Acquisition Model
# Feature selection for binary classification

X_acq = df[['Acq_Expense', 'Acq_Expense_SQ', 'Industry', 'Revenue', 'Employees']]
y_acq = df['Acquisition']


# Train-test split

X1_train, X1_test, y1_train, y1_test = train_test_split(X_acq, y_acq, test_size=0.2, random_state=42)

#Step 4: Fit Logistic Regression (Acquisition Prediction)

acq_model = LogisticRegression(max_iter=1000)
acq_model.fit(X1_train, y1_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

```
# Predict on full data

df['Acq_Pred'] = acq_model.predict(X_acq)


# Evaluation of acquisition model

print("\n Acquisition Model Evaluation:")
y1_pred = acq_model.predict(X1_test)
print("Accuracy:", accuracy_score(y1_test, y1_pred))
print("ROC AUC:", roc_auc_score(y1_test, acq_model.predict_proba(X1_test)[:, 1]))
print(confusion_matrix(y1_test, y1_pred))
print(classification_report(y1_test, y1_pred))
```

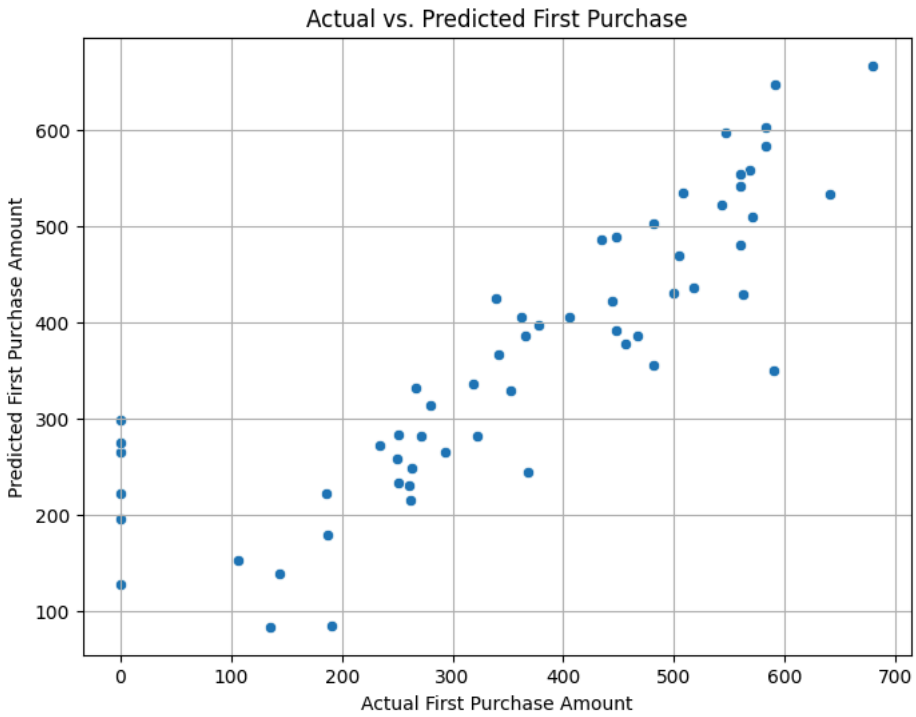
```
 Acquisition Model Evaluation:
Accuracy: 0.91
ROC AUC: 0.9616483068135454
[[38  5]
 [ 4 53]]
              precision    recall  f1-score   support

           0       0.90      0.88      0.89        43
           1       0.91      0.93      0.92        57

    accuracy                           0.91       100
   macro avg       0.91      0.91      0.91       100
weighted avg       0.91      0.91      0.91       100
```

```
#Prepare for First Purchase Regression (Step 2)
# Filter only customers predicted to be acquired
df_acq = df[df['Acq_Pred'] == 1].copy()

# Feature selection for regression model
X_fp = df_acq[['Acq_Expense', 'Acq_Expense_SQ', 'Industry', 'Revenue', 'Employees']]
y_fp = df_acq['First_Purchase']

# Train-test split for regression
X2_train, X2_test, y2_train, y2_test = train_test_split(X_fp, y_fp, test_size=0.2, random_state=42)
```

```
#Fit Linear Regression (First Purchase Prediction)
reg_model = LinearRegression()
reg_model.fit(X2_train, y2_train)

# Predict
y2_pred = reg_model.predict(X2_test)


#Evaluate Regression Model

mse = mean_squared_error(y2_test, y2_pred)
r2 = r2_score(y2_test, y2_pred)
print("\n First Purchase Regression Evaluation:")
print("Mean Squared Error:", mse)
print("R² Score:", r2)
```

```
 First Purchase Regression Evaluation:
Mean Squared Error: 9389.286183869051
R² Score: 0.7245894234555078
```

```
#Scatter Plot

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y2_test, y=y2_pred)
plt.xlabel("Actual First Purchase Amount")
plt.ylabel("Predicted First Purchase Amount")
plt.title("Actual vs. Predicted First Purchase")
plt.grid(True)
plt.show()
```




```
#Step 8 (Fixed): Save Combined Results to CSV
df_acq_final = df_acq.copy()
df_acq_final['Predicted_First_Purchase'] = reg_model.predict(X_fp)

# Export only relevant columns
df_acq_final[['Acquisition', 'First_Purchase', 'Predicted_First_Purchase']].to_csv(
    "/content/Predicted_First_Purchase.csv", index=False
)

print("\n Saved predictions to: /content/Predicted_First_Purchase.csv")
```

```
 Saved predictions to: /content/Predicted_First_Purchase.csv
```

Customer Lifetime Value (CLV) using a Tobit

```
pip install py4etrics
```

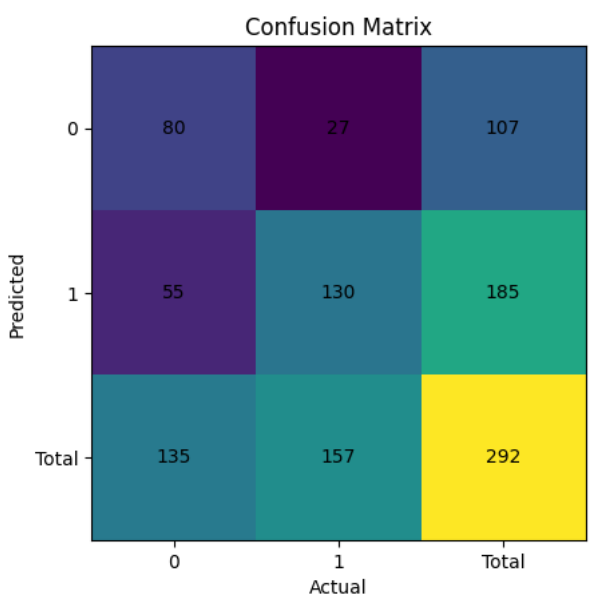
```
Collecting py4etrics
  Downloading py4etrics-0.1.9-py2.py3-none-any.whl.metadata (1.8 kB)
Downloading py4etrics-0.1.9-py2.py3-none-any.whl (19 kB)
Installing collected packages: py4etrics
Successfully installed py4etrics-0.1.9
```

```
# 1. Mount Google Drive and import libraries
from google.colab import drive
import pandas as pd
import numpy as np
import statsmodels.api as sm
from py4etrics.tobit import Tobit

#drive.mount('/content/drive')

# 2. Load and clean data
file_path = '/content/drive/MyDrive/Chapter 3 - Customer Acquisition Data.xlsx'
df = pd.read_excel(file_path, sheet_name='Customer Acquisition Data')
df = df.dropna(subset=['Duration', 'Censor'])
df = df[df['Duration'] > 0].reset_index(drop=True)

# 3. Define censoring and cap Duration at 730 days
censor_limit = 730
df['Duration_Cap'] = df['Duration'].clip(upper=censor_limit)
cens = np.where(df['Duration'] >= censor_limit, 1, 0)  # 1 = right-censored, 0 = observed

# 4. Prepare feature matrix and dependent variable
y = df['Duration_Cap'].astype(float)
X = df[['Ret_Expense', 'Ret_Expense_SQ', 'Crossbuy',
        'Frequency', 'Frequency_SQ', 'Industry',
        'Revenue', 'Employees']].astype(float)
X = sm.add_constant(X)

# 5. Fit Tobit model
model = Tobit(endog=y, exog=X, cens=cens, left=0, right=censor_limit)
res = model.fit()

# 6. Predict (latent) durations and enforce censor limit
beta = res.params[:-1]
XB = np.dot(X, beta)
df['Predicted_Duration'] = np.minimum(XB, censor_limit)

# 7. Compute MAD for customers who churned (Censor == 0)
churned = df[df['Censor'] == 0]
mad_model     = np.mean(np.abs(churned['Predicted_Duration'] - churned['Duration']))
benchmark_mu  = churned['Duration'].mean()
mad_benchmark = np.mean(np.abs(churned['Duration'] - benchmark_mu))

# 8. Build churn classification table
df['Actual_Churn']    = (df['Censor'] == 0).astype(int)
df['Predicted_Churn'] = (df['Predicted_Duration'] < censor_limit).astype(int)
confusion = pd.crosstab(df['Predicted_Churn'],
                        df['Actual_Churn'],
                        rownames=['Predicted'],
                        colnames=['Actual'])
confusion['Total'] = confusion.sum(axis=1)
confusion.loc['Total'] = confusion.sum()

# 9. Print results
print(f"Mean Absolute Deviation (Tobit Model): {mad_model:.2f}")
print(f"Mean Absolute Deviation (Benchmark):     {mad_benchmark:.2f}\n")
print("Churn Classification Table:")
print(confusion)

# 10. Print Tobit model coefficients
print("\nTobit Coefficients:")
print(res.params)
```

```
Optimization terminated successfully.
         Current function value: 4.179053
         Iterations: 3810
         Function evaluations: 5353
Mean Absolute Deviation (Tobit Model): 256.80
Mean Absolute Deviation (Benchmark):     165.47
```

```
Churn Classification Table:
Actual       0    1  Total
Predicted
0           80   27    107
1           55  130    185
Total      135  157    292

Tobit Coefficients:
[-6.25789001e+02  4.78640372e-01 -7.01439336e-05  8.87721404e+00
  3.66052790e+01 -4.46637027e-01  1.93342835e+02  5.20962889e+00
  1.30276551e-01  5.81491247e+00]
```

```
import matplotlib.pyplot as plt
import numpy as np

# Assuming 'confusion' DataFrame is already defined in the environment
cm = confusion.values
actual_labels = confusion.columns.tolist()
predicted_labels = confusion.index.tolist()

fig, ax = plt.subplots()
im = ax.imshow(cm)

# Set tick labels
ax.set_xticks(np.arange(len(actual_labels)))
ax.set_yticks(np.arange(len(predicted_labels)))
ax.set_xticklabels(actual_labels)
ax.set_yticklabels(predicted_labels)

# Labels and title
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title('Confusion Matrix')

# Annotate each cell with its value
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, int(cm[i, j]), ha='center', va='center')

plt.show()
```




```
# 1. Mount Google Drive and import libraries
# from google.colab import drive
# drive.mount('/content/drive')

import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.duration.hazard_regression import PHReg

# 2. Load and clean data
file_path = '/content/drive/MyDrive/Chapter 3 - Customer Acquisition Data.xlsx'
df = pd.read_excel(file_path, sheet_name='Customer Acquisition Data')
df = df.dropna(subset=['Duration', 'Censor'])
df = df[df['Duration'] > 0].copy()
```

```
# 3. Log-transform the duration
df['Log_Duration'] = np.log(df['Duration'])

# 4. Prepare regressors and status indicator
X_cols = [
    'Ret_Expense', 'Ret_Expense_SQ', 'Crossbuy',
    'Frequency', 'Frequency_SQ', 'Industry',
    'Revenue', 'Employees'
]
X = sm.add_constant(df[X_cols])
y = df['Log_Duration']
status = 1 - df['Censor']  # 1 = event observed (churned), 0 = censored

# 5. Fit AFT model (log-normal)
model = PHReg(endog=y, exog=X, status=status, ties='breslow', dist='lognorm')
result = model.fit()

# 6. Predict durations
pred_log = result.predict(X).predicted_values
df['Predicted_Duration'] = np.exp(pred_log)

# 7. Compute MAD vs. benchmark
churned = df[df['Censor'] == 0]
mad_model     = np.mean(np.abs(churned['Predicted_Duration'] - churned['Duration']))
benchmark_mu  = churned['Duration'].mean()
mad_benchmark = np.mean(np.abs(churned['Duration'] - benchmark_mu))

# 8. Build confusion matrix for churn classification
df['Actual_Churn']    = (df['Censor'] == 0).astype(int)
df['Predicted_Churn'] = (df['Predicted_Duration'] < 730).astype(int)
confusion = pd.crosstab(
    df['Predicted_Churn'],
    df['Actual_Churn'],
    rownames=['Predicted'],
    colnames=['Actual']
)
confusion['Total'] = confusion.sum(axis=1)
confusion.loc['Total'] = confusion.sum()

# 9. Print results
print(f"Mean Absolute Deviation (AFT model):   {mad_model:.2f}")
print(f"Mean Absolute Deviation (Benchmark):   {mad_benchmark:.2f}\n")
print("Churn Classification Table:")
print(confusion)

print("\nAFT Model Summary:")
print(result.summary())
```

```
Mean Absolute Deviation (AFT model):   327.38
Mean Absolute Deviation (Benchmark):   165.47

Churn Classification Table:
Actual       0    1  Total
Predicted
1          135  157    292
Total      135  157    292

AFT Model Summary:
                           Results: PHReg
==================================================================
Model:                     PH Reg             Sample size:       292
Dependent variable:        Log_Duration       Num. events:       157
Ties:                      Breslow
------------------------------------------------------------------
                log HR log HR SE   HR      t    P>|t|  [0.025 0.975]
------------------------------------------------------------------
const          -0.0000       nan 1.0000     nan    nan    nan    nan
Ret_Expense    -0.0017    0.0002 0.9983 -9.9702 0.0000 0.9979 0.9986
Ret_Expense_SQ  0.0000    0.0000 1.0000  2.5847 0.0097 1.0000 1.0000
Crossbuy       -0.0728    0.0476 0.9298 -1.5311 0.1257 0.8470 1.0206
Frequency      -0.1321    0.0682 0.8762 -1.9374 0.0527 0.7666 1.0015
Frequency_SQ    0.0014    0.0040 1.0014  0.3521 0.7248 0.9935 1.0094
Industry       -0.6987    0.1645 0.4972 -4.2465 0.0000 0.3602 0.6864
Revenue        -0.0197    0.0047 0.9805 -4.1632 0.0000 0.9714 0.9896
Employees      -0.0005    0.0002 0.9995 -2.6596 0.0078 0.9992 0.9999
==================================================================
Confidence intervals are for the hazard ratios
/usr/local/lib/python3.11/dist-packages/statsmodels/duration/hazard_regression.py:1239: RuntimeWarning: invalid value encountered in
  ret_val.standard_errors = np.sqrt(va)
/usr/local/lib/python3.11/dist-packages/statsmodels/duration/hazard_regression.py:1423: RuntimeWarning: invalid value encountered in
  return np.sqrt(np.diag(self.cov_params()))
```

```
# 1. Mount Google Drive and import libraries
# from google.colab import drive
# drive.mount('/content/drive')

import pandas as pd
import numpy as np
import statsmodels.api as sm
from scipy.stats import norm

# 2. Load and clean data
file_path = '/content/drive/MyDrive/Chapter 3 - Customer Acquisition Data.xlsx'
df = pd.read_excel(file_path, sheet_name='Customer Acquisition Data')

# 3. Keep only acquired prospects
df = df[df['Acquisition'] == 1].copy()

# 4. Ensure numeric inputs
X1_cols = [
    'Acq_Expense', 'Acq_Expense_SQ',
    'Ret_Expense', 'Ret_Expense_SQ',
    'First_Purchase',
    'Crossbuy',
    'Frequency', 'Frequency_SQ',
    'Industry', 'Revenue', 'Employees'
]
df[X1_cols] = df[X1_cols].apply(pd.to_numeric, errors='coerce')
df = df.dropna(subset=X1_cols + ['Censor', 'CLV'])

# 5. Stage 1: Probit model for Censor
X1 = sm.add_constant(df[X1_cols])
y1 = df['Censor']  # 1 = still a customer, 0 = churned
probit = sm.Probit(y1, X1)
probit_res = probit.fit(disp=False)

# 6. Compute inverse Mills ratio λ = ϕ(Xβ)/Φ(Xβ)
XB1 = probit_res.predict(X1)
df['IMR'] = norm.pdf(XB1) / norm.cdf(XB1)

# 7. Stage 2: OLS regression on CLV for active customers
df2 = df[df['Censor'] == 1].copy()
X2_cols = X1_cols + ['IMR']
X2 = sm.add_constant(df2[X2_cols])
y2 = df2['CLV']
ols = sm.OLS(y2, X2)
ols_res = ols.fit()

# 8. Predict CLV and compute accuracy metrics
df2['Predicted_CLV'] = ols_res.predict(X2)
mad_model     = np.mean(np.abs(df2['Predicted_CLV'] - df2['CLV']))
mape_model    = np.mean(np.abs((df2['Predicted_CLV'] - df2['CLV']) / df2['CLV'])) * 100
benchmark     = df2['CLV'].mean()
mad_benchmark = np.mean(np.abs(df2['CLV'] - benchmark))
mape_bench    = np.mean(np.abs((df2['CLV'] - benchmark) / df2['CLV'])) * 100

# 9. Print results
print(f"MAD (Model):      {mad_model:.2f}")
print(f"MAPE (Model):     {mape_model:.2f}%")
print(f"MAD (Benchmark):  {mad_benchmark:.2f}")
print(f"MAPE (Benchmark): {mape_bench:.2f}%\n")

print("Probit Model Summary:")
print(probit_res.summary())

print("\nCLV OLS Model Summary:")
print(ols_res.summary())
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
```

```
Possibly complete quasi-separation: A fraction 0.69 of observations can be
```

```
Possibly complete quasi-separation: A fraction 0.09 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.

CLV OLS Model Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                    CLV   R-squared:                       0.855
Model:                            OLS   Adj. R-squared:                  0.841
Method:                 Least Squares   F-statistic:                     59.93
Date:                Fri, 25 Apr 2025   Prob (F-statistic):           2.93e-45
Time:                        12:18:59   Log-Likelihood:                -56.051
No. Observations:                 135   AIC:                             138.1
Df Residuals:                     122   BIC:                             175.9
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const             -2.7469      0.971     -2.830      0.005      -4.669      -0.825
Acq_Expense        0.0087      0.003      3.018      0.003       0.003       0.014
Acq_Expense_SQ -1.035e-05   2.65e-06     -3.902      0.000   -1.56e-05    -5.1e-06
Ret_Expense        0.0036      0.000      8.791      0.000       0.003       0.004
Ret_Expense_SQ -6.763e-07   1.12e-07     -6.043      0.000   -8.98e-07   -4.55e-07
First_Purchase     0.0026      0.001      2.370      0.019       0.000       0.005
Crossbuy           0.2028      0.020     10.072      0.000       0.163       0.243
Frequency          0.1539      0.034      4.504      0.000       0.086       0.221
Frequency_SQ      -0.0048      0.002     -2.657      0.009      -0.008      -0.001
Industry           0.5649      0.079      7.194      0.000       0.409       0.720
Revenue            0.0082      0.004      2.213      0.029       0.001       0.016
Employees          0.0002      0.000      0.619      0.537      -0.000       0.001
IMR                0.5941      0.523      1.135      0.259      -0.442       1.630
==============================================================================
Omnibus:                        0.454   Durbin-Watson:                   1.634
Prob(Omnibus):                  0.797   Jarque-Bera (JB):                0.170
Skew:                           0.046   Prob(JB):                        0.918
Kurtosis:                       3.147   Cond. No.                     1.13e+08
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.13e+08. This might indicate that there are
```

Logistic Regression: Customer Repurchase Prediction

```
#Import Required Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    classification_report,
    roc_auc_score,
    roc_curve
)


#loading datasets

df = pd.read_excel('/content/drive/MyDrive/Chapter 4 - Customer Retention Data (1).xls')
df.head()
df_transactions = pd.read_excel('/content/drive/MyDrive/Chapter 4 - Customer Retention Data (1).xls', sheet_name=0)
df_transactions.head()
df_demographics = pd.read_excel('/content/drive/MyDrive/Chapter 4 - Customer Retention Data (1).xls', sheet_name=1)
df_demographics.head()
```

	Customer	Gender	Married	Income	First_Purchase	Loyalty	SOW	CLV
0	1	1	0	4	33.55	1	46	759.12
1	2	1	0	3	94.01	1	28	160.77
2	3	1	0	3	157.55	1	30	486.42
3	4	0	0	2	211.49	0	82	1983.20
4	5	0	0	4	165.22	1	100	2445.97

Next steps: Generate code with df_demographics | View recommended plots | New interactive sheet

```
#Merge Datasets on 'Customer'
df = pd.merge(df_transactions, df_demographics, on='Customer', how='inner')

#Basic Data Overview
print(" Dataset Overview:")
print(df.head())
print("\n Dataset Info:")
print(df.info())
```

```
 Dataset Overview:
   Customer  Quarter  Purchase  Order_Quantity  Crossbuy  Ret_Expense  \
0         1        1         1          285.46         1          0.0
1         1        2         1          363.88         1          0.0
2         1        3         1          305.86         1         16.2
3         1        4         1          260.63         1          0.0
4         1        5         0            0.00         0          0.0

   Ret_Expense_SQ  Gender  Married  Income  First_Purchase  Loyalty  SOW  \
0            0.00       1        0       4           33.55        1   46
1            0.00       1        0       4           33.55        1   46
2          262.44       1        0       4           33.55        1   46
3            0.00       1        0       4           33.55        1   46
4            0.00       1        0       4           33.55        1   46

      CLV
0  759.12
1  759.12
2  759.12
3  759.12
4  759.12

 Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
```

```
 0   Customer        6000 non-null   int64
 1   Quarter         6000 non-null   int64
 2   Purchase        6000 non-null   int64
 3   Order_Quantity  6000 non-null   float64
 4   Crossbuy        6000 non-null   int64
 5   Ret_Expense     6000 non-null   float64
 6   Ret_Expense_SQ  6000 non-null   float64
 7   Gender          6000 non-null   int64
 8   Married         6000 non-null   int64
 9   Income          6000 non-null   int64
 10  First_Purchase  6000 non-null   float64
 11  Loyalty         6000 non-null   int64
 12  SOW             6000 non-null   int64
 13  CLV             6000 non-null   float64
dtypes: float64(5), int64(9)
memory usage: 656.4 KB
None
```

```
#Create Lag Feature
# Lag_Purchase indicates whether customer purchased in the previous quarter

df["Lag_Purchase"] = df.groupby("Customer")["Purchase"].shift(1).fillna(0).astype(int)
```

```
#Define Features and Target

features = ["Lag_Purchase", "Order_Quantity", "Ret_Expense", "Ret_Expense_SQ",
            "Gender", "Married", "Income", "First_Purchase", "Loyalty", "SOW", "CLV"]
X = df[features]
y = df["Purchase"]
```

```
#Split Data into Train and Test Sets

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
#Standardize Features

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
print("\n Standardization complete.")
```

```
Standardization complete.
```

```
#Train Logistic Regression Model

model = LogisticRegression(solver='liblinear', random_state=42)
model.fit(X_train, y_train)
print(" Logistic Regression model training complete!")
```

```
Logistic Regression model training complete!
```

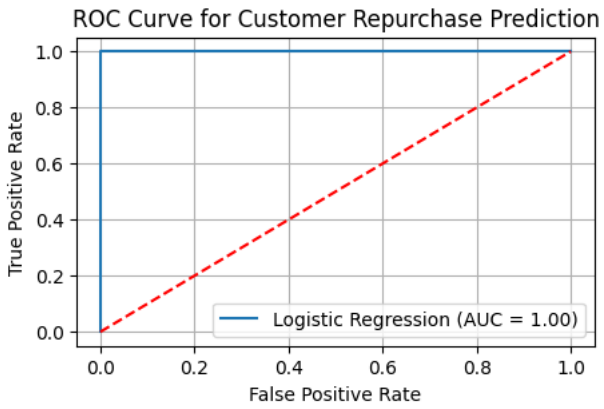
```
#Predict on Test Data

y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]
```

```
#Evaluate Model Performance

accuracy = accuracy_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("\n Model Performance Metrics:")
print(f" Accuracy: {accuracy:.4f}")
print(f" AUC-ROC Score: {roc_auc:.4f}")
print("\n Confusion Matrix:")
print(conf_matrix)
print("\n Classification Report:")
print(class_report)
```

```
Model Performance Metrics:
Accuracy: 0.9975
AUC-ROC Score: 1.0000
```

```
Confusion Matrix:
[[566   0]
 [  3 631]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       566
           1       1.00      1.00      1.00       634

    accuracy                           1.00      1200
   macro avg       1.00      1.00      1.00      1200
weighted avg       1.00      1.00      1.00      1200
```

```
#ROC Curve Plot

fpr, tpr, _ = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(5, 3))
plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], linestyle='dashed', color='red')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Customer Repurchase Prediction')
plt.legend()
plt.grid(True)
plt.show()

# Step 14: Create Final Prediction DataFrame
customers_test = df.iloc[y_test.index]["Customer"].values
quarters_test = df.iloc[y_test.index]["Quarter"].values

predictions_df = pd.DataFrame({
    "Customer_ID": customers_test,
    "Quarter": quarters_test,
    "Actual_Purchase": y_test.values,
    "Predicted_Purchase": y_pred,
    "Predicted_Probability": y_pred_proba
})

print("\n Sample Predictions:")
print(predictions_df.head())
```




```
 Sample Predictions:
   Customer_ID  Quarter  Actual_Purchase  Predicted_Purchase  \
0          149        7                0                   0
1          327        6                0                   0
2           19        6                0                   0
3          178       12                0                   0
4          436        5                1                   1

   Predicted_Probability
0               0.003127
1               0.014047
2               0.010536
3               0.012532
4               0.999998
```

```
#Save Predictions to CSV

output_path = "/content/drive/MyDrive/Customer_Repurchase_Predictions.csv"
predictions_df.to_csv(output_path, index=False)
print(f"\n Predictions saved to: {output_path}")
```

KNN for SOW - Store Prediction

Import Libraries

```
# === Import Required Libraries === #
import pandas as pd
import numpy as np
import random

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
```

Load Dataset

```
# === Load the CSV Data === #
file_path = "/content/drive/MyDrive/Customer_Store_Purchase_Data__300_rows_.csv"
df = pd.read_csv(file_path)

# Preview data
df.head()
```

	Customer_ID	Store_ID	Purchase_Amount
0	C0399	Store_2	202.62
1	C0179	Store_1	145.50
2	C0051	Store_2	241.32
3	C0170	Store_2	106.76
4	C0249	Store_4	31.10

Next steps: Generate code with df | View recommended plots | New interactive sheet

Aggregate Customer Features

```
# === Aggregate customer-level purchase history === #
agg_df = df.groupby('Customer_ID').agg(
    Spend_Sum=('Purchase_Amount', 'sum'),
    Spend_Avg=('Purchase_Amount', 'mean'),
    Visit_Count=('Store_ID', 'count')
).reset_index()

# Identify most frequently visited store per customer
store_freq = df.groupby(['Customer_ID', 'Store_ID']).size().reset_index(name='Freq')
store_freq_sorted = store_freq.sort_values(['Customer_ID', 'Freq'], ascending=[True, False])
main_store = store_freq_sorted.drop_duplicates('Customer_ID')

# Merge target store with aggregated features
final_df = pd.merge(agg_df, main_store[['Customer_ID', 'Store_ID']], on='Customer_ID')
final_df.head()
```

	Customer_ID	Spend_Sum	Spend_Avg	Visit_Count	Store_ID
0	C0001	91.91	91.910	1	Store_4
1	C0003	114.04	114.040	1	Store_3
2	C0006	61.16	30.580	2	Store_2
3	C0008	351.75	175.875	2	Store_1
4	C0009	225.43	112.715	2	Store_3

Next steps: Generate code with final_df | View recommended plots | New interactive sheet

Encode Target and Split Data

```
# === Encode Store_ID as labels === #
label_encoder = LabelEncoder()
final_df['Store_Label'] = label_encoder.fit_transform(final_df['Store_ID'])

# Features and target
X = final_df[['Spend_Sum', 'Spend_Avg', 'Visit_Count']]
y = final_df['Store_Label']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
```

Scale and Train KNN Model

```
# === Feature scaling and model training === #
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_scaled, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

Predict and Evaluate

```
# === Prediction and Evaluation === #
y_pred = knn_model.predict(X_test_scaled)
predicted_store = label_encoder.inverse_transform(y_pred)
true_store = label_encoder.inverse_transform(y_test)

# Evaluation
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred, target_names=label_encoder.classes_))
```

```
Accuracy Score: 0.2391304347826087

Classification Report:

              precision    recall  f1-score   support

     Store_1       0.24      0.38      0.29        13
     Store_2       0.27      0.33      0.30         9
     Store_3       0.40      0.22      0.29         9
     Store_4       0.00      0.00      0.00         8
     Store_5       0.25      0.14      0.18         7

    accuracy                           0.24        46
   macro avg       0.23      0.22      0.21        46
weighted avg       0.24      0.24      0.23        46
```

Display Prediction Results

```
# === Create and display a result dataframe === #
result_df = pd.DataFrame({
    'Customer_ID': final_df.iloc[y_test.index]['Customer_ID'].values,
    'Actual_Store': true_store,
    'Predicted_Store': predicted_store
})
result_df.head()
```

	Customer_ID	Actual_Store	Predicted_Store
0	C0093	Store_3	Store_5
1	C0130	Store_3	Store_1
2	C0149	Store_1	Store_1
3	C0022	Store_2	Store_1
4	C0095	Store_1	Store_1

Artificial Neural Network for Churn Prediction

```
# Install the xlrd package to read .xls files
!pip install xlrd
```

```
Requirement already satisfied: xlrd in /usr/local/lib/python3.11/dist-packages (2.0.1)
```

Import Libraries

```
# Import all required libraries
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, accuracy_score

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

Load and Preprocess Dataset python Copy

```
# Load the dataset
file_path = "/content/drive/MyDrive/Chapter 6 - Customer Churn Data.xls"
df = pd.read_excel(file_path)
print(df.head())

# Drop missing values
df.dropna(inplace=True)

# Encode categorical (string) columns using Label Encoding
for col in df.select_dtypes(include='object').columns:
    df[col] = LabelEncoder().fit_transform(df[col])

# Create binary target variable for churn prediction
df['Churn'] = 1 - df['Censor']  # Churn (1) = censor = 0, Active (0) = censor = 1

# Define features (X) and target (y)
X = df.drop(columns=['Censor', 'Churn'])  # Use all other features
y = df['Churn']
```

```
   Customer  Duration  Censor  Avg_Ret_Exp  Avg_Ret_Exp_SQ  Industry  Revenue  \
0         1       500       0        89.61       8029.9521         1    30.16
1         2       730       1        49.89       2489.0121         0    39.80
2         3       730       1        40.70       1656.4900         0    54.93
3         4       340       0        85.76       7354.7776         0    45.83
4         5       730       1        31.90       1017.6100         0    69.03

   Employees  Total_Crossbuy  Total_Freq  Total_Freq_SQ
0       1240               6          16            256
1        166               6          10            100
2       1016               2          14            196
3        122               2          15            225
4        313               1           9             81
```

Split and Scale Data

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Feature scaling (important for Neural Networks)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Build the Neural Network Model

```
# Define the neural network architecture
model = Sequential([
```

```
    Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')  # Binary classification
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Display model summary
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
dense_9 (Dense)	(None, 128)	1,408
dropout_6 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 64)	8,256
dropout_7 (Dropout)	(None, 64)	0
dense_11 (Dense)	(None, 1)	65

Total params: 9,729 (38.00 KB)
Trainable params: 9,729 (38.00 KB)
Non-trainable params: 0 (0.00 B)

Train the Neural Network

```
# Train the model
history = model.fit(X_train_scaled, y_train, epochs=50, batch_size=32, validation_split=0.2)
```

```
Epoch 1/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.5872 - loss: 0.6546 - val_accuracy: 0.8750 - val_loss: 0.5575
Epoch 2/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.7525 - loss: 0.5706 - val_accuracy: 0.8625 - val_loss: 0.4736
Epoch 3/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 0.8293 - loss: 0.4832 - val_accuracy: 0.8875 - val_loss: 0.4031
Epoch 4/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.8363 - loss: 0.4250 - val_accuracy: 0.9125 - val_loss: 0.3449
Epoch 5/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - accuracy: 0.8873 - loss: 0.3648 - val_accuracy: 0.9250 - val_loss: 0.2990
Epoch 6/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 61ms/step - accuracy: 0.8373 - loss: 0.3656 - val_accuracy: 0.9000 - val_loss: 0.2654
Epoch 7/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 54ms/step - accuracy: 0.9115 - loss: 0.2819 - val_accuracy: 0.9000 - val_loss: 0.2387
Epoch 8/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 58ms/step - accuracy: 0.9059 - loss: 0.2536 - val_accuracy: 0.9500 - val_loss: 0.2174
Epoch 9/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 61ms/step - accuracy: 0.9129 - loss: 0.2447 - val_accuracy: 0.9250 - val_loss: 0.2061
Epoch 10/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.8916 - loss: 0.2447 - val_accuracy: 0.9500 - val_loss: 0.1941
Epoch 11/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.8927 - loss: 0.2232 - val_accuracy: 0.9500 - val_loss: 0.1845
Epoch 12/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 50ms/step - accuracy: 0.9233 - loss: 0.1959 - val_accuracy: 0.9500 - val_loss: 0.1760
Epoch 13/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 75ms/step - accuracy: 0.8973 - loss: 0.2364 - val_accuracy: 0.9500 - val_loss: 0.1716
Epoch 14/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 59ms/step - accuracy: 0.9239 - loss: 0.1914 - val_accuracy: 0.9500 - val_loss: 0.1643
Epoch 15/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 55ms/step - accuracy: 0.9265 - loss: 0.2127 - val_accuracy: 0.9500 - val_loss: 0.1562
Epoch 16/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 76ms/step - accuracy: 0.9477 - loss: 0.1422 - val_accuracy: 0.9500 - val_loss: 0.1505
Epoch 17/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 77ms/step - accuracy: 0.9426 - loss: 0.1794 - val_accuracy: 0.9500 - val_loss: 0.1474
Epoch 18/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.9204 - loss: 0.1883 - val_accuracy: 0.9500 - val_loss: 0.1432
Epoch 19/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - accuracy: 0.9366 - loss: 0.1731 - val_accuracy: 0.9500 - val_loss: 0.1384
Epoch 20/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 49ms/step - accuracy: 0.9076 - loss: 0.1915 - val_accuracy: 0.9500 - val_loss: 0.1374
Epoch 21/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.9098 - loss: 0.1713 - val_accuracy: 0.9500 - val_loss: 0.1326
Epoch 22/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step - accuracy: 0.9432 - loss: 0.1413 - val_accuracy: 0.9500 - val_loss: 0.1258
Epoch 23/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - accuracy: 0.9479 - loss: 0.1441 - val_accuracy: 0.9500 - val_loss: 0.1236
```

```
Epoch 24/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - accuracy: 0.9434 - loss: 0.1329 - val_accuracy: 0.9500 - val_loss: 0.1201
Epoch 25/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9294 - loss: 0.1302 - val_accuracy: 0.9625 - val_loss: 0.1182
Epoch 26/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.9415 - loss: 0.1288 - val_accuracy: 0.9625 - val_loss: 0.1156
Epoch 27/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.9645 - loss: 0.1090 - val_accuracy: 0.9375 - val_loss: 0.1173
Epoch 28/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.9333 - loss: 0.1748 - val_accuracy: 0.9500 - val_loss: 0.1106
Epoch 29/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 55ms/step - accuracy: 0.9572 - loss: 0.1293 - val_accuracy: 0.9500 - val_loss: 0.1063
```

Evaluate the Model

```
# Predict on test data
y_pred_prob = model.predict(X_test_scaled)
y_pred = (y_pred_prob > 0.5).astype(int)

# Evaluate the model
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```
WARNING:tensorflow:5 out of the last 9 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distribu
1/4 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/stepWARNING:tensorflow:6 out of the last 12 calls to <function TensorFlowTrainer.make_predict_funct
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
Accuracy Score: 0.95

Classification Report:

              precision    recall  f1-score   support

           0       0.92      1.00      0.96        54
           1       1.00      0.89      0.94        46

    accuracy                           0.95       100
   macro avg       0.96      0.95      0.95       100
weighted avg       0.95      0.95      0.95       100
```

Results and Visualization

```
# Create and display a result dataframe
result_df = pd.DataFrame({
    'Customer': df.iloc[y_test.index]['Customer'].values,
    'Actual_Store': y_test.values,
    'Predicted_Store': y_pred.flatten()
})
result_df.head()
```

	Customer	Actual_Store	Predicted_Store
0	301	1	1
1	365	0	0
2	490	1	1
3	250	0	0
4	428	0	0

Hazard Model for Churn Duration

```
# Step 1: Install required libraries
!pip install lifelines --quiet
!pip install tensorflow --quiet
```

```
# Step 1: Import Libraries
import pandas as pd
import numpy as np
from lifelines import LogNormalAFTFitter
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error
```

```
# Step 2: Load the dataset
file_path = '/content/drive/MyDrive/Chapter 6 - Customer Churn Data.xls'
df = pd.read_excel(file_path)
df.head()
```

	Customer	Duration	Censor	Avg_Ret_Exp	Avg_Ret_Exp_SQ	Industry	Revenue	Employees	Total_Crossbuy	Total_Freq	Total_Freq_SQ
0	1	500	0	89.61	8029.9521	1	30.16	1240	6	16	256
1	2	730	1	49.89	2489.0121	0	39.80	166	6	10	100
2	3	730	1	40.70	1656.4900	0	54.93	1016	2	14	196
3	4	340	0	85.76	7354.7776	0	45.83	122	2	15	225
4	5	730	1	31.90	1017.6100	0	69.03	313	1	9	81

```
# Check column names
print(df.columns)
```

```
Index(['Customer', 'Duration', 'Censor', 'Avg_Ret_Exp', 'Avg_Ret_Exp_SQ',
       'Industry', 'Revenue', 'Employees', 'Total_Crossbuy', 'Total_Freq',
       'Total_Freq_SQ'],
      dtype='object')
```

```
# Step 3: Preprocessing
# Handle missing values by filling them with mean values
df.fillna(df.mean(), inplace=True)

# Select features for prediction
X = df[['Avg_Ret_Exp', 'Avg_Ret_Exp_SQ', 'Industry', 'Revenue', 'Employees',
        'Total_Crossbuy', 'Total_Freq', 'Total_Freq_SQ']]  # Independent variables
y = df['Duration']  # Dependent variable: Duration before churn or censor

# Censoring variable: 1 if the customer stayed, 0 if churned
censor = df['Censor']

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 4: Train-Test Split
X_train, X_test, y_train, y_test, censor_train, censor_test = train_test_split(
    X_scaled, y, censor, test_size=0.2, random_state=42)
```

```
# Step 3: Preprocessing (Ensure no NaN values)
# Create train_df from X_train, y_train, and censor_train
train_df = pd.DataFrame(X_train, columns=X.columns)  # Create DataFrame from X_train
train_df['Duration'] = y_train  # Add Duration column
train_df['Censor'] = censor_train  # Add Censor column

# Check if there are any NaN values in the dataset
if train_df.isnull().sum().any():
    print("NaN values detected in the training dataset, filling with mean values.")
    # Fill NaN values with the mean of the column
    train_df.fillna(train_df.mean(), inplace=True)

# Verify that no NaN values remain
print("Missing values after filling:", train_df.isnull().sum().sum())

# Step 5: Fit the Log-Normal AFT Model
```

```
aft_model = LogNormalAFTFitter(penalizer=0.01)

# Fit the model using the training data
aft_model.fit(train_df, duration_col='Duration', event_col='Censor')

# Step 6: Predict Duration for Churned Customers
# Create a DataFrame for prediction with the test data
test_df = pd.DataFrame(X_test, columns=X.columns)

# Predict the expected duration (time to churn)
predicted_duration = aft_model.predict_expectation(test_df)

# Print predicted duration for first few test samples
print(predicted_duration.head())
```

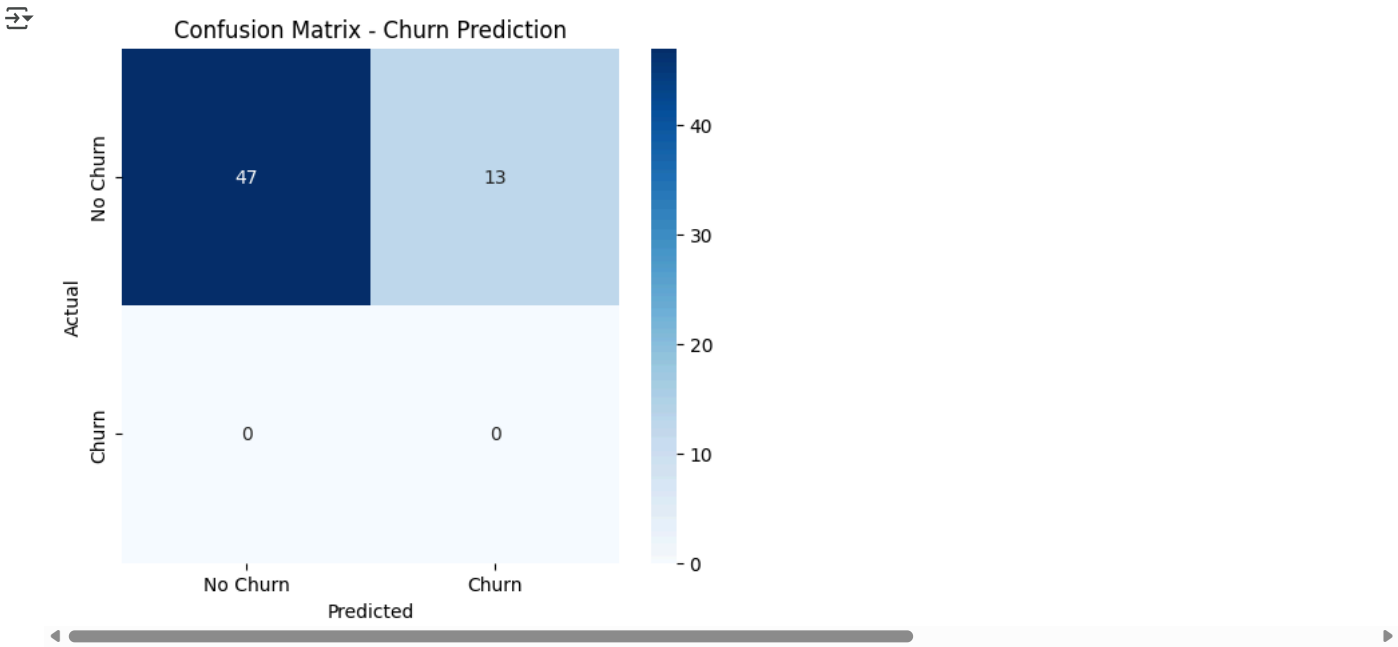
```
NaN values detected in the training dataset, filling with mean values.
Missing values after filling: 0
0    700.732465
1    704.621507
2    702.015826
3    709.003220
4    713.809169
dtype: float64
```

```
# Step 7: Evaluate the Model

# Align the predicted_duration with the indices of y_test to avoid the IndexingError
predicted_duration = pd.Series(predicted_duration, index=y_test.index)  # Ensures proper alignment

# Compare predicted duration with actual duration for churned customers
# Using .iloc to ensure that we only consider churned customers
predicted_duration_churned = predicted_duration[censor_test == 1]
actual_duration_churned = y_test[censor_test == 1]

# Calculate Mean Absolute Error (MAD) for churned customers
mad = np.mean(np.abs(predicted_duration_churned - actual_duration_churned))
print(f"Mean Absolute Error (MAD) for churned customers: {mad:.2f} days")

# Calculate Mean Absolute Percentage Error (MAPE) for churned customers
mape = np.mean(np.abs((predicted_duration_churned - actual_duration_churned) / actual_duration_churned)) * 100
print(f"Mean Absolute Percentage Error (MAPE) for churned customers: {mape:.2f}%")

# Step 8: Compare Actual vs Predicted Duration for Churned Customers
# Create a DataFrame to compare
comparison_df = pd.DataFrame({
    'Actual_Duration': actual_duration_churned,
    'Predicted_Duration': predicted_duration_churned
})

# Print first few rows of the comparison
print("\nActual vs Predicted Duration for Churned Customers:")
print(comparison_df.head())

# Step 9: Further Analysis - Classification of Churn Prediction
# Convert continuous predicted durations to binary classification (churn or not)
predicted_churn = (predicted_duration_churned < 730).astype(int)  # Threshold at 730 days (censored value)
actual_churn = (actual_duration_churned < 730).astype(int)  # True churn (1) or not (0)

# Calculate Confusion Matrix and Classification Report
from sklearn.metrics import confusion_matrix, classification_report

# Confusion Matrix
print("\nConfusion Matrix:")
print(confusion_matrix(actual_churn, predicted_churn))

# Classification Report
print("\nClassification Report:")
print(classification_report(actual_churn, predicted_churn))
```

```
Mean Absolute Error (MAD) for churned customers: 28.44 days
Mean Absolute Percentage Error (MAPE) for churned customers: 3.90%

Actual vs Predicted Duration for Churned Customers:
     Actual_Duration  Predicted_Duration
361              730                 NaN
73               730          702.265653
374              730                 NaN
155              730                 NaN
104              730                 NaN

Confusion Matrix:
```

```
[[47 13]
 [ 0  0]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.78      0.88        60
           1       0.00      0.00      0.00         0

    accuracy                           0.78        60
   macro avg       0.50      0.39      0.44        60
weighted avg       1.00      0.78      0.88        60

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and b
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and b
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-defined and b
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Step 9: Confusion Matrix Visualization
# Generate confusion matrix
cm = confusion_matrix(actual_churn, predicted_churn)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', xticklabels=['No Churn', 'Churn'], yticklabels=['No Churn', 'Churn'])

# Add labels, title, and axis
plt.title("Confusion Matrix - Churn Prediction")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```




```
# Step 3: Predict Duration for Non-Churned Customers (Censor = 0)
predicted_duration_non_churned = predicted_duration[censor_test == 0]
actual_duration_non_churned = y_test[censor_test == 0]

# Calculate Mean Absolute Error (MAD) for non-churned customers
mad_non_churned = np.mean(np.abs(predicted_duration_non_churned - actual_duration_non_churned))
print(f"Mean Absolute Error (MAD) for non-churned customers: {mad_non_churned:.2f} days")

# Calculate Mean Absolute Percentage Error (MAPE) for non-churned customers
mape_non_churned = np.mean(np.abs((predicted_duration_non_churned - actual_duration_non_churned) / actual_duration_non_churned)) * 100
print(f"Mean Absolute Percentage Error (MAPE) for non-churned customers: {mape_non_churned:.2f}%")
```

```
Mean Absolute Error (MAD) for non-churned customers: 204.42 days
Mean Absolute Percentage Error (MAPE) for non-churned customers: 90.62%
```

Artificial Neural Network for SCLV

```
!pip install -q xlrd scikit-learn
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
from google.colab import files
```

```
# Load the dataset
file_path = "/content/drive/MyDrive/Chapter 7 - Customer Win-back Data.xls"  # Replace with actual path in Colab
df = pd.read_excel(file_path)

df.head()
```

	Customer	Reacquire	Duration_2	SCLV	Duration_1	Offer	Duration_lapse	Price_Change	Gender	Age
0	1	1	766	1404.899622	609	30	152	-8.97	0	37
1	2	1	405	874.556786	411	20	73	16.66	1	32
2	3	0	0	-5.000000	205	30	170	9.13	1	54
3	4	1	623	1155.006000	664	20	120	9.85	0	69
4	5	0	0	-5.000000	227	25	161	-42.15	1	67

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
y = df["SCLV"]                          # target
X = pd.get_dummies(df.drop(columns="SCLV"), drop_first=True)  # one-hot

scaler   = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_tr, X_te, y_tr, y_te = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42
)
```

```
mlp = MLPRegressor(
    hidden_layer_sizes=(128, 64),
    activation="relu",
    solver="adam",
    learning_rate_init=0.001,
    max_iter=500,
    random_state=42,
    early_stopping=True,
    n_iter_no_change=20
)
mlp.fit(X_tr, y_tr)
```

```
MLPRegressor(early_stopping=True, hidden_layer_sizes=(128, 64), max_iter=500,
             n_iter_no_change=20, random_state=42)
```

```
print("Layers (input + hidden + output):", mlp.n_layers_)
print("Hidden layer sizes:", mlp.hidden_layer_sizes)
print("Iterations run:", mlp.n_iter_)
print("Final training loss:", mlp.loss_)
```

```
Layers (input + hidden + output): 4
Hidden layer sizes: (128, 64)
Iterations run: 354
Final training loss: 1799.0580390422147
```

```
y_pred = mlp.predict(X_te)
mse  = mean_squared_error(y_te, y_pred)
mae  = mean_absolute_error(y_te, y_pred)
print(f"Test MSE {mse:.4f}   MAE {mae:.4f}")
```

```
Test MSE 6616.6323   MAE 58.3965
```

```
# Cell 8 – R² “accuracy” for regression
from sklearn.metrics import r2_score

r2 = r2_score(y_te, y_pred)
print(f"Test R² (variance explained): {r2:.4f}")
```

```
Test R² (variance explained): 0.9884
```

Start coding or generate with AI.

6. SUMMARY AND OUTLOOK

This thesis has systematically bridged the gap between marketing theory and quantitative modeling. Through rigorous mathematical understanding of statistical, econometric, machine learning, and deep learning techniques, the thesis systematically addressed key marketing objectives: customer acquisition, retention, churn, and win-back strategies.

Each stage of the customer lifecycle was matched with appropriate modeling approaches—logistic regression for acquisition, survival and hazard models for churn and retention timing, and neural networks for predicting reacquisition. Tools like the Tobit model addressed censoring in Customer Lifetime Value (CLV) estimation, while methods such as KNN and ANN illustrated practical deployment of these analytics.

Empirical validation using real-world data confirmed the models' relevance and performance. By operationalizing insights into targeted CRM interventions, this thesis emphasizes how firms can use analytics to balance investments across acquisition and retention, ultimately optimizing customer profitability.

Despite these, challenges persist including data dependency, static behavioral assumptions, and computational demands. Future work should consider dynamic, real-time learning, incorporate causal inference frameworks, and integrate ethical standards to guide automated decisions.

In conclusion, mathematical modeling is not just a theoretical pursuit but it is essential for strategic decision-making. This work provides a foundation for firms to evolve from product-centric operations to data-driven, customer-centric strategies that foster sustainable growth and competitive advantage.

BIBLIOGRAPHY

[1] Viba Kumar and J Andrew Petersen. *Statistical methods in customer relationship management*. John Wiley & Sons, 2012.

[2] Philip Kotler, Kevin Lane Keller, Mairead Brady, Malcolm Goodman, and Torben Hansen. *Marketing Management 3rd edn PDF eBook*. Pearson Higher Ed, 2016.

[3] Michael John Baker and Susan J Hart. *The marketing book*, volume 195. Butterworth-Heinemann Oxford, 2003.

[4] OpenStax. *Principles of Marketing*. Rice University, 2023. Access for free at openstax.org.

[5] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, 2014.

[6] Achim Klenke. *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013.

[7] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. Available at http://sebastianruder.com/optimizing-gradient-descent/.

[8] Peter McCullagh and John A Nelder. *Generalized Linear Models*. Chapman and Hall/CRC, 2nd edition, 1989.

[9] Erich L Lehmann and George Casella. *Theory of Point Estimation*. Springer, 2nd edition, 2006.

[10] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

[11] Kenneth E. Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2nd edition, 2009.

[12] Siddhartha Chib. Markov chain monte carlo methods: Computation and inference. In *Handbook of Econometrics*, volume 5, pages 3569–3649. Elsevier, 2001.

[13] Stephen P Brooks. Markov chain monte carlo method and its application. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(1):69–100, 1998.

[14] Bruce Walsh. Markov chain monte carlo and gibbs sampling: Lecture notes for eeb 581. https://nitro.biosci.arizona.edu/zdownload/mcmc.pdf, 2004. Version: April 26, 2004.

[15] Christian P Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Science & Business Media, 2004.

[16] Art B Owen. *Monte Carlo Theory, Methods and Examples*. 2013. https://statweb.stanford.edu/~owen/mc/.

[17] Luke Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728, 1994.

[18] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[19] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[20] Alan E Gelfand and Adrian F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990.

[21] George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[22] James Tobin. Estimation of relationships for limited dependent variables. *Econometrica: Journal of the Econometric Society*, 26(1):24–36, 1958.

[23] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[24] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[25] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[26] Kevin Gurney. *An introduction to neural networks*. CRC press, 2018.

[27] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[28] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[29] David G. Kleinbaum and Mitchel Klein. *Survival Analysis: A Self-Learning Text*. Springer Science & Business Media, 2nd edition, 2005.

[30] John D. Kalbfleisch and Ross L. Prentice. *The Statistical Analysis of Failure Time Data*. John Wiley & Sons, 2nd edition, 2011.

[31] Rupert G. Miller. *Survival Analysis*. John Wiley & Sons, 2011.

[32] Jerald F. Lawless. *Statistical Models and Methods for Lifetime Data*. John Wiley & Sons, 2011.

[33] D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

[34] Paul D. Allison. Discrete-time methods for the analysis of event histories. *Sociological Methodology*, 13:61–98, 1982.

[35] Shenyang Guo and Mark W. Fraser. *Survival Analysis: A Practical Approach*. Oxford University Press, 2011.

[36] Francis Buttle and Stan Maklan. *Customer Relationship Management: Concepts and Technologies*. Routledge, 2015.

[37] Harvard Business Review. The value of keeping the right customers, 2014.

[38] EWT Ngai. Customer relationship management research (1992–2002): An academic literature review and classification. *Marketing Intelligence & Planning*, 23(6):582–605, 2005.

[39] Werner J. Reinartz, Manfred Krafft, and Wayne D. Hoyer. The customer relationship management process: Its measurement and impact on performance. *Journal of Marketing Research*, 41(3):293–305, 2004.

[40] Julian Villanueva, Shijin Yoo, and Dominique M. Hanssens. The impact of marketing-induced versus word-of-mouth customer acquisition on customer equity growth. *Journal of Marketing Research*, 45(1):48–59, 2008.

[41] Behram Hansotia and Pan Wang. Analytical crm: Making it happen. *Customer Inter@ction Solutions*, 16(7):24–28, 1997.

[42] Julian Villanueva and Dominique M Hanssens. Customer equity: Measurement, management and research opportunities. *Foundations and Trends in Marketing*, 1(1):1–95, 2008.

[43] B. Hansotia and P. Wang. Analytical modeling for customer acquisition strategies. *Journal of Database Marketing & Customer Strategy Management*, 5(1):29–36, 1997.

[44] Werner Reinartz and V. Kumar. The impact of customer relationship characteristics on profitable lifetime duration. *Journal of Marketing*, 67(1):77–99, 2003.

[45] Robert C. Blattberg, Byung-Do Kim, and Scott A. Neslin. *Database Marketing: Analyzing and Managing Customers*. Springer, 2009.

[46] C.B. Bhattacharya. When customers go rogue: Modeling customer defection behavior. *Journal of Marketing Research*, 45(3):335–347, 2008.

[47] Ran Kivetz, Oleg Urminsky, and Yuhuang Henry Zheng. Ego depletion and goal gradient in the self-regulation of vicarious goal pursuits. *Journal of Consumer Research*, 33(4):540–552, 2006.

[48] P. B. Seetharaman and Pradeep K. Chintagunta. Consumer preference structures for multiattribute products: A hierarchical bayes application. *Marketing Science*, 22(2):184–195, 2003.

[49] Sharad Borle, Siddharth S Singh, Dipak C Jain, and Neeraj Arora. The impact of email marketing on customer profitability: Evidence from a field experiment. *Information Systems Research*, 19(3):370–389, 2008.

[50] Thomas Cover and Peter Hart. The nearest neighbor algorithm. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[51] Werner Reinartz and V. Kumar. Using customer lifetime value to segment retail customers. *Journal of Marketing*, 69(1):50–60, 2008.

[52] Amy Gallo. The value of keeping the right customers. *Harvard Business Review*, 2014.

[53] Ramon A. Bolton and James B. K. C. Does customer satisfaction really matter? *Journal of Marketing*, 64(2):15–24, 2000.

[54] et al. Edward C. Malthouse. Customer engagement with company-generated content: The impact of crm and firm's marketing initiatives. *Journal of Marketing Research*, 50(5):657–668, 2013.

[55] Scott A. Neslin, Sunil Gupta, Wagner A. Kamakura, Jiwoong Lu, and Charlotte H. Mason. Defection detection: Measuring and understanding the predictive accuracy of customer churn models. *Journal of Marketing Research*, 43(2):204–211, 2006.

[56] Wouter Verbeke, David Martens, and Bart Baesens. Predictive modeling with big data: Is the hype justified? *Applied Economics*, 44(9):1121–1135, 2012.

[57] Kristin Anderson and Carol Kerr. Customer relationship management: A databased approach. *Journal of Database Marketing & Customer Strategy Management*, 14(3):153–159, 2007.