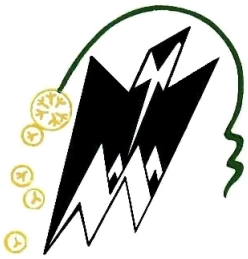
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY MOULOUD MAMMERI OF TIZI-OUZOU



Faculty of Sciences
Department of Mathematics

Master's Thesis
For the attainment of Master's degree in Mathematics

Branch: Operational Research

Topic:

Game theory in Binary Classification: A Nash Equilibrium Decision Tree Model

Presented by: ATLAOUI AMEL

Defended on 07/07/2025 before the committee members composed of:

President:	Mr. Hamaz Abdalghani	Dr.	Univ. UMMTO
Supervisor:	Mrs. Achemine Farida	Pr.	Univ. UMMTO
Examiner:	Mrs. Fahem Karima	Pr	Univ. UMMTO
Examiner:	Mrs. Djebara Sabiha	Dr	Univ. Boumerdes

Academic year: 2024-2025

Dedication

To my dad and mom, words will never fully express the depth of my gratitude for the love, wisdom, and strength you have given me. You are my foundation, my heart, and my Home.

To my sisters and my brother, I'm forever grateful for the love and laughter we share.

To my two aunts, your love is a gift. I cherish every day.

To my little angel, to my precious niece, you are a little star, lighting up the world around you.

To my self. I am my own hero, and I will continue to rise.

You mean everything to me.

Acknowledgments

*To ATLAOUI Family, "Family is not an important thing. It's everything"(**Michel J.Fox**).*

Your love, support, and guidance have been the foundation of my strength. Through every trial your belief in me has helped me become who I am. I carry your love in my heart, always.

*To my teacher **Achemine Farida**, "The influence of a good teacher can never be erased".*

Thank you for your patience, wisdom, and belief in me, your encouragement pushed me beyond my limits, and my guidance was the compass that directed my path.

To myself, I am proud of how far. I have come, and I am excited to keep evolving.

*You are braver than you believe, stronger than you seem, and smarter than you think" (**A.A. Milne**).*

*I would like to express my deepset gratitude to the numbers of the jury " **Mrs Fahem karima , Mr Hamaz abdalghan and Mrs Djebara sabiha**" for their valuable time, insightful comments, and constructive feedback on this work.*

Thank you to everyone who has been part of my work with all my love and gratitude, this is for you.

List of Figures

Contents

list of Anonyms

USA	United States of America
USSR	Union of Soviet Socialist Republics
AI	Artifcial Intelligence
IBM	International Buses Machines
ACC	Classification Accuracy
Err	Classification error rate
RMSE	Root Mean Square Error
RAE	Root Absolute Error
NE	Nash Equilibrium

Notations

$\mathrm{err}(m)$: A error (Variance or mean squared error) for region / leaf m.
g^m : A prediction value for region m.
x_i : A vector sample / a time series.
$\hat{y}_i$: A predicted labele (discrete or Continuous)
y_i : A label (dicrete or Continuous).
$X = \{x_i, y_i\}_{i=1}^{n}$: A training set of $n \in \mathbb{N}$ labeled time series.
f : function to learn.
P : Number of metric measures considered in metric learning.
C : Hyper parameter of trade -off.
V : Number of folds in Goss-Validation.
X_{op} : An operational set of $L \in \mathbb{N}$ labeled time series.
X_{TEST} : A test set of $m \in \mathbb{N}$ labeled time series.

General Introduction

In a modern world where human and economic interactions are becoming increasingly complex, strategic issues are ubiquitous. Game theory, a branch of applied mathematics, provides an analytical framework to model and understand the behaviors of agents in situations of conflict or cooperation. By stating individuals' choices, game theory allows us to anticipate the outcomes of strategic interactions, whether in economic, political, or social contexts. At the same time, artificial intelligence (AI) has emerged as a revolutionary tool. AI can analyze vast amounts of data and simulate complex decision-making processes. The integration of these two disciplines opens new perspectives for solving complex problems and optimizing decision-making.

The field of machine learning is rapidly evolving, with new algorithms enabling the analysis of large and complex datasets. As these capabilities grow, data analysis tools are becoming increasingly relevant. They are applied across diverse fields such as medicine, physics, chemistry, agriculture, economics, and the social sciences. These tools help extract actionable insights and support decision-making. Among these tools, classification methods are widely applied, often using tailored adaptations of classical approaches to suit domain-Specific needs.

Decision trees are particularly popular due to their interpretability and ease of use, providing clear rules that help explain data patterns. However, to improve their accuracy, decision trees are often used as base classifiers in ensemble methods such as boosting and bagging.

Binary classification, the task of categorizing data into two distinct classes, is a cornerstone of machine learning. Its simplicity reduces complexity, facilitates clearer interpretation, and serves as a foundation for more advanced tasks such as multi-class classification.

This project is structured around three main chapters. The first chapter will focus on the fundamental concepts of game theory. We will explore essential concepts such as non-cooperative games and the notion of Nash equilibrium. This theoretical framework will lay the necessary foundations for understanding strategic interactions and modeling agent's behaviors.

In the second chapter, we will dive into the field of machine learning. We will examine different approaches to machine learning including supervised, unsupervised, and reinforcement learning. This chapter will highlight how these techniques can be used to improve strategic decisions by learning from past data

and adapting strategies in real time.

Finally, the last chapter will be dedicated to exploring recent studies that illustrate the joint application of game theory and artificial intelligence. We will analyze concrete case studies where these two fields intersect, highlighting the advantages and challenges associated with this integration.

In conclusion, this project aims to demonstrate that the synergy between game theory and artificial intelligence is not merely a juxtaposition of concepts but constitutes an innovative and promising approach to addressing complex issues in various fields.

CHAPTER 1

BASICS CONCEPTS OF GAME THEORY

1.1 Introduction

According to history, game theory has its roots in the early twentieth century, primarily developed by the mathematician John Von Neumann. His pivotal work began with the publication of the book "Theory of Games and Economic Behavior" in 1944 coauthored with economist Oskar Morgenstern. This work established game theory as a formal mathematical discipline and explored its applications in economics, particularly in understanding competitive behaviors and strategic decision-making. In the years that followed, game theory expanded beyond economics into various fields, including political science, biology, and psychology. The introduction of the Nash Equilibrium concept by John Nash in 1950 further advanced the field, providing a crucial solution concept for non cooperative games where players strategies are independent. Nash's work demonstrated that even in competitive scenarios, stable outcomes could emerge, where no player has an incentive to deviate from their chosen strategy. Throughout the latter half of the 20th century, game theory continued to evolve, with researchers exploring its applications in areas such as negotiation and evolutionary biology. The development of computational game theory in the 21st century has also allowed for more complex models and simulations, making it a vital tool for analyzing strategic interactions in diverse contexts.

In this chapter, we will recall the general concepts of game theory. For more details, see [3, 5, 11, 12].

1.2 Game theory - what is it?

The principles of game theory offer a framework for defining, organizing, analyzing, and ultimately comprehending various strategic situations. Essentially, game theory focuses on scenarios of conflict and the interplay between agents and their choices.

In this context, "game" involves a finite number of participants who interact under specific rules. These participants can be individuals, groups, businesses, or organizations. Their interactions influence not only their individual results, but also the collective outcome of the group, highlighting their interdependence.

To elaborate, a game is defined by its players and the strategies available to them within the established rules. This leads to a comprehensive definition of game theory: it examines situations where the outcome of a player is affected not just by their own choices but also by the decisions made by others.

The theory of games is based on fundamental concepts that are crucial for analyzing strategic interactions. Here are some Key concepts that form the basics of this theory:

- **Players** Players are the participants in a game who make decisions or take actions. These players can be individuals, organizations, nations, or any entity involved in a strategic interaction.
- **Strategies**

  Strategies refer to the possible choices or actions available to each player. A strategy represents a player's action plan, detailing how they will respond to different situations within the game.
- **Payoffs**

  Payoffs are the outcomes or rewards that players receive based on the combinations of strategies chosen by each player.
- **Classes of Games**

  Games can be classified according to certain characteristics.

1.2.1 Cooperative Games and non Cooperative Games

Game can be divided into two categories: cooperative and non-cooperative games.

1. **Cooperative game:** Cooperative games involve negotiating legally enforceable contracts that allow players plan joint state strategies. The most popular cooperative games deals with how coalitions of players might establish and improve their positions in a game.
2. **Non-Cooperative game:** When every participant (player) in the game acts in their own best interests, it is said to be non-cooperative. In these games, each player is a unit analysis. The Cooperative game implies that players may reach certain payoffs by making cooperative agreements and uses groups of players as the unit of analysis. Negotiation and enforcement of legally binding agreements are impossible in non cooperative games.

1.2.2 Games with Perfect Information and Games with Imperfect Information

Definition 1.2.1. *If each player when it is turn to choose a strategy, knows exactly the previous decisions of other players in the game, then the game is said to be a perfect information game. And game is referred to as an imperfect information game if a player has no knowledge about other players actions.*

1.2.3 Games with Complete information and Games with incomplete information

Definition 1.2.2. *when a game has complete information, every player knows every aspect of the game. In other words, every player in the game is totally aware of every other player, their choices and strategies, and their own payoffs. However, in incomplete information games, players lack information about other players, which prevents them from foreseeing how their actions would affect other players.*

1.2.4 Normal Form Games and Extensive form Games

Depending on the order in which players announce their strategies, there are two main forms of representation of the game.

1. **Normal form games (or Strategic form):** A normal form game consists of three objects:
$$\left\langle I, X, \{f_i\}_{i\in I} \right\rangle \tag{1.1}$$
   - I: set of players, each player is characterized by an index $i \in I = \{1, 2, \ldots, N\}, N \geqslant 2$.
   - $X_i \subset \mathbb{R}^{n_i}, i \in \mathbb{N}$: The i^{th} player strategy set and $n_i \in \mathbb{N}$.
   $$x = (x_i, x_{-i}) \in X = \prod_{i\in I} X_i$$
   where X_i is the strategy set of player $i \in I$.
   $$x_{-i} = x_{I-\{i\}} = (x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_N)$$
   situation of the game which contains the strategies of all the players except that of the i^{th} player.
   - $f_i : X \to \mathbb{R}$: the gain (payoff) function of i^{th} player.
   - $x = (x_1, x_2, \ldots, x_N) \in X$ is called a strategy profile. If x is played, player i receives $f_i(x)$.
   - The players choose a strategy simultaneously, an outcome is obtained.

- Each player knows the strategy sets and payoff function of all the other players (complete information).
- The aim of each player is to maximize his payoff.

Definition 1.2.3. *The game in (1.1) is to be a zero-sum game if and only if*

$$\forall x \in X, \sum_{i \in I} f_i(x) = 0.$$

In a zero-sum game one player's gain is exactly balanced by the losses of the other players. In other words, the total payoff for all players remains constant.

Player 1 / Player 2	L	R
U	(1 ; 3)	(2 ; 4)
D	(1 ; 0)	(3 ; 3)

Table 1.1: Payoff Matrix for a Two-Player Normal Form Game

Example 1.2.1. *In this example, player 1 (shown vertically) has two strategies: up* (U) *and Down* (D)*. Player 2 (shown horizontally) has also two Strategies: Left* (L) *and Right* (R)*. The matrix elements represent the payoffs for each combination of strategies (supposing, player 1 chooses strategy* D *and player 2 chooses strategy* L*, the outcome is* $(1; 0)$*. The payoff for player 1 is*1 *and for player 2 is* 0*).*

2. **Extensive Form Games:** In contrast to normal form games, extensive form games are defined by rules that allow players to make their moves one after the other. These games are illustrated using a game tree, where each node represents a potential stage in the game's progression. The initial node indicates the start of the game, while terminal nodes, which have only one edge, signify the end of the game and correspond to specific strategy profiles. Each non terminal node is linked to a player, indicating that it is that player's turn to act. The edges of the tree present the tree represent the possible actions available to players. Each terminal node is associated with payoffs for all players, which reflect the rewards they would receive in the actions leading to that terminal node are carried out.

Example 1.2.2. .

In Figure 1.1. IF player 1 selects strategy U *and player 2 Chooses Strategy* D'*, player 1 will Earn a payoff of 2 and player* 2 *will earn a payoff of* 1*.*

1.3 Game Theory-where Is It Applied?

In simpler terms, game theory is a part of mathematics that help us describe different types of games, we have already seen that Von Neumann applied game

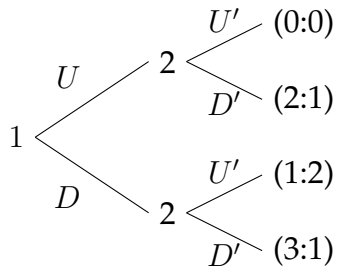


Figure 1.1: A Game in extensive Form

theory to economics. When there is competition for a resource, game theory can either help us understand current behaviors or find better strategies.

The first application is often used in fields that look at long-term situations, such as biology or sociology. For example, in the animal world, we can see cases where cooperation has formed to benefit everyone involved. The second application is especially important in economics, where businesses use game theory to enhance their strategic position in the marketplace.

Even though von Neumann gained significant insights from applying game theory to economics, his main interest was in applying these methods to political and military contexts. He modeled the Cold War dynamics between the U.S. and the USSR, viewing them as two players in a zero-sum game. From this, he developed a mathematical model of the conflict and concluded that the Allies would prevail, using game-theoretic methods to support his predictions.

Moreover, there are numerous applications of game theory across various sciences, including sociology, philosophy, psychology, and cultural anthropology.

1.4 Nash Equilibrium

The Nash Equilibrium is a solution concept in normal form games, it is proposed by John Forbes Nash in 1950 [11]. It is a situation where each player correctly predicts the choices of others, and each player maximize their gain, considering this prediction.

A Nash equilibrium is a set of strategies (one per player) such that no player can obtain additional gain by unilaterally changing their strategy

Definition 1.4.1. *[11] A strategy profile* $(x^* = (x_1^*, x_2^*, \ldots, x_n^*)) \in X$ *is a Nash Equilibrium of the normal for the game (1.1) if and only if:*

$$\forall i \in I, \forall x_i \in X_i \quad f_i\left(x_{-i}, x_i^*\right) \leq f_i\left(x_i^*, x_{-i}^*\right).$$

Remark 1.4.1. *In the case where the players in game (1.1) are minimizing, the inequality*

in Definition 1.5.1 becomes

$$\forall i \in I, \forall x_i \in X_i \quad f_i\left(x_{-i}, x_i^*\right) \geq f_i\left(x_i^*, x_{-i}^*\right).$$

Example 1.4.1. *(The prisoner's dilemma)*

Two criminals, named Alice and Bob, are arrested for theft. The police do not have enough evidence to convict them of the main crime. So, the police offer them a deal:

1. *If both remain silent (cooperate), they each serve one year in prison.*
2. *If one betrays the other (defects) while the other stays silent, the betrayer goes free, and the silent one serves ten years.*
3. *If both betray each other, they each serve five years in prison.*

		Prisoner 2	
		To remain silent (S)	To betray (B)
Prisoner 1	To remain silent (S)	$(-1;-1)$	$(-10;0)$
	To betray (B)	$(0;-10)$	$(-5;-5)$

Table 1.2: A prisoner's dilemma

For Prisoner 1: $U_1(S,S) = -1 < U_1(B,S) = 0$.
For Prisoner 2: $U_2(S,S) = -1 < U_2(S,B) = 0$.
Then (S,S) *is not a Nash Equilibrium. But (* $B;B$ *) is a Nash Equilibrium. In a similar way, we can verify that* (S,B) *and* (B,S) *are not Nash equilibria.*
The Nash equilibrium in this game is (B,B).

1.4.1 Best Response

A Strategy x_i^* is a best response of player i to strategy profile $x_{-i} \in I - \{i\}$ selected by all other players if it gives player i a larger payoff than any of available strategies $x_i \in X_i$ [11] :

$$f_i\left(x_i^*, x_{-i}\right) \geq f_i\left(x_i, x_{-i}\right) \text{ for all } x_i \in X_i.$$

Best response correspondence :

$$C_i(x_{-i}) = \left\{ x_i^* \in X_i / \sup_{x_i \in X_i} f_i(x_i, x_{-i}) = f_i(x_i^*, x_{-i}) \right\}.$$

In the case of two player, N=2, we have two players 1 and 2.
Let x_1 be the strategy set for player 1. Then, strategy $x_1^* \in X_1$ is a best response for player 2 to player 2's strategy X_2 if:

$$f_2\left(x_1^*, x_2\right) \geqslant f_2\left(x_1, x_2\right).$$

Example 1.4.2 (Prisoner's Dilemma (See Example 1.5.1))**.** *The payoff matrix below represents a classic Prisoner's Dilemma, where two players must independently decide whether to cooperate (S) or betray (B). Their payoffs depend on the joint decision:*

Player 1 \ Player 2	***S***	***B***
S	$(-1,-1)$	$(-10,\boxed{0})$
B	$(\boxed{0},-10)$	$(\boxed{\text{-5}},\boxed{\text{-5}})$

Table 1.3: *Payoff Matrix for the Prisoner's Dilemma Game*

$\boxed{\text{Best response}}$ *: Each red value corresponds to a player's best response given the other's strategy.*

So, we have:

- $C_1(S) = B$ *and* $C_1(B) = S$
- $C_2(S) = B$ *and* $C_2(B) = S$

A mutual best response constitutes a Nash equilibrium when both players choose strategies that are optimal given the strategy of the other.

Conclusion: *The Nash equilibrium occurs at* (B, B) *neither player can improve their outcome by deviating alone.*

Definition 1.4.2 (Fixed Point)**.** *we say that* $x \in X$ *is a fixed point of the correspondence* $C(.)$*. iff* $\quad x \in C(x)$*, where*

$$\begin{aligned} C : X &\to 2^X \\ x &\mapsto C(x) = \prod_{i \in I} C_i(x_i). \end{aligned}$$

Lemma 1.4.1. *A game outcome* $\bar{x}$ *is a Nash equilibrium for game (1.1) if and only if* $\bar{x}$ *is a fixed point:* $\bar{x} \in C(\bar{x})$

Proof. Let $\bar{x} \in X$ a Nash equilibrium in game (1.1), then we have the equivalences: $\bar{x}$ is a Nash equilibrium $\Leftrightarrow$

$$\begin{aligned} \forall i \in I, \forall x_i \in X_i, f_i(x_i, \bar{x}_{-i}) &\leqslant f_i(\bar{x}_i, \bar{x}_{-i}) \\ \Leftrightarrow \forall i \in I, \bar{x}_i &\in C_i(\bar{x_{-i}}) \\ \Leftrightarrow \bar{x} &\in C(\bar{x}) \end{aligned}$$

□

1.4.2 Existence Problem of Nash Equilibrium

The existence problem of Nash equilibrium is a difficult mathematical problem. To address this, Nash used a powerful mathematical analysis tool, the fixed point theorem, to establish sufficient conditions for the existence of this equilibrium.

In this paragraph, we will limit ourselves to stating an existence theorem whose demonstration will be based on the notion of the fixed point of a multi-valued application. The theorem providing a sufficient existence conditions of a Nash equilibrium is given below.

Theorem 1.4.3. *[11] [Nash Theorem, 1951] Assume that in the game* (1.1)*, the following conditions are verified:*

1. $X_i, i = \overline{1,n}$*, are non-empty, convex, and compact.*
2. *The functions* $x \mapsto f_i(x)$ *are continuous on* X*.*
3. *The functions* $x_i \mapsto f_i(x_i, x_{-i})$ *are quasi-Concave* $\forall x_{-i} \in X_{-i}$*.*
   Then the game (1.1) *processes at least one Nash Equilibrium.*

Proof. According to lemma 1.5.1, to demonstrate the existence of a Nash Equilibrium, it suffices to show that the multi-valued map $C(.)$ has at least one fixed point. For this, we must verify the conditions of Kakutani's theorem.
We recall Kakutani's fixed point theorem. □

Theorem 1.4.4. *[8] [Kakutani's fixed. Point Theorem, 1941]. Let* X *be a non-empty, convex, and compact set, and let* $C : X_{\mapsto} \to 2^X$ *be a correspondence with non-empty, convex, and compact values. Then* C *has at least one fixed point:*

$$\exists x^* \in X, \quad x^* \in C(x^*).$$

we must verify the following conditions:

1. Since $X_i, i \in I$ are non empty, convex and compact, then $X = X_1 \times X_2 \times X_3 \times \ldots \times X_n \neq \varnothing$ is convex and compact.
2. Let $x \in X$, Show that $C(x) = \prod_{i \in I} C_i(x_i) \neq \varnothing$

   Indeed, We have$C(x) = \prod_{i \in I}$,
   $C_i(\underline{x}_i) \neq \phi \Leftrightarrow \forall i \in N, C_i(x_i) \neq \phi$,

   Therefore, it aims to showing that:

   $$C_i(x_{-i}) \neq \phi, \forall i \in I.$$

   By definition, we have

   $$C_i(x_{-i}) = \left\{ \bar{x}_i \in X_i, f_i(\bar{x}_i, x_{-i}) = \sup_{x_i \in X_i} f_i(x_i, x_{-i}) \right\}$$

Since $x_i \to f_i(x)$ is continuous on X_i and X_i is compact, then f_i attains its maximum:

$$\exists \bar{x}_i \in X_i, f_i\left(\bar{x}_i, x_{-i}\right) = \sup_{x_i \in X_i} f_i(x_i, x_i).$$

In other words $\bar{x}_i \in C_i\left(x_{-i}\right)$. Therefore, $\forall i \in N, C_i\left(x_i\right) \neq \phi$.

3. Let $x \in X$, show that $\forall x \in X, C(x)$ is convex.

   Let $z, t \in C(x), \lambda \in [0,1]$, we have to prove that $\lambda z + (1-\lambda)t \in C(n)$. Which means

   $$\lambda z_i + (1-\lambda)t_i \in C_i\left(x_i\right), \quad \forall i \in I.$$

   We have,

   $$\begin{array}{l} \forall i \in I, z_i \in C_i\left(x_{-i}\right) \Leftrightarrow \forall x_i \in X_i f_i\left(z_i, x_{-i}\right) \geqslant f_i\left(x_i, x_{-i}\right) \\ \forall i \in I, t_i \in C_i\left(x_{-i}\right) \Leftrightarrow \forall x_i \in X_i f_i\left(t_i, x_{-i}\right) \geqslant f_i\left(x_i, x_{-i}\right) \end{array}.$$

   Since f_i is quasi-concave, then,

   $$\begin{aligned} f_i\left(\lambda z_i + (1-\lambda)t_i, x_{-i}\right) &\geqslant \min\left\{f_i\left(z_i, x_{-i}\right), f_i\left(t_i, x_{-i}\right)\right\} \\ &\geqslant f_i\left(x_i, x_i\right), \forall x_i \in X_i \\ &\Rightarrow \lambda z_i + (1-\lambda)t_i \in C_i\left(x_{-i}\right), \forall i \in I. \end{aligned}$$

   We deduce that $(\lambda Z + (1-\lambda)t \in C(x)$.

   Consequently, we have $C(x)$ is convex $\forall x \in X$.

4. Show that $\forall x \in X, C(x)$ is compact.

   Since X is Compact, it suffices to Show that the set $C(x)$ is closed.

   $$C_i\left(x_i\right) = \left\{\bar{x}_i \in X_i \text{ such that } f_i\left(\bar{x}_i, x_{-i}\right) \geqslant f_i\left(x_i, x_{-i}\right), \forall x_i \in X_i\right\}, i \in I.$$

   We consider a sequence $\left(\bar{x}_i^k\right)$ in $C_i\left(x_{-i}\right)$, such that

   $$\left(\bar{x}_i^k\right) \mapsto x_i^* \text{ for } k \mapsto +\infty.$$

   Show that $x_i^* \in C_i\left(x_i\right)$.
   Using the continuity of f_i, we have

   $$\begin{aligned} \forall i \in I, \bar{x}_i \in C_i\left(x_{-i}\right) \Rightarrow & f_i\left(\bar{x}_i^k, x_{-i}\right) \geqslant f_i\left(x_i, x_{-i}\right), \quad \forall x_i \in X_i y_i \in x_i \\ \Rightarrow & \forall x_i \in X_i, \lim_{k \to +\infty} f_i\left(\bar{x}_i^k, x_{-i}\right) \geqslant f_i\left(x_i, x_{-i}\right) \\ \Rightarrow & \forall x_i \in X_i, f_i\left(\lim_{k \to +\infty}\left(\bar{x}_i^k\right), x_{-i}\right) \geqslant f_i\left(x_i, x_{-i}\right) \end{aligned}.$$

$$\Rightarrow \forall y_i \in X_i, f_i\left(x_i^*, x_{-i}\right) \geqslant f_i\left(y_i, x_{-i}\right)$$
$$\Rightarrow x_i^* \in C_i\left(x_{-i}\right)$$

Thus $C_i\left(x_{-i}\right)$ is closed, then $\Rightarrow C(x)$ is closed.

All the conditions of Kakutani's fixed-Point Theorem are verified by $C(.)$, So the multi-valued application $C(.)$ has at least a fixed point $\bar{x} \in C(\bar{x})$. According to the Lemma 1.5.1, $\bar{x}$ is a Nash equilibrium in the game (1.1).

Remark 1.4.2. *The conditions in Theorem 1.5.3 are sufficient but not necessary. A Nash equilibrium can exist even if they are not satisfied.*

1.5 Conclusion

This chapter introduces key concepts such as players, strategies, payoffs, and Nash equilibrium. These fundamental principles provide a framework for analyzing strategic interactions and predicting outcomes, with applications across various fields.

CHAPTER 2

MACHINE LEARNING

2.1 Introduction

It has been over 20 years since a computer program defeated the reigning world chess champion, marking a significant milestone in the field of artificial intelligence (AI) and Machine Learning (ML). International Blues Machines' Deep Blue program defeated Garry Kasparov in 1997, drawing widespread attention to the rapidly evolving fiats of ML. Today, ML is a mature technology with applications in almost every aspect of life. Iran recommend toys to toddlers, suggest books to readers, and even predict stack market trends. ML is used in healthcare to diagnose diseases, in energy management to optimize consumption, and in transportation to develop self-driving cars.

Google has been at the forefront of ML research, with projects: The Google brain and self-driving cars. The company's foams on ML has made it an integral part in our daily lives.

But where did ML begin? The foundation of ML dates back to the 18th and 19th centuries, with the work of Thomas Bayes and Pierre-Simon Laplace. The method least squares, developed in 1805, laid the groundwork of regression analysis. Andrey Markov's work on Markov chains in 1913 also contributed to the development of ML.

However, the modern era of machine learning began with Alan Turing's seminal paper in 1950, which proposed the idea of machines learning from experience. Arthur Samuel's work on machine learning programs at IBM (International Business Machines Corporation) in the 1950s, along with Frank Rosenblatt's development of the first neural network model in 1957, further advanced the field.

Over the past 50 years, ML has evolved rapidly, with the development of new algorithms and techniques. The latest breakthroughs, such as Google's AlphaGo program, have demonstrated the power of ML in complex decision-marking tasks.

2.2 Machine learning - what is it?

Machine learning is a subset of artificial intelligence (AI) that focuses on solving problems using historical or previous examples. Unlike traditional AI applications its are systems that rely on predefined rules, logical inference, and symbolic

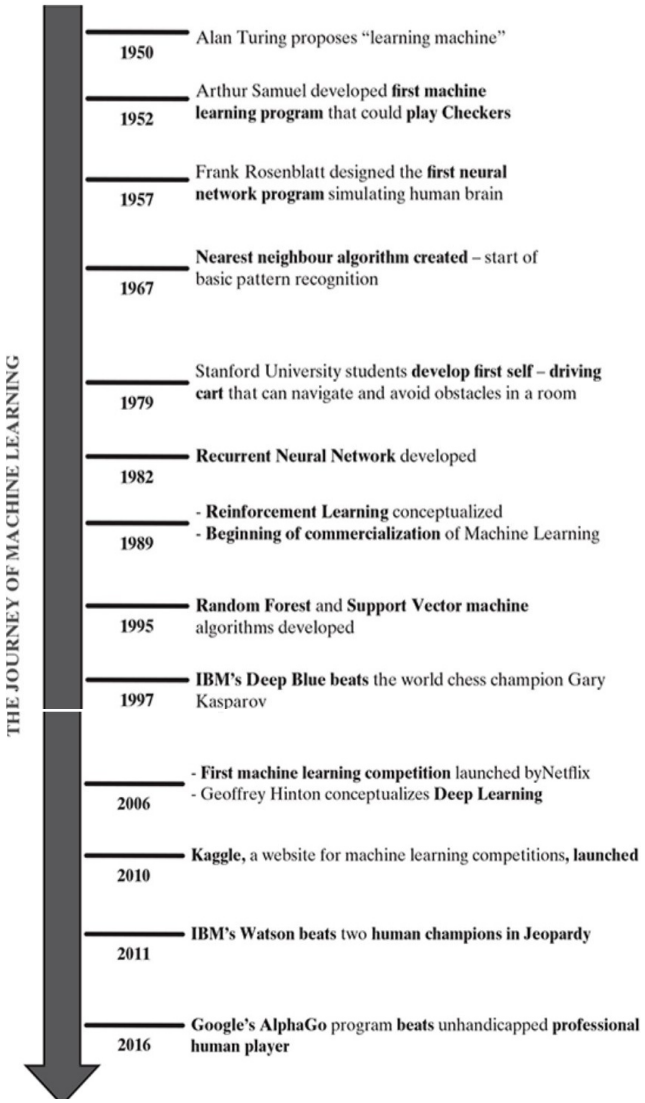


Figure 2.1: Evolution of Machine Learning [6]

representations to simulate intelligent behavior, often using expert knowledge encoded manually by humans , machine learning involves discovering hidden patterns within data (data mining) and using those patterns to classify or predict events related to the problem. In essence, machine learning provides intelligent machines with the knowledge they need to function effectively.

Machine learning algorithms are embedded in machines and data streams, allowing them to extract knowledge and information, and feed it back into the system for further and more efficient process management While all machine learning algorithms are also AI techniques, not all AI methods qualify as machine learning algorithms.

Machine learning algorithms can be broadly classified into two main categories: supervised and unsupervised learning. Some authors also recognize reinforcement learning, which involves learning from data through interaction with an environment. However, most research considers supervised and unsupervised learning as the primary categories.[6]
Think of it this way:

- **Traditional programming**: You give the computer a set of instructions, and it follows them precisely.
- **Machine learning:** You give the computer a bunch of examples, and it figures out the rules on its own.

Example 2.2.1. *[6]*
We can analyze a customer based on their invoices. A simple, classic algorithm—namely addition is sufficient for this task, and no learning algorithm is required.
Now, suppose we want to use these invoices to determine which products the customer is most likely to purchase in a given month. While this question is related, we clearly do not have all the necessary information to answer it directly. However, if we have the purchase history of a large number of individuals, it becomes possible to use a machine learning algorithm to extract a predictive model that enables us to answer this question.

2.2.1 Basic machine learning

- **Data Input:** Past data or information is used as the foundation for future decision-making.
- **Abstraction:** The input data is transformed into a more general representation through the underlying algorithm.
- **Generalization:** The abstracted representation is generalized to form a framework for making decisions.

1. **Abstraction** : During the machine learning process, knowledge is fed into the system in the form of input data. However, this data cannot be used in its

raw form; it requires abstraction to derive a conceptual model. This model represents a summarized form of knowledge extracted from the data and can take various forms, such as: Computational blocks (e.g., if/else rules), Mathematical equations, Specific data structures (e.g., tries or graphs), and Logical groupings of similar observations.
The choice of model depends on several factors, including:

- The type of problem to be solved (e.g., forecasting, trend analysis, or segmentation)
- The nature of the captured data (e.g., completeness, data types, and missing values)
- The domain of the problem (e.g., business-critical domains requiring immediate inference, such as fraud detection in banking)

Once the model is chosen, the next step is to fit the model to the input data. This involves finding the optimal values for the model's parameters. For example, in a simple linear regression model

$$(y = C_1 + C_2 x)\,,$$

we need to find the values of C_1 and C_2 that best fit the input data. This process of fitting the model to the data is known as training, and the input data used for this purpose is called the training data.

2. **Generalization** : The machine learning process involves two crucial steps.

- The first step is abstraction, where knowledge from input data is distilled into a model. This process, also known as training the model, forms the foundation of learning but represents only half the challenge.

- The second step is generalization, where the abstracted knowledge is refined into a form that can be applied to make future decisions. Generalization is particularly challenging because the model is trained on a finite dataset with limited characteristics.

When applying the model to previously unseen test data, two main problems may occur:

Overfitting – when the model performs well on training data but poorly on new, unseen data because it has learned noise or specific patterns that do not generalize.

Underfitting – when the model fails to capture the underlying structure of the data, resulting in poor performance on both training and test data.[17]

In such cases, an approximate or heuristic approach, similar to human intuition or gut feeling, must be employed.This approach carries the risk of making incorrect decisions, as assumptions may not hold true in reality. Nevertheless, humans also make mistakes when relying on intuition or gut feelings in situations where reason-based decision-making is not possible.

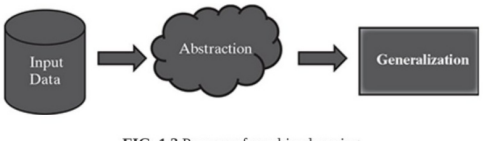


Figure 2.2: Process of machine learning [6]

2.3 Machine Learning Principle

Machine learning (ML) aims to replicate the way living organisms learn from experience, using algorithms that process data instead of following explicit instructions. For example, to teach a child the alphabet, we show them multiple labeled examples of letters (such as 'A', 'B', and 'C') in different fonts and styles not a rigid set of rules about their shapes. Through this training phase, the child learns to recognize patterns. Later, during a testing phase, we evaluate their ability to recognize letters they've seen before and generalize to new, unseen variations (e.g., a novel handwriting style).[17]

ML systems operate similarly: they extract patterns from training data to make predictions or decisions on new data, thereby mimicking the human capacity for adaptive learning.

Let $X = \{(x_i, y_i)\}_{i=1}^{n}$ be a training set of n samples, where:

- $x_i \in \mathbb{R}^p$ is a p-dimensional feature vector,
- y_i is the label (or output) associated with x_i.

The goal of supervised learning is to infer a model

$$f : \mathbb{R}^p \to Y$$

that captures the relationship between the samples x_i and their labels y_i from the training data. The model f is optimized to fit the training data by minimizing prediction errors on the observed examples (x_i, y_i), and to generalize well to unseen data. For new instances x_j (not in x, the model should produce accurate predictions $y_i = f(x_i)$.

The type of machine learning problem depends on the nature of the labels y_i:

- **Classification:** When y_i represents discrete categories (such as letters 'A', 'B', or 'C'), the goal is to assign inputs to predefined classes.
- **Regression:** When y_i is a continuous numerical value (like energy usage measurements), the model predicts quantities rather than categories.
- **Semi-Supervised Learning:** If only some y_i labels are available during training, the algorithm learns from both labeled and unlabeled data. Semi-supervised problem are out of the scope of this chapter)
- **Unsupervised Learning (Clustering):** When no labels y_i are provided, the system identifies hidden patterns or groups in the data on its own.

2.4 Types of machine learning

There are three types of machine learning :

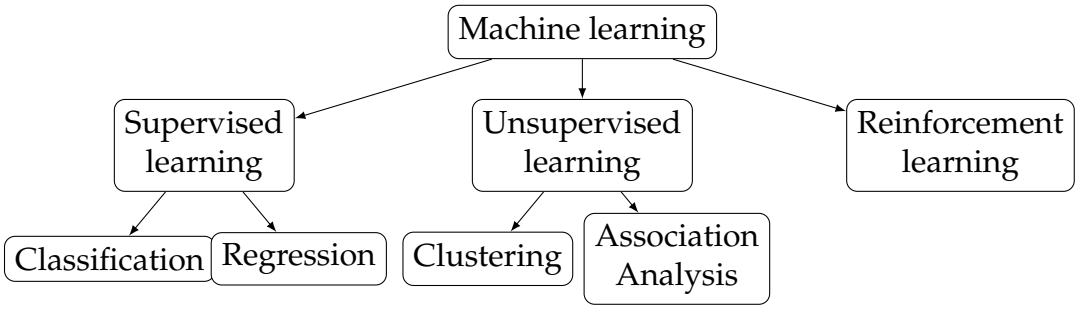


Figure 2.3: Types of Machine Learning[6]

2.4.1 Supervised Learning

The primary goal of supervised learning is to learn from past information. This past information is known as experience in the context of machine learning

What is Supervised learning

A supervised learning algorithm is trained using a labeled dataset (also known as a training set), which contains examples comprising both input variables x_i (features) and corresponding output values y_i (targets). The goal is to infer knowledge from this data to make predictions on new inputs with unknown outputs.

Each example, also referred to as an instance, is a pair of input and output: $(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n) \in X \times Y$.

– X represents the set of attributes (explanatory variables).

– Y represents the set of output values (the target or dependent variable), which can be continuous or discrete.

When the algorithm accurately determines the output y for inputs X that were not part of the training set, the function or model is considered optimal.

Example 2.4.1. *(Examples of Supervised Learning)*
Some examples of supervised learning include:

- *Predicting game results.*
- *Predicting whether a tumor is malignant or benign.*
- *Predicting real estate prices or stock prices.*
- *Classifying text, such as spam or non-spam emails.*

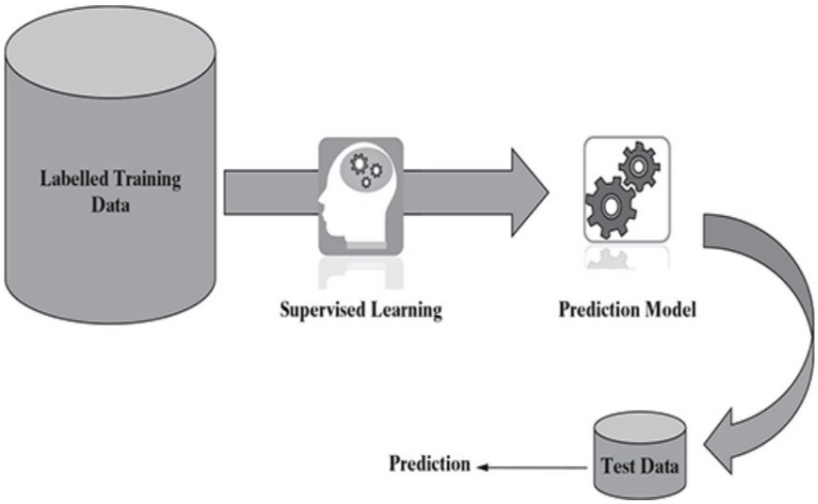


Figure 2.4: Supervised Learning [6]

The supervised learning process can be outlined in the following steps:
- Input Labeled Data: The labeled training data is fed into the machine learning algorithm.
- Model Training: The algorithm analyzes the data to identify patterns and relationships, constructing a predictive model.
- Prediction: The trained model is then used to make predictions on new, unseen data, classifying inputs based on learned patterns.

In supervised learning, two primary types of tasks are distinguished: Clasification and Regression.

1) Classification

A classifier is a method used to construct models that categorize data based on input features (attribute). These methods rely on specific learning algorithms to develop models that reveal the relation between input attributes and their associated class labels. A well-trained model should not only represent the training data accurately but also be capable of making reliable predictions on previously unseen instances.

Definition 2.4.1. *[4] Classification is the task of learning a target function f that maps each attribute set x to one of the predicted class labels y.*

Definition 2.4.2. *A binary classification is the task of classifying the elements of a dataset into two classes.*

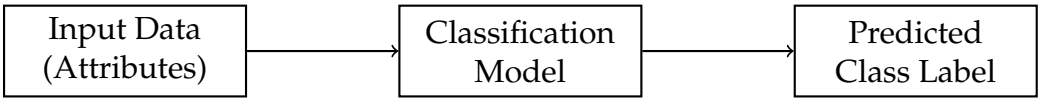


Figure 2.5: Diagram of a classification process

Remark 2.4.1. *Some popular machine learning algorithms for classification include: Support vector machines, Rule-based systems, Neural networks, Naive Bayes algorithms, and Decision Tree.*

Remark 2.4.2. *Classification has numerous real-world applications, including:*

- ***Fraud Detection:*** *identifying potential fraudulent transactions in banking.*
- ***Image classification:*** *Classifying images into categories such as objects, scenes, or actions.*
- ***Disease Predictions:*** *Predicting the likelihood of a diseax based on patient data.*
- ***Win-Loss Prediction:*** *Predicting the outcome of games or Sports events.*
- ***Natural Calamity Prediction:*** *Predicting the likelihood of natural disasters such as earthquakes or floods.*
- ***Handwriting Recognition:*** *Recognizing and classifying handwritten text.*

Example 2.4.2. *(Binary classification problem)*
The goal is to predict whether a tumor is benign (label 0) or malignant (label 1) based on patient features such as age, height, and weight.

Instance (Patient)	Age	Height (cm)	Weight (kg)	Benign (0) / Malignant (1)
1	48	175	69	0
2	52	160	81	1
3	34	182	74	0
4	45	185	94	1

Table 2.1: Example dataset for binary classification

Key terms: *We aim to explain what each entry in the previous table represents :*

- *Instance (or example): A single row in the dataset representing one observation. Example: A patient's data with age, height, weight, and tumor status.*

Instance (Patient)	Age	Height (cm)	Weight (kg)	Benign (0) / Malignant (1)
1	48	175	69	0

- *Attribute (or feature): A measurable characteristic used to describe each instance. Example: Age, height, and weight.*
- *Label (or target): The value to predict, typically provided in supervised learning. Example: Tumor status:* 1 *for malignant,* 0 *for benign.*

Summary Table of Terms:

Term	Also Called	Example
Instance	Example, Record	One patient's row
Attribute	Feature, Variable	Age, Height, Weight
Label	Target, Output	Tumor Status: 0 or 1

Table 2.2: Summary Table of Terms

1) Regression

Regression is a type of supervised learning where the objective is to predict numerical features, such as real estate prices, stock prices, temperature, or sales revenue. In linear regression, a straight line relationship is "fitted" between the predictor variables and the target variables using the statistical concept of least squares method. The goal is to minimize the sum of squared errors between actual and predicted Values.

There are Two types of Linear regression:

- **Simple linear Regression:** One predictor variable is use to predict the target variable.
- **Multiple Linear Regression:** Multiple predictor variables are used to predict the target variable.

Example 2.4.3 (Sales Prediction)**.** *[6]*
A sales manager needs to predict sales revenue for the next year based on past sales data and investment. A Simpler linear regression model can be applied with investment as the predictor Variable and sales revenue as the target variable.

Figure 2.6 shows a typical simpler regression model, where regression line is fitted to minimize error between predicted and actual target values with respect to different values of predictor variable. A typical linear regression model can be represented in the form

$$y = \alpha + \beta x$$

x: the predictor variable.
y: the target variable.

2.4.2 Unsupervised Learning

Unsupervised learning is a type of machine learning where there is no labeled training data to learn from and no specific prediction to be made. The objective is to discover natural groupings or patterns within the data. Unsupervised learning

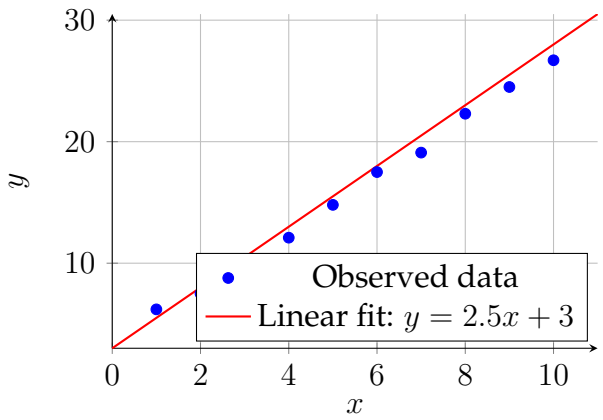


Figure 2.6: Linear it of data points

is often referred to as a descriptive model, and the process is known as pattern discovery or knowledge discovery.

There are two main types of unsupervised learning:

1. **Clustering:**
   Clustering involves grouping similar objects together based on Their characteristics. The goal is to discover the intrinsic grouping of unlabeled data into clusters. This method uses distance measures to determine the similarity between data items, grouping similar items together and separating dissimilar ones. See Figure 2.7.
   Before clustering, the data points are unlabelled and appear uniform, typically shown in black to indicate the lack of any grouping. Unsupervised learning algorithms, such as clustering methods, analyze the intrinsic similarities between these data points and group them into clusters based on their characteristics. After clustering, each group is assigned a distinct color to visually separate and identify the clusters. This process helps reveal hidden structures in unlabeled data by organizing it into meaningful groups.

2. **Association Analysis**
   Association analysis involves identifying relationships between data elements. Let's try to understand the approach of association analysis in the context of one of the most common examples: market basket analysis.

Market Basket Analysis is a common example of association analysis, where the goal is to identify items that are frequently purchased together in order to uncover strong associations between data elements and use this information to inform business decisions.
For example, in a grocery store, it can reveal that customers who buy bread

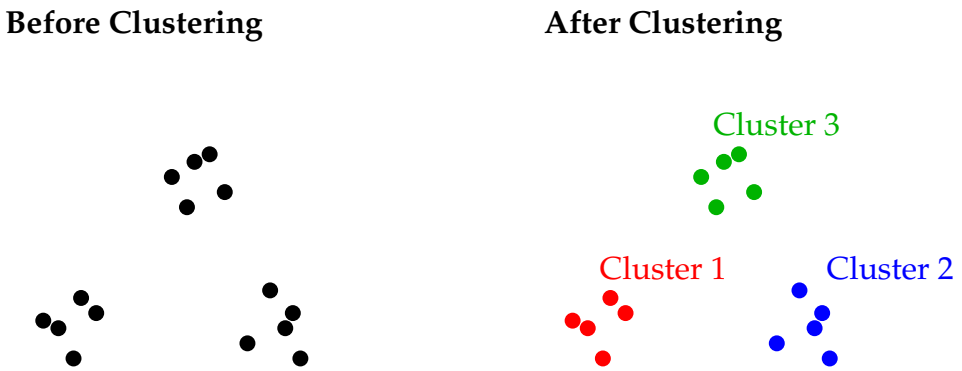


Figure 2.7: Unsupervised learning: Data points before and after clustering

often also buy butter. This information helps businesses understand purchasing patterns, optimize product placement, and design targeted marketing strategies.

2.4.3 Reinforcement Learning

Reinforcement learning is a type of machine learning where an agent learns to perform tasks autonomously by interacting with an environment and receiving rewards or penalties based on its actions.

Example 2.4.4. *Let's try to understand this with the example of a child learning to walk. First, the child observes others walking perhaps their parents or people around them. They understand that legs must be moved one at a time to take a step. While walking, the child sometimes falls after hitting an obstacle, and at other times, manages to walk smoothly while avoiding bumps.*

When the child successfully overcomes an obstacle, the parents are delighted and reward them with loud claps or perhaps even chocolates. However, when the child falls, they don't receive any applause or reward. Over time, with repeated attempts and feedback, the child learns to walk with ease.

In the same way, machines can learn to accomplish tasks autonomously. In this analogy:
The action is walking,
The child is the agent,
The environment is the space with obstacles.
The agent (child or machine) tries to improve its performance through trial and error. When a sub-task is completed successfully, a reward is given. When a sub-task fails, no reward is provided. This process continues until the machine is able to perform the full task efficiently.

This method of learning through feedback from the environment is known as reinforcement learning .

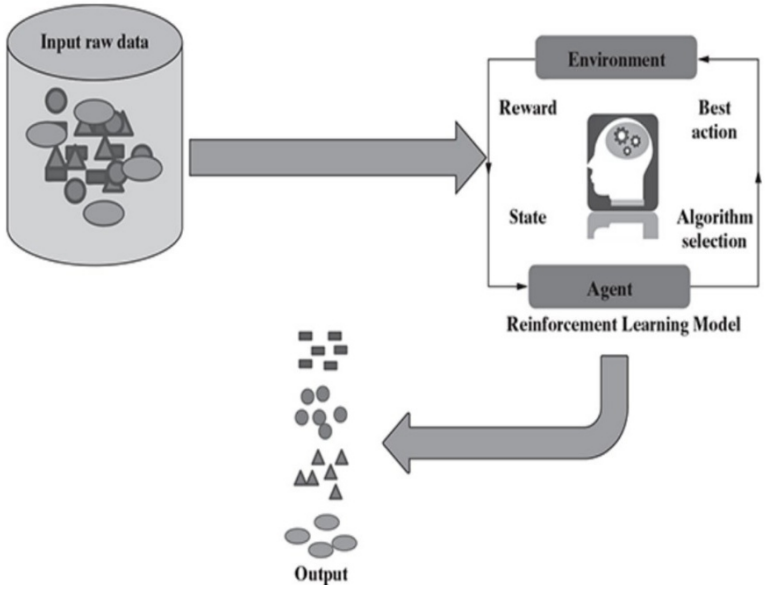


Figure 2.8: Reinforcement Learning[6]

2.5 Classification Evaluation

The performance of a classification model depends on how accurately it predicts test samples x_j. These predictions are summarized in a matrix, which records the counts of correct and incorrect classifications. For a binary classification problem (see Table 2.3), each entry e_{ij} represents the number of samples from the true class i predicted as class j.

The total correct predictions are calculated as the sum of diagonal entries $\sum_{i=1}^{K} C_i e_{ii}$, where K is the number of classes.

		Predicted ed	class
		class=1	class=0
Actual class	cLass=1	e_{11}	e_{10}
	class=0	e_{01}	e_{00}

Table 2.3: Confusion matrix for a 2-class problem

In binary classification, the matrix elements are defined as:

- **True Positives** $(TP = e_{11})$**:** Correctly classified positive instances.
- **False Negatives** $(FN = e_{10})$**:** Positive instances misclassified as negative.
- **False Positives** $(FP = e_{01})$**:** Negative instances misclassified as positive.
- **True Negatives** $(TN = e_{00})$**:** Correctly classified negative instances.

For model evaluation, summary metrics are typically used:

- **Accuracy (Acc):** Overall correct prediction rate.
- **Error Rate (Err):** Overall incorrect prediction rate.

Note that

$$Err = 1 - Acc$$

This single-value representation enables straightforward model comparison.

$$Acc = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

$$Acc = \frac{\sum_{i=1}^{K} e_{ii}}{\sum_{i,j=1}^{K} e_{ij}} \tag{2.1}$$

$$Err = 1 - Acc \tag{2.2}$$

$$Err = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}}$$

$$Err = \frac{\sum_{i,j=1,\, i \neq j}^{K} e_{ij}}{\sum_{i,j=1}^{K} e_{ij}} \tag{2.3}$$

Another measure of accuracy is given in equation 2.4 [15]. It is the percentage of the correct predictions when compared with actual classes among the test set.

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

Performance metrics enable systematic comparison between different classifiers (f). These comparisons reveal whether one learning algorithm demonstrates superior performance to another on a specific learning task when evaluated against a given test set X_{test}.

2.6 Decision Tree Classifier

2.6.1 Definitions

A decision tree is a supervised machine learning model that can be used for both classification and regression tasks. It functions by recursively dividing the dataset into subsets based on feature values, forming a tree-like structure that is straightforward to interpret.

We recall here the key terms of decision trees.

- **Root node:** Represent the entire dataset before any spilt.
- **Internal Nodes:** These are points in the tree where the feature space is split.
- **Terminal nodes (or leaves):** These are nodes that Terminal nodes (or leaves). These are nodes that are no longer divided and represent a final decision.
- **Branches:** These are lines connecting different nodes (terminal or internal).
- **Parent and Child Node:** A node becomes a parent node when it is split according to a particular criteria or feature, and the nodes that emerge from this split are known as child nodes.

Decisions and their potential outcomes are represented by nodes that are connected in a hierarchical structure within a decision tree. Each child node represents a potential consequence or further choice route resulting from the parent node, which represents a decision point or test condition. The recursive division of the dataset made possible by this parent-child connection allows the model to classify or predict using the input features.

Remark 2.6.1. *How a Decision Tree Works?*
A decision tree is a predictive model that guides input data through a sequence of decisions to produce an outcome. Here's how it functions, step by step:

Beginning at the Root:
The process starts at the top of the tree, known as the root node. At this stage, the algorithm chooses the most informative feature to divide the dataset. This choice is based on specific criteria like Gini impurity.

Creating Branches by Splitting:
Based on the chosen feature, the data is split into smaller groups. Each group follows a different branch, leading to new decision points (nodes).

Recursive Splitting:
The procedure is repeated for each new node: the best feature is selected again to split the subgroup. This continues, forming deeper levels in the tree, until no meaningful split can be made, or a stopping rule is met.

Arriving at a Leaf Node:
Eventually, each path ends at a leaf node, which provides the final prediction. In classification, this is usually the most common class within that group. In regression, it is typically

the average of the target values.

Making a Prediction:
To make a prediction for a new input, the model checks its features one by one, following the path in the tree according to the decisions at each node. Once a leaf is reached, the associated value is returned as the output.

Example 2.6.1. *Example (Classification):*
This decision tree is a simple classification model that predicts the weather condition based on two features: Temperature and Humidity.
The root node asks the question: Is the temperature greater than 20°C?
If the answer is No (temperature is 20°C or below), the model predicts the class Sunny directly.
If the answer is Yes (temperature is above 20°C), the model proceeds to the next decision node.
The second node asks: Is the humidity greater than 50
If Yes, the model predicts the class Rain.
If No, the model predicts the class Sunny.
In summary, the tree uses simple yes/no questions about temperature and humidity to classify the weather as either Rain or Sunny. This illustrates how decision trees split the data step-by-step based on feature thresholds to arrive at a classification.

Condition 1	**Condition 2**	**Predicted Class**
Temperature $>$ 20°C	Humidity $>$ 50%	Rainy
Temperature $>$ 20°C	Humidity $\leq$ 50%	Sunny
Temperature $\leq$ 20°C	-	Sunny

Table 2.4: Decision rules for weather prediction

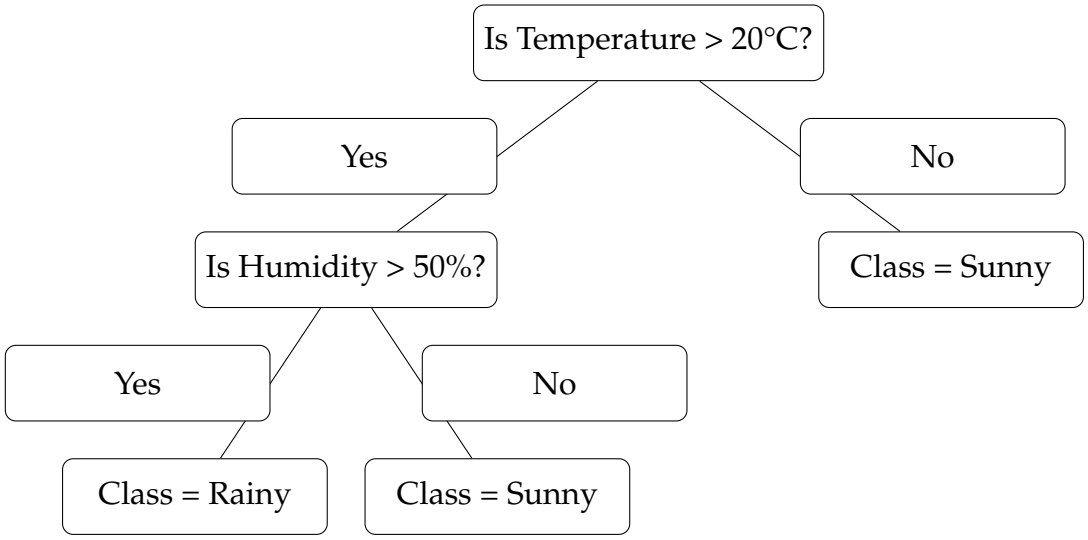


Figure 2.9: Decision tree used to predicts the weather condition based on two features: Temperature and Humidity.

2.6.2 Mathematical Model

- The nodes represent subspaces (regions) of the input space X.
- The root node contains the entire space X, while the leaves represent unit regions.
- Between the root and leaves, the internal nodes represent nested regions, such that:

$$X = X^1 \cup X^2 \cup \ldots \cup X^m \cup \ldots \cup X^{p'} \quad \text{with} \quad n' \leq n,$$

  and each region X^m can be further decomposed into subregions.
- Each node m is associated with a region $X^m \subseteq X$ and a decision function f_m, which for an element $x \in X^m$ determines a subspace $X^{m'} \subseteq X^m$ for the next split.
- Each leaf node corresponds to an element of the output space Y:
  - For classification problems, this is a discrete class label.
  - For regression problems, this is a continuous real value.
- All inputs x belonging to the same leaf region share the same predicted output value y.
- The decision function f_m at each node is typically simple, but the combined function of the entire tree forms a complex decision function.

- No assumptions are made about the probability distribution of the classes.
- The tree structure is not fixed in advance; nodes, branches, and leaves are added dynamically based on the data.
- Methods in this family differ mainly based on:
  - The choice of the decision function f_m used to split the data.
  - The criteria used to evaluate the quality of these splits.

A decision tree is a model that recursively partitions the feature space into disjoint regions to make predictions. It can be thought of as a function that maps an input vector $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ to a predicted output $\bar{y}$.

It operates in two main phases:

1-**Construction Phase**:
- During this phase, the data space is recursively divided into increasingly homogeneous subgroups based on a selected criterion—commonly information gain or the Gini index.
-Each internal node in the tree performs a test on a feature (e.g., $x \geq$ threshold?), and the data is split accordingly into branches.
- This recursive process continues until a stopping condition is met such as achieving class purity or reaching a minimum number of samples per node.
- Each leaf node represents a final prediction: a class label (in classification) or a mean value (in regression).

2.-**Pruning Phase**:
- The purpose of pruning is to enhance the model's generalization ability by removing branches that contribute little to predictive performance.
- This involves replacing certain subtrees with leaf nodes when doing so leads to a lower estimated error on a validation set or test data.

2.6.3 Equation of a Decision Tree

For an imput vector $x \in X$, the prediction function of a decision tree given as [1, 2]:

$$\bar{y}(x) = \sum_{m=1}^{t} C_m \cdot I(\mathbf{x} \in R_m), \tag{2.5}$$

where:

- t is the number of terminal nodes (leaves) in the tree,

- C_m is the prediction value assigned to region X^m (e.g., the most common class for classification or the mean for regression),
- X^m is the m-th region (leaf) of the feature space,
- $I(x \in X^m)$ is the indicator function, which equals 1 if $x \in X^m$ and 0 otherwise.

This function means that for a given input x, the tree traverses its structure until it reaches a leaf, where the prediction C_m is made based on the values in that region.

2.6.4 Classification Trees

Decision trees are used in classification tasks to give each region a class label. The most prevalent class in the training data inside region X^m is determined to be the constant C_m [1]:

$$C_m = \arg \max_{k \in \{0,1,\ldots,K\}} \sum_{x_i \in X^m} I(y_i = k), \tag{2.6}$$

where the number of classes is denoted by K. According to this equation, the class k with the greatest number of samples in region X is the projected class C_m.
This equation expresses an essential rule in decision trees for classification: the predicted class in each leaf of the tree is the majority class.

2.6.5 Splitting Criteria

A splitting criterion is used to decide how to divide the data at each node in order to build the tree.

Gini Index

The Gini Index is a measure of impurity (non-homogeneity). It aims to measure the probability of misclassifying a randomly chosen element from the dataset.

The greater the value of the Gini Index, the greater the chances of having misclassifications.

In classification trees, impurity at a node is measured using the Gini index [1, 2].

$$\text{Gini}(s) = 1 - \sum_{i=1}^{K} p_i^2, \tag{2.7}$$

where p_i is the proportion of samples in class i at node s.

A purer node is indicated by a lower Gini index.
In the case where there are only two classes :

$$\text{Gini}(s) = 1 - \sum_{i=1}^{2} p_i^2 = 1 - (p_1^2 + p_2^2) = 1 - (p_1^2 + (1 - p_1)^2).$$

Entropy

The degree of uncertainty or randomness in the class distribution at a node s is measured by entropy [1]:

$$\text{Entropy}(s) = -\sum_{l=1}^{K} p_l \log_2(p_l), \tag{2.8}$$

where p_l is the proportion of samples in class l.

Samples from a single class make up the majority of the node when the entropy value is lower. Entropy is the quantity of information needed to categorize a fresh sample and is derived from information theory.

2.6.6 Types of Decision Tree

The Hunt algorithm, which originated in the 1960s as a model of human learning in psychology, serves as the basis for many widely-used decision tree algorithms. Some important examples include:

ID3 (Iterative Dichotomiser 3): Developed by Ross Quinlan, ID3 employs entropy and information gain to assess how well a dataset is split. Quinlan's foundational work on this method dates back to 1986.

C4.5: Seen as a refinement of ID3, this algorithm is created by Quinlan. It uses either information gain or gain ratio to determine the best split points within decision trees.

CART (Classification and Regression Trees): CART was introduced by Leo Breiman. It commonly relies on the Gini impurity measure to select the most suitable attribute for splitting.

2.6.7 The advantages of decision trees

They are simple to understand and interpret. The trees can be visualized. Also, the results obtained can be easily explained. They can work on data with little preparation. For example, they do not require data normalization. They accept both numeric and nominal data. Other learning algorithm specialize in a single type of data. They perform well even if their assumptions are somewhat violated by the actual model form which the data were generated.

2.7 Conclusion

Chapter 2 provided an overview of machine learning, explaining its core principles and methodologies. It covered supervised, unsupervised, and reinforcement learning, along with key techniques like decision trees and their evaluation metrics. The discussion highlighted how machine learning enables data-driven decision-making, offering powerful tools for classification, prediction, and pattern recognition. These concepts form the foundation for more advanced applications in artificial intelligence .

CHAPTER 3

NASH EQUILIBRIUM IN A DECISION TREE

3.1 Introduction

In this Chapter, we consider the paper in [13]. It introduces a novel binary classification approach combining decision trees with game theory. Traditional decision trees divide data step by step, with each node representing a region and each split creating smaller sub-regions. The method in [13] proposes a new splitting based on Nash Equilibrium.

3.2 Problem Statement - Binary Classification

We consider a binary classification setting where each data point is represented in a d-dimensional feature space. Given a dataset $X \in \mathbb{R}^{n \times d}$ with n instances and corresponding labels $y \in \{0, 1\}^n$, the objective is to construct a classification rule $\mathcal{R}$ that maps each input $x_i \in X$ to a predicted label $\hat{y}_i \in \{0, 1\}$.

The goal is for the predicted labels to closely match the true labels, i.e., $\hat{y}_i \approx y_i$ for all i, while ensuring that the learned rule $\mathcal{R}$ generalizes well to new, unseen data drawn from the same distribution.

3.3 Nash Equilibrium - Decision trees

The proposed method employs the Nash equilibrium concept to determine the optimal parameters for the splitting hyper-plane at each node. This involves formulating a strategic game between the two classes, where the objective is to identify a hyper-plane that: maximizes the separation between instances of different class. By doing so, the approach ensures that data points belonging to the same class are clustered together white being discriminability and simplifying the classification task.

3.3.1 Splitting game

The game $\Gamma(x, y \mid j)$ is a two-player framework designed to a partition a dataset X with labels y, based on attribute j [13]:

- **the players** are represented by two sub-nodes, Left (L) and Right (R);
- **the strategy** of each player (L and R) proposes a hyper-plane parameter, $\mathcal{B}^L$ and $\mathcal{B}^R$, respectively, where $B^L, B^R \in \mathbb{R}^2$;
- Each player minimizes a dual objective function that:
  1. Maximizes separation between classes.
  2. Minimizes hyper-plane complexity (via parameter norm).

$$\begin{cases} u^L\left(B^L, B^R\right) = \sum_{x_i \in X, y_i = 0} \left(x_{ij} B_{j,1} + B_{j,0}\right) + \left\|B^L\right\|^2 \\ u^R\left(B^L, B^R\right) = -\sum_{x_i \in x, y_i = 1} \left(x_{ij} B_{j,1} + B_{j,0}\right) + \|B^R\|^2 \end{cases} \tag{3.1}$$

where $B_j = (B_{j,0}, B_{j,1}) \in \mathbb{R}^2$ and

$$B_j = \frac{1}{2}\left(B^2 + B^R\right).$$

And the squared euclidean norm $\|B\|^2 = \|(B_0, B_2)\|^2 = B_0^2 + B_1^2$.
In the proposed game, the payoff functions are structured so that:

- Player L (Left) selects a strategy B^L to minimizes the dot product $\langle x_i, B^L \rangle$ for all instances labeled 0.
- Player R (right) chooses a strategy B^R to maximize the dot product $\left\langle x_i, B^R \right\rangle$ for all instances labeled 1.

This setup encourages each player to push dater points of their assigned label as far as possible from the opposing class. The final splitting hyper-plane is derived as a Nash Equilibrium, defined by the average of their strategies:

$$B^* = \frac{B^L + B^R}{2}.$$

This equilibrium ensures neither player can unilaterally improve his payoff without the other adapting.

3.3.2 Nash Equilibrium

The Nash equilibrium [11] is a widely recognized solution in game theory, prized for its ability in competitive situations. It describes a game state where each player has chosen a strategy, and no player can benefit by unilaterally changing their strategy while the others keep their unchanged.

For a game Γ_L, a Nash equilibrium is represented by strategies B^L and B^R, where neither player can achieve a better outcome by altering their strategy if the other player's choice remains fixed.

Remark 3.3.1. *In their study, the authors [13] consider the the Nash equilibrium in the sense of minimization, meaning that each player's strategy aims to minimize their individual cost or loss function.*

The Nash equilibrium for game $\Gamma(x, y/j)$ is calculated by solving the optimization problems:

$$\min_{B^L} u^L\left(B^L, B^R\right). \tag{3.2}$$

and

$$\min_{B^R} u^R\left(B^L, B^R\right). \tag{3.3}$$

From the equation $B_j = \frac{1}{2}\left(B^L + B^R\right)$, we obtain $B_{j,0} = \frac{1}{2}\left(B_0^L + B_0^R\right)$ and $B_{j,1} = \frac{1}{2}\left(B_1^L + B_1^R\right)$. These relations allow the reformulation of the payoffs as follows:

$$u^L\left(B^L, B^R\right) = \frac{n_0}{2}\left(B_0^L + B_0^R\right) + \frac{1}{2}\sum_{\substack{x_i \in X \\ y_i = 0}} x_{ij}\left(B_1^L + B_1^R\right) + \left(B_1^L\right)^2 + (B_0^L)^2$$

$$u^R\left(B^L, B^R\right) = -\frac{n_1}{2}\left(B_0^L + B_0^R\right) - \frac{1}{2}\sum_{\substack{x_i \in X \\ y_i = 1}} x_{ij}\left(B_1^L + B_1^R\right) + \left(B_1^R\right)^2 + \left(B_0^R\right)^2,$$

where n_0 is the number of instances in X with label 0, and n_1 is the number of instances with label 1.

In the left node, we write the partial derivatives with respect to B_0^L and B_1^L:

$$u_0^{L'}\left(B^L, B^R\right) = \frac{n_0}{2} + 2B_0^L.$$

and

$$u_1^{L'}\left(B^L, B^R\right) = \frac{1}{2}\sum_{\substack{x_i \in X \\ y_i = 0}} x_{ij} + 2B_1^L$$

The hessian matrix H of the function $B^L = (B_0^L, B_1^L) \to u^L(B^L, B^R)$ is convex. Indeed,

$$H = \begin{bmatrix} u_0^{L"} & u_{0,1}^{L"} \\ u_{1,0}^{L"} & u_1^{L"} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix},$$

the determinant of H which verify $det(H) > 0$ and the second partial derivative with respect to B_0^L, $u_0^{L"} > 0$.

We know that for convex functions: A local minimum is a global minimum. Then, the optimal (the minimum) B^{L^*} is given bellow.
The critical point is:

$$B_0^{L^*} = -\frac{n_0}{4}\text{ , and } B_1^{L^*} = -\frac{1}{4}\sum_{\substack{x_i \in X \\ y_i = 0'}} x_{ij} \tag{3.4}$$

At this point, we have $det(H) > 0$ and $u_0^{L"} > 0$, which make it an optimal (minimum) solution.

With the same reasoning, for the right node we have B^{R^*}:

$$B_0^{R^*} = \frac{n_2}{4}\text{ ; and } B_2^{R^*} = \frac{1}{4}\sum_{\substack{x_i \in X, \\ y_i = 1}} x_{ij} \tag{3.5}$$

Consequently, the Nash equilibrium of the game $\Gamma(x, y, j)$ is $\left(B^{L^*}, B^{R^*}\right)$.

And we obtain $B_j^* = \left(B_{j,0}^*, B_{j,1}^*\right)$, which are computed as $B_j^* = \frac{1}{2}\left(B^{L^*} + B^{R^*}\right)$ and has the Components:

$$\begin{cases} B_{j,0}^* = \frac{1}{2}(B_0^{L*} + B_0^{R*}) = \frac{1}{8}(n_1 - n_0) \\ B_{j,1}^* = \frac{1}{2}(B_1^{L*} + B_1^{R*}) = \frac{1}{8}\left(\sum_{x_i \in X, y_i = 1} x_{ij} - \sum_{x_i \in X, y_i = 0} x_{ij}\right) = \frac{1}{8}\sum_{x_i \in X}(2x_{ij}y_i - x_{ij}) \end{cases}$$

The game $\Gamma(x, y/j)$ determines the equilibrium parameters B_j^* which define the optimal splitting hyper-plane for the node data.

In [13], the split is based on $X_j B_{j,1}^* + B_{j,0}^*$, where the coefficients are derived from the Nash equilibrium.

3.3.3 Splitting hyper-plane

Remark 3.3.2. *In classification methods, a splitting hyperplane is used to separate data points from different classes. This hyperplane can be defined by the following linear equation: $w^T x + b = 0$, where:*
1) $x \in \mathbb{R}^n$ is the feature vector,
2) $w \in \mathbb{R}^n$ is the weight vector (normal to the hyperplane),
3) $b \in \mathbb{R}$ is the bias or intercept term.
Points satisfying $w^T x + b > 0$ are classified into one class, and those with $w^T x + b < 0$ into another.

In the work of Suciu and Lung [13], the Nash equilibrium derived from the game $\Gamma(x, y/j)$ ensures maximal separation between instances of different labels. To construct the splitting hyper-plane for a node with data X, y and attribute j:

1. Compute the equilibrium parameters B_j by solving $\Gamma(x, y/j)$.
2. Associate to each instance x_i, the line defined by $x_{ij}B_{j,1} + B_{j,0}$.

Consider the sets:

$$\widetilde{X}_j^0 = \{x_{ij}B_{j,1} + B_{j,0} \mid y_i = 0\}$$

is the set of transformed feature values for data instances with label 0, computed using a splitting hyperplane derived from a Nash equilibrium.

$$\widetilde{X}_j^1 = \{x_{ij}B_{j,1} + B_{j,0} \mid y_i = 1\}$$

is the set of transformed feature values for data instances with label 1, computed using a splitting hyperplane derived from a Nash equilibrium.

The game Nash equilibrium for the game $\Gamma(x, y/j)$, B_j ensures that the sum of elements in $\widetilde{X_j^1}$ is minimized while the sum of elements in $\widetilde{X}_j^1$ is maximized.

This reflects the game's objective of separating the products of the form $x_{ij}B_{j,1}+$ $B_{j,0}$ for instances with different labels.

To split the node data, based on X_j and B_j, the authors define the splitting point as the average of representative points from $\widetilde{X}_j$ and $\widetilde{X}_j^1$.

We recall the following definitions, which are useful for the understanding of the rest of this chapter.

Definition 3.3.1. *[7] An outlier in a classification problem is a data point that significantly differs from other instances in the same class or does not conform to the general pattern of the dataset.*

Definition 3.3.2. *[16]*

Let a sorted dataset be given as $v_1 \leq v_2 \leq \cdots \leq v_n$. The k-th percentile P_k is defined as:

$$P_k = v_{\lceil k \cdot n \rceil}$$

where:

- $k \in [0, 1]$ *is the desired percentile,*
- n *is the total number of observations,*
- $\lceil \cdot \rceil$ *denotes the ceiling function. The ceiling function returns the smallest integer greater than or equal to a given number.*

- v_i *are the ordered values of the sample.*

To reduce the influence of outliers, the percentiles are used:
The k^{th} percentile, $P_k(\widetilde{X}_j^0)$ for $\widetilde{X}_j^0$ and the $(1-k)^{th}$ percentile $P_{(1-k)}(\widetilde{X}_j^1)$ for $\widetilde{X}_j^1$.

The final splitting point $\widetilde{B}_j$ is computed as:

$$\widetilde{B}_j = \frac{1}{2}\left(P_k\left(\tilde{X}_j^0\right) + P_{(1-k)}\left(\tilde{X}_j^1\right)\right) \tag{3.6}$$

and by using B_j and $\widetilde{B}_j$ the splitting (axis parallel) hyper-plane can be defined as

$$B_{j,1}x + B_{j,0} = \widetilde{B_j}.$$

Instead of simply splitting at a raw value of x, the method calculates a splitting threshold $\widetilde{B_j}$ based on percentiles of the transformed values for each class (to reduce outlier effects). This threshold acts as the decision boundary in the transformed space.
So the hyper-plan $B_{j,1}x + B_{j,0} = \widetilde{B_j}$ acts as the boundary that splits the data into left and right child nodes.

The rule for splitting data: Given a dataset X at a node, and an attribute j, the data is split into two sub nodes: the left sub-node X_j^L and the and the right sub-node X_j^R.:

$$X_j^L = \left\{x \in X | B_{j,1}x + B_{j,0} \leq \widetilde{B}_j\right\}$$
$$X_j^R = \left\{x \in X \mid B_{j,1}x + B_{j,0} \geqslant \widetilde{B}_j\right\},$$

with the Corresponding set of labels:

$$y_j^L = \left\{y \in Y \mid B_{j,1}x + B_{j,0} \leq \widetilde{B}_j\right\}$$
$$y_j^R = \left\{y \in Y \mid B_{j,1}x + B_{j,0} \geqslant \widetilde{B}_j\right\}$$

The dataset X is split into two subsets X_j^L and X_j^R based on a linear condition involving the attribute j. Specifically, for each sample $x \in X$, we compute the value

$$B_{j,1}x + B_{j,0},$$

where $B_{j,1}$ and $B_{j,0}$ are parameters related to attribute j.
The left sub-node X_j^L consists of all samples for which this value is less than or equal to a threshold $\widetilde{B}_j$:

$$X_j^L = \left\{x \in X \mid B_{j,1}x + B_{j,0} \leq \widetilde{B}_j\right\}.$$

Similarly, the right sub-node X_j^R contains all samples where the value is strictly greater than the threshold:

$$X_j^R = \left\{ x \in X \mid B_{j,1}x + B_{j,0} > \widetilde{B}_j \right\}.$$

This linear decision boundary defines how the data is partitioned into left and right subsets based on attribute j.

GameSplit – Selects the best feature to split on using a Nash Equilibrium-inspired method [13].

Algorithm 1: GameSplit(X, y, k): Split node data (X, y) using percentile k

for each feature j **do**
  $n_0 \leftarrow$ number of samples with $y_i = 0$;
  $n_1 \leftarrow$ number of samples with $y_i = 1$;
  $B_{j,0} \leftarrow \frac{1}{8}(n_1 - n_0)$;
  $B_{j,1} \leftarrow \frac{1}{8} \sum_{x_i \in X} (2x_{ij}y_i - x_{ij})$;
  $\tilde{X}_j^0 \leftarrow \{x_{ij}B_{j,1} + B_{j,0} \mid y_i = 0\}$;
  $\tilde{X}_j^1 \leftarrow \{x_{ij}B_{j,1} + B_{j,0} \mid y_i = 1\}$;
  $\tilde{B}_j \leftarrow \frac{1}{2}\left(P_k(\tilde{X}_j^0) + P_{1-k}(\tilde{X}_j^1)\right)$;
  $X_j^L \leftarrow \left\{x \in X \mid B_{j,1}x + B_{j,0} \leq \tilde{B}_j\right\}$;
  $X_j^R \leftarrow \left\{x \in X \mid B_{j,1}x + B_{j,0} \geq \tilde{B}_j\right\}$;
  $y_j^L \leftarrow \left\{y_i \in y \mid B_{j,1}x_i + B_{j,0} \leq \tilde{B}_j\right\}$;
  $y_j^R \leftarrow \left\{y_i \in y \mid B_{j,1}x_i + B_{j,0} \geq \tilde{B}_j\right\}$;
  Compute $C(X, y, j)$ Evaluate the quality of the split (e.g., entropy or Gini index);
end for
$j^* \leftarrow \arg\max_j C(X, y, j)$;
return j^*, B_{j^*}, $\tilde{B}_{j^*}$, $X_{j^*}^L$, $y_{j^*}^L$, $X_{j^*}^R$, $y_{j^*}^R$.

NE-DT: Builds a recursive decision tree using GameSplit at each node [13].

Algorithm 2: NE-DT$(X, y; k, \text{depth}, \text{MaxDepth})$: Nash equilibria - Decision Tree Algorithm - outline.

Input: X node data; y corresponding labels;
Parameters:k percentiles depth - the node depth (starts at the root with 0)
Max Depth - maximum tree depth
$n \leftarrow$ size of X ;
$n_i \leftarrow$ number of instances with label i in X ;
if $n \leq 1$ or depth = MaxDepth or all $\exists i : n_i = n$ **then**
  $c^* \leftarrow \text{argmax}_{c_i} \frac{n_i}{n}$ (majority class) ;
  Create leaf node, label with class c^*, and corresponding probability $\frac{n_{c^*}}{n}$;
  return;
end if
$B_{j^*}, \tilde{B}_{j^*}, X^L_{j^*}, y^L_{j^*}, X^R_{j^*}, y^R_{j^*} \leftarrow \text{GameSplit}(X, y, \kappa)$ (Algorithm 1) ;
NE-DT $\left(X^L_{j^*}, y^L_{j^*,k};\ \text{depth } + 1;\ \text{MaxDepth}\right)$;
NE-DT $\left(X^R_{j^*}, y^R_{j^*}, k;\ \text{depth } + 1;\ \text{MaxDepth}\right)$;

MaxDepth:Maximum number of levels in the tree.

MATLAB Code: `GameSplit`

```
%caption=GameSplit Function]
function [j_star, beta_j_star, beta_tilde_j_star, XL, yL, XR, yR] =
    GameSplit(X, y, k)

% GameSplit: Determines the best attribute to split a node using Nash
    Equilibrium
% Inputs:

%   X      - data matrix (n x d)
%   y      - binary labels (0 or 1)
%   k  - percentile threshold (e.g., 0.2)
% Outputs:
%   j_star - index of the best split attribute
%   beta_j_star - [beta0, beta1] for the chosen split
%   beta_tilde_j_star - split threshold value
%   XL, yL - data and labels for the left child
%   XR, yR - data and labels for the right child

% Initialization
[n, d] = size(X);    % n = number of samples, d = number of
                     %features
n0 = sum(y == 0);    % number of negative class samples
n1 = sum(y == 1);    % number of positive class samples
C = zeros(1, d);     % to store split quality (Gini index) for
                     % each attribute


    for j = 1:d
         % Compute B parameters
        B0 = (1/8) * (n1 - n0);
        B1 = (1/8) * sum(2 * X(:, j) .* y - X(:, j));
        B = [B0, B1];

        %  Project data using beta
        x_tilde = X(:, j) * B1 + B0;
        % Separate projected data by class
        X_tilde0 = x_tilde(y == 0);
        X_tilde1 = x_tilde(y == 1);

        %Compute split threshold using percentiles
        Pk0 = prctile(X_tilde0, k * 100);
        Pk1 = prctile(X_tilde1, (1 - k) * 100);
        B_tilde = 0.5 * (Pk0 + Pk1);

        %Partition the data based on the threshold
        idxL = x_tilde <= B_tilde;
        idxR = x_tilde >= B_tilde;

        XL_j = X(idxL, :);
        yL_j = y(idxL);
```

```
        XR_j = X(idxR, :);
        yR_j = y(idxR);

        % Compute Gini impurity for each split
        % pL: positiveRatioLeft
        % pR: positiveRatioRight
        pL = sum(yL_j == 1) / max(1, length(yL_j));
        pR = sum(yR_j == 1) / max(1, length(yR_j));
        giniL = 1 - (pL^2 + (1 - pL)^2);
        giniR = 1 - (pR^2 + (1 - pR)^2);
        C(j) = (length(yL_j) * giniL + length(yR_j) * giniR) / n;

        %Store results for current attribute j
        B_all{j} = B;
        B_tilde_all(j) = B_tilde;
        XL_all{j} = XL_j;
        XR_all{j} = XR_j;
        yL_all{j} = yL_j;
        yR_all{j} = yR_j;
    end

   %Select attribute with minimum impurity
    [~, j_star] = min(C); % minimize the impurity

    %Output best parameters and split subsets
    B_j_star = B_all{j_star};
    B_tilde_j_star = B_tilde_all(j_star);
    XL = XL_all{j_star};
    XR = XR_all{j_star};
    yL = yL_all{j_star};
    yR = yR_all{j_star};
end
```

MATLAB Code: `NE_DT`

```
%caption={Recursive NE\_DT function for Nash Equilibrium Decision Tree}
function tree = NE_DT(X, y, k, depth, MaxDepth)
    tree = struct();

% NE_DT: Builds a decision tree using Nash Equilibrium-based splitting
% Inputs:
%   X        - feature matrix (n x d)
%   y        - binary labels (0 or 1)
%   k    - percentile for threshold in GameSplit
%   depth    - current depth of the tree
%   MaxDepth - maximum depth allowed
% Output:
%   tree     - struct representing a node in the decision tree

    % Stopping conditions
    if size(X, 1) <= 1 || depth == MaxDepth || all(y == y(1))
        tree.isLeaf = true;
        tree.class = mode(y);
        return;
    end

    % Call GameSplit to find the best split attribute and threshold
    [j_star, B, beta_tilde, XL, yL, XR, yR] = GameSplit(X, y, k);

    % Store split information in current node

    tree.isLeaf = false;
    tree.attribute = j_star;
    tree.B = beta;
    tree.B_tilde = beta_tilde;

        % Recursively build left and right subtrees
    tree.left = NE_DT(XL, yL, k, depth + 1, MaxDepth);
    tree.right = NE_DT(XR, yR, k, depth + 1, MaxDepth);
end
```

3.4 Application Example: A Simplified Dataset

To demonstrate the practical application of the proposed "Nash Equilibrium Decision Tree (NE-DT) model, we present a simplified binary classification example using a small subset of well-known Iris dataset.

The original Iris dataset contains three classes of Iris plants, each described by four numerical attributes. For this example, we select only two classes (Iris setosa and Iris versicolar) and include four representative instances to facilitate top by step analysis and clarity of computation.

The selected subset is Shown below:

notance	Sepal length	Sepal width	Petal length	Petal Width	Class
1	5,1	3,5	1,4	0,2	Iris setosa
2	4,9	3,0	1,4	0,2	Iris setosa
3	6,6	2,9	4,6	1,3	Iris versicolor
4	5,2	2,7	3,9	1,4	Iris versicolor

For binary classification, the class labels are encoded as follower:

Class	Encoded & label
Iris setosa	0
Iris versicolor	1

Purpose of This Example

This small dataset was chosen to clearly illustrate how the NE-DT algorithm works, both mathematically and logically. The use of only two classes ensures the applicability of binary classification logic, while the small number of instances allows for a detailed, manual Computation of all key steps in the algorithm, including:

- Calculation of the Nash-based Splitting parameters β.
- Transformation of attribute values.
- Determination of the optimal Splitting threshold $\widetilde{\beta_j}$.
- Data partitioning based on the decision rule.

Dataset Description

Number of instances: 4.

Features: 4 Continuous numerical attributes.

Target classes: 0 (Iris setosa), 1 (Iris versicolor).

Tash: Binary classification using the Nash equilibrium Decision True (NE-DT) model.

3.4.1 Mathematical Analysis of the Splitting Process

Computation of B parameters

The decision Split in NE-DT is based on a two player game representing the left and right child nodes. The Splitting rule is obtained by computing the Nash Equilibrium of this game.

The parameters $B_{j,0}$ and $B_{j,1}$ for each attribute j as computed as:

$$B_{j,0} = \frac{1}{8}(n_1 - n_0)$$
$$B_{j,1} = \frac{1}{8}\sum_{i=1}^{n}(2y_i - 1)\,x_{ij}$$

with:

$n_0 = 2$ (Iris setosa)
$n_1 = 2$ (Iris versicolor)

Thus $B_{j,0} = 0$ because $B_{j,0} = \frac{1}{8}(2-2) = 0$.
We apply the game $\Gamma(x, y/j)$ for each feature.
For Sepal length:

$$B_{j,1} = \frac{1}{8}[(2\times 0-1)\times 5,1 + (2\times 0-1)\times 4,9 + (2\times 1-1)\times 6,6 + (2\times 1-1)\times 5,2]$$
$$B_{j,1} = \frac{1}{8}[-5,1 - 4,9 + 6,6 + 5,2]$$
$$B_{j,1} = 0,225$$

For Sepal width:

$$B_{j,1} = \frac{1}{8}[(2\times 0-1)\times 3,5 + (2\times 0-1)\times 3,0 + (2\times 1-1)\times 2,9 + (2\times 1-1)\times 2,7)$$
$$B_{j,1} = \frac{1}{8}[-3,5 - 3,0 + 2,9 + 2,7]$$
$$B_{j1} = -0,1125$$

For Petal length:

$$B_{j,1} = \frac{1}{8}[(2\times 0-1)\times 1,4 + (2\times 0-1)\times 1,4 + (2\times 1,1)\times 4,6 + (2\times 1-1)\times 3,9)$$
$$B_{j,1} = \frac{1}{8}[-1,4 - 1,4 + 4,6 + 3,9]$$
$$B_{j,1} = 0,7125$$

For Petal width:

$$B_{j,1} = \frac{1}{8}[(2\times 0-1)\times 2 + (2\times 0-1)\times 0,2 + (2\times 1-1)\times 1,3 + (2\times 1-1)\times 1,4]$$

$$B_{j,1} = \frac{1}{8}[-0,2,2+1,3+1,4]$$
$$B_{j,1} = 0,2875$$

The computed values are as follows:

Attribute	$B_{j,0}$	$B_{j,1}$
Sepal length	0	$0,2225$
Sepal width	0	-0,1125
Petal length	0	0,7125
Petal width	0	0,2875

Each data point is transformed using:

$$\tilde{x}_{ij} = x_{ij} \cdot B_{j,1} + B_{j,0}$$

where:

$$B_{j,0} = 0$$

Then:

$$\tilde{x}_{ij} = x_{ij} \cdot \beta_{j,1}$$

this transformation shifts the instances in a way that emphasizes class separation under the Nash Equilibrium:

Instance	Class	Sepal length	Sepal width	Petal length	Petal width
1	Iris setosa (0)	$5,1 \times 0,225 = 1,1475$	$3,5 \times -0,1125 = -0,3938$	$1,4 \times 0,7125 = 0,9975$	$0,2 \times 0,28 = 0,0575$
2	Iris setosa (0)	$4,9 \times 0,225 = 1,1025$	$3,0 \times -0,1125 = -0,3375$	$1,4 \times 0,7125 = 0,9975$	$0,2 \times 0,287 = 0,0575$
3	Iris versi-color (1)	$6,6 \times 0,225 = 1,4850$	$2,9 \times -0,1125 = -0,3263$	$4,6 \times 0,7125 = 3,2775$	$1,3 \times 0,287 = 0,3738$
4	Iris versi-color (1)	$5,2 \times 0,225 = 1,1700$	$2,7 \times -0,1125 = -0,3038$	$3, \times 0,7125 = 2,7790$	$1,4 \times 0,28 = 0,4025$

For each attribute, the Splitting point is defined as:

$$\tilde{B}_j = \frac{1}{2} P_{0,5}\left(\tilde{X}_0\right) + P_{0,5}\left(\tilde{X}_1\right)$$

where:

$\tilde{X}_0$: transformed values for class 0.

$\tilde{X}_1$: transformed values for class 1.

$P_{0,5}$: median function.

Attribute	Median $(\tilde{X}_0 = 0)$	Median $(\tilde{X}_1 = 1)$	$\tilde{\beta}_j$
Sepal length	$(1,1475 + 1,1025)/2$ $= 1,1250$	$(1,4850 + 1,1700)/2$ $= 1,3275$	$(1,1250 + 1,3275)/2$ $= 1,2263$
Sepal with	$(-0,3938 - 0,3375)/2$ $= -0,3656$	$(-0,3263 - 0,3038)/2$ $= -0,3150$	$(-0,3656 - 0,3150)/2$ $= -0,3403$
Petal length	$(0,9975 + 0,9975)/2$ $= 0,9975$	$(3,2775 + 2,7790)/2$ $= 3,0283$	$(0,9975 + 3,023)$ $= 2,0229$
Petal	$(0,0575 + 0,0575)/2$ $= 0,0575$	$(0,3738 + 0,4025)/2$ $= 0,3881$	$(0,0575 + 0,3831)$ $= 0,2228$

In this example, the Petal length attribute allows a perfect separation between the two classes. According to purity measures (e.g entropy or Gini index), this attribute would be selected to split the root node.

For a given attribute j, and for each data instance i, the Splitting decision is made based on the following condition:

Left Branch: $\tilde{x}_{ij} \leqslant \tilde{B}_j$.

Right Branch: $\tilde{x}_{ij} > \tilde{B}_j$.

Based on the Petal length transformation:

Instance	Transformed value	Class	Decision Branch
1	0,9975	Iris setosa (0)	left
2	0,9975	Iris setosa (0)	left
3	3,2775	Iris versicolor (1)	Right
4	2,7790	Iris versicolor (1)	Right

Resulting: Decision Tree

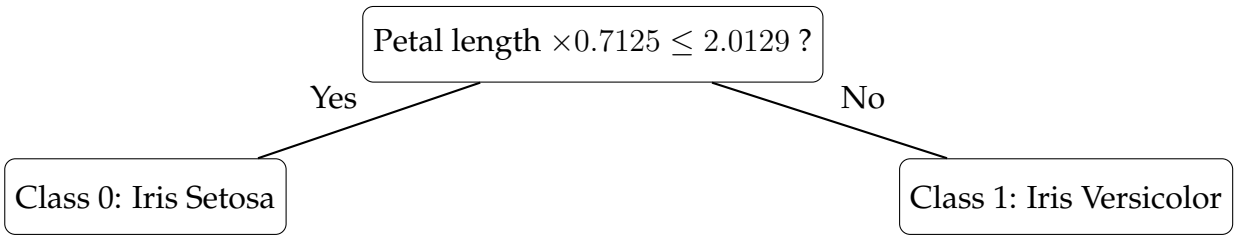


Figure 3.1: Decision tree resulting from Nash-based split on Petal length

Discussion of the Result (Figure 3.1): Figure 3.1 illustrates the result of applying the Nash Equilibrium Decision Tree (NE-DT) method to a binary classification task, where the first split is performed on the feature *Petal Length*. The result demonstrates that this feature provides a strong discriminatory power between the two classes. Unlike traditional decision tree algorithms that rely on greedy heuristics such as Gini impurity or information gain, the NE-DT approach formulates the splitting process as a two-player game between the classes. Each class acts as a player, proposing a strategy (hyperplane) to maximize its separation from the other while minimizing classification complexity. The equilibrium is reached when neither class can improve its strategy unilaterally, resulting in a final splitting point defined by the average of both strategies. The clear split shown in the figure highlights the effectiveness of this game theoretic approach, which yields a balanced and optimal division of the data. This result confirms that the Nash-based splitting criterion can capture meaningful patterns in the data, reduce classification errors, and enhance the generalization capability of the model.

Code Matlab

```
% --- Dataset (first 4 rows: attributes + binary labels) ---
X = [
    5.1, 3.5, 1.4, 0.2;  % Iris-setosa
    4.9, 3.0, 1.4, 0.2;  % Iris-setosa
    6.6, 2.9, 4.6, 1.3;  % Iris-versicolor
    5.2, 2.7, 3.9, 1.4   % Iris-versicolor
];

y = [0; 0; 1; 1];  % Binary labels: 0 = Iris-setosa, 1 = Iris-
   versicolor

% --- Initialization ---
n0 = sum(y == 0);
n1 = sum(y == 1);
beta_0 = (n1 - n0) / 8;  % \beta_{j,0} is constant for all attributes

[n, d] = size(X);
beta_1 = zeros(1, d);
split_point = zeros(1, d);

% --- Nash Equilibrium Calculation for each attribute ---
for j = 1:d
    % Step 2: Compute \beta_{j,1}
    beta_1(j) = (1/8) * sum((2 * y - 1) .* X(:, j));

    % Step 3: Compute transformed values
    x_tilde = X(:, j) * beta_1(j) + beta_0;

    % Step 4: Class separation
    X0 = x_tilde(y == 0);  % Transformed values for class 0
    X1 = x_tilde(y == 1);  % Transformed values for class 1

    % Compute medians (k = 0.5)
    med0 = median(X0);
    med1 = median(X1);

    % Step 4: Calculate the optimal split point
    split_point(j) = 0.5 * (med0 + med1);

    fprintf("Attribute %d - \beta_1 = %.4f, Split point = %.4f\n", j,
       beta_1(j), split_point(j));
end
```

```
% --- Choose the most discriminating attribute (e.g., Petal.Length =
    column 3) ---
j = 3;  % Example: using Petal.Length

beta_j1 = beta_1(j);
split = split_point(j);

fprintf('\n--- Splitting with Attribute %d (\beta_1 = %.4f, Split = %.4
    f) ---\n', j, \beta_j1, split);

% --- Apply the splitting rule to all instances ---
for i = 1:n
    x_transformed = X(i, j) * beta_j1 + beta_0;

    if x_transformed <= split
        side = 'Left';
    else
        side = 'Right';
    end

    fprintf('Instance %d (Class %d) \to %.4f \to %s\n', i, y(i),
       x_transformed, side);
end
```

Remark 3.4.1. *This illustrative example demonstrates the NE-DT model's ability to generate stable and interpretable decision boundaries through a game theoric lens. Despite the dataset's simplicity, the method yields a split that align's perfectly with the underlying class distribution confirming the relevance and efficiency of using Nash Equilibrium.*

Figure 3.2 :Shows how the instances of a feature are shifted to improve class separation.The left represents original data points for two classes (red: class 0 , blue: class 1) with overlap and the right one the same data shifted using Nash equilibrium making the classes more separable , the black dashed line is the splitting point , computed from the transformed data to optimally separate the two classes.

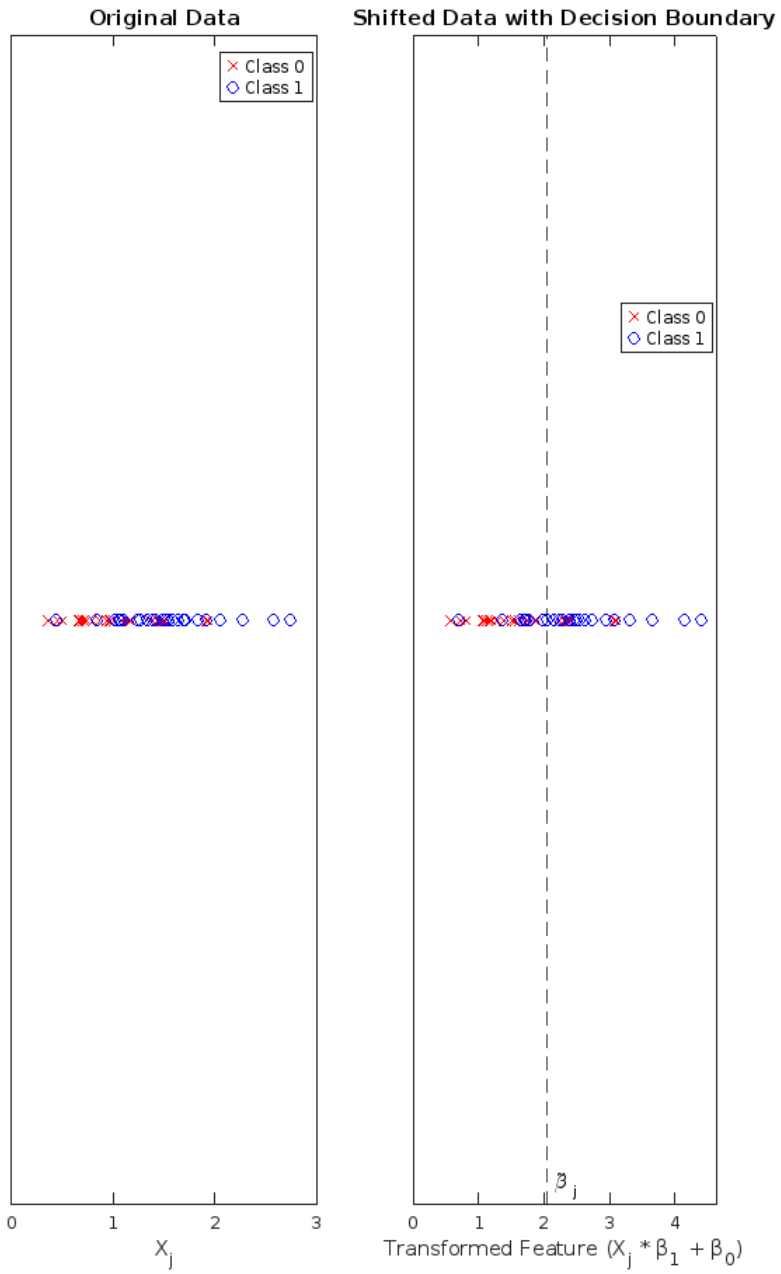


Figure 3.2: Nash Equilibrium Decision Boundary on Transformed Data

3.5 Conclusion

Chapter 3 introduced an innovative approach to binary classification by integrating game theory with decision trees through the Nash Equilibrium Decision Tree (NE-DT) model. By framing the splitting process as a strategic game between sub-nodes, the method ensures optimal class separation while maintaining computational efficiency. The analytical derivation of equilibrium parameters and the robust percentile-based thresholding enhance both accuracy and resilience to outliers. The NE-DT algorithm exemplifies how game-theoretic principles can refine machine learning techniques, offering a structured and interpretable solution for classification tasks. This fusion of disciplines not only advances decision tree methodologies but also paves the way for future explorations in hybrid models for complex decision-making scenarios.

General Conclusion

This memoir has explored an intersection of game theory and machine learning, culminating in the development of the Nash Equilibrium Decision Tree (NE-DT) model for binary classification [13]. Beginning with foundational concepts in game theory such as strategic interactions, Nash equilibrium, and their applications the work established the theoretical groundwork for integrating these principles into machine learning. The discussion then transitioned to core machine learning techniques, with a focus on decision trees and their adaptability for classification tasks.

The NE-DT model, introduced in [13], represents the key element of the master's thesis as it synthesizes the main steps of the approach developed in [13]. By framing the decision tree splitting process as a non-cooperative game, the model leverages Nash equilibrium to derive optimal hyperplanes that maximize class separation. This approach not only enhances interpretability but also improves robustness against outliers. The algorithmic implementation of NE-DT further highlights its practicality for real-world applications.

Beyond its technical contributions, this work underscores the broader potential of interdisciplinary research, illustrating how game theory can refine and advance machine learning methodologies. Future directions could explore extensions to multi-class classification, ensemble methods, or dynamic environments where adaptive strategies are critical.

Bibliography

[1] M. N. Alonso. The Mathematics of Decision Trees and Ensembles: Random Forests and Boosting. Artificial Intelligence Finance Institute. 2024. https://ssrn.com/abstract=4999900.

[2] L. Breiman, J. H. Friedman, R. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth and Brooks/Cole Advanced Books and Software, 1984.

[3] S. Djebara. Contribution to the analyses of a matrix game with uncertain Linear Constraints. These Doctora LMD on Mathematiques option Recherche operationnelle. Mouloud MAAMERI University Tizi-uzou (2024).

[4] C. T. Do. Multi-modal and multi-scale Temporal MetricLearn ing for robust nearst neighbors classification. Doctoral Thesis university of Grenoble. 2016.

[5] B. Drivine, Econs 424 Strategy and game theory - School of Economic Sciences, (WASHiNTON State University).

[6] S. Dutt, S. Chandramouli and A. K. Das. Machine learning, 63-64. (2018).

[7] D. M. Hawkins. Identification of Outliers. Chapman and Hall, 1980.

[8] S. Kakutani, A generalization of Brouwer's Fixed point theorem, *Duke Math. J.* **8** (1941) 457–459.

[9] R. Kumasi and S. K. Srivastava. Machine learning: A review on binary classification. Int J. Comput. App. 160 (7) (2017).

[10] M. MaScheler, S. Zamir, E. Solan. (2020) Game Theory, 2^{nd} edn. Cambridge University Press, New York. Lrtps:/ll doi.org/10. 1017/9781108636049.

[11] J. Nash. Equilibrium points in n-person games. Proc. Nat. Acad. Sci. U.S.A., 36, 48-49 (1950). doi.org/10.1073/pnas.36.1.48

[12] O. Raoof and H. Al-Raweshidy. Theory of Games: an Introduction. Game Theory (2010).

[13] M. A. Suciu and R .L. Lung . A Nash equilibria decision tree for binary classification. Appl Intell 55, 192 (2025). https://doi.org/10.1007/s10489-024-06132-3

[14] P. N. Tan, M. Steinbach and V. Kumar. Introduction to Data Mining. Classification: Basic Concepts, Decision Trees, and Model Evaluation, Addison-Wesley, 2005.

[15] J. Thongkam, G. Xu, Y. Zhang, and F. Huang, "Support Vector Machine for Outlier Detection in Breast Cancer Survivability Prediction," in APWeb 2008 Workshops, Y. Ishikawa et al., Eds., LNCS 4977, page 99–109, Springer, 2008.

[16] L. Wasserman. All of Statistics: A Concise Course in Statistical Inference. Springer, 2004. ISBN: 978-0387402727

[17] Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, 2019,2nd, 9781492032649

Abstract

In this thesis, we present the method in [13] which combines game theory and machine learning to develop a new decision tree model for binary classification, called the Nash Equilibrium Decision Tree (NE-DT). Unlike standard decision trees, NE-DT treats the splitting process as a game between two players (left and right branches), where each tries to best separate the data. The Nash equilibrium is a key concept in game theory ,is used to find the optimal split, improving accuracy and robustness.

The work explains basic game theory and machine learning concepts, then details the NE-DT algorithm with clear mathematical steps. A simple example using the Iris dataset shows how the model works in practice. The results prove that NE-DT effectively separates classes while being easy to understand.

Keywords: Game theory, Nash equilibrium, Decision trees, Binary classification, Machine learning.